Batched High-dimensional Bayesian Optimization
via Structural Kernel Learning

Zi Wang [* 1] **Chengtao Li** [* 1] **Stefanie Jegelka** [1] **Pushmeet Kohli** [2]

Abstract

Optimization of high-dimensional black-box functions is an extremely challenging problem. While Bayesian optimization has emerged as a popular approach for optimizing black-box functions, its applicability has been limited to low-dimensional problems due to its computational and statistical challenges arising from high-dimensional settings. In this paper, we propose to tackle these challenges by (1) assuming a latent additive structure in the function and inferring it properly for more efficient and effective BO, and (2) performing multiple evaluations in parallel to reduce the number of iterations required by the method. Our novel approach learns the latent structure with Gibbs sampling and constructs batched queries using determinantal point processes. Experimental validations on both synthetic and real-world functions demonstrate that the proposed method outperforms the existing state-of-the-art approaches.

1. Introduction

Optimization is one of the fundamental pillars of modern machine learning. Considering that most modern machine learning methods involve the solution of some optimization problem, it is not surprising that many recent breakthroughs in this area have been on the back of more effective techniques for optimization. A case in point is deep learning, whose rise has been mirrored by the development of numerous techniques like batch normalization.

While modern algorithms have been shown to be very

---

*Equal contribution [1]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Massachusetts, USA [2]DeepMind, London, UK. Correspondence to: Zi Wang <ziw@csail.mit.edu>, Chengtao Li <ctli@mit.edu>, Stefanie Jegelka <stefje@csail.mit.edu>, Pushmeet Kohli <pushmeet@google.com>.

effective for convex optimization problems defined over continuous domains, the same cannot be stated for non-convex optimization, which has generally been dominated by stochastic techniques. During the last decade, Bayesian optimization has emerged as a popular approach for optimizing black-box functions. However, its applicability is limited to low-dimensional problems because of computational and statistical challenges that arise from optimization in high-dimensional settings.

In the past, these two problems have been addressed by assuming a simpler underlying structure of the black-box function. For instance, Djolonga et al. (2013) assume that the function being optimized has a low-dimensional effective subspace, and learn this subspace via low-rank matrix recovery. Similarly, Kandasamy et al. (2015) assume additive structure of the function where different constituent functions operate on disjoint low-dimensional subspaces. The subspace decomposition can be partially optimized by searching possible decompositions and choosing the one with the highest GP marginal likelihood (treating the decomposition as a hyper-parameter of the GP). Fully optimizing the decomposition is, however, intractable. Li et al. (2016) extended (Kandasamy et al., 2015) to functions with a projected-additive structure, and approximate the projective matrix via projection pursuit with the assumption that the projected subspaces have the same and known dimensions. The aforementioned approaches share the computational challenge of learning the groups of decomposed subspaces without assuming the dimensions of the subspaces are known. Both (Kandasamy et al., 2015) and subsequently (Li et al., 2016) adapt the decomposition by maximizing the GP marginal likelihood every certain number of iterations. However, such maximization is computationally intractable due to the combinatorial nature of the partitions of the feature space, which forces prior work to adopt randomized search heuristics.

In this paper, we develop a new formulation of Bayesian optimization specialized for high dimensions. One of the key contributions of this work is a new formulation that interprets prior work on high-dimensional Bayesian optimization (HDBO) through the lens of structured kernels, and places a prior on the kernel structure. Thereby, our

formulation enables simultaneous learning of the decomposition of the function domain.

Prior work on latent decomposition of the feature space considers the setting where exploration/evaluation is performed once at a time. This approach makes Bayesian optimization time-consuming for problems where a large number of function evaluations need to be made, which is the case for high dimensional problems. To overcome this restriction, we extend our approach to a batched version that allows multiple function evaluations to be performed in parallel (Desautels et al., 2014; González et al., 2016; Kathuria et al., 2016). Our second contribution is an approach to select the batch of evaluations for structural kernel learning-based HDBO.

Other Related Work. In the past half century, a series of different acquisition functions was developed for sequential BO in relatively low dimensions (Kushner, 1964; Močkus, 1974; Srinivas et al., 2012; Hennig & Schuler, 2012; Hernández-Lobato et al., 2014; Kawaguchi et al., 2015; Wang et al., 2016a; Kawaguchi et al., 2016; Wang & Jegelka, 2017). More recent developments address high dimensional BO by making assumptions on the latent structure of the function to be optimized, such as low-dimensional structure (Wang et al., 2016b; Djolonga et al., 2013) or additive structure of the function (Li et al., 2016; Kandasamy et al., 2015). Duvenaud et al. (2013) explicitly search over kernel structures.

While the aforementioned methods are sequential in nature, the growth of computing power has motivated settings where at once a *batch* of points is selected for observation (Contal et al., 2013; Desautels et al., 2014; González et al., 2016; Snoek et al., 2012; Wang et al., 2017). For example, the UCB-PE algorithm (Contal et al., 2013) exploits that the posterior variance of a Gaussian Process is independent of the function mean. It greedily selects points with the highest posterior variance, and is able to update the variances without observations in between selections. Similarly, B-UCB (Desautels et al., 2014) greedily chooses points with the highest UCB score computed via the out-dated function mean but up-to-date function variances. However, these methods may be too greedy in their selection, resulting in points that lie far from an optimum. More recently, Kathuria et al. (2016) tries to resolve this issue by sampling the batch via a diversity-promoting distribution for better randomized exploration, while Wang et al. (2017) quantifies the goodness of the batch with a submodular surrogate function that trades off quality and diversity.

2. Background

Let $f : \mathcal{X} \to \mathbb{R}$ be an unknown function and we aim to optimize it over a compact set $\mathcal{X} \subseteq \mathbb{R}^D$. Within as few function evaluations as possible, we want to find

$$f(x^*) = \max_{x \in \mathcal{X}} f(x).$$

Following (Kandasamy et al., 2015), we assume a latent decomposition of the feature 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 $i \neq j$, $i, j \in [D]$. Further, f can be decomposed into the following additive form:

$$f(x) = \sum_{m \in [M]} f_m(x^{A_m}).$$

To make the problem tractable, we assume that each f_m is drawn independently from $\mathcal{GP}(0, k^{(m)})$ for all $m \in [M]$. The resulting f will also be a sample from a GP: $f \sim \mathcal{GP}(\mu, k)$, where the priors are $\mu(x) = \sum_{m \in [M]} \mu_m(x^{A_m})$ and $k(x, x') = \sum_{m \in [M]} k^{(m)}(x^{A_m}, x'^{A_m})$. Let $\mathcal{D}_n = \{(x_t, y_t)\}_{t=1}^{n}$ be the data we observed from f, where $y_t \sim \mathcal{N}(f(x_t), \sigma)$. The log data likelihood for $\mathcal{D}_n$ is

$$\log p(\mathcal{D}_n | \{k^{(m)}, A_m\}_{m \in [M]}) \tag{2.1}$$
$$= -\frac{1}{2}(\boldsymbol{y}^{\mathrm{T}}(\boldsymbol{K}_n + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{y} + \log |\boldsymbol{K}_n + \sigma^2 \boldsymbol{I}| + n \log 2\pi)$$

where $\boldsymbol{K}_n = \left[\sum_{m=1}^{M} k^{(m)}(x_i^{A_m}, x_j^{A_m})\right]_{i \leq n, j \leq n}$ is the gram matrix associated with $\mathcal{D}_n$, and $\boldsymbol{y} = [y_t]_{t \leq n}$ are the concatenated observed function values. Conditioned on the observations $\mathcal{D}_n$, we can infer the posterior mean and covariance function of the function component $f^{(m)}$ to be

$$\mu_n^{(m)}(x^{A_m}) = \boldsymbol{k}_n^{(m)}(x^{A_m})^{\mathrm{T}}(\boldsymbol{K}_n + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{y},$$
$$k_n^{(m)}(x^{A_m}, x'^{A_m}) = k^{(m)}(x^{A_m}, x'^{A_m})$$
$$\quad - \boldsymbol{k}_n^{(m)}(x^{A_m})^{\mathrm{T}}(\boldsymbol{K}_n + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{k}_n^{(m)}(x'^{A_m}),$$

where $\boldsymbol{k}_n^{(m)}(x^{A_m}) = [k^{(m)}(x_t^{A_m}, x^{A_m})]_{t \leq n}$.

