Computationally Efficient High-Dimensional Bayesian Optimization via Variable Selection

Author Name
Affiliation
pcchair@ijcai-22.org

Abstract

Bayesian Optimization (BO) is a method for globally optimizing black-box functions. While BO has been successfully applied to many scenarios, developing effective BO algorithms that scale to functions with high-dimensional domains is still a challenge. Optimizing such functions by vanilla BO is extremely time-consuming. Alternative strategies for high-dimensional BO that are based on the idea of embedding the high-dimensional space to one with low dimension are sensitive to the choice of the embedding dimension, which needs to be pre-specified. We develop a new computationally efficient high-dimensional BO method that exploits variable selection. Our method is able to automatically learn axis-aligned sub-spaces, i.e. spaces containing selected variables, without the demand of any pre-specified hyperparameters. We analyze the computational complexity of our algorithm. We empirically show the efficacy of our method on several synthetic and real problems.

1 Introduction

We study the problem of globally maximizing a black-box function $f(\mathbf{x})$ with an input domain $\mathcal{X} = [0, 1]^D$, where the function has some special properties: (1) It is hard to calculate its first and second-order derivatives, therefore gradient-based optimization algorithms are not useful; (2) It is expensive to evaluate the function, hence some classical global optimization algorithms such as evolutionary algorithms (EA) are not applicable.

Bayesian optimization (BO) is a popular global optimization method to solve the problem described above. It aims to obtain the input $\mathbf{x}^*$ that maximizes the function f by sequentially acquiring queries that are likely to achieve the maximum and evaluating the function on these queries. BO has been successfully applied in many scenarios such as hyperparameter tuning [Snoek *et al.*, 2012; Klein *et al.*, 2017], reinforcement learning [Brochu *et al.*, 2010; Marco *et al.*, 2017], robotics [Calandra *et al.*, 2016; Berkenkamp *et al.*, 2016], and chemical design [Griffiths and Hernández-Lobato, 2017]. However, most problems described above that have been solved by BO successfully have black-box functions with low-dimensional domains, typically with $D \leq 20$ [Frazier, 2018]. Scaling BO to high-dimensional black-box functions is challenging because of the following two reasons: (1) Due to the curse of dimensionality, the global optima is harder to find as D increases; and (2) computationally, vanilla BO is extremely time consuming on functions with large D. As global optimization for high-dimensional black-box function has become a necessity in several scientific fields such as algorithm configuration [Hutter *et al.*, 2010], computer vision [Bergstra *et al.*, 2013] and biology [Gonzalez *et al.*, 2015], developing new BO algorithms that can effectively optimize black-box functions with high dimensions is important for practical applications.

A large class of algorithms for high-dimensional BO is based on the assumption that the black-box function has an effective subspace with dimension $d_e \ll D$ [Djolonga *et al.*, 2013; Wang *et al.*, 2016; Moriconi *et al.*, 2019; Nayebi *et al.*, 2019; Letham *et al.*, 2020]. Therefore, these algorithms first embed the high-dimensional domain $\mathcal{X}$ to a space with the embedding dimension d pre-specified by users, perform vanilla BO in the embedding space to obtain the new query, and then project the query back and evaluate the function f. These algorithms are time efficient since BO is done in a low-dimensional space. Wang [2016] proves that if $d \geq d_e$, then theoretically with probability 1 the embedding space contains the global optimum. However, since d_e is usually not known, it is difficult for users to set a suitable d. Previous work such as Eriksson [2021] shows that different settings of d will impact the performance of embedding-based algorithms, and there has been little work on how to choose d heuristically. Letham [2020] also points out that when projecting the optimal point in the embedding space back to the original space, it is not guaranteed that this projection point is inside $\mathcal{X}$, hence algorithms may fail to find an optimum within the input domain.

We develop a new algorithm, called VS-BO (Variable Selection Bayesian Optimization), to solve issues mentioned above. Our method is based on the assumption that all the D variables (elements) of the input $\mathbf{x}$ can be divided into two disjoint sets $\mathbf{x} = \{\mathbf{x}_{ipt}, \mathbf{x}_{nipt}\}$: (1) $\mathbf{x}_{ipt}$, called important variables, are variables that have significant effect on the output value of f; (2) $\mathbf{x}_{nipt}$, called unimportant variables, are variables that have no or little effect on the output. Previous work such as Hutter [2014] shows that the performance of

many machine learning methods is strongly affected by only a small subset of hyperparameters, indicating the rationality of this assumption. We propose a robust strategy to identify $\mathbf{x}_{ipt}$ and perform BO on the space of $\mathbf{x}_{ipt}$ to reduce time consumption. In particular, our method is able to learn the dimension of $\mathbf{x}_{ipt}$ automatically, hence there is no need to pre-specify the hyperparameter d as with embedding-based algorithms. Since the space of $\mathbf{x}_{ipt}$ is axis-aligned, issues caused by the space projection do not exist in our method. We analyze the computational complexity of VS-BO, showing that our method can decrease the computational complexity of both steps of fitting the Gaussian Process (GP) and optimizing the acquisition function. We empirically show the good performance of VS-BO on synthetic and real problems.

