Bayesian Optimization with Exponential Convergence

Authored by:

Tom?s Lozano-P?rez
Leslie Pack Kaelbling
Kenji Kawaguchi

Abstract

This paper presents a Bayesian optimization method with exponential convergence without the need of auxiliary optimization and without the delta-cover sampling. Most Bayesian optimization methods require auxiliary optimization: an additional non-convex global optimization problem, which can be time-consuming and hard to implement in practice. Also, the existing Bayesian optimization method with exponential convergence requires access to the delta-cover sampling, which was considered to be impractical. Our approach eliminates both requirements and achieves an exponential convergence rate.

1 Paper Body

We consider a general global optimization problem: maximize f (x) subject to x ? ? ? RD where f : ? ? R is a non-convex black-box deterministic function. Such a problem arises in many realworld applications, such as parameter tuning in machine learning [3], engineering design problems [4], and model parameter fitting in biology [5]. For this problem, one performance measure of an algorithm is the simple regret, rn , which is given by rn = supx?? f (x) ? f (x+) where x+ is the best input vector found by the algorithm. For brevity, we use the term ?regret? to mean simple regret. The general global optimization problem is known to be intractable if we make no further assumptions [6]. The simplest additional assumption to restore tractability is to assume the existence of a bound on the slope of f . A well-known variant of this assumption is Lipschitz continuity with a known Lipschitz constant, and many algorithms have been proposed in this setting [7, 8, 9]. These algorithms successfully guaranteed certain bounds on the regret. However appealing from a theoretical point of view, a practical concern was soon raised regarding the assumption that a tight Lipschitz constant is known. Some researchers relaxed this somewhat strong assumption by proposing procedures to estimate a Lipschitz constant during the

optimization process [10, 11, 12]. Bayesian optimization is an efficient way to relax this assumption of complete knowledge of the Lipschitz constant, and has become a well-recognized method for solving global optimization problems with non-convex black-box functions. In the machine learning community, Bayesian optimization? especially by means of a Gaussian process (GP)?is an active research area [13, 14, 15]. With the requirement of the access to the ?-cover sampling procedure (it samples the function uniformly such that the density of samples doubles in the feasible regions at each iteration), de Freitas et al. [1] recently proposed a theoretical procedure that maintains an exponential convergence rate (exponential regret). However, as pointed out by Wang et al. [2], one remaining problem is to derive a GP-based optimization method with an exponential convergence rate without the ?-cover sampling procedure, which is computationally too demanding in many cases. In this paper, we propose a novel GP-based global optimization algorithm, which maintains an exponential convergence rate and converges rapidly without the ?-cover sampling procedure.
1

2

Gaussian Process Optimization

In Gaussian process optimization, we estimate the distribution over function f and use this information to decide which point of f should be evaluated next. In a parametric approach, we consider a parameterized function f (x; ?), with ? being distributed according to some prior. In contrast, the nonparametric GP approach directly puts the GP prior over f as f (?) ? GP (m(?), ?(?, ?)) where m(?) is the mean function and ?(?, ?) is the covariance function or the kernel. That is, m(x) = E[f (x)] and ?(x, x0) = E[(f (x) ? m(x))(f (x0) ? m(x0))T]. For a finite set of points, the GP model is simply a joint Gaussian: f (x1:N) ? N (m(x1:N), K), where Ki,j = ?(xi , xj) and N is the number of data points. To predict the value of f at a new data point, we first consider the joint distribution over f of the old data points and the new data point:

f (x1:N) f (xN +1)

?N

m(x1:N) m(xN +1) ,

K kT

k ?(xN +1 , xN +1)

where k = ?(x1:N , xN +1) ? RN ?1 . Then, after factorizing the joint distribution using the Schur complement for the joint Gaussian, we obtain the conditional distribution, conditioned on observed entities DN := {x1:N , f (x1:N)} and xN +1 , as: f (xN +1)—DN , xN +1 ? N (?(xN +1 —DN), ? 2 (xN +1 —DN))

where ?(xN +1 —DN) = m(xN +1) + kT K?1 (f (x1:N) ? m(x1:N)) and ? 2 (xN +1 —DN) = ?(xN +1 , xN +1) ? kT K?1 k. One advantage of GP is that this closed-form solution simplifies both its analysis and implementation. To use a GP, we must specify the mean function and the covariance function. The mean function is usually set to be zero. With this zero mean function, the conditional mean ?(xN +1 —DN) can still be flexibly specified by the covariance function, as shown in the above equation for ?. For the covariance

function, there are several common choices, including the Matern kernel and the Gaussian

