Distance Exploration for Scalable Batch Bayesian Optimization

Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, Svetha Venkatesh
Centre of Pattern Recognition and Data Analytics (PRaDA), Deakin University
Email: v.nguyen@deakin.edu.au

Abstract

Bayesian optimization (BO) is posed as a sequential problem where each experiment is completed before selecting a next one. However, it is often desirable to simultaneously explore using batches of parameters, especially when parallel processing facilities are available. Existing works have addressed batch BO in different ways. Still, the existing approaches are not scalable to large batch size or when the function evaluations are cheap. In this paper, we propose a computationally efficient batch Bayesian optimization based on a new exploration strategy using geometric distance. Our relaxation reduces the complexity in computing batch BO and also provides an alternative way for exploration, selecting a point far from the already observed locations. We theoretically formulate that our new strategy is a special case of the standard GP variance. We derive convergence analysis for the proposed batch BO approach. We present extensive experiments to show that our distance-based approach outperforms the state-of-the-art methods in both computational efficiency and performance.

1 Introduction

The optimization of the expensive black-box functions based on noisy observations is a fundamental problem in various real-world domains, e.g., material design [2] and machine learning hyper-parameter optimization [17]. In recent years, Bayesian optimization (BO) has received considerable attention in tuning hyper-parameters for complex models and algorithms in machine learning, robotics and experimental designs [3, 16, 13, 5, 14, 12].

The standard Bayesian optimization suggests one evaluation at a time. However, in many situations, it is desirable to run multiple evaluations in parallel so that we can speed up the process of finding the optimal setting. Such scenarios appear, for instance, in optimizing computer models where several cores are available to run in parallel. Another example is in wet-lab experiments, wherein the need for batch experiments is more pronounced as the cost of testing one experimental design is the same as testing a batch of them.

Related work. Existing work has addressed batch Bayesian optimization in multiple ways. Constant liar (CL) [7] iteratively constructs a batch of q points. Another direction [4, 6] in batch BO exploits a crucial fact about GP: the predictive variance depends only on the feature $\boldsymbol{x}$, but not the outcome values $\boldsymbol{y}$. BUCB algorithm [6] extends the sequential UCB to a batch setting. BUCB selects a first point in a batch using a standard UCB. Then, it updates the predictive variance using the new point which in turn alters the acquisition function to select a next point. Slightly different from BUCB, UCB-PE [4] only select a point from a relevance region which is potentially containing the higher value. The relevant region is sequentially updated after each iteration. Thus, this computation makes UCB-PE more expensive than BUCB.

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

Recently, UCB-PE [4] is shown to be equivalent to the determinantal point process (DPP) [10, 9] that models the diversity of all elements in a batch via a greedy algorithm. Another batch approach is local penalization (LP) [8] which iteratively penalizes the current peak in the acquisition function to find the next peak. LP depends on the estimation of Lipschitz constant to flexibly penalize the peaks. However, the Lipschitz constant L is unknown in general and not trivial to estimate. Batch BO can also be developed using simulation matching [1], information-based policies [15], infinite GMM [13] and parallel knowledge gradient [18].

Contribution. To improve the scalability in batch BO, we propose an alternative view which utilizes the geometric distance of the observations for exploration rather than the standard predictive variance. Our intuition is based on the fact that the best location for exploration should not be close to the existing observations. Based on this view, we present a simple, but effective strategy for batch Bayesian optimization. Our method reduces the computational complexity required to (repeatedly) compute the predictive variance in Gaussian process. We show theoretically that our distance exploration is a special case of using the GP predictive variance and derive the sublinear convergence analysis for the proposed approach.

2 Distance Exploration for Scalable Batch Bayesian Optimization

We present our intuition for exploration using geometric distance and show that this view is a special case of the standard GP variance. We present some theoretical properties and convergence analysis.