2 Related work

The basic framework of BO has two steps for each iteration: First, GP is used as the surrogate to model f based on all the previous query-output pairs $\left(\mathbf{x}^{1:n}, y^{1:n}\right)$:

$$y^{1:n} \sim \mathcal{N}\left(\mathbf{0}, K(\mathbf{x}^{1:n}, \Theta) + \sigma_0^2 \mathbf{I}\right) \tag{1}$$

Here $y^{1:n} = [y^1, \ldots, y^n]$ is a n-dimensional vector, $y^i = f(\mathbf{x}^i) + \epsilon^i$ is the output of f with random noise $\epsilon^i \sim \mathcal{N}(0, \sigma_0^2)$, and $K(\mathbf{x}^{1:n}, \Theta)$ is a $n \times n$ covariance matrix where its entry $K_{i,j} = k(\mathbf{x}^i, \mathbf{x}^j, \Theta)$ is the value of a kernel function k in which $\mathbf{x}^i$ and $\mathbf{x}^j$ are the i-th and j-th queries respectively. Θ and σ_0 are parameters of GP that will be optimized each iteration, and $\mathbf{I}$ is the $n \times n$ identity matrix. A detailed description of GP and its applications can be found in Williams [2006].

Given a new input $\mathbf{x}$, we can compute the posterior distribution of $f(\mathbf{x})$ from GP, which is again a Gaussian distribution with mean $\mu(\mathbf{x} \mid \mathbf{x}^{1:n}, y^{1:n})$ and variance $\sigma^2(\mathbf{x} \mid \mathbf{x}^{1:n})$ that have the following forms:

$$\mu(\mathbf{x} \mid \mathbf{x}^{1:n}, y^{1:n}) = \mathbf{k}(\mathbf{x}, \mathbf{x}^{1:n})[K(\mathbf{x}^{1:n}, \Theta) + \sigma_0^2 \mathbf{I}]^{-1}(y^{1:n})^\top \tag{2}$$

$$\begin{aligned}\sigma^2(\mathbf{x} \mid \mathbf{x}^{1:n}) = &- \mathbf{k}(\mathbf{x}, \mathbf{x}^{1:n})[K(\mathbf{x}^{1:n}, \Theta) + \sigma_0^2 \mathbf{I}]^{-1}\mathbf{k}(\mathbf{x}, \mathbf{x}^{1:n})^\top \\ &+ k(\mathbf{x}, \mathbf{x}, \Theta)\end{aligned} \tag{3}$$

Here, $\mathbf{k}(\mathbf{x}, \mathbf{x}^{1:n}) = [k(\mathbf{x}, \mathbf{x}^1, \Theta), \ldots, k(\mathbf{x}, \mathbf{x}^n, \Theta)]$ is a n-dimensional vector.

The second step of BO is to use μ and σ to construct an acquisition function acq and maximize it to get the new query $\mathbf{x}^{new}$, on which the function f is evaluated to obtain the new pair $(\mathbf{x}^{new}, y^{new})$:

$$\mathbf{x}^{new} = \text{argmax}_{\mathbf{x} \in \mathcal{X}}\ acq(\mu(\mathbf{x} \mid \mathbf{x}^{1:n}, y^{1:n}), \sigma(\mathbf{x} \mid \mathbf{x}^{1:n})) \tag{4}$$

A wide variety of methods have been proposed that are related to high-dimensional BO, and most of them are based on some extra assumptions on intrinsic structures of the domain $\mathcal{X}$ or the function f. As mentioned in the previous section, a considerable body of algorithms is based on the assumption that the black-box function has an effective subspace with a significantly smaller dimension than $\mathcal{X}$. Among them, REMBO [Wang *et al.*, 2016] uses a randomly generated matrix as the projection operator to embed $\mathcal{X}$ to a low-dimensional subspace. SI-BO [Djolonga *et al.*, 2013], DSA [Ulmasov *et al.*, 2016] and MGPC-BO [Moriconi *et al.*, 2019] propose different ways to learn the projection operator from data, of which the major shortcoming is that a large number of data points are required to make the learning process accurate. HeSBO [Nayebi *et al.*, 2019] uses a hashing-based method to do subspace embedding. Finally, ALEBO [Letham *et al.*, 2020] aims to improve the performance of REMBO with several novel refinements.