T kernel. For example, the Gaussian kernel is defined as ?(x, x0) = exp ? 12 (x ? x0) ??1 (x ? x0)

where ??1 is the kernel parameter matrix. The kernel parameters or hyper-parameters can be estimated by empirical Bayesian methods [16]; see [17] for more information about GP.

The flexibility and simplicity of the GP prior make it a common choice for continuous objective functions in the Bayesian optimization literature. Bayesian optimization with GP selects the next query point that optimizes the acquisition function generated by GP. Commonly used acquisition functions include the upper confidence bound (UCB) and expected improvement (EI). For brevity, we consider Bayesian optimization with UCB, which works as follows. At each iteration, the UCB function U is maintained as U (x—DN) = ?(x—DN) + ??(x—DN) where ? ? R is a parameter of the algorithm. To find the next query xn+1 for the objective function f , GP-UCB solves an additional non-convex optimization problem with U as xN +1 = arg maxx U (x—DN). This is often carried out by other global optimization methods such as DIRECT and CMA-ES. The justification for introducing a new optimization problem lies in the assumption that the cost of evaluating the objective function f dominates that of solving additional optimization problem. For deterministic function, de Freitas et al. [1] recently presented a theoretical procedure that maintains exponential convergence rate. However, their own paper and the follow-up research [1, 2] point out that this result relies on an impractical sampling procedure, the ?-cover sampling. To overcome this issue, Wang et al. [2] combined GP-UCB with a hierarchical partitioning optimization method, the SOO algorithm [18], providing a regret bound with polynomial dependence on the number of function evaluations. They concluded that creating a GP-based algorithm with an exponential convergence rate without the impractical sampling procedure remained an open problem.

3 3.1

Infinite-Metric GP Optimization Overview

The GP-UCB algorithm can be seen as a member of the class of bound-based search methods, which includes Lipschitz optimization, A* search, and PAC-MDP algorithms with optimism in the face of uncertainty. Bound-based search methods have a common property: the tightness of the bound determines its effectiveness. The tighter the bound is, the better the performance becomes. 2

However, it is often difficult to obtain a tight bound while maintaining correctness. For example, in A* search, admissible heuristics maintain the correctness of the bound, but the estimated bound with admissibility is often too loose in practice, resulting in a long period of global search. The GP-UCB algorithm has the same problem. The bound in GP-UCB is represented by UCB, which has the following property: f (x) ? U (x—D) with some probability. We formalize this property in the analysis of our algorithm. The problem is essentially due to the difficulty of obtaining a tight bound U (x—D) such that f (x) ? U (x—D)

and f (x) ? U (x—D) (with some probability). Our solution strategy is to first admit that the bound encoded in GP prior may not be tight enough to be useful by itself. Instead of relying on a single bound given by the GP, we leverage the existence of an unknown bound encoded in the continuity at a global optimizer. Assumption 1. (Unknown Bound) There exists a global optimizer x? and an unknown semi-metric ‘ such that for all x ? ?, f (x?) ? f (x) + ‘ (x, x?) and ‘ (x, x?) ¡ ?. In other words, we do not expect the known upper bound due to GP to be tight, but instead expect that there exists some unknown bound that might be tighter. Notice that in the case where the bound by GP is as tight as the unknown bound by semi-metric ‘ in Assumption 1, our method still maintains an exponential convergence rate and an advantage over GP-UCB (no need for auxiliary optimization). Our method is expected to become relatively much better when the known bound due to GP is less tight compared to the unknown bound by ‘. As the semi-metric ‘ is unknown, there are infinitely many possible candidates that we can think of for ‘. Accordingly, we simultaneously conduct global and local searches based on all the candidates of the bounds. The bound estimated by GP is used to reduce the number of candidates. Since the bound estimated by GP is known, we can ignore the candidates of the bounds that are looser than the bound estimated by GP. The source code of the proposed algorithm is publicly available at http://lis.csail.mit.edu/code/imgpo.html. 3.2