2.1 Geometric distance for exploration (DE)

Intuitively, the highest uncertainty point should not be close to already observed locations. If a data point stays closer to an observation, the less uncertainty it gets, and vice versa. Thus, we can employ this distance as the metric for exploration. That is, we find the location $\boldsymbol{x}_t$ s.t. the distance from $\boldsymbol{x}_t$ to its nearest observation $\boldsymbol{x}_i \in \mathcal{D}_t$ is maximized, i.e. $\boldsymbol{x}_t = \arg\max_{\boldsymbol{x}\in\mathcal{X}} ||\boldsymbol{x} - [\boldsymbol{x}]||^2$ where we denote the nearest observation to $\boldsymbol{x}$ as $[\boldsymbol{x}] = \arg\min_{\boldsymbol{x}_i\in\mathcal{D}_t} ||\boldsymbol{x} - \boldsymbol{x}_i||^2$. The example of distance exploration is illustrated in Fig. 1.

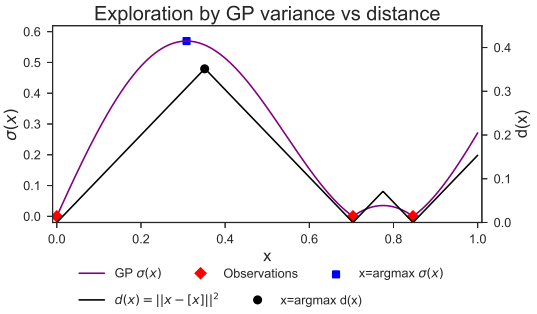

Figure 1: Illustration of exploration by GP variance and the proposed DE. The former selects the blue square points while the latter selects a black **circle**. $[\boldsymbol{x}]$ denotes the nearest observation to $\boldsymbol{x}$. DE offers the alternative exploration to GP variance while DE gets rid of cubic operation in GP.

2.2 Connection to GP Predictive Variance

We derive that using the geometric distance for exploration is a special case of using the GP variance so that the predictive variance value from an arbitrary location $\boldsymbol{x} \in \mathcal{X}$ - denoted as $\sigma^2_{\text{DE}}(\boldsymbol{x})$ - is only influenced by its nearest observation $\boldsymbol{x}_i \in \mathcal{D}_t$, instead of depending on the entire $\forall \boldsymbol{x}_i \in \mathcal{D}_t$. However, this $\sigma^2_{\text{DE}}(\boldsymbol{x})$ is not used by the algorithm.

Lemma 1. *Let denote $[\boldsymbol{x}] = \arg\min_{\boldsymbol{x}_i\in\mathcal{D}_t} ||\boldsymbol{x} - \boldsymbol{x}_i||$, the GP predictive variance at $\boldsymbol{x}$ defined by the nearest observation $[\boldsymbol{x}]$ is computed as $\sigma^2_{DE}(\boldsymbol{x}) = 1 - \exp^2\left(-||x - [x]||^2/\sigma_l^2\right)$.*

Theorem 2. *The distance exploration (DE) selects a data point $\boldsymbol{x}_t$ by maximizing the distance from $\boldsymbol{x}_t$ to its nearest observation $\boldsymbol{x}_i \in \mathcal{D}_t$. The DE selection is equivalent to maximizing the predictive variance in standard GP defined by the nearest observation as*

$$\boldsymbol{x}_t = \underset{\boldsymbol{x}\in\mathcal{X}}{argmax} \left[\underset{\boldsymbol{x}_i\in\mathcal{D}_t}{argmin} ||\boldsymbol{x} - \boldsymbol{x}_i||^2\right] = \underset{\boldsymbol{x}\in\mathcal{X}}{argmax}\, \sigma^2_{DE}(\boldsymbol{x}).$$

2.3 Convergence Analysis

We present the key theoretical results and refer to the supplement for the proofs.

Algorithm 1 UCB-Distance Exploration (DE) for Scalable Batch Bayesian Optimization.