Another assumption is that the black-box function has an additive structure. Kandasamy [2015] first develops a high-dimensional BO algorithm called Add-GP by adopting this assumption. They derive a simplified acquisition function and prove that the regret bound is linearly dependent on the dimension. Their framework was subsequently generalized by Li [2016], Wang [2017] and Rolland [2018].

Several approaches attempt to solve the high-dimensional BO problem by developing more efficient methods to optimize the acquisition function instead of adding extra assumptions. For example, Rana [2017] builds a sequence of GPs and optimizes a series of acquisition functions to make the gradient-based methods applicable even in the region where the acquisition function is flattened, and Kirschner [2019] develops a method called LineBO to optimize the acquisition function on a one-dimensional line each time.

Our method is based on the assumption that some variables are more "important" than others, which is similar to an axis-aligned subspace embedding. Several previous works propose different methods to choose axis-aligned subspaces in high-dimensional BO. Li [2016] uses the idea of dropout, i.e, for each iteration of BO, a subset of variables are randomly chosen and optimized, while our work chooses variables that are important in place of the randomness. Gupta [2020] proposes to optimize the acquisition function on a finite set of axis-aligned subspaces, while the dimension of the subspace is still pre-specified. Eriksson [2021] develops a method called SAASBO, which uses the idea of Bayesian inference. SAASBO defines a prior distribution for each parameter in the kernel function k, and for each iteration the parameters are sampled from posterior distributions and used in the step of optimizing the acquisition function. Since those priors restrict parameters to concentrate near zero, the method is able to learn a sparse axis-aligned subspace (SAAS) during the BO process. The main drawback of SAASBO is that it is very time consuming. While traditionally it is assumed that the function f is very expensive to evaluate so that the runtime of BO itself does not need to be considered, previous work such as Ulmasov [2016] points out that in some applications the runtime of BO cannot be neglected. Spagnol [2019] proposes a similar framework of high-dimensional BO as us; they use Hilbert Schmidt Independence criterion (HSIC) to select variables and use the chosen variables to do BO. However, they use simple heuristics to handle unimportant variables, which affects the performance of the overall algorithm, and they do not analyze the computational complexity of their method. In addition, they do not provide a comprehensive comparison with other high-dimensional BO methods: their method is only compared with the method in Li [2016] on several synthetic functions.

Algorithm 1 VS-BO

Input: $f(\mathbf{x})$, $\mathcal{X} = [0,1]^D$, N_{init}, N, N_{vs}
Output: $\mathbf{x}^{max}$

1: Initialize the set of $\mathbf{x}_{ipt}$ to be all variables in $\mathbf{x}$, $\mathbf{x}_{ipt} = \mathbf{x}$, and $\mathbf{x}_{nipt} = \emptyset$
2: Uniformly sample N_{init} points $\mathbf{x}^i$ and evaluate $y^i = f(\mathbf{x}^i)$, let $\mathcal{D} = \{(\mathbf{x}^i, y^i)\}_{i=1}^{N_{init}}$
3: Initialize the distribution $p(\mathbf{x} \mid \mathcal{D})$
4: **for** $t = N_{init}+1, N_{init}+2, \dots N_{init}+N$ **do**
5: **if** mod$(t - N_{init}, N_{vs}) = 0$ **then**
6: Variable selection to update $\mathbf{x}_{ipt}$ and let $\mathbf{x}_{nipt} = \mathbf{x} \setminus \mathbf{x}_{ipt}$ (Algorithm 2)
7: Update $p(\mathbf{x} \mid \mathcal{D})$, then derive the conditional distribution $p(\mathbf{x}_{nipt} \mid \mathbf{x}_{ipt}, \mathcal{D})$
8: **end if**
9: Fit a GP to $\mathcal{D}_{ipt} := \{(\mathbf{x}_{ipt}^i, y^i)\}_{i=1}^{t-1}$
10: Maximize the acquisition function to obtain $\mathbf{x}_{ipt}^t$.
11: Sample $\mathbf{x}_{nipt}^t$ from $p(\mathbf{x}_{nipt} \mid \mathbf{x}_{ipt}^t, \mathcal{D})$
12: Evaluate $y^t = f(\mathbf{x}^t) + \epsilon^t = f(\{\mathbf{x}_{ipt}^t, \mathbf{x}_{nipt}^t\}) + \epsilon^t$ and update $\mathcal{D} = \mathcal{D} \cup \{(\mathbf{x}^t, y^t)\}$
13: **end for**
14: **return** $\mathbf{x}^{max}$ which is equal to $\mathbf{x}^i$ with maximal y^i

3 Framework of VS-BO

Given the black-box function $f(\mathbf{x}) : \mathcal{X} \to \mathbb{R}$ in the domain $\mathcal{X} = [0,1]^D$ with a large D, the goal of high-dimensional BO is to find the maximizer $\mathbf{x}^* = \text{argmax}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x})$ efficiently. VS-BO is based on the assumption that all variables in $\mathbf{x}$ can be divided into important variables $\mathbf{x}_{ipt}$ and unimportant variables $\mathbf{x}_{nipt}$, and the algorithm uses different strategies to decide values of the variables from two different sets.