Description of Algorithm

Figure 1 illustrates how the algorithm works with a simple 1-dimensional objective function. We employ hierarchical partitioning to maintain hyperintervals, as illustrated by the line segments in the figure. We consider a hyperrectangle as our hyperinterval, with its center being the evaluation point of f (blue points in each line segment in Figure 1). For each iteration t, the algorithm performs the following procedure for each interval size: (i) Select the interval with the maximum center value among the intervals of the same size. (ii) Keep the interval selected by (i) if it has a center value greater than that of any larger interval. (iii) Keep the interval accepted by (ii) if it contains a UCB greater than the center value of any smaller interval. (iv) If an interval is accepted by (iii), divide it along with the longest coordinate into three new intervals. (v) For each new interval, if the UCB of the evaluation point is less than the best function value found so far, skip the evaluation and use the UCB value as the center value until the interval is accepted in step (ii) on some future iteration; otherwise, evaluate the center value. (vi) Repeat steps (i)?(v) until every size of intervals are considered Then, at the end of each iteration, the algorithm updates the GP hyperparameters. Here, the purpose of steps (i)?(iii) is to select an interval that might contain the global optimizer. Steps (i) and (ii) select the possible intervals based on the unknown bound by ‘, while Step (iii) does so based on the bound by GP. We now explain the procedure using the example in Figure 1. Let n be the number of divisions of intervals and let N be the number of function evaluations. t is the number of iterations. Initially, there is only one interval (the center of the input region ? ? R) and thus this interval is divided, resulting in the first diagram of Figure 1. At the beginning of iteration t = 2 , step (i) selects the third interval from the left side in the first diagram (t = 1,

n = 2), as its center value is the maximum. Because there are no intervals of different size at this point, steps (ii) and (iii) are skipped. Step (iv) divides the third interval, and then the GP hyperparameters are updated, resulting in the second 3

Figure 1: An illustration of IMGPO: t is the number of iteration, n is the number of divisions (or splits), N is the number of function evaluations. diagram (t = 2, n = 3). At the beginning of iteration t = 3, it starts conducting steps (i)?(v) for the largest intervals. Step (i) selects the second interval from the left side and step (ii) is skipped. Step (iii) accepts the second interval, because the UCB within this interval is no less than the center value of the smaller intervals, resulting in the third diagram (t = 3, n = 4). Iteration t = 3 continues by conducting steps (i)?(v) for the smaller intervals. Step (i) selects the second interval from the left side, step (ii) accepts it, and step (iii) is skipped, resulting in the forth diagram (t = 3, n = 4). The effect of the step (v) can be seen in the diagrams for iteration t = 9. At n = 16, the far right interval is divided, but no function evaluation occurs. Instead, UCB values given by GP are placed in the new intervals indicated by the red asterisks. One of the temporary dummy values is resolved at n = 17 when the interval is queried for division, as shown by the green asterisk. The effect of step (iii) for the rejection case is illustrated in the last diagram for iteration t = 10. At n = 18, t is increased to 10 from 9, meaning that the largest intervals are first considered for division. However, the three largest intervals are all rejected in step (iii), resulting in the division of a very small interval near the global optimum at n = 18. 3.3

Technical Detail of Algorithm

We define h to be the depth of the hierarchical partitioning tree, and ch,i to be the center point of the ith hyperrectangle at depth h. Ngp is the number of the GP evaluations. Define depth(T) to be the largest integer h such that the set Th is not empty. To compute UCB U , we use ?M = p 2 log(? 2 M 2 /12?) where M is the number of the calls made so far for U (i.e., each time we use U , we increment M by one). This particular form of ?M is to maintain the property of f (x) ? U (x—D) during an execution of our algorithm with probability at least 1 ? ?. Here, ? is the parameter of IMGPO. ?max is another parameter, but it is only used to limit the possibly long computation of step (iii) (in the worst case, step (iii) computes UCBs 3?max times although it would rarely happen). The pseudocode is shown in Algorithm 1. Lines 8 to 23 correspond to steps (i)-(iii). These lines compute the index i?h of the candidate of the rectangle that may contain a global optimizer for each depth h. For each depth h, non-null index i?h at Line 24 indicates the remaining candidate of a rectangle that we want to divide. Lines 24 to 33 correspond to steps (iv)-(v) where the remaining candidates of the rectangles for all h are divided. To provide a simple executable division scheme (line 29), we assume ? to be a hyperrectangle (see the last paragraph of section 4 for a general case). Lines 8 to 17 correspond to steps (i)-(ii). Specifically, line 10 implements step (i) where a single candidate is selected for each depth, and lines 11 to 12 conduct step (ii) where some candidates are screened out. Lines 13 to 17 resolve the the temporary dummy values computed by GP. Lines 18 0 (ch,i?h) to 23 correspond to step (iii) where

