Batched Large-scale Bayesian Optimization in High-dimensional Spaces

Zi Wang [†] **Clement Gehring** [†] **Pushmeet Kohli** [‡] **Stefanie Jegelka** [†]

[†] MIT CSAIL {ziw,gehring,stefje}@csail.mit.edu [‡] DeepMind pushmeet@google.com

Abstract

Bayesian optimization (BO) has become an effective approach for black-box function optimization problems. Despite recent successes of applying BO to high-dimensional problems, it is extremely challenging to use these methods for large scale observations, because of the expensive computations involving Gaussian process (GP) posterior inference and the inference of the function structures. We propose *ensemble Bayesian optimization* (EBO) to address three current challenges in BO *simultaneously*: (1) large-scale observations; (2) high dimensional input spaces; and (3) selections of batch queries that balance quality and diversity. The key idea of EBO is to operate on an ensemble of additive Gaussian process models, each of which possesses a randomized strategy to divide and conquer. We show unprecedented, previously impossible results of scaling up BO to tens of thousands of observations within minutes of computation.

1 Introduction

Global optimization of black-box and non-convex functions is an important component of modern machine learning and has wide applications in many areas of science and engineering [2, 19, 6]. In the past decade, Bayesian optimization has become a popular approach for optimizing black-box non-convex functions with assumptions usually expressed by a Gaussian process (GP) prior. Recent work on Bayesian optimization addresses better query strategies [13, 17, 21, 8, 10, 22], techniques for batch queries [3, 4, 7, 12], algorithms for high dimensional problems [24, 5, 15, 11, 23], and alternative models for scalability [20, 14, 16]. Despite these successes, Bayesian optimization is typically limited to merely a few thousand observations [14]. Yet, reliable search and estimation for complex functions in very high-dimensional spaces may well require more evaluations. With the increasing availability of parallel computing resources, large number of function evaluations are also possible if the underlying approach can leverage the parallelism.

In this paper, we propose ensemble Bayesian optimization (EBO), a global optimization method targeted to high dimensional, large scale parameter search problems whose queries are parallelizable. EBO relies on two main ideas that are implemented at multiple levels: (1) we use efficient partition-based function approximators (across both data *and* features) that simplify and accelerate search and optimization; (2) we enhance the expressive power of these approximators by using ensembles and a stochastic approach. We maintain an evolving (posterior) distribution over the (infinite) ensemble and, in each iteration, draw one member to perform search and estimation. The model estimation and query selection can be parallelized across blocks in the Gram matrix sampled by a Mondrian process. Empirically, we demonstrate the ability of EBO to handle sample-intensive hard optimization problems by applying it to a simulated control problem with tens of thousands of observations. Our code will be publicly available at `https://github.com/zi-w/Ensemble-Bayesian-Optimization`.

2 Notations

Consider a simple but high-dimensional search space $\mathcal{X} = [0, R]^D \subseteq \mathbb{R}^D$. We aim to find a maximizer $x^* \in \arg\max_{x \in \mathcal{X}} f(x)$ of a black-box function $f : \mathcal{X} \to \mathbb{R}$. Gaussian processes

(GPs) are popular priors for modeling the function f in Bayesian optimization. We denote a Gaussian process as $\mathcal{GP}(\mu, \kappa)$ where $\mu(\cdot)$ is the mean function and $\kappa(\cdot, \cdot)$ is the covariance (kernel) function. Assume f is a function sampled from $\mathcal{GP}(0, \kappa)$. Given observations $\mathcal{D}_n = \{(\boldsymbol{x}_t, y_t)\}_{t=1}^n$ where $y_t \sim \mathcal{N}(f(\boldsymbol{x}_t), \sigma)$, we denote the posterior mean function to be $\mu_n(\cdot)$ and the posterior variance function to be $\sigma_n^2(\cdot)$. The log data likelihood for $\mathcal{D}_n$ is given by $\log p(\mathcal{D}_n) = -\frac{1}{2}\boldsymbol{y}_n^{\mathrm{T}}(\boldsymbol{K}_n + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}_n - \frac{1}{2}\log|\boldsymbol{K}_n + \sigma^2\boldsymbol{I}| - \frac{n}{2}\log 2\pi$, where $\boldsymbol{K}_n = [\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}_n}$ [18].