We use regret to evaluate the BO algorithms, both in the sequential and the batch selection case. For the sequential selection, let $\tilde{r}_t = \max_{x \in \mathcal{X}} f(x) - f(x_t)$ denote the immediate regret at iteration t. We are interested in both the averaged cumulative regret $R_T = \frac{1}{T}\sum_t \tilde{r}_t$ and the simple regret $r_T = \min_{t \leq T} \tilde{r}_t$ for a total number of T iterations. For batch evaluations, $\tilde{r}_t = \max_{x \in \mathcal{X}, b \in [B]} f(x) - f(x_{t,b})$ denotes the immediate regret obtained by the batch at iteration t. The averaged cumulative regret of the batch setting is $R_T = \frac{1}{T}\sum_t \tilde{r}_t$, and the simple regret $r_T = \min_{t \leq T} \tilde{r}_t$. We use the averaged cumulative regret in the bandit setting, where each evaluation of the function incurs a cost. If we simply want to optimize the function, we use the simple regret to capture the minimum gap between the best point found and the global optimum of the black-box function f. Note that the averaged cumulative regret upper bounds the simple regret.

3. Learning Additive Kernel Structure

We take a Bayesian view on the task of learning the latent structure of the GP kernel. The decomposition of the input space $\mathcal{X}$ will be learned simultaneously with optimization as more and more data is observed. Our generative model draws mixing proportions $\theta \sim \text{DIR}(\alpha)$. Each dimension j is assigned to one out of M groups via the decomposition assignment variable $z_j \sim \text{MULTI}(\theta)$. The objective function is then $f(x) = \sum_{m=1}^{M} f_m(x^{A_m})$, where $A_m = \{j : z_j = m\}$ is the set of support dimensions for function f_m, and each f_m is drawn from a Gaussian Process. Finally, given an input x, we observe $y \sim \mathcal{N}(f(x), \sigma)$. Figure 1 illustrates the corresponding graphical model.

Given the observed data $\mathcal{D}_n = \{(x_t, y_t)\}_{t=1}^{n}$, we obtain a posterior distribution over possible decompositions z (and mixing proportions θ) that we will include later in the BO process:

$$p(z, \theta \mid \mathcal{D}_n; \alpha) \propto p(\mathcal{D}_n \mid z)p(z \mid \theta)p(\theta; \alpha).$$

Marginalizing over θ yields the posterior distribution of the decomposition assignment

$$p(z \mid \mathcal{D}_n; \alpha) \propto p(\mathcal{D}_n \mid z) \int p(z \mid \theta)p(\theta; \alpha) \, d\theta$$

$$\propto p(\mathcal{D}_n \mid z) \frac{\Gamma(\sum_m \alpha_m)}{\Gamma(D + \sum_m \alpha_m)} \prod_m \frac{\Gamma(|A_m| + \alpha_m)}{\Gamma(\alpha_m)}$$

where $p(\mathcal{D}_n|z)$ is the data likelihood (2.1) for the additive GP given a fixed structure defined by z. We learn the posterior distribution for z via Gibbs sampling, choose the decomposition among the samples that achieves the highest data likelihood, and then proceed with BO. The Gibbs sampler repeatedly draws coordinate assignments z_j according to

$$p(z_j = m \mid z_{\neg j}, \mathcal{D}_n; \alpha) \propto p(\mathcal{D}_n \mid z)p(z_j \mid z_{\neg j})$$
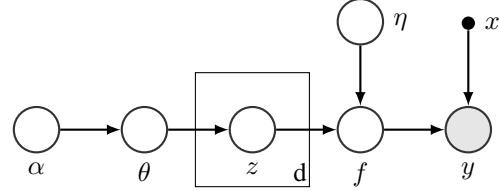$$\propto p(\mathcal{D}_n \mid z)(|A_m| + \alpha_m) \propto e^{\phi_m},$$

where

$$\phi_m = -\frac{1}{2}\boldsymbol{y}^{\text{T}}(\boldsymbol{K}_n^{(z_j = m)} + \sigma^2 \boldsymbol{I})^{-1}\boldsymbol{y}$$
$$-\frac{1}{2}\log|\boldsymbol{K}_n^{(z_j = m)} + \sigma^2 \boldsymbol{I}| + \log(|A_m| + \alpha_m)$$

and $\boldsymbol{K}_n^{(z_j = m)}$ is the gram matrix associated with the observations $\mathcal{D}_n$ by setting $z_j = m$. We can use the Gumbel trick to efficiently sample from this categorical distribution. Namely, we sample a vector of i.i.d standard Gumbel variables ω_i of length M, and then choose the sampled decomposition assignment $z_j = \arg\max_{i \leq M} \phi_i + \omega_i$.

With a Dirichlet process, we could make the model non-parametric and the number M of possible groups in the decomposition infinite. Given that we have a fixed number of input dimension D, we set $M = D$ in practice.

Figure 1. Graphical model for the structured Gaussian process; η is the hyperparameter of the GP kernel; z controls the decomposition for the input space.

4. Diverse Batch Sampling

In real-world applications where function evaluations translate into time-intensive experiments, the typical sequential exploration strategy – observe one function value, update the model, then select the next observation – is undesirable. *Batched Bayesian Optimization (BBO)* (Azimi et al., 2010; Contal et al., 2013; Kathuria et al., 2016) instead selects a batch of B observations to be made in parallel, then the model is updated with all simultaneously.

Extending this scenario to high dimensions, two questions arise: (1) the acquisition function is expensive to optimize and (2), by itself, does not sufficiently account for exploration. The additive kernel structure improves efficiency for (1). For batch selection (2), we need an efficient strategy that enourages observations that are both informative and non-redundant. Recent work (Contal et al., 2013; Kathuria et al., 2016) selects a point that maximizes the acquisition function, and adds additional batch points via a diversity criterion. In high dimensions, this diverse selection becomes expensive. For example, if each dimension has a finite number of possible values[1], the cost of sampling batch points via a *Determinantal Point Process (DPP)*, as proposed in (Kathuria et al., 2016), grows exponentially with the number of dimensions. The same obstacle arises with the approach by Contal et al. (2013), where points are selected greedily. Thus, naïve adoptions of these approaches in our setting would result in intractable algorithms. Instead, we propose a general approach that explicitly takes advantage of the structured kernel to enable relevant, non-redundant high-dimensional batch selection.

We describe our approach for a single decomposition sampled from the posterior; it extends to a distribution of decompositions by sampling a set of decompositions from the posterior and then sampling points for each decomposition individually. Given a decomposition z, we define a separate Determinantal Point Process (DPP) on each group of A_m dimensions. A set S of points in the subspace $\mathbb{R}^{|A_m|}$ is sampled with probability proportional to $\det(\mathbf{K}_n^{(m)}(S))$,

---

[1]While we use this discrete categorical domain to illustrate the batch setting, our proposed method is general and is applicable to continuous box-constrained domains.

where $\mathbf{K}_n^{(m)}$ is the posterior covariance matrix of the m-th group given n observations, and $\mathbf{K}(S)$ is the submatrix of $\mathbf{K}$ with rows and columns indexed by S. Assuming the group sizes are upper-bounded by some constant, sampling from each such DPP individually implies an exponential speedup compared to using the full kernel.

Sampling vs. Greedy Maximization The determinant $\det(\mathbf{K}_n^{(m)}(S))$ measures diversity, and hence the DPP assigns higher probability to diverse subsets S. An alternative to sampling is to directly maximize the determinant. While this is NP-hard, a greedy strategy gives an approximate solution, and is used in (Kathuria et al., 2016), and in (Contal et al., 2013) as Pure Exploration (PE). We too test this strategy in the experiments. In the beginning, if the GP is not approximating the function well, then greedy may perform no better than a stochastic combination of coordinates, as we observe in Fig. 6.