the candidates are further screened out. At line 21, Th+? indicates the set of all center points of a fully expanded tree until depth h + ? within the region 0 (ch,i?h) contains the nodes of covered by the hyperrectangle centered at ch,i?h . In other words, Th+? the fully expanded tree rooted at ch,i?h with depth ? and can be computed by dividing the current rectangle at ch,i?h and recursively divide all the resulting new rectangles until depth ? (i.e., depth ? from ch,i?h , which is depth h + ? in the whole tree). 4

Algorithm 1 Infinite-Metric GP Optimization (IMGPO) Input: an objective function f , the search domain ?, the GP kernel ?, ?max ? N+ and ? ? (0, 1) 1: Initialize the set Th = {?} ?h ? 0 2: Set c0,0 to be the center point of ? and T0 ? {c0,0 } 3: Evaluate f at c0,0 : g(c0,0) ? f (c0,0)

4: f + ? g(c0,0), D ? {(c0,0 , g(c0,0))} 5: n, N ? 1, Ngp ? 0, ? ? 1 6: for t = 1, 2, 3, ... do 7: ?max ? ?? for h = 0 to depth(T) do # for-loop for steps (i)-(ii) 8: 9: while true do 10: i?h ? arg maxi:ch,i ?Th g(ch,i) if g(ch,i?h) ¡ ?max then 11: 12: i?h ? ?, break 13: else if g(ch,i?h) is not labeled as GP-based then 14: ?max ? g(ch,i?h), break else 15: 16: g(ch,i?h) ? f (ch,i?h) and remove the GP-based label from g(ch,i?h) 17: N ? N + 1, Ngp ? Ngp ? 1 18: D ? {D, (ch,i?h , g(ch,i?h))} 19: for h = 0 to depth(T) do # for-loop for step (iii) if i?h 6= ? then 20: 21: ? ? the smallest positive integer s.t. i?h+? 6= ? and ? ? min(?, ?max) if exists, and 0 otherwise 0 22: z(h, i?h) = maxk:ch+?,k ?Th+? (ch,i?) U (ch+?,k —D) h

23: 24: 25: 26: 27: 28: 29:

30: 31: 32: 33: 34: 35: 36: 37: 38: 39:

3.4

if ? 6= 0 and z(h, i?h) ¡ g(ch+?,i?h+?) then i?h ? ?, break ?max ? ?? for h = 0 to depth(T) do # for-loop for steps (iv)-(v) if i?h 6= ? and g(ch,i?h) ? ?max then n ? n + 1. Divide the hyperrectangle centered at ch,i?h along with the longest coordinate into three new hyperrectangles with the following centers: S = {ch+1,i(lef t) , ch+1,i(center) , ch+1,i(right) } Th+1 ? {Th+1 , S} Th ? Th ch,i?h , g(ch+1,i(center)) ? g(ch,i?h) for inew = {i(lef t), i(right)} do if U (ch+1,inew —D) ? f + then g(ch+1,inew) ? f (ch+1,inew) D ? {D, (ch+1,inew , g(ch+1,inew))} N ? N + 1, f + ? max(f + , g(ch+1,inew)), ?max = max(?max , g(ch+1,inew))

else

g(ch+1,inew) ? U (ch+1,inew —D) and label g(ch+1,inew) as GP-based. Ngp ? Ngp + 1 Update ?: if f + was updated, ? ? ? + 22 , and otherwise, ? ? max(? ? 2?1 , 1) Update GP hyperparameters by an empirical Bayesian method