Input: $\mathcal{D}_0$, T, batch size B

for $t = 1$ to T **do**
  Obtain the first element from UCB $\boldsymbol{x}_{t,1} = \arg\max_{x \in \mathcal{X}} \alpha_t^{\text{UCB}}(x)$, and $\mathcal{D}_{t,1} = \mathcal{D}_{t-1} \cup \boldsymbol{x}_{t,1}$.
  for $i = 2$ to B **do**
    Obtain $\boldsymbol{x}_{t,i} = \arg\max_{x \in \mathcal{X}} \left[\arg\min_{x_i \in \mathcal{D}_{t,i}} ||\boldsymbol{x} - \boldsymbol{x}_i||\right]$ and $\mathcal{D}_{t,i} = \mathcal{D}_{t,i-1} \cup \boldsymbol{x}_{t,i}$.
  end for
  Evaluate in parallel $y_{t,b} = f(\boldsymbol{x}_{t,b}), \forall b \leq B$ and $\mathcal{D}_t = \mathcal{D}_{t-1} \cup (\boldsymbol{x}_{t,b}, y_{t,b})_{b=1}^B$.
end for

Output: $\boldsymbol{x}_{\max}, y_{\max}$

Lemma 3. *We bound the maximum information gain of the Distance Exploration using sublinear term as* $\gamma^{DE} \leq \mathcal{O}\left(\sum_{t=1}^{T} t^{-\frac{2}{d}}\right)$ *and* $\lim_{T \to \infty} \frac{\gamma^{DE}}{T} = 0$.

Theorem 4. *Let* γ_{TB} *be the maximum information gain of DE,* $\delta \in (0, 1)$ *and define* $\beta_T = 2\log\left(|\mathcal{X}|\pi^2 T^2/6\delta\right)$, $C = 32/\log\left(1 + \sigma^{-2}\right)$ *then with probability at least* $1-\delta$ *the cumulative batch regret of our UCB-DE obtains the sublinear regret as* $R_T^B \leq \sqrt{\frac{T}{B} C \beta_T \gamma_{TB}}$ *and* $\lim_{T \to \infty} R_T^B / T = 0$.

Similar to the sublinear regret bound of the previous batch approaches, our proposed UCB-DE achieves a desirable asymptotic property of an algorithm which is no-regret, i.e. $\lim_{T \to \infty} R_T^B / T = 0$ and ensures the algorithm converged better than linear rate (e.g., of the random algorithm). It should be noted that our regret bound may be worse than the bounds of BUCB, UCB-PE and DPP by a constant factor because the DE offers a greater level of exploration than SE kernel and thus $\gamma_{TB} > \gamma_{TB}^{\text{SE}} \sim O(d(\ln T))$ [9]. However, unlike multi-arm bandits, for Bayesian optimization the performance achieved at the end of the optimization process, i.e. $\max_{\forall x_i \in \mathcal{D}_t} f(x_t)$, is more important than the cumulative regrets, occurred throughout the process to get that performance.

2.4 Computational Complexity

Let denote T #iteration, B #batch size, d #dimension, $N = TB$ #observation. Let consider the number of evaluations in global optimization step be 10^d (grows exponentially with the dimension). We consider the Cholesky decomposition for matrix inversion with the cost of $\mathcal{O}(N^2)$. Both BUCB and our approach are similar in computing a first point in a batch with the complexity of $\mathcal{O}(10^d T N^2)$. The difference in computation is from selecting $B - 1$ points in T iterations. In BUCB, once we insert a new element into a batch, we need to perform Cholesky update for the inverse covariance matrix. After multiplying with $T(B-1)$ and noting that $N = TB$, we have $O(10^d N^3)$ for BUCB and $O(10^d N^2)$ for us. Thus, our approach is one order cheaper than BUCB in selecting $B - 1$ points in a batch. Then, our DE offers scalable solution for batch BO with larger batch size B, higher dimension d and more observation N.

2.5 Connection to Determinantal Point Process and UCB-PE

Next, we relate our strategy to k-DPP which is a distribution over all discrete subsets $A \in \mathcal{A}$ [10] with the probability $p(A) \sim \det(K_A)$. The goal is to pick a subset A with k elements such that $p(A)$ is maximized [10]. Thus, all of the elements in set A are diverse due to the property of $\det(K_A)$.