Sample Combination Now we have chosen a diverse subset $\mathcal{X}_m = \{x_i^{(m)}\}_{i \in [B-1]} \subset \mathbb{R}^{|A_m|}$ of size $(B-1)$ for each group A_m. We need to combine these subspace points to obtain $B-1$ final batch query points in $\mathbb{R}^D$. A simple way to combine samples from each group is to do it *randomly without replacement*, i.e., we sample one $x_i^{(m)}$ from each $\mathcal{X}_m$ uniformly randomly without replacement, and combine the parts, one for each $m \in [M]$, to get one sample in $\mathbb{R}^D$. We repeat this procedure until we have $(B-1)$ points. This retains diversity across the batch of samples, since the samples are diverse within each group of features.

Besides this random combination, we can also combine samples greedily. We define a *quality function* $\psi_t^{(m)}$ for each group $m \in [M]$ at time t, and combine samples to maximize this quality function. Concretely, for the first point, we combine the maximizers $x_*^{(m)} = \arg\max_{x^{(m)} \in \mathcal{X}_m} \psi_t^{(m)}(x^{(m)})$ from each group. We remove those used parts, $\mathcal{X}_m \leftarrow \mathcal{X}_m \setminus \{x_*^{(m)}\}$, and repeat the procedure until we have $(B-1)$ samples. In each iteration, the sample achieving the highest quality score gets selected, while diversity is retained.

Both selection strategies can be combined with a wide range of existing quality and acquisition functions.

Add-UCB-DPP-BBO We illustrate the above framework with GP-UCB (Srinivas et al., 2012) as both the acquisition and quality functions. The Upper Confidence Bound $(f_t^{(m)})^+$ and Lower Confidence Bound $(f_t^{(m)})^-$ with parameter β_t for group m at time t are

$$(f_t^{(m)})^+(x) = \mu_{t-1}^{(m)}(x) + \beta_t^{1/2}\sigma_t^{(m)}(x); \qquad (4.1)$$
$$(f_t^{(m)})^-(x) = \mu_{t-1}^{(m)}(x) - \beta_t^{1/2}\sigma_t^{(m)}(x),$$

and combine the expected value $\mu_{t-1}^{(m)}(x)$ of $f_t^{(m)}$ with the uncertainty $\beta_t^{1/2}\sigma_t^{(m)}(x)$. We set both the acquisition function and quality function $\psi_t^{(m)}$ to be $(f_t^{(m)})^+$ for group m at time t.

To ensure that we select points with high acquisition function values, we follow (Contal et al., 2013; Kathuria et al., 2016) and define a relevance region $\mathcal{R}_t^{(m)}$ for each group m as

$$\mathcal{R}_t^{(m)} = \{x \in \mathcal{X}_m \mid$$
$$\mu_{t-1}^{(m)}(x) + 2\sqrt{\beta_{t+1}^{(m)}}\sigma_{t-1}^{(m)}(x) \geq (y_t^{(m)})^\bullet\},$$

where $(y_t^{(m)})^\bullet = \max_{x^{(m)} \in \mathcal{X}_m}(f_t^{(m)})^-(x^{(m)})$. We then use $\mathcal{R}_t^{(m)}$ as the ground set to sample with PE/DPP. The full algorithm is shown in the appendix.

5. Empirical Results

We empirically evaluate our approach in two parts: First, we verify the effectiveness of using our Gibbs sampling algorithm to learn the additive structure of the unknown function, and then we test our batch BO for high dimensional problems with the Gibbs sampler. Our code is available at https://github.com/zi-w/Structural-Kernel-Learning-for-HDBBO.

5.1. Effectiveness of Decomposition Learning

We first probe the effectiveness of using the Gibbs sampling method described in Section 3 to learn the decomposition of the input space. More details of the experiments including sensitivity analysis for α can be found in the appendix.

Recovering Decompositions First, we sample test functions from a known additive Gaussian Process prior with zero-mean and isotropic Gaussian kernel with bandwidth $= 0.1$ and scale $= 5$ for each function component. For $D = 2, 5, 10, 20, 50, 100$ input dimensions, we randomly sample decomposition settings that have at least two groups in the decomposition and at most 3 dimensions in each group.

Table 1. Empirical posterior of any two dimensions correctly being grouped together by Gibbs sampling.

N / D	50	150	250	450
5	0.81 ± 0.28	0.91 ± 0.19	1.00 ± 0.03	1.00 ± 0.00
10	0.21 ± 0.13	0.54 ± 0.25	0.68 ± 0.25	0.93 ± 0.15
20	0.06 ± 0.06	0.11 ± 0.08	0.20 ± 0.12	0.71 ± 0.22
50	0.02 ± 0.03	0.02 ± 0.02	0.03 ± 0.03	0.06 ± 0.04
100	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.02

We set the burn-in period to be 50 iterations, and the total number of iterations for Gibbs sampling to be 100. In Tables 4 and 5, we show two quantities that are closely related
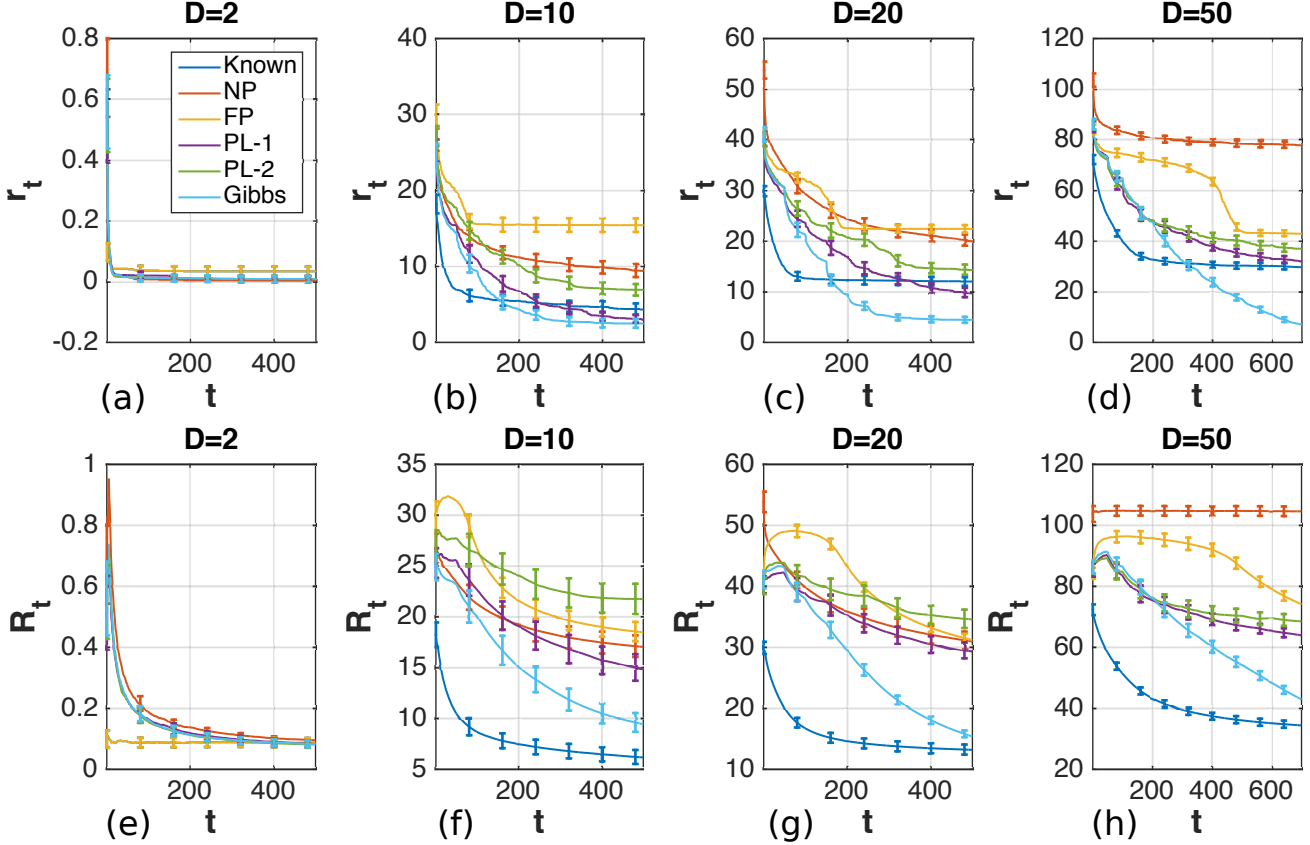
Figure 2. The simple regrets (r_t) and the averaged cumulative regrets (R_t) for setting input space decomposition with Known, NP, FP, PL-1, PL-2, and Gibbs on 2, 10, 20, 50 dimensional synthetic additive functions. Gibbs achieved comparable results to Known. Comparing PL-1 and PL-2 we can see that sampling more settings of decompositions did help to find a better decomposition. But a more principled way of learning the decomposition using Gibbs can achieve much better performance than PL-1 and PL-2.