Relationship to Previous Algorithms

The most closely related algorithm is the BaMSOO algorithm [2], which combines SOO with GPUCB. However, it only achieves a polynomial regret bound while IMGPO achieves a exponential regret bound. IMGPO can achieve exponential regret because it utilizes the information encoded in the GP prior/posterior to reduce the degree of the unknownness of the semi-metric ‘. The idea of considering a set of infinitely many bounds was first proposed by Jones et al. [19]. Their DIRECT algorithm has been successfully applied to real-world problems

[4, 5], but it only maintains the consistency property (i.e., convergence in the limit) from a theoretical viewpoint. DIRECT takes an input parameter to balance the global and local search efforts. This idea was generalized to the case of an unknown semi-metric and strengthened with a theoretical support (finite regret bound) by 5

Munos [18] in the SOO algorithm. By limiting the depth of the search tree with a parameter hmax , the SOO algorithm achieves a finite regret bound that depends on the near-optimality dimension.

4

Analysis

In this section, we prove an exponential convergence rate of IMGPO and theoretically discuss the reason why the novel idea underling IMGPO is beneficial. The proofs are provided in the supplementary material. To examine the effect of considering infinitely many possible candidates of the bounds, we introduce the following term. Definition 1. (Infinite-metric exploration loss). The infinite-metric exploration loss ?t is the number of intervals to be divided during iteration t. Pdepth(T) 1(i?h 6= ?) at line The infinite-metric exploration loss ?? can be computed as ?t = h=1 25. It is the cost (in terms of the number of function evaluations) incurred by not committing to any particular upper bound. If we were to rely on a specific bound, ?? would be minimized to 1. For example, the DOO algorithm [18] has ?t = 1 ?t ? 1. Even if we know a particular upper bound, relying on this knowledge and thus minimizing ?? is not a good option unless the known bound is tight enough compared to the unknown bound leveraged in our algorithm. This will be clarified in our analysis. Let ??t be the maximum of the averages of ?1:t0 for t0 = 1, 2, ..., t (i.e., P t0 ??t ? max({ t10 ? =1 ?? ; t0 = 1, 2, ..., t}).

Assumption 2. There exist L ¿ 0, ? ¿ 0 and p ? 1 in R such that for all x, x0 ? ?, ‘(x0 , x) ? L——x0 ? x——? p.

In Theorem 1, we show that the exponential convergence rate O ?N +Ngp with ? ¡ 1 is achieved. We define ?n ? ?max to be the largest ? used so far with n total node expansions. For simplicity, we assume that ? is a square, which we satisfied in our experiments by scaling original ?. ?

Theorem 1. Assume Assumptions 1 and 2. Let ? = supx,x0 ?? 12 kx ? x0 k? . Let ? = 3? 2CD??t ¡ 1. Then, with probability at least 1 ? ?, the regret of IMGPO is bounded as

N + Ngp rN ? L(3?D 1/p)? exp ?? ? ?n ? 2 ln 3 = O ?N +Ngp . 2CD ??t

Importantly, our bound holds for the best values of the unknown L, ? and p even though these values are not given. The closest result in previous work is that of BaMSOO [2], which obtained 2? ? ? D(4??)) with probability 1 ? ? for ? = {1, 2}. As can be seen, we have improved the regret O(n bound. Additionally, in our analysis, we can see how L, p, and ? affect the bound, allowing us to view the inherent difficulty of an objective function in a theoretical perspective. Here, C is a constant in N and is used in previous work [18, 2]. For example, if we conduct 2D or 3D ? 1 function evaluations per node-expansion and if p = ?, we have that C = 1. We note that ? can get close to one as input dimension D increases, which suggests that there is a remaining

challenge in scalability for higher dimensionality. One strategy for addressing this problem would be to leverage additional assumptions such as those in [14, 20].

Remark 1. (The effect of the tightness of UCB by GP) UCB computed by If GP is ?useful? such N +N that N/? ?t = ?(N), then our regret bound becomes O exp ? 2CDgp ? ln 3 . If the bound due to Pt up to O(N/t) UCB by GP is too loose (and thus useless), ??tcan increase

(due to ??t ? i=1 i/t ? t(1+Ngp /N) ? ln 3 , which can be bounded O(N/t)), resulting in the regret bound of O exp ? 2CD

N +N by O exp ? 2CDgp max(?1N , Nt)? ln 3 1 . This is still better than the known results.