One of the intermediate steps in proving Theorem 4 (see the supplement and also discussed in [9]) that

$$\sum_{b=1}^{B} r_{t,b} \leq \sum_{b=1}^{B} \sigma_{t-1,b}^2(x_{t,b}) \leq \frac{1}{\log(1 + \sigma^{-2})} \log \prod_{b=1}^{B} \left(1 + \sigma^{-2} \sigma_{t-1,b}(x_{t,b})\right). \tag{1}$$

At an iteration t after choosing the first element, let the kernel matrix be $K_{t,1} = I + \sigma^{-2}\left[k_{t,1}(p_i, p_j)\right]_{i,j}$ over all possible points $p_i, p_j \in \mathcal{X}$. We can see that the product of the last $B - 1$ terms in Eq. (1) is exactly $B - 1$ principal minor of the kernel matrix $K_{t,1}$ formed by the indices corresponding to $A = \{x_{t,b}\}_{b=2}^B$. Thus, we conclude that our UCB-DE is a special case of the k-DPP, where $k = B - 1$, via a greedy algorithm.

Table 1: Best-found-value comparison on benchmark functions with $T = 10d$ and $B = 5$.

Functions		Hartmann	Ackley	Alpine2	Hartmann	gSobol
Dim		3D	5D	5D	6D	10D
CL	UCB	**-3.862(.00)**	***10.53(1.51)***	-46.46(30.8)	-2.747(0.05)	678.2(304)
LP	UCB	-3.833(.03)	15.52(2.31)	**-63.54(21.6)**	***-2.987(0.03)***	500.8(347)
BUCB		-3.488(.26)	**10.21(0.62)**	-45.93(15.1)	-2.847(0.09)	432.9(476)
UCB-PE (DPP)		-3.691(.13)	12.90(1.15)	-48.86(7.53)	-2.903(0.06)	***320.8(340)***
UCB-DE		**-3.862(.00)**	11.41(2.17)	***-52.84(0.0)***	**-3.098(0.07)**	**228.3(224)**

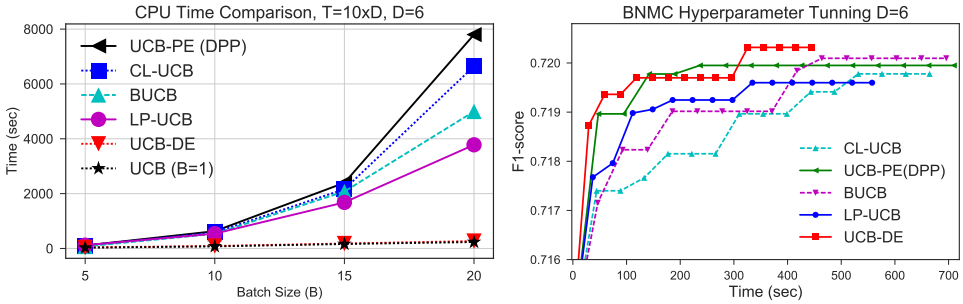

Figure 2: Left: Time comparison with different batch size. Right: Comparison on real experiments.

3 Experiments

The proposed UCB-DE outperforms the baselines in CPU time whilst achieving comparable results.

Comparison on benchmark functions. We compare the performances using benchmark objective functions which the dimensions range from 3 to 10. We set the batch size $B = 5$ so that all of the methods have a similar budget. We present the result in Table 1 that all of the methods perform generally well and competitive. This is because all methods are quite similar in selecting the first element in a batch. In addition, our proposed UCB-DE performs superior to the baselines for 4 over 5 cases. This is because the selected points by DE is more explorative and far from the observed location than the choices of using GP variance. However, in situation that requires more exploitation, such as Ackley function, our UCB-DE fails against the BUCB and CL.