to the learned empirical posterior of the decompositions with different numbers of randomly sampled observed data points (N). Table 4 shows the probability of two dimensions being correctly grouped together by Gibbs sampling in each iteration of Gibbs sampling after the burn-in period, namely, $(\sum_{i<j\leq D} \mathbb{1}_{z_i^g \equiv z_j^g \wedge z_i \equiv z_j})/(\sum_{i<j\leq D} \mathbb{1}_{z_i \equiv z_j})$. Table 5 reports the probability of two dimensions being correctly separated in each iteration of Gibbs sampling after the burn-in period, namely, $(\sum_{i<j\leq D} \mathbb{1}_{z_i^g \neq z_j^g \wedge z_i \neq z_j})/(\sum_{i<j\leq D} \mathbb{1}_{z_i \neq z_j})$. The results show that the more data we observe, the more accurate the learned decompositions are. They also suggest that the Gibbs sampling procedure can converge to the ground truth decomposition with enough data for relatively small numbers of dimensions. The higher the dimension, the more data we need to recover the true decomposition.

Effectiveness of Learning Decompositions for Bayesian Optimization To verify the effectiveness of the learned decomposition for Bayesian optimization, we tested on 2, 10, 20 and 50 dimensional functions sampled from a zero-mean Add-GP with randomly sampled decomposi-

Table 2. Empirical posterior of any two dimensions correctly being separated by Gibbs sampling.

N \ D	50	150	250	450
2	0.30 ± 0.46	0.30 ± 0.46	0.90 ± 0.30	1.00 ± 0.00
5	0.87 ± 0.17	0.80 ± 0.27	0.60 ± 0.32	0.50 ± 0.34
10	0.88 ± 0.05	0.89 ± 0.06	0.89 ± 0.07	0.94 ± 0.07
20	0.94 ± 0.02	0.94 ± 0.02	0.94 ± 0.02	0.97 ± 0.02
50	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.01	0.98 ± 0.01
100	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00

tion settings (at least two groups, at most 3 dimensions in each group) and isotropic Gaussian kernel with bandwidth $= 0.1$ and scale $= 5$. Each experiment was repeated 50 times. An example of a 2-dimensional function component is shown in the appendix. For Add-GP-UCB, we used $\beta_t^{(m)} = |A_m| \log 2t$ for lower dimensions ($D = 2, 5, 10$), and $\beta_t^{(m)} = |A_m| \log 2t/5$ for higher dimensions ($D = 20, 30, 50$). We show parts of the results on averaged cumulative regret and simple regret in Fig. 10, and the rest in the appendix. We compare Add-GP-UCB with known additive structure (Known), no partitions (NP), fully partitioned with one dimension for each group (FP)

and the following methods of learning the decomposition: Gibbs sampling (Gibbs), randomly sampling the same number of decompositions sampled by Gibbs and select the one with the highest data likelihood (PL-1), randomly sampling 5 decompositions and selecting the one with the highest data likelihood (PL-2). For the latter two learning methods are referred to as "partial learning" in (Kandasamy et al., 2015). The learning of the decomposition is done every 50 iterations. Fig. 3 shows the improvement of learning decompositions with Gibbs over optimizing without partitions (NP).

Overall, the results show that Gibbs outperforms both of the partial learning methods, and for higher dimensions, Gibbs is sometimes even better than Known. Interestingly, similar results can be found in Fig. 3 (c) of (Kandasamy et al., 2015), where different decompositions than the ground truth may give better simple regret. We conjecture that this is because Gibbs is able to explore more than Known, for two reasons:

1. Empirically, Gibbs changes the decompositions across iterations, especially in the beginning. With fluctuating partitions, even exploitation leads to moving around, because the supposedly "good" points are influenced by the partition. The result is an implicit "exploration" effect that is absent with a fixed partition.

2. Gibbs sometimes merges "true" parts into larger parts. The parameter β_t in UCB depends on the size of the part, $|A_m|(\log 2t)/5$ (as in (Kandasamy et al., 2015)). Larger parts hence lead to larger β_t and hence more exploration.

Of course, more exploration is not always better, but Gibbs was able to find a good balance between exploration and exploitation, which leads to better performance. Our preliminary experiments indicate that one solution to ensure that the ground truth decomposition produces the best result is to tune β_t. Hyperparameter selection (such as choosing β_t) for BO is, however, very challenging and an active topic of research (e.g. (Wang et al., 2016a)).

Next, we test the decomposition learning algorithm on a real-world function, which returns the distance between a designated goal location and two objects being pushed by two robot hands, whose trajectory is determined by 14 parameters specifying the location, rotation, velocity, moving direction etc. This function is implemented with a physics engine, the Box2D simulator (Catto, 2011). We use add-GP-UCB with different ways of setting the additive structure to tune the parameters for the robot hand so as to push the object closer to the goal. The regrets are shown in Fig. 4. We observe that the performance of learning the decomposition with Gibbs dominates all existing
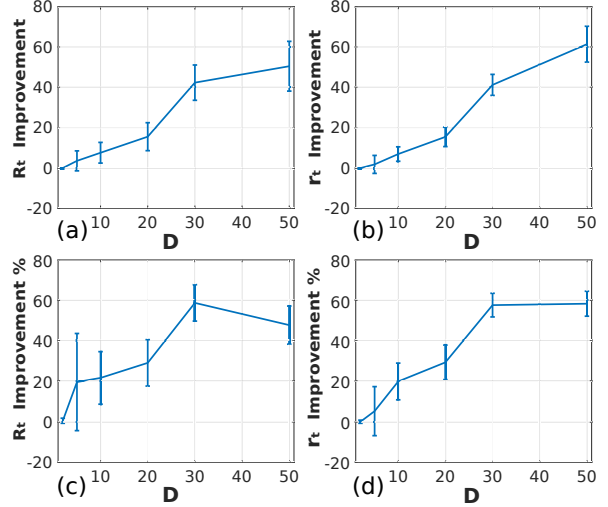

Figure 3. Improvement made by learning the decomposition with Gibbs over optimizing without partitions (NP). (a) averaged cumulative regret; (b) simple regret. (c) averaged cumulative regret normalized by function maximum; (d) simple regret normalized by function maximum. Using decompositions learned by Gibbs continues to outperform BO without Gibbs.

alternatives including partial learning. Since the function we tested here is composed of the distance to two objects, there could be some underlying additive structure for this function in certain regions of the input space, e.g. when the two robots hands are relatively distant from each other so that one of the hands only impacts one of the objects. Hence, it is possible for Gibbs to learn a good underlying additive structure and perform effective BO with the structures it learned.
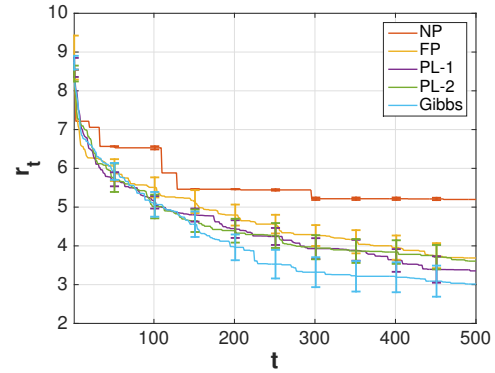

