Batch Bayesian Optimization via Local Penalization

Javier González
Department of Computer Science
University of Sheffield
j.h.gonzalez@sheffield.ac.uk

Zhenwen Dai
Department of Computer Science
University of Sheffield
z.dai@sheffield.ac.uk

Philipp Henning
MPI for Intelligent Systems
Tubingen, Germany
phennig@tuebingen.mpg.de

Neil D. Lawrence
Department of Computer Science
University of Sheffield
n.lawrence@sheffield.ac.uk

Abstract

Most proposed Bayesian optimization approaches only allow the exploration of the parameter space to occur *sequentially*. In this work we we investigate a highly effective heuristic for batch Bayesian optimization based on an estimate of the function's *Lipschitz constant* that captures the most important aspect of this interaction—local repulsion—at negligible computational overhead. A *penalized acquisition function* is used to collect batches of points minimizing the non-parallelizable computational effort. The resulting algorithm compares very well, in run-time, with much more elaborate alternatives.

1 Introduction

The task is to solve the global optimization problem of finding $\mathbf{x}_M = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$. We assume that f is a *black-box* from which only perturbed evaluations of the type $y_i = f(\mathbf{x}_i) + \epsilon_i$, with $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, are available. We will assume that the objective of interest can be described well by a L-Lipschitz continuous function $f : \mathcal{X} \to \mathbb{R}$ defined on a compact subset $\mathcal{X} \subseteq \mathbb{R}^d$. In sequential BO the goal is to make a series of evaluations $\mathbf{x}_1, \dots, \mathbf{x}_N$ of f such that the maximum of f is evaluated as quickly as possible. To this end, a surrogate probabilistic model for f is calculated. This is typically a Gaussian Process (GP) $p(f) = \mathcal{GP}(\mu; k)$ with mean function μ and a covariance function k. The posterior of the GP is used to form the acquisition function $\alpha(\mathbf{x}; \mathcal{I}_n)$, where $\mathcal{I}_n$ represents the available data set $\mathcal{D}_n$ and the GP structure (kernel, likelihood and parameter values) when n data points are available. The next evaluation is placed at the (numerically estimated) global maximum $\mathbf{x}_{n+1}$ of this acquisition function [14, 13, 10, 11].

In this work, we focus on cases in which the cost of evaluating f in a batch of points of size n_b is the same as evaluating f in a single point. Such scenarios appear, for instance, in the optimization of computer models where several cores are available to run in parallel, or in wet-lab experiments when the cost of testing one experimental design is the same as testing a batch of them. In these settings, the set of available pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ can be augmented with the evaluations of f on batches of n_b data points $\mathcal{B}_t^{n_b} = \{\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,nb}\}$, for $t = 1, \dots, m$, rather than on single observations. The goal of any batch criterion is to build a batch by mimicking the decisions that would be made under the equivalent (optimal) sequential policy. Consider the choice of selecting $\mathbf{x}_{t,k}$, the k-th element of the t-th batch: the decision about where to collect $\mathbf{x}_{t,k}$ has to incorporate the uncertainty about the locations $\mathbf{x}_{t,1}, \dots, \mathbf{x}_{t,k-1}$, and the outcomes of the evaluation of f there, which is intractable even for small batch-sizes, due to the optimization-marginalization loop required to obtain $\mathbf{x}_{t,k}$. The literature has tried to avoid this computational burden by means of different strategies [15, 6, 3, 2, 4, 5, 9, 1, 7, 12] most of which requiere to update the GP after each element in the batch is collected. Unfortunately, this has computational overhead of $\mathcal{O}(n^3)$.