Remark 2. (The effect of GP) Without the use of GP, our regret bound would be as follows: rN ? N 1 L(3?D 1/p)? exp(??[2CD ?t ? ??t is the infinite-metric exploration loss without ??t ?2] ln 3), where ?

?). Our proof works with this This can be done by limiting the depth of search tree as depth(T) = O(N ? additional mechanism, but results in the regret bound with N being replaced by N . Thus, if we assume to ? have at least ?not useless? UCBs such that N/? ?t = ?(N), this additional mechanism can be disadvantageous. Accordingly, we do not adopt it in our experiments. 1

6

GP. Therefore, the use of GP reduces the regret bound by increasing Ngp and decreasing ??t , but may potentially increase the bound by increasing ?n ? ?. Remark 3. (The effect of infinite-metric optimization) To understand the effect of considering all the possible upper bounds, we consider the case without GP. If we consider all the possible bounds, N 1 ? 2] ln 3) for the best unknown L, ? and p. we have the regret bound L(3?D1/p)? exp(??[2CD ? ?t 0 0 For standard optimization with a estimated bound, we have L0 (3?D1/p)? exp(??0 [2CN0 D ? 2] ln 3) 0 0 0 for an estimated L , ? , and p . By algebraic manipulation, considering all the possible bounds has a better regret when ???1 t ?

2CD N N ln 3? ((2C 0 D

0

0

1/p0 ?0

(3?D) ? 2) ln 3? + 2 ln 3? ? ln LL(3?D 1/p)?). For an intuitive 0

?/p0

Cc2 D LD insight, we can simplify the above by assuming ?0 = ? and C 0 = C as ???1 t ? 1 ? N ln LD ?/p . Because L and p are the ones that achieve the lowest bound, the logarithm on the right-hand side is always non-negative. Hence, ??t = 1 always satisfies the condition. When L0 and p0 are not tight enough, the logarithmic term increases in magnitude, allowing ??t to increase. For example, if the second term on the right-hand side has a magnitude of greater than 0.5, then ??t = 2 satisfies the inequality. Therefore, even if we know the upper bound of the function, we can see that it may be better not to rely on this, but rather take the infinite many possibilities into account.

One may improve the algorithm with different division procedures than one presented in Algorithm 1. Accordingly, in the supplementary material, we derive an abstract version of the regret bound for IMGPO with a family of division procedures that satisfy some assumptions. This information could be used to design a new division procedure.

5

Experiments

In this section, we compare the IMGPO algorithm with the SOO, BaMSOO, GP-PI and GP-EI algorithms [18, 2, 3]. In previous work, BaMSOO and GP-UCB were tested with a pair of a handpicked good kernel and hyperparameters for each function [2]. In our experiments, we assume that the knowledge of good kernel and hyperparameters is unavailable, which is usually the case in practice. Thus, for IMGPO, BaMSOO, GP-PI and GP-EI, we simply used one of the p most popular kernels, 0 the isotropic Matern kernel with ? = 5/2. This is given by ?(x, x) = g(5——x ? x0 ——2 /l), where g(z) = ? 2 (1 + z + z 2 /3) exp(?z). Then, we blindly initialized the hyperparameters to ? = 1

(a) Sin1: [1, 1.92, 2]

(b) Sin2: [2, 3.37, 3]

(c) Peaks: [2, 3.14, 4]

(d) Rosenbrock2: [2, 3.41, 4]

(e) Branin: [2, 4.44, 2]

(f) Hartmann3: [3, 4.11, 3]

(g) Hartmann6: [6, 4.39, 4]

(h) Shekel5: [4, 3.95, 4]

(i) Sin1000: [1000, 3.95, 4]

Figure 2: Performance Comparison: in the order, the digits inside of the parentheses [ ] indicate the dimensionality of each function, and the variables ??t and ?n at the end of computation for IMGPO. 7

Table 1: Average CPU time (in seconds) for the experiment with each test function Algorithm GP-PI GP-EI SOO BaMSOO IMGPO