Figure 4. Simple regret of tuning the 14 parameters for a robot pushing task. Learning decompositions with Gibbs is more effective than partial learning (PL-1, PL-2), no partitions (NP), or fully partitioned (FP). Learning decompositions with Gibbs helps BO to find a better point for this tuning task.
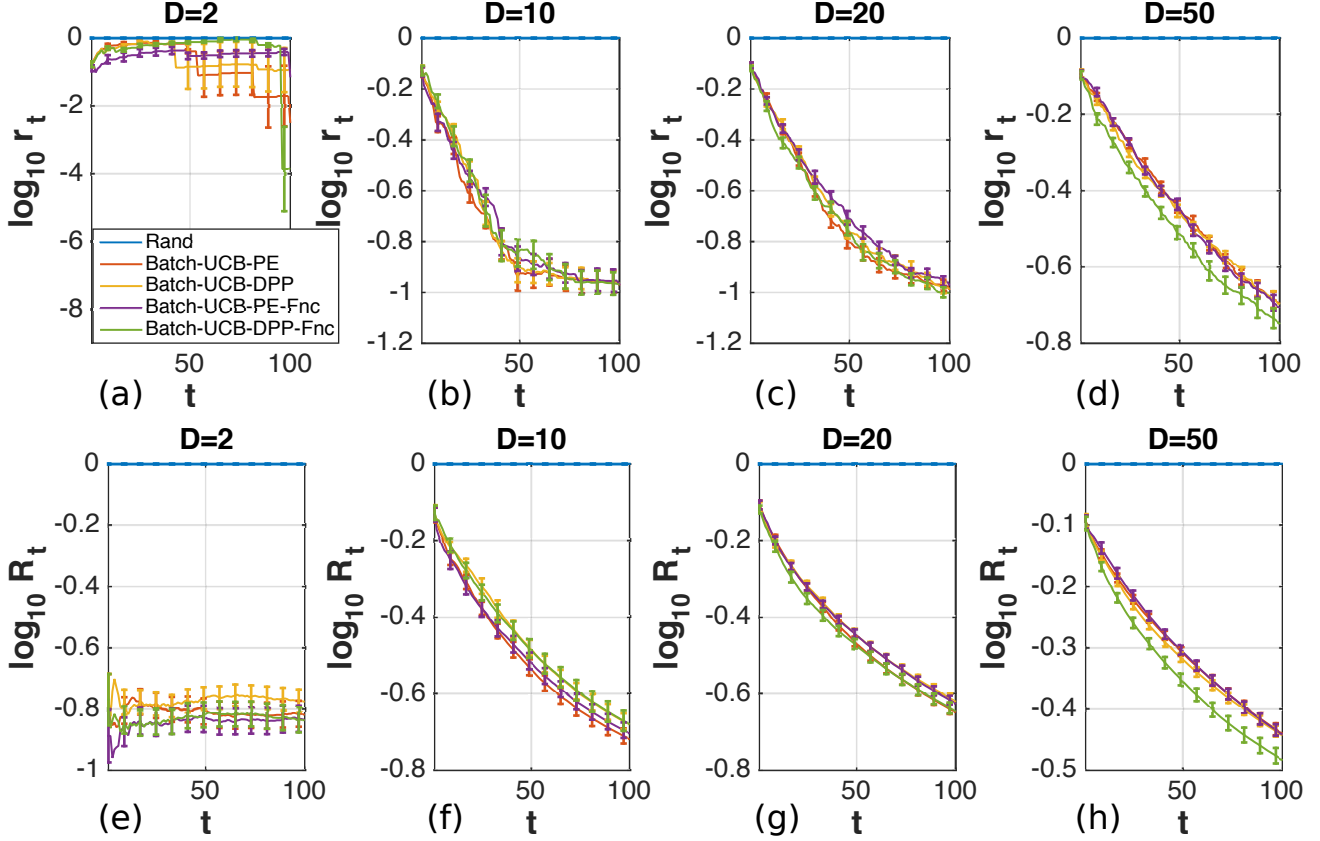
Figure 5. Scaled simple regrets (r_t) and scaled averaged cumulative regrets (R_t) on synthetic functions with various dimensions when the ground truth decomposition is known. The batch sampling methods (`Batch-UCB-PE`, `Batch-UCB-DPP`, `Batch-UCB-PE-Fnc` and `Batch-UCB-DPP-Fnc`) perform comparably well and outperform random sampling (`Rand`) by a large gap.

5.2. Diverse Batch Sampling

Next, we probe the effectiveness of batch BO in high dimensions. In particular, we compare variants of the Add-UCB-DPP-BBO approach outlined in Section 4, and a baseline:

- `Rand`: All batch points are chosen uniformly at random from $\mathcal{X}$.

- `Batch-UCB-*`: $* \in \{\text{PE}, \text{DPP}\}$. All acquisition functions are UCB (Eq. 4.1). Exploration is done via PE or DPP with posterior covariance kernels for each group. Combination is via sampling without replacement.

- `*-Fnc`: $* \in \{\text{Batch-UCB-PE}, \text{Batch-UCB-DPP}\}$. All quality functions are also UCB's, and combination is done by maximizing the quality functions.

A direct application of existing batch selection methods is very inefficient in the high-dimensional settings where they differ more, algorithmically, from our approach that ex-

ploits decompositions. Hence, we only compare to uniform sampling as a baseline.

Effectiveness We tested on 2, 10, 20 and 50-dimensional functions sampled the same way as in Section 5.1; we assume the ground-truth decomposition of the feature space is known. Since `Rand` performs the worst, we show relative averaged cumulative regret and simple regret of all methods compared to `Rand` in Fig. 5. Results for absolute values of regrets are shown in the appendix. Each experiment was repeated for 20 times. For all experiments, we set $\beta_t^m = |A_m| \log 2t$ and $B = 10$. All diverse batch sampling methods perform comparably well and far better than `Rand`, although there exist slight differences. While in lower dimensions ($D \in \{2, 10\}$), `Batch-UCB-PE-Fnc` performs among the best, in higher dimensions ($D \in \{20, 50\}$), `Batch-UCB-DPP-Fnc` performs better than (or comparable to) all other variants. We will see a larger performance gap in later real-world experiments, showing that biasing the combination towards higher quality functions while retaining diversity across the batch of samples provides a better exploration-exploitation trade-off.

For a real-data experiment, we tested the diverse batch sampling algorithms for BBO on the `Walker` function which returns the walking speed of a three-link planar bipedal walker implemented in Matlab (Westervelt et al., 2007). We tune 25 parameters that may influence the walking speed, including 3 sets of 8 parameters for the ODE solver and 1 parameter specifying the initial velocity of the stance leg. We discretize each dimension into 40 points, resulting in a function domain of $|\mathcal{X}| = 40^{25}$. This size is very inefficient for existing batch sampling techniques. We learn the additive structure via Gibbs sampling and sample batches of size $B = 5$. To further improve efficiency, we limit the maximum size of each group to 2. The regrets for all methods are shown in Fig. 6. Again, all diverse batch sampling methods outperform `Rand` by a large gap. Moreover, `Batch-UCB-DPP-Fnc` is a bit better than other variants, suggesting that a selection by quality functions is useful.
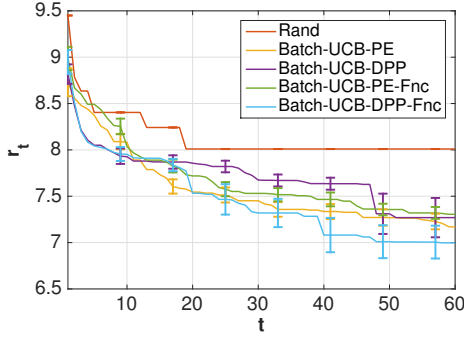

Figure 6. The simple regrets (r_t) of batch sampling methods on `Walker` data where $B = 5$. Four diverse batch sampling methods (`Batch-UCB-PE`, `Batch-UCB-DPP`, `Batch-UCB-PE-Fnc` and `Batch-UCB-DPP-Fnc`) outperform random sampling (`Rand`) by a large gap. `Batch-UCB-DPP-Fnc` performs the best among the four diverse batch sampling methods.