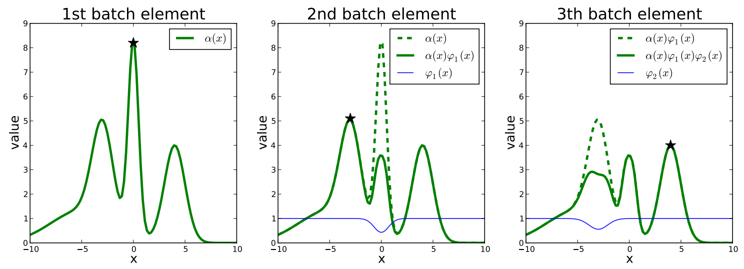


Figure 1: Illustration of three iterations of the *maximization-penalization* loop. The main task of good batch design is to explore the modes of the acquisition function, achieved by iterative maximization (black stars) and penalization (using $\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x})$) of the acquisition function $\alpha(\mathbf{x})$.

The motivation of this work is to develop a *heuristic* for batch BO at lower computational cost, while incorporating information about global properties of f from the GP model into the batch design. Our approach rests on the hypothesis that f is a Lipschitz continuous function[1], which is a common assumption in global optimization [8]. As explained below, the information provided by the Lipschitz constant can be used to define policies to collect a batch of points multiple steps ahead without evaluating f, by mimicking the hypothesized behavior of a sequential policy.

2 Maximization-Penalization Strategy for Batch Design

The intuition behind our approach is that for most GP priors in practical use for BO, the dominant effect of a function evaluation on the acquisition function is a *local exclusion* around the new evaluation. This shape of the acquisition function will be modeled through the Lipschitz properties of f, to distribute the elements in each batch. This should be understood as a heuristic to the shape of $\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$ if all previous observations were available, mimicking the effect a sequential policy.

In the sequel we will say that a function $\varphi(\mathbf{x}; \mathbf{x}_j)$, $\mathbf{x} \in \mathcal{X}$, is a *local penalizer* of a generic acquisition function $\alpha(\mathbf{x})$ at $\mathbf{x}_j$ if $\varphi(\mathbf{x}; \mathbf{x}_j)$ is differentiable, $0 \leq \varphi(\mathbf{x}; \mathbf{x}_j) \leq 1$ and $\varphi(\mathbf{x}; \mathbf{x}_j)$ is an non-decreasing function in $\|\mathbf{x} - \mathbf{x}_j\|$. We propose to replace the *maximization-marginalization* loop required in batch BO by a *maximization-penalization* strategy: while the optimization is carried out in a similar fashion, the marginalization step is replaced by the direct penalization of $\alpha(\mathbf{x}; \mathcal{I}_{t,k-1})$ around its most recent maximum, *i.e*, the previous batch element. Figure 1 gives a graphical illustration. The maximization-penalization strategy selects $\mathbf{x}_{t,k}$ as

$$\mathbf{x}_{t,k} = \arg\max_{x \in \mathcal{X}} \left\{ g(\alpha(\mathbf{x}; \mathcal{I}_{t,0})) \prod_{j=1}^{k-1} \varphi(\mathbf{x}; \mathbf{x}_{t,j}) \right\}, \tag{1}$$

where $\varphi(\mathbf{x}; \mathbf{x}_{t,j})$ are local local penalizers centered at $\mathbf{x}_{t,j}$ and $g : \mathbb{R} \to \mathbb{R}^+$ is a differentiable transformation of $\alpha(\mathbf{x})$ that keeps it strictly positive without changing the location of its extrema. Here, we will use the *soft-plus* transformation $g(z) = \ln(1+e^z)$. This does not require to re-estimate the GP model after each location is selected, just a new optimization of the penalized utility.