To reduce the complexity of the vanilla GP, we assume a latent decomposition of the input dimensions $[D] = \{1, \ldots, D\}$ into disjoint subspaces, namely, $\bigcup_{m=1}^M A_m = [D]$ and $A_i \cap A_j = \emptyset$ for all $i \neq j$, $i, j \in [M]$. As a result, the function f decomposes as $f(x) = \sum_{m\in[M]} f_m(x^{A_m})$ [11]. If each component f_m is drawn independently from $\mathcal{GP}(\mu^{(m)}, \kappa^{(m)})$ for all $m \in [M]$, the resulting f will also be a sample from an additive GP: $f \sim \mathcal{GP}(\mu, \kappa)$, with $\mu(x) = \sum_{m\in[M]} \mu_m(x^{A_m})$, $\kappa(x, x') = \sum_{m\in[M]} \kappa^{(m)}(x^{A_m}, x'^{A_m})$.

3 Ensemble Bayesian Optimization

We outline our approach, *Ensemble Bayesian optimization (EBO)*, in Alg.1. At a high level, EBO uses a (stochastic) series of Mondrian trees to partition the input space, learn the kernel parameters of a GP locally, and aggregate these parameters. Our forest hence spans across BO iterations.

In the t-th iteration of EBO in Alg. 1, we use a Mondrian process to randomly partition the search space into J parts (line 4), where J can be dependent on the size of the observations $\mathcal{D}_{t-1}$. For the j-th partition, we have a subset $\mathcal{D}_{t-1}^j$ of observations. From those observations, we learn a local GP with random tile coding *and* additive structure, via Gibbs sampling (line 6). For conciseness, we refer to such GPs as TileGPs. The probabilistic tile coding can be replaced by a Mondrian grid that approximates a Laplace kernel [1]. Once a TileGP is learned locally, we can run BO with the acquisition function η in each partition to generate a candidate set of points, and, from those, select a batch that is both informative (high-quality) and diverse (line 14). In this paper, we use an acquisition function from [22], and FILTER selects the batch by maximizing a surrogate function $\xi(X) = \log\det K_X + \sum_{b=1}^B \eta(\boldsymbol{x}_b)$ where $X = \{\boldsymbol{x}_b\}_{b=1}^B$.

Algorithm 1 Ensemble Bayesian Optimization (EBO)

1: **function** EBO $(f, \mathcal{D}_0)$
2: Initialize z, k
3: **for** $t = 1, \cdots, T$ **do**
4: $\{\mathcal{X}_j\}_{j=1}^J \leftarrow$MONDRIAN$([0, R]^D, z, k, J)$
5: **parfor** $j = 1, \cdots, J$ **do**
6: $z^j, k^j \leftarrow$ GIBBSSAMPLING$(z, k \mid \mathcal{D}_{t-1}^j)$
7: $\eta_{t-1}^j(\cdot) \leftarrow$ACQUISITION $(\mathcal{D}_{t-1}^j, z^j, k^j)$
8: $\{A_m\}_{m=1}^M \leftarrow$ DECOMPOSITION(z^j)
9: **for** $m = 1, \cdots, M$ **do**
10: $\boldsymbol{x}_{tj}^{A_m} \leftarrow \arg\max_{\boldsymbol{x}\in\mathcal{X}_j^{A_m}} \eta_{t-1}^j(\boldsymbol{x})$
11: **end for**
12: **end parfor**
13: $z \leftarrow$ SYNC$(\{z^j\}_{j=1}^J)$, $k \leftarrow$ SYNC$(\{k^j\}_{j=1}^J)$
14: $\{\boldsymbol{x}_{tb}\}_{b=1}^B \leftarrow$ FILTER $(\{\boldsymbol{x}_{tj}\}_{j=1}^J \mid z, k)$
15: **parfor** $b = 1, \cdots, B$ **do**
16: $y_{tb} \leftarrow f(\boldsymbol{x}_{tb})$
17: **end parfor**
18: $\mathcal{D}_t \leftarrow \mathcal{D}_{t-1} \cup \{\boldsymbol{x}_{tb}, y_{tb}\}_{b=1}^B$
19: **end for**
20: **end function**

Since, in each iteration, we draw an input space partition and update the kernel width and the additive structure, the algorithm may be viewed as implicitly and stochastically running BO on an ensemble of GP models. In the appendix, we describe the procedures of Alg. 1 in detail and show an illustration how EBO optimizes a 2D function. In the following, we focus on one important component of EBO: TileGP.