Batch Sizes Finally, we show how the batch size B affects the performance of the proposed methods. We test the algorithms on the 14-dimensional `Robot` dataset with $B \in \{5, 10\}$. The regrets are shown in Fig. 4. With larger batches, the differences between the batch selection approaches become more pronounced. In both settings, `Batch-UCB-DPP-Fnc` performs a bit better than other variants, in particular with larger batch sizes.

6. Conclusion

In this paper, we propose two novel solutions for high dimensional BO: inferring latent structure, and combining it with batch Bayesian Optimization. The experimental results demonstrate that the proposed techniques are effective at optimizing high-dimensional black-box functions.
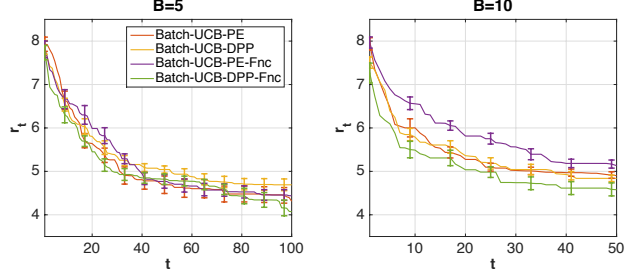

Figure 7. Simple regret when tuning the 14 parameters of a robot pushing task with batch size 5 and 10. Learning decompositions with Gibbs sampling and diverse batch sampling are employed simultaneously. In general, `Batch-UCB-DPP-Fnc` performs a bit better than the other four diverse batch sampling variants. The gap increases with batch size.

Moreover, their gain over existing methods increases as the dimensionality of the input grows. We believe that these results have the potential to enable the increased use of Bayesian optimization for challenging black-box optimization problems in machine learning that typically involve a large number of parameters.

Acknowledgements

References

Azimi, Javad, Fern, Alan, and Fern, Xiaoli Z. Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems (NIPS)*, 2010.

Catto, Erin. Box2D, a 2D physics engine for games. http://box2d.org, 2011.

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

Desautels, Thomas, Krause, Andreas, and Burdick, Joel W. Parallelizing exploration-exploitation tradeoffs in Gaus-

sian process bandit optimization. *Journal of Machine Learning Research*, 2014.

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

Duvenaud, David, Lloyd, James Robert, Grosse, Roger, Tenenbaum, Joshua B., and Ghahramani, Zoubin. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning (ICML)*, 2013.

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

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

Hernández-Lobato, José Miguel, Hoffman, Matthew W, and Ghahramani, Zoubin. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

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

Kathuria, Tarun, Deshpande, Amit, and Kohli, Pushmeet. Batched Gaussian process bandit optimization via determinantal point processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Kawaguchi, Kenji, Kaelbling, Leslie Pack, and Lozano-Pérez, Tomás. Bayesian optimization with exponential convergence. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Kawaguchi, Kenji, Maruyama, Yu, and Zheng, Xiaoyu. Global continuous optimization with error bound and fast convergence. *Journal of Artificial Intelligence Research*, 56(1):153–195, 2016.

Kushner, Harold J. 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.

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

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

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

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

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

Wang, Zi, Zhou, Bolei, and Jegelka, Stefanie. Optimization as estimation with Gaussian processes in bandit settings. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016a.

Wang, Zi, Jegelka, Stefanie, Kaelbling, Leslie Pack, and Pérez, Tomás Lozano. Focused model-learning and planning for non-Gaussian continuous state-action systems. In *International Conference on Robotics and Automation (ICRA)*, 2017.

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

Westervelt, Eric R, Grizzle, Jessy W, Chevallereau, Christine, Choi, Jun Ho, and Morris, Benjamin. *Feedback control of dynamic bipedal robot locomotion*, volume 28. CRC press, 2007.

A. Add-UCB-DPP-BBO Algorithm

We present four variants of Add-UCB-DPP-BBO in Algorithm 1. The algorithm framework is general in that, one can plug in other acquisition and quality functions other than UCB to get different algorithms.

B. Additional experiments

In this section, we provide more details in our experiments.

B.1. Optimization of the Acquisition Functions

We decompose the acquisition function into M subacquisition functions, one for each part, and optimize those separately. We randomly sample 10000 points in the low dimensional space, and then choose the one with the best value to start gradient descent in the search space (i.e. the range of the box on $R^{|A_m|}$). In practice, we observe this approach optimizes low-dimensional (< 5 dimensions) functions very well. As the number of dimensions grows, the known difficulties of high dimensional BO (and global nonconvex optimization) arise.

B.2. Effectiveness of Decomposition Learning

Recovering Decompositions In Table 6, Table 4 and Table 5, we show three quantities which may imply the quality of the learned decompositions. The first quantity , reported in Table 6, is the Rand Index of the decompositions learned by Gibbs sampling, namely,
$$\frac{\sum_{i<j\leq D} \mathbb{1}_{z_i^g \equiv z_j^g \wedge z_i \equiv z_j} + \sum_{i<j\leq D} \mathbb{1}_{z_i^g \neq z_j^g \wedge z_i \neq z_j}}{\binom{D}{2}}.$$ The second quantity, reported in Table 4, is the probability of two dimensions being correctly grouped together by Gibbs sampling in each iteration of Gibbs sampling after the burn-in period, namely, $\frac{\sum_{i<j\leq D} \mathbb{1}_{z_i^g \equiv z_j^g \wedge z_i \equiv z_j}}{\sum_{i<j\leq D} \mathbb{1}_{z_i \equiv z_j}}$. The third quantity, reported in Table 5, is the probability of two dimensions being correctly separated by Gibbs sampling in each iteration of Gibbs sampling after the burn-in period, namely, $\frac{\sum_{i<j\leq D} \mathbb{1}_{z_i^g \neq z_j^g \wedge z_i \neq z_j}}{\sum_{i<j\leq D} \mathbb{1}_{z_i \neq z_j}}$.

Sensitivity Analysis for α Empirically, we found that the quality of the learned decompositions is not very sensitive to the scale of α (see Table 6), because the log data likelihood plays a much more important role than $\log(|A_m| + \alpha)$ when α is less than the total number of dimensions. The reported results correspond to alpha = 1 for all the partitions.

BO for Synthetic Functions We show an example of a 2 dimensional function component in the additive synthetic function in Fig. 8. Because of the numerous local maxima, it is very challenging to achieve the global optimum even for 2 dimensions, let alone maximizing an additive
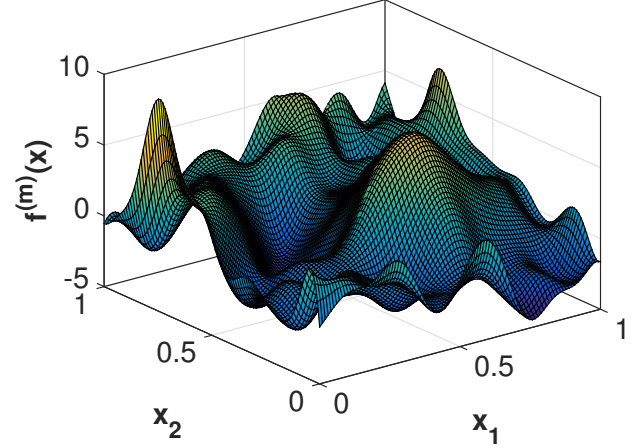

Figure 8. An example of a 2 dimensional function component of the synthetic function.

sum of them, only by observing their sum. The full results of the simple and cumulative regrets for the synthetic functions comparing Add-GP-UCB with known additive structure (Known), no partitions (NP), fully partitioned with one dimension for each group (FP) and the following methods of learning partition: Gibbs sampling (Gibbs), random sampling the same number of partitions sampled by Gibbs and select the one with the highest data likelihood (PL-1), random sampling 5 partitions and select the one with the highest data likelihood (PL-2) are shown in Fig. 10. The learning was done every 50 iterations, starting from the first iteration. For $D = 20, 30$, it is quite obvious that when a new partition is learned from the newly observed data (e.g. at iteration 100 and 150), the simple regret gets a boost.