The effect of a local penalizer is to smoothly reduce the value of the acquisition function in a neighborhood of $\mathbf{x}_j$. A 'good' local penalizer centered at $\mathbf{x}_j$ should reflect the belief about the distance from $\mathbf{x}_j$ to $\mathbf{x}_M$: If we suspect that $\mathbf{x}_M$ is far from $\mathbf{x}_j$, a broad $\varphi(\mathbf{x}; \mathbf{x}_j)$ will discard a large portion of $\mathcal{X}$ in which we don't need to collect any sample. On the other hand, if we believe that $\mathbf{x}_M$ and $\mathbf{x}_j$ are close, ideally we want to minimize the penalization of $\alpha(\mathbf{x})$ and keep collecting samples is a close neighborhood. This local penalization mimics the acquisition function's dynamics under a sequential policy in the following sense: the modes of the acquisition functions correspond to regions in which either $\mu_n(\mathbf{x})$ or $\sigma_n^2(\mathbf{x})$ (or both) are large. Evaluating, for instance, where $\sigma_n(\mathbf{x})$ is large will reduce uncertainty in that region, decreasing $\alpha(\mathbf{x})$ in a neighborhood. The functions $\varphi(\mathbf{x}; \mathbf{x}_j)$ are surrogates for this neighborhood.

[1] $f : \mathcal{X} \to \mathbb{R}$ on a compact subset $\mathcal{X} \subseteq \mathbb{R}^d$ of the d-dimensional real vector space is said to be L-Lipschitz if it satisfies $|f(\mathbf{x}_1) - f(\mathbf{x}_2)| \leq L\|\mathbf{x}_1 - \mathbf{x}_2\|$, $\forall\, \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ where L is a global positive constant, and $\|\cdot\|$ is the ℓ^2-norm on $\mathbb{R}^d$

Algorithm 1 Batch Bayesian Optimization with Local Penalization (BBO-LP)

```
Input: dataset $\mathcal{D}_1 = \{\mathbf{x}_i, y_i\}_{i=1}^n$, batch size $n_b$, iteration budget $m$, acquisition transformation $g$.
for $t = 1$ to $m$ do
    Fit a GP to $\mathcal{D}_t$ and build the acquisition function $\alpha(\mathbf{x}, \mathcal{I}_{t,0})$ using the current GP.
    Take $\tilde{\alpha}_{t,0}(\mathbf{x}) \leftarrow g(\alpha(\mathbf{x}, \mathcal{I}_{t,0}))$ and $\hat{L} \leftarrow \max_{\mathcal{X}} \|\mu_{\nabla}(\mathbf{x})\|$
    for $j = 1$ to $n_b$ do
        1. Maximization-step: $\mathbf{x}_{t,j} \leftarrow \arg\max_{x \in \mathcal{X}} \{\tilde{\alpha}_{t,j-1}(\mathbf{x})\}$.
        2. Penalization-step: $\tilde{\alpha}_{t,j}(\mathbf{x}) \leftarrow \tilde{\alpha}_{t,0}(\mathbf{x}) \prod_{j=1}^{k} \varphi(\mathbf{x}; \mathbf{x}_{t,j}, \hat{L})$.
    end for
    $\mathcal{B}_t^{n_b} \leftarrow \{\mathbf{x}_{t,1}, \ldots, \mathbf{x}_{t,n_b}\}$.
    $y_{t,1}, \ldots, y_{t,n_b} \leftarrow$ Parallel evaluations of $f$ at $\mathcal{B}_t^{n_b}$.
    $\mathcal{D}_{t+1} \leftarrow \mathcal{D}_t \cup \{(\mathbf{x}_{t,j}, y_{t,j})\}_{j=1}^{n_b}$.
end for
Returns: $\hat{\mathbf{x}}_M = \arg\max_{x \in \mathcal{X}} \{\mu(\mathbf{x})\}$.
```