Sin1 29.66 12.74 0.19 43.80 1.61

Sin2 115.90 115.79 0.19 4.61 3.15

Peaks 47.90 44.94 0.24 7.83 4.70

Rosenbrock2 921.82 893.04 0.744 12.09 11.11

Branin 1124.21 1153.49 0.33 14.86 5.73

Hartmann3 573.67 562.08 0.30 14.14 6.80

Hartmann6 657.36 604.93 0.25 26.68 13.47

Shekel5 611.01 558.58 0.29 371.36 15.92

and l = 0.25 for all the experiments; these values were updated with an empirical Bayesian method after each iteration. To compute the UCB by GP, we used ? = 0.05 for IMGPO and BaMSOO. For IMGPO, ?max was fixed to be 22 (the effect of selecting ? different values is discussed later). For BaMSOO and SOO, the parameter hmax was set to n, according to Corollary 4.3 in [18]. For GP-PI and GP-EI, we used the SOO algorithm and a local optimization method using gradients to solve the auxiliary optimization. For SOO, BaMSOO and IMGPO, we used the corresponding deterministic division procedure (given

?, the initial point is fixed and no randomness exists). For GP-PI and GP-EI, we randomly initialized the first evaluation point and report the mean and one standard deviation for 50 runs. The experimental results for eight different objective functions are shown in Figure 2. The vertical axis is log10 (f (x?) ? f (x+)), where f (x?) is the global optima and f (x+) is the best value found by the algorithm. Hence, the lower the plotted value on the vertical axis, the better the algorithm?s performance. The last five functions are standard benchmarks for global optimization [21]. The first two were used in [18] to test SOO, and can be written as fsin1 (x) = (sin(13x) sin +1)/2 for Sin1 and fsin2 (x) = fsin1 (x1)fsin1 (x2) for Sin2. The form of the third function is given in Equation (16) and Figure 2 in [22]. The last function is Sin2 embedded in 1000 dimension in the same manner described in Section 4.1 in [14], which is used here to illustrate a possibility of using IMGPO as a main subroutine to scale up to higher dimensions with additional assumptions. For this function, we used REMBO [14] with IMGPO and BaMSOO as its Bayesian optimization subroutine. All of these functions are multimodal, except for Rosenbrock2, with dimensionality from 1 to 1000. As we can see from Figure 2, IMGPO outperformed the other algorithms in general. SOO produced the competitive results for Rosenbrock2 because our GP prior was misleading (i.e., it did not model the objective function well and thus the property f (x) ? U (x—D) did not hold many times). As can be seen in Table 1, IMGPO is much faster than traditional GP optimization methods although it is slower than SOO. For Sin 1, Sin2, Branin and Hartmann3, increasing ?max does not affect IMGPO because ?n did not reach ?max = 22 (Figure 2). For the rest of the test functions, we would be able to improve the performance of IMGPO by increasing ?max at the cost of extra CPU time.

6

Conclusion

We have presented the first GP-based optimization method with an exponential convergence rate

O ?N +Ngp (? ¡ 1) without the need of auxiliary optimization and the ?-cover sampling. Perhaps more importantly in the viewpoint of a broader global optimization community, we have provided a practically oriented analysis framework, enabling us to see why not relying on a particular bound is advantageous, and how a non-tight bound can still be useful (in Remarks 1, 2 and 3). Following the advent of the DIRECT algorithm, the literature diverged along two paths, one with a particular bound and one without. GP-UCB can be categorized into the former. Our approach illustrates the benefits of combining these two paths. As stated in Section 3.1, our solution idea was to use a bound-based method but rely less on the estimated bound by considering all the possible bounds. It would be interesting to see if a similar principle can be applicable to other types of bound-based methods such as planning algorithms (e.g., A* search and the UCT or FSSS algorithm [23]) and learning algorithms (e.g., PAC-MDP algorithms [24]). Acknowledgments The authors would like to thank Dr. Remi Munos for his thoughtful comments and suggestions. We gratefully acknowledge support from NSF grant 1420927, from ONR grant N00014-14-1-0486, and

from ARO grant W911NF1410433. Kenji Kawaguchi was supported in part by the Funai Overseas Scholarship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors. 8