BO for Real-world functions In addition to be 14 parameter robot pushing task, we tested on the walker function which returns the walking speed of a three-link planar bipedal walker implemented in Matlab (Westervelt et al., 2007). We tune 25 parameters that may influence the walking speed, including 3 sets of 8 parameters for the ODE solver and 1 parameter specifying the initial velocity of the stance leg. To our knowledge, this function does not have an additive structure. The regrets of each decomposition learning methods are shown in Fig. 9. In addition to Gibbs, we test learning decomposition via constrained Gibbs sampling (Gibbs-L), where the maximum size of each group of dimensions does not exceed 2. Because the function does not have additive structure, Gibbs performed poorly since it groups together many dimensions of the input. As a result, its performance is similar to that of no partition (NP). However, Gibbs-L appears to learn a good decomposition with the group size limit, and manages to achieve a slightly lower regret than other methods.

Algorithm 1 Add-UCB-DPP-BBO Variants

Input: $\mathcal{X}$, N_{init}, N_{cyc}, T, B, M

Observe function values of N_{init} points chosen randomly from $\mathcal{X}$

Get the initial decomposition of feature space via Gibbs sampling and get corresponding $\mathcal{X}_m$'s

for $t = 1$ to T **do**

  if $(t \mod N_{cyc} = 0)$ **then**

    Learn the decomposition via Gibbs sampling and get corresponding $\mathcal{X}_m$'s

  end if

  Choose x_0 by maximizing UCB (acquisition function) for each group and combine them

  for $m = 1$ to M **do**

    Compute $(\mathcal{R}_t^{(m)})^+$ and $\mathbf{K}_{(t-1)B+1}^{(m)}$

    Sample $\{x_i^{(m)}\}_{i \in [B-1]} \subseteq \mathcal{X}_m$ via PE or DPP with kernel $\mathbf{K}_{(t-1)B+1}^{(m)}$

  end for

  Combine $\{x_i^{(m)}\}_{i \in [B-1], m \in [M]}$ either randomly or by maximizing UCB (quality function) without replacement to get $\{x_i\}_{i \in [B-1]}$

  Observe (noisy) function values for $\{x_i\}$ for $i \in \{0, \ldots, B-1\}$.

end for

Table 3. Rand Index of the decompositions computed by Gibbs sampling.

D \ N	50	150	250	350	450
5	0.85 ± 0.20	0.83 ± 0.23	0.71 ± 0.18	0.68 ± 0.16	0.66 ± 0.18
10	0.78 ± 0.06	0.85 ± 0.08	0.86 ± 0.10	0.89 ± 0.12	0.95 ± 0.06
20	0.88 ± 0.02	0.88 ± 0.02	0.89 ± 0.02	0.92 ± 0.02	0.95 ± 0.04
50	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01
100	0.98 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00	0.97 ± 0.00

Table 4. Empirical posterior of any two dimensions correctly being grouped together by Gibbs sampling.

d_x \ N	50	150	250	350	450
5	0.81 ± 0.28	0.91 ± 0.19	1.00 ± 0.03	0.97 ± 0.08	1.00 ± 0.00
10	0.21 ± 0.13	0.54 ± 0.25	0.68 ± 0.25	0.81 ± 0.27	0.93 ± 0.15
20	0.06 ± 0.06	0.11 ± 0.08	0.20 ± 0.12	0.43 ± 0.17	0.71 ± 0.22
50	0.02 ± 0.03	0.02 ± 0.02	0.03 ± 0.03	0.04 ± 0.03	0.06 ± 0.04
100	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.02

Table 5. Empirical posterior of any two dimensions correctly being separated by Gibbs sampling.

d_x \ N	50	150	250	350	450
2	0.30 ± 0.46	0.30 ± 0.46	0.90 ± 0.30	0.90 ± 0.30	1.00 ± 0.00
5	0.87 ± 0.17	0.80 ± 0.27	0.60 ± 0.32	0.55 ± 0.29	0.50 ± 0.34
10	0.88 ± 0.05	0.89 ± 0.06	0.89 ± 0.07	0.91 ± 0.08	0.94 ± 0.07
20	0.94 ± 0.02	0.94 ± 0.02	0.94 ± 0.02	0.95 ± 0.02	0.97 ± 0.02
50	0.98 ± 0.00	0.98 ± 0.00	0.98 ± 0.01	0.98 ± 0.00	0.98 ± 0.01
100	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00

`Gibbs`, `PL-1`, `PL-2` and `FP` all performed relatively well in for this function, indicating that using the additive structure may benefit the BO procedure even if the function itself is not additive.

B.3. Diverse Batch Sampling

In Fig. 11, we show the full results of the simple and the cumulative regrets on the synthetic functions described in Section 5.2 of the paper.

Table 6. Rand Index of the decompositions learned by Gibbs sampling for different values of α.

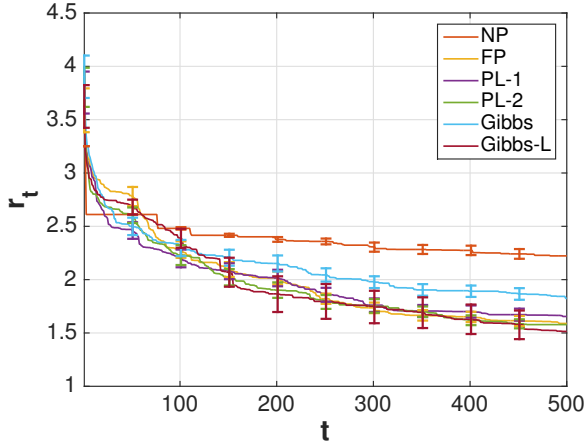| α \ N	50	150	250	350	450
0.2	0.87811 ± 0.019002	0.90126 ± 0.022394	0.95284 ± 0.047111	0.98811 ± 0.02602	0.98811 ± 0.026322
0.5	0.88211 ± 0.019893	0.90305 ± 0.024574	0.95295 ± 0.046232	0.98947 ± 0.025872	0.99505 ± 0.013881
1	0.88211 ± 0.016947	0.90326 ± 0.024935	0.95305 ± 0.043878	0.98558 ± 0.034779	0.98053 ± 0.035843
2	0.88084 ± 0.016972	0.9 ± 0.023489	0.95463 ± 0.042968	0.97989 ± 0.038818	0.98832 ± 0.023592
5	0.88337 ± 0.015784	0.90158 ± 0.02203	0.96126 ± 0.037045	0.98716 ± 0.030949	0.99316 ± 0.015491