2.1 Choosing Local Penalizers $\varphi(\mathbf{x}; \mathbf{x}_j)$

We now construct penalizing functions $\varphi(\mathbf{x}; \mathbf{x}_j)$ that incorporate into $\alpha(\mathbf{x})$ the current belief about the distance from the batch locations to $\mathbf{x}_M$. Take $M = \max_{\mathbf{x} \in \mathcal{X}} f(\mathbf{x})$, and a valid Lipschitz constant L. Consider the ball $B_{r_j}(\mathbf{x}_j) = \{\mathbf{x} \in \mathcal{X} : \|\mathbf{x}_j - \mathbf{x}\| \leq r_j\}$ where $r_j = (M - f(\mathbf{x}_j))/L$. To simplify the notation we write $r_j = r(\mathbf{x}_j)$ for the radius of the ball around $\mathbf{x}_j$. If f is the true optimization objective, then $\mathbf{x}_M \notin B_{r_j}(\mathbf{x})$—otherwise the Lipschitz condition would be violated. The size of $B_{r_j}(\mathbf{x}_j)$ depends on L, M and the value of f at $\mathbf{x}_j$. Both large variability in f (large L) and proximity of $f(\mathbf{x}_j)$ to the optimum M shrink $B_{r_j}(\mathbf{x}_j)$.

In the BO context, under the assumption $f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, we choose $\varphi(\mathbf{x}; \mathbf{x}_j)$ as the probability that $\mathbf{x}$, any point in $\mathcal{X}$ that is a potential candidate to be a maximum, does not belong to $B_{r_j}(\mathbf{x}_j)$: $\varphi(\mathbf{x}; \mathbf{x}_j) = 1 - p(\mathbf{x} \in B_{r_j}(\mathbf{x}_j))$. The functions $\varphi(\mathbf{x}; \mathbf{x}_j)$, that can be computed in closed form, thus create exclusion zones whose size is governed by L. If $\mu_n(\mathbf{x}_j)$ is close to M, then $\varphi(\mathbf{x}; \mathbf{x}_j)$ will have a smaller and more localized effect on $\alpha(\mathbf{x})$ (a smaller exclusion area). On the other hand, if $\mu_n(\mathbf{x}_j)$ is far from M, $\varphi(\mathbf{x}; \mathbf{x}_j)$ will produce a wider yet less intense correction on $\alpha(\mathbf{x})$. The value of L also affects the size of the effect of $\varphi(\mathbf{x}; \mathbf{x}_j)$ on $\alpha(\mathbf{x})$, decreasing it as L increases. Of course, the values of M and L are unknown in general. To approximate M, we take $\hat{M} = \max_i\{y_i\}$. We approximate L by taking $L_{\nabla} = \max_{\mathbf{x} \in \mathcal{X}} \|\nabla f(\mathbf{x})\|$ for $\nabla f(\mathbf{x})$ the gradient of the GP at $\mathbf{x}$.

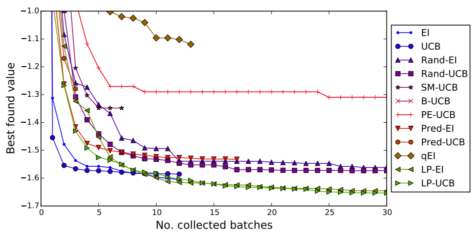
3 Experimental Section

We perform (i) a simulation in which the performance of the algorithms is compared for a fixed time budget across different problem dimensions, batch sizes and acquisition functions and (ii) a comparison of the *running time* in two objective functions with different evaluation costs. In all the experiments the exponentiated quadratic (EQ) covariance $k(\mathbf{x}, \mathbf{x}') = \theta \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$, $\theta, \gamma > 0$ is used in the GP model. The parameters of the GP are optimized by maximizing the marginal likelihood from the best of 10 random initializations. The results are taken over 20 replicates with different initial values. All the simulations were done on Amazon EC2 servers with Intel Xeon E5-2666 processors and 2 virtual CPUs except the SVR tuning with 16 virtual CPUs.