Computation comparison. As our focus is to speed up batch BO by offering a simpler but equivalent solution to the existing techniques. In Fig. 2, we compare the CPU time w.r.t. different batch sizes $B = 5, 10, 15, 20$ on Hartmann 6D function. UCB-PE (DPP) takes the most computation cost because it repeatedly computes relevant regions and search for the highest variance location in these regions. BUCB only updates the predictive variance while it keeps the mean function fixed. Hence, BUCB is cheaper than CL which requires updating both mean and variance.

Our proposed approach outperforms all baselines in computation. Especially, when a batch size B increases, our UCB-DE computation seems invariant and surpasses the others in an order of magnitude. Because increasing a batch size results in more observations, the number of GP updates (after each element is added into a batch) increases and also each GP update is more costly $\mathcal{O}\left(N^3\right)$ due to large observations. In contrast, computing a distance in DE is much cheaper and our UCB-DE cost is almost similar to the cost of the sequential setting ($B = 1$).

Hyperparameter tuning. We optimize the hyper-parameters for the BNMC [11] on Scene dataset using the public code. There are 6 hyper-parameters to tune so that the F1-score is maximized.

We compare the performances under the axis of running time (both optimization and evaluation). The experiments show that our method achieves generally the best performance (in terms of F1-score and strength) within the shortest time (see Fig. 2). This is because the proposed DE can prevent the complexity of GP and thus run much faster. In addition, our approach also offers a greater level of exploration based on geometric distance.

References

[1] J. Azimi, A. Fern, and X. Z. Fern. Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems*, pages 109–117, 2010.

[2] P. V. Balachandran, D. Xue, J. Theiler, J. Hogden, and T. Lookman. Adaptive strategies for materials design using uncertainties. *Scientific reports*, 6, 2016.

[3] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[4] E. Contal, D. Buffoni, A. Robicquet, and N. Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

[5] T. Dai Nguyen, S. Gupta, S. Rana, V. Nguyen, S. Venkatesh, K. J. Deane, and P. G. Sanders. Cascade Bayesian optimization. In *Australasian Joint Conference on Artificial Intelligence*, pages 268–280. Springer, 2016.

[6] T. Desautels, A. Krause, and J. W. Burdick. Parallelizing exploration-exploitation tradeoffs in gaussian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, 2014.

[7] D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. Springer, 2010.

[8] J. González, Z. Dai, P. Hennig, and N. D. Lawrence. Batch Bayesian optimization via local penalization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 648–657, 2016.

[9] T. Kathuria, A. Deshpande, and P. Kohli. Batched gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 4206–4214, 2016.

[10] A. Kulesza and B. Taskar. k-dpps: Fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1193–1200, 2011.

[11] V. Nguyen, S. Gupta, S. Rana, C. Li, and S. Venkatesh. A Bayesian nonparametric approach for multi-label classification. In *Proceedings of The 8th Asian Conference on Machine Learning (ACML)*, pages 254–269, 2016.

[12] V. Nguyen, S. Gupta, S. Rana, C. Li, and S. Venkatesh. Bayesian optimization in weakly specified search space. In *IEEE 17th International Conference on Data Mining (ICDM)*, 2017.

[13] V. Nguyen, S. Rana, S. Gupta, C. Li, and S. Venkatesh. Budgeted batch Bayesian optimization. In *IEEE 16th International Conference on Data Mining (ICDM)*, pages 1107–1112, 2016.

[14] S. Rana, C. Li, S. Gupta, V. Nguyen, and S. Venkatesh. High dimensional Bayesian optimization with elastic gaussian process. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2883–2891, 2017.

[15] A. Shah and Z. Ghahramani. Parallel predictive entropy search for batch global optimization of expensive objective functions. In *Advances in Neural Information Processing Systems*, pages 3312–3320, 2015.

[16] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[17] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[18] J. Wu and P. Frazier. The parallel knowledge gradient method for batch Bayesian optimization. In *Advances In Neural Information Processing Systems*, pages 3126–3134, 2016.