Figure 9. Simple regret of tuning the 25 parameters for optimizing the walking speed of a bipedal robot. We use the vanilla Gibbs sampling algorithm (Gibbs) and a Gibbs sampling algorithm with partition size limit set to be 2 (Gibbs-L) to compare with partial learning (PL-1, PL-2), no partitions (NP), and fully partitioned (FP). Gibbs-L performed slightly better than PL-2 and FP. This function does not have an additive structure, and as a result, Gibbs does not perform well for this function because the sizes of the groups it learned tend to be large .
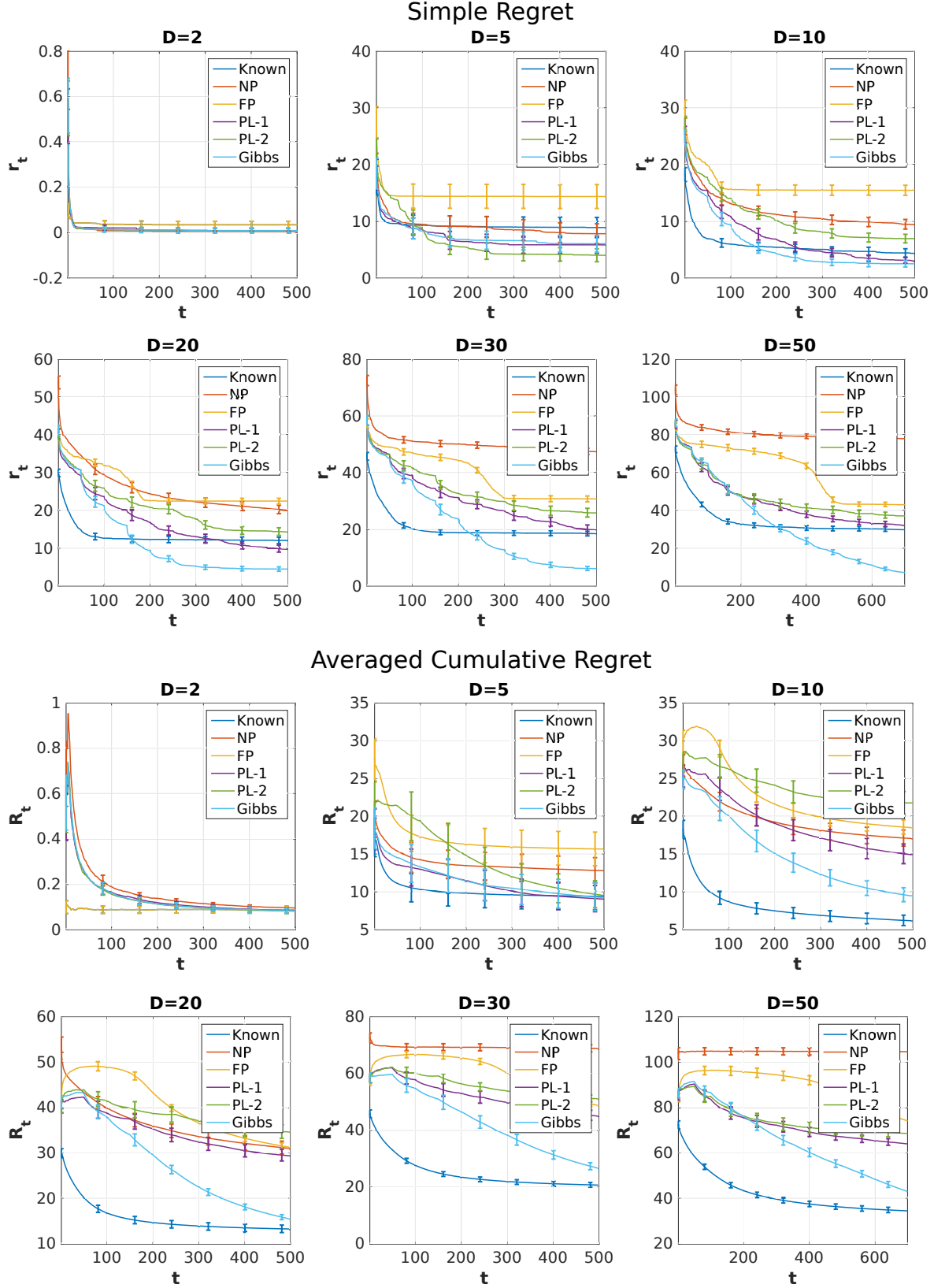
Simple Regret

Averaged Cumulative Regret

Figure 10. The simple regrets (r_t) and the averaged cumulative regrets (R_t) and for `Known` (ground truth partition is given), `Gibbs` (using Gibbs sampling to learn the partition), `PL-1` (randomly sample the same number of partitions sampled by `Gibbs` and select the one with highest data likelihood), `PL-2` (randomly sample 5 partitions and select the one with highest data likelihood), `FP` (fully partitioned, each group with one dimension) and `NP` (no partition) on 10, 20, 50 dimensional functions. `Gibbs` achieved comparable results to `Known`. Comparing `PL-1` and `PL-2` we can see that sampling more partitions did help to find a better partition. But a more principled way of learning partition using `Gibbs` can achieve much better performance than `PL-1` and `PL-2`.

Simple Regret
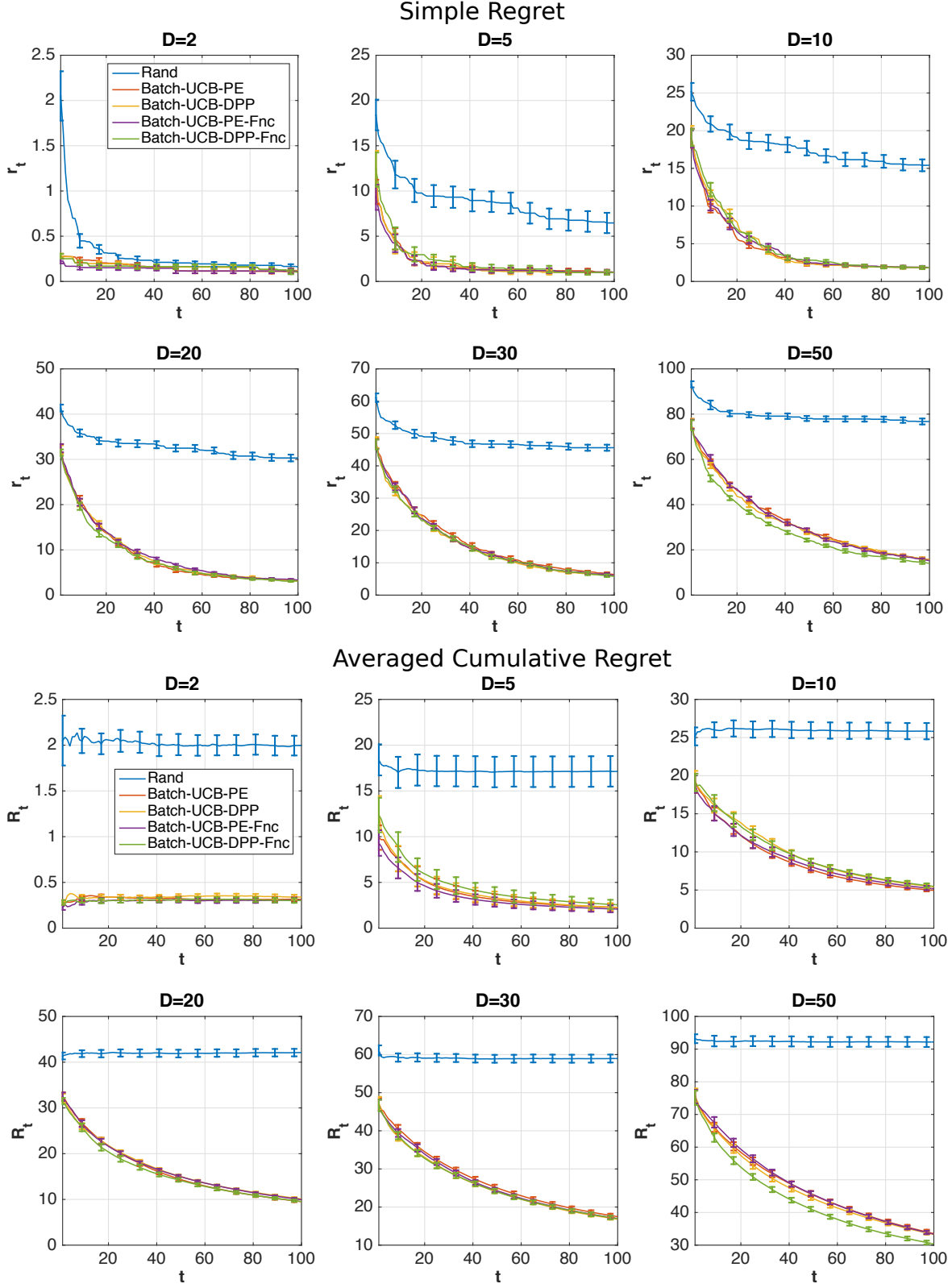

Averaged Cumulative Regret

Figure 11. The simple regrets (r_t) and the averaged cumulative regrets (R_t) on synthetic functions with various dimensions when the ground truth partition is known. Four batch sampling methods (`Batch-UCB-PE`, `Batch-UCB-DPP`, `Batch-UCB-PE-Fnc` and `Batch-UCB-DPP-Fnc`) perform comparably well and outperform random sampling (`Rand`) by a large gap.