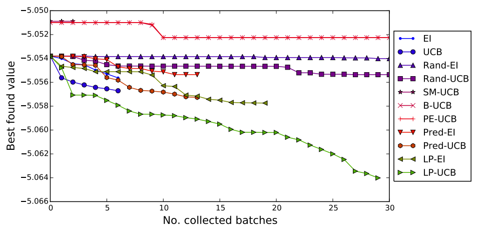
We label the methods used by means of the batch design type followed by the acquisition used: Rand is used when the first element in the batch is collected maximizing the acquisition and the remaining ones randomly, B and PE are the exploratory approaches in [15] and [6], Pred is used in cases when the model is used to generate 'fake' batch obervations as in [3], SM identifies the simulating and matching method [1] and LP stands for our local penalization method [2]. The multi-point expected improvement [5] is denoted by qEI. Two acquisition functions are used: the expected improvement (EI) and the Upper Confidence Bound (UCB). To run the B, PE, SM, methods, we use the available MATLAB code[3]. The implementation of these methods optimize f by searching its optimum in a fine grid, which is an advantage computationally but a drawback in terms of precision. The qEI

[2]https://github.com/SheffieldML/GPyOpt.

[3]http://econtal.perso.math.cnrs.fr/software/



(a) Results for the Cosines objective.



(b) Results for the wet-lab objective.

Figure 2: Results for the two functions used in the experimental section of this work. Geometric figures on top of the lines represent the moments in which the batches are evaluated.

d	n_b	EI	UCB	Rand-EI	Rand-UCB	SM-UCB	B-UCB
	5			0.32±0.05	**0.31±0.05**	1.86±1.06	0.56±0.03
2	10	0.31±0.03	0.32±0.06	0.65±0.32	0.79±0.42	4.40±2.97	**0.59±0.00**
	20			0.67±0.31	0.75±0.32	-	0.57±0.01
	5			9.19±5.32	10.59±5.04	137.2±113.0	**6.01±0.00**
5	10	8.84±3.69	11.89±9.44	1.74±1.47	2.20±1.85	108.7±74.38	3.77±0.00
	20			2.18±2.30	2.76±3.06	-	2.53±0.00
	5			**690.5±947.5**	1825±2149	9e+04±7e+04	2098±0.00
10	10	559.1±1014	1463±1803	200.9±455.9	1149±1830	9e+04±1e+05	857.8±0.00
	20			639.4±1204	385.9±642.9	-	1656±0.00

d	n_b	PE-UCB	Pred-EI	Pred-UCB	qEI	LP-EI	LP-UCB
	5	0.99±0.74	0.41±0.15	0.45±0.16	1.53±0.86	0.35±0.11	**0.31±0.06**
2	10	0.66±0.29	1.16±0.70	1.26±0.81	3.82±2.09	0.66±0.48	0.69±0.51
	20	0.75±0.44	1.28±0.93	1.34±0.77	-	**0.50±0.21**	0.58±0.21
	5	123.5±81.43	10.43±4.88	11.77±9.44	15.70±8.90	11.85±5.68	10.85±8.08
5	10	120.8±78.56	9.58±7.85	11.66±11.48	17.69±9.04	3.88±4.15	**1.88±2.46**
	20	98.60±82.60	8.58±8.13	10.86±10.89	-	6.53±4.12	**1.44±1.93**
	5	2e+05±2e+05	793.0±1226	1412±3032	-	1881±1176	1194±1428
10	10	6e+04±8e+04	442.6±717.9	1725±3205	-	1042±1562	**100.4±338.7**
	20	5e+04±4e+04	1091±1724	2231±3110	-	1249±1570	**20.75±50.12**

Table 1: Results for the gSobol function across different dimensions, batch sizes and methods. For each algorithm, the mean and standard deviation are shown. Best results among the batch methods are highlighted in bold. '-' represents that the method could not complete the first iteration within the time budget. The value of f at the minimum is always zero.

was taken from the R-package DiceOptim[4]. Unless specified otherwise, the default implemented settings of all the previous methods are used.