3.1 Learning a local TileGP via Gibbs sampling

We use the acronym "TileGP" to denote the Gaussian process model that uses additive kernels, with each component represented by tilings. For each Mondrian partition $\mathcal{X}_j = [l_1^j, h_1^j] \times \cdots \times [l_D^j, h_D^j]$, we use a TileGP to model the function f locally. We show the graphical model in Fig. 3.1 with fixed hyper-parameters α, β_0, β_1. The main difference to the additive GP model used in [23] is that TileGP constructs a hierarchical model for the random features (and hence, the kernels), while [23] do not consider the kernel parameters to be part of the random variables. To generate a TileGP, we first draw the mixing proportions $\theta \sim \textsc{Dir}(\alpha)$. Then for each dimension $d = 1, \cdots, D$, we draw additive decomposition $z_d \sim \textsc{Multi}(\theta)$ and a Poisson rate parameter $\lambda_d \sim \textsc{Gamma}(\beta_0, \beta_1)$. For each tiling layer $i = 1, \cdots, L$, the number of cuts k_{di} is generated by a Poisson process parameterized by λ_d. On the i-th layer of the tilings, we have two options to place the cuts: if we use tile coding, it samples the offset δ from a uniform distribution $U[0, \frac{h_d^j - l_d^j}{k_{di}}]$ and places the cuts uniformly starting at $\delta + l_d^j$; if we use Mondrian grids, it samples k_{di} cut locations uniformly randomly from $[l_d^j, h_d^j]$.

We can use Gibbs sampling to efficiently learn the cut parameter k and decomposition parameter z by marginalizing out λ and θ. Notice that both k and z take discrete values; hence, unlike other continuous GP parameterizations, we only need to sample discrete variables for Gibbs sampling. where $|k_d| = \sum_{i=1}^{L} k_{di}$. Hence, we only need to sample k and z when learning the hyperparameters of the TileGP kernel. For each dimension d, we sample the group assignment z_d according to

$$p(z_d = m \mid \mathcal{D}_{t-1}, k, z_{\neg d}; \alpha) \propto p(\mathcal{D}_{t-1} \mid z, k) p(z_d \mid z_{\neg d}) \propto p(\mathcal{D}_{t-1} \mid z, k)(|A_m| + \alpha_m). \quad (3.1)$$

We sample the number of cuts k_{di} for each dimension d and each layer i from the posterior

$$p(k_{di} \mid \mathcal{D}_{t-1}, k_{\neg di}, z; \beta) \propto p(\mathcal{D}_{t-1} \mid z, k) p(k_{di} \mid k_{\neg di}) \propto \frac{p(\mathcal{D}_n \mid z, k)\Gamma(\beta_1 + |k_d|)}{(\beta_0 + L)^{k_{di}} k_{di}!}. \quad (3.2)$$

If distributed computing is available, each Mondrian partition of the input space is assigned a worker to manage all the computations within this partition. On each worker, we use the above Gibbs sampling method to learn the additive structure and kernel bandwidth jointly. Conditioned on the observations associated with the partition on the worker, we use the learned posterior TileGP to select the most promising input point in this partition, and eventually send this candidate input point back to the main process together with the learned decomposition parameter z and the cut parameter k.

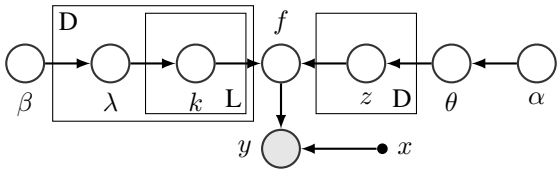

Figure 1: The graphical model for TileGP, a GP with additive and tile kernel partitioning structure. The parameter λ controls the rate for the number of cuts k of the tilings (inverse of the kernel bandwidth); the parameter z controls the additive decomposition of the input feature space.