2 References

[1] N. De Freitas, A. J. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In Proceedings of the 29th International Conference on Machine Learning (ICML), 2012. [2] Z. Wang, B. Shakibi, L. Jin, and N. de Freitas. Bayesian Multi-Scale Optimistic Optimization. In Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTAT), pages 1005?1014, 2014. [3] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In Proceedings of Advances in Neural Information Processing Systems (NIPS), pages 2951?2959, 2012. [4] R. G. Carter, J. M. Gablonsky, A. Patrick, C. T. Kelley, and O. J. Eslinger. Algorithms for noisy problems in gas transmission pipeline optimization. Optimization and engineering, 2(2):139?157, 2001. [5] J. W. Zwolak, J. J. Tyson, and L. T. Watson. Globally optimised parameters for a model of mitotic control in frog egg extracts. IEEE Proceedings-Systems Biology, 152(2):81?92, 2005. [6] L. C. W. Dixon. Global optima without convexity. Numerical Optimisation Centre, Hatfield Polytechnic, 1977. [7] B. O. Shubert. A sequential method seeking the global maximum of a function. SIAM Journal on Numerical Analysis, 9(3):379?388, 1972. [8] D. Q. Mayne and E. Polak. Outer approximation algorithm for nondifferentiable optimization problems. Journal of Optimization Theory and Applications, 42(1):19?30, 1984. [9] R. H. Mladineo. An algorithm for finding the global maximum of a multimodal, multivariate function. Mathematical Programming, 34(2):188?200, 1986. [10] R. G. Strongin. Convergence of an algorithm for finding a global extremum. Engineering Cybernetics, 11(4):549?555, 1973. [11] D. E. Kvasov, C. Pizzuti, and Y. D. Sergeyev. Local tuning and partition strategies for diagonal GO methods. Numerische Mathematik, 94(1):93?106, 2003. [12] S. Bubeck, G. Stoltz, and J. Y. Yu. Lipschitz bandits without the Lipschitz constant. In Algorithmic Learning Theory, pages 144?158. Springer, 2011. [13] J. Gardner, M. Kusner, K. Weinberger, and J. Cunningham. Bayesian Optimization with Inequality Constraints. In Proceedings of The 31st International Conference on Machine Learning (ICML), pages 937? 945, 2014. [14] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. De Freitas. Bayesian optimization in high dimensions via random embeddings. In Proceedings of the Twenty-Third international joint conference on Artificial Intelligence, pages 1778?1784. AAAI Press, 2013. [15] N. Srinivas, A. Krause, M. Seeger, and S. M. Kakade. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In Proceedings of the 27th International Conference on Machine Learning (ICML), pages 1015?1022, 2010. [16] K. P. Murphy. Machine learning: a probabilistic perspective. MIT press,

page 521, 2012. [17] C. E. Rasmussen and C. Williams. Gaussian Processes for Machine Learning. MIT Press, 2006. [18] R. Munos. Optimistic optimization of deterministic functions without the knowledge of its smoothness. In Proceedings of Advances in neural information processing systems (NIPS), 2011. [19] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Applications, 79(1):157?181, 1993. [20] K. Kandasamy, J. Schneider, and B. Poczos. High dimensional Bayesian optimisation and bandits via additive models. arXiv preprint arXiv:1503.01673, 2015. [21] S. Surjanovic and D. Bingham. Virtual library of simulation experiments: Test functions and datasets. Retrieved November 30, 2014, from http://www.sfu.ca/?ssurjano, 2014. [22] D. B. McDonald, W. J. Grantham, W. L. Tabor, and M. J. Murphy. Global and local optimization using radial basis function response surface models. Applied Mathematical Modelling, 31(10):2095?2110, 2007. [23] T. J. Walsh, S. Goschin, and M. L. Littman. Integrating Sample-Based Planning and Model-Based Reinforcement Learning. In Proceedings of the 24th AAAI conference on Artificial Intelligence (AAAI), 2010. [24] A. L. Strehl, L. Li, and M. L. Littman. Reinforcement learning in finite MDPs: PAC analysis. The Journal of Machine Learning Research (JMLR), 10:2413?2444, 2009.

9