Table 1 shows the results for the gSobol function[5] for dimensions $d = 2, 5, 10$ and batch sizes, $n_b = 5, 10, 20$. For methods using the UCB, κ was fixed to 2, which allows us to compare the different batch designs using the same acquisition function. For dimension 2, 5 and 10, we use a time budget of 1, 5 and 10 mins. respectively. The overall best technique is the LP-UCB, that achieves the best results in 5 of the 9 cases. It is also notable that it exhibits fairly small standard deviations compared with the rest of the methods and it is coherent accumulating information about the optimum of f in terms of the batch size: as n_b increases the results are consistently better. Figure 2 shows the comparison of the methods in two scenarios in terms of the running time. The first experiment uses the function $f(\mathbf{x}) = 1 - \sum_{i=1}^{2}(g(x_i) - r(x_i))$ where $g(x_i) = (1.6x_i - 0.5)^2$ and $r(x_i) = 0.3\cos(3\pi(1.6x_i - 0.5))$ in $[0, 5]^2$. The second experiment is motivated by a wet-lab experimental design. We work with a surface that emulates the performance of mammalian cells in protein production. The function has dimension 71 and it is moderately expensive to evaluate since it corresponds to the predictive mean of a GP trained over 1,500 data instances[6]. In both experiments the Local penalization approach shows the best performance.

[4]http://cran.r-project.org/web/packages/DiceOptim/index.html

[5]See http://www.sfu.ca/ ssurjano/gfunc.html.

[6]The qEI was not used in this experiment due to the dimensionality of the problem

4 Discussion

We have investigated a new heuristic for batch BO, BBO-LP, that significantly reduces the computational burden of non-parallelizable tasks. The resulting method can be used with any acquisition function and it is able to make fast and appropriate decisions about the locations where f should be evaluated. When the batch evaluations of f are parallelizable this is an important advantage, meaning that they don't lead to considerable additional computational overhead.

References

[1] Javad Azimi, Alan Fern, and Xiaoli Fern. Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems*, pages 109–117, 2010.

[2] Javad Azimi, Ali Jalali, and Xiaoli Fern. Dynamic batch Bayesian optimization. *CoRR*, abs/1110.3347, 2011.

[3] Javad Azimi, Ali Jalali, and Xiaoli Zhang Fern. Hybrid batch Bayesian optimization. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[4] James Bergstra, Rémy Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *NIPS'2011*, 2011.

[5] Clment Chevalier and David Ginsbourger. Fast computation of the multi-points expected improvement with applications in batch selection. In Giuseppe Nicosia and Panos M. Pardalos, editors, *LION*, volume 7997 of *LNCS*, pages 59–69. Springer, 2013.

[6] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel Gaussian process optimization with upper confidence bound and pure exploration. *CoRR*, abs/1304.5350, 2013.

[7] Thomas Desautels, Andreas Krause, and Joel W. Burdick. Parallelizing exploration-exploitation tradeoffs with Gaussian process bandit optimization. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[8] Christodoulos A. Floudas and Panos M. Pardalos, editors. *Encyclopedia of Optimization, Second Edition*. Springer, 2009.

[9] P. I. Frazier. Parallel global optimization using an improved multi-points expected improvement criterion. In *INFORMS Optimization Society Conference, Miami FL*, 2012.

[10] Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13, 2012.

[11] José M. Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems 27*, pages 918–926. Curran Associates, Inc., 2014.

[12] Ali Jalali, Javad Azimi, Xiaoli Fern, and Ruofei Zhang. A lipschitz exploration-exploitation scheme for Bayesian optimization. In *Machine Learning and Knowledge Discovery in Databases*, pages 210–224, 2013.

[13] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

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

[15] Matthias Schonlau, William J. Welch, and Donald R. Jones. *Global versus local search in constrained optimization of computer models*, volume Volume 34 of *Lecture Notes–Monograph Series*, pages 11–25. Institute of Mathematical Statistics, Hayward, CA, 1998.