3.2 Efficient data likelihood computation and parameter synchronization

For the random features, we use tile coding due to its sparsity and efficiency. Since non-zero features can be found and computed by binning, the computational cost for encoding a data point scales linearly with dimensions and number of layers. The resulting representation is sparse and convenient to use. Additionally, the number of non-zero features is quite small, which allows us to efficiently compute a sparse Cholesky decomposition of the inner product (Gram matrix) or the outer product of the data. This allows us to efficiently compute the data likelihoods.

In each iteration t, after the batch workers return the learned decomposition indicator z^b and the number of tiles $k^b, b \in [B]$, we synchronize these two parameters (line 13 of Alg. 1). For the number of tiles k, we set k_d to be the rounded mean of $\{k_d^b\}_{b=1}^B$ for each dimension $d \in [D]$. For the decomposition indicator, we use correlation clustering to cluster the input dimensions.

4 Experiments

We empirically verify the scalability of EBO and the effectiveness of random Mondrian partitions.

Scalability We compare EBO with a recent, state-of-the-art additive kernel learning algorithm, Structural Kernel Learning (SKL) [23]. EBO can make use of parallel resources both for Gibbs sampling and BO query selections, while SKL can only parallelize query selections but not sampling. Because the kernel learning part is the computationally dominating factor of large scale BO, we compare the time each method needs to run 10 iterations of Gibbs sampling with 100 to 50000 observations in 20 dimensions. We show the timing results for the Gibbs samplers in Fig. 2(a), where EBO uses 240 cores via the Batch Service of Microsoft Azure. For EBO, the maximum number of Mondrian partitions is set to be 1000 and the minimum number of data points in each Mondrian partition is 100. Due to a time limit we imposed, we did not finish SKL for more than 1500 observations. EBO runs more than 390 times faster than SKL when the observation size is 1500. Comparing the quality of learned parameter z for the additive structure, SKL has a Rand Index of 96.3% and EBO has a Rand Index of 96.8%, which are similar. In Fig. 2(b), we show speed-ups for different number of cores. EBO with 500 cores is not significantly faster than with 240 cores because EBO runs synchronized parallelization, whose runtime is decided by the slowest core. It is often the case that most of the cores have finished while the program is waiting for the slowest 1 or 2 cores to finish.

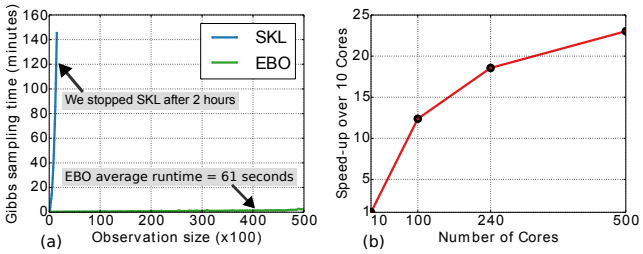

Figure 2: (a) Timing for the Gibbs sampler of EBO and SKL. EBO is significantly faster than SKL when the observation size N is relatively large. (b) Speed-up of EBO with 100, 240, 500 cores over EBO with 10 cores on 30,000 observations. Running EBO with 240 cores is almost 20 times faster than with 10 cores.

Effectiveness of Mondrian ensembles We verify the effectiveness of using ensemble models for BO on 4 functions randomly sampled from a 50-dimensional GP with an additive Laplace kernel. The hyperparameter of the Laplace kernel is known. In each iteration, each algorithm evaluates a batch of parameters of size B in parallel. We denote $\tilde{r}_t = \max_{x\in\mathcal{X}} f(x) - \max_{b\in[B]} f(x_{t,b})$ as the immediate regret obtained by the batch at iteration t, and $r_T = \min_{t\leq T} \tilde{r}_t$ as the regret, which captures the minimum gap between the best point found and the global optimum of the black-box function f.

We compare BO using SVI [9] (BO-SVI), BO using SVI with an additive GP (BO-Add-SVI) and a distributed version of BO with a fixed partition (PBO) against EBO with a randomly sampled partition in each iteration. PBO has the same 1000 Mondrian partitions in all the iterations while EBO can have at most 1000 Mondrian partitions. BO-SVI uses a Laplace isotropic kernel without any additive structure, while BO-Add-SVI, PBO, EBO all use the known prior. More detailed experimental settings can be found in the appendix. Our experimental results in Fig. 3 shows that EBO is able to find a good point much faster than BO-SVI and BO-Add-SVI; and, randomization and the ensemble of partitions matters: EBO is much better than PBO.

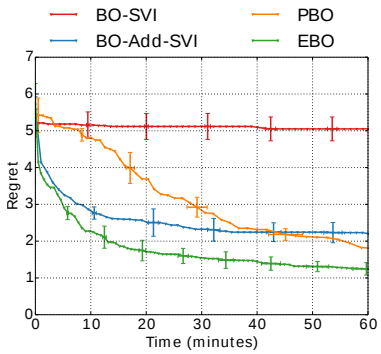

Figure 3: Averaged results of the regret of BO-SVI, BO-Add-SVI, PBO and EBO on functions drawn from a 50D additive GP. Using an additive GP within SVI (BO-Add-SVI) significantly improves over the full kernel (BO-SVI). In general, EBO finds a good point faster than the other methods.

5 Conclusion

We propose a novel framework, ensemble Bayesian optimization, to tackle the scaling problem of Bayesian optimization. To achieve this, we proposed a new GP model based on tile coding and additive structure. Our empirical results showed that EBO is able to explore the space smartly and find better solutions than its competitors.

References

[1] Matej Balog and Yee Whye Teh. The mondrian process for machine learning. *arXiv preprint arXiv:1507.05181*, 2015.

[2] Roberto Calandra. *Bayesian Modeling for Optimization and Control in Robotics*. PhD thesis, Technische Universität, 2017.

[3] Emile Contal, David Buffoni, Alexandre Robicquet, and Nicolas Vayatis. Parallel gaussian process optimization with upper confidence bound and pure exploration. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 225–240. Springer, 2013.

[4] Thomas Desautels, Andreas Krause, and Joel W Burdick. Parallelizing exploration-exploitation tradeoffs in Gaussian process bandit optimization. *Journal of Machine Learning Research*, 2014.

[5] Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional Gaussian process bandits. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[6] Javier González, Joseph Longworth, David C James, and Neil D Lawrence. Bayesian optimization for synthetic gene design. *arXiv preprint arXiv:1505.01627*, 2015.

[7] Javier González, Zhenwen Dai, Philipp Hennig, and Neil D Lawrence. Batch bayesian optimization via local penalization. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[8] Philipp Hennig and Christian J Schuler. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13:1809–1837, 2012.

[9] James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. *arXiv preprint arXiv:1309.6835*, 2013.

[10] José Miguel 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 (NIPS)*, 2014.

[11] Kirthevasan Kandasamy, Jeff Schneider, and Barnabas Poczos. High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning (ICML)*, 2015.

[12] Kirthevasan Kandasamy, Akshay Krishnamurthy, Jeff Schneider, and Barnabas Poczos. Asynchronous parallel bayesian optimisation via thompson sampling. *arXiv preprint arXiv:1705.09236*, 2017.

[13] Harold J Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Fluids Engineering*, 86(1):97–106, 1964.

[14] Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. Mondrian forests for large-scale regression when uncertainty matters. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[15] Chun-Liang Li, Kirthevasan Kandasamy, Barnabás Póczos, and Jeff Schneider. High dimensional bayesian optimization via restricted projection pursuit models. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[16] Mitchell McIntire, Daniel Ratner, and Stefano Ermon. Sparse Gaussian processes for bayesian optimization. In *Uncertainty in Artificial Intelligence (UAI)*, 2016.

[17] J. Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, 1974.

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

[19] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[20] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

[21] Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias W Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 2012.

[22] Zi Wang and Stefanie Jegelka. Max-value entropy search for efficient bayesian optimization. In *International Conference on Machine Learning (ICML)*, 2017.

[23] Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional bayesian optimization via structural kernel learning. In *International Conference on Machine Learning (ICML)*, 2017.

[24] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.