The high-level framework of VS-BO is described in Algorithm 1. For every N_{vs} iterations VS-BO will update $\mathbf{x}_{ipt}$ and $\mathbf{x}_{nipt}$ (line 8 in Algorithm 1), and for every BO iteration t only variables in $\mathbf{x}_{ipt}$ are used to fit GP (line 11 in Algorithm 1), and the new query of important variables $\mathbf{x}_{ipt}^t$ is obtained by maximising the acquisition function (line 12 in Algorithm 1). VS-BO learns a conditional distribution $p(\mathbf{x}_{nipt} \mid \mathbf{x}_{ipt}, \mathcal{D})$ from the existing query-output pairs $\mathcal{D}$ (line 5, 9 in Algorithm 1). This distribution is used for choosing the value of $\mathbf{x}_{nipt}$ to make $f(\mathbf{x})$ large when $\mathbf{x}_{ipt}$ is fixed. Hence, once $\mathbf{x}_{ipt}^t$ is obtained, the algorithm samples $\mathbf{x}_{nipt}^t$ from $p(\mathbf{x}_{nipt} \mid \mathbf{x}_{ipt}^t, \mathcal{D})$ (line 13 in Algorithm 1), concatenates it with $\mathbf{x}_{ipt}^t$ and evaluates $f(\{\mathbf{x}_{ipt}^t, \mathbf{x}_{nipt}^t\})$.

Compared to Spagnol [2019], our method is new on the following two aspects: First, we propose a new variable selection method that takes full advantage of the information in the fitted GP model, and there is no hyperparameter that needs to be pre-specified in this method; Second, we integrate an evolutionary algorithm into the framework of BO to make the sampling of unimportant variables more precise. The following subsections introduce these two points in detail.

3.1 Variable selection

The variable selection step in VS-BO (Algorithm 2 in the appendix) can be further separated into two substeps: (1) calculate the importance score (IS) of each variable (line 3 in Algorithm 2), and (2) do the stepwise-forward variable selection [Derksen and Keselman, 1992] according to the importance scores.

For step one, we extended a gradient-based IS calculation method, called Grad-IS, based on Paananen [2019]. Intuitively, if the partial derivative of the function f with respect to one variable is large on average, then the variable ought to be important. Since the derivative of f is unknown, VS-BO instead estimates the expectation of the gradient of posterior mean from a fitted GP model, normalized by the posterior standard deviation:

$$\begin{aligned} IS &= \mathbb{E}_{\mathbf{x}\sim Unif(\mathcal{X})}\left[\frac{\nabla_{\mathbf{x}}\mathbb{E}_{p(f(\mathbf{x})|\mathbf{x},\mathcal{D})}\left[f(\mathbf{x})\right]}{\sqrt{Var_{p(f(\mathbf{x})|\mathbf{x},\mathcal{D})}\left[f(\mathbf{x})\right]}}\right] \\ &= \mathbb{E}_{\mathbf{x}\sim Unif(\mathcal{X})}\left[\frac{\nabla_{\mathbf{x}}\mu(\mathbf{x} \mid \mathcal{D})}{\sigma(\mathbf{x} \mid \mathcal{D})}\right] \\ &\approx \frac{1}{N_{is}}\sum_{k=1}^{N_{is}}\frac{\nabla_{\mathbf{x}}\mu(\mathbf{x}^k \mid \mathcal{D})}{\sigma(\mathbf{x}^k \mid \mathcal{D})} \quad \mathbf{x}^k \overset{i.i.d}{\sim} Unif(\mathcal{X}). \end{aligned} \tag{5}$$

Here, both $\nabla_{\mathbf{x}}\mu(\cdot \mid \mathcal{D})$ and $\sigma(\cdot \mid \mathcal{D})$ have explicit forms. Both the Grad-IS and Kullback-Leibler Divergence (KLD)-based methods in Paananen [2019] are estimations of $\mathbb{E}_{\mathbf{x}\sim Unif(\mathcal{X})}\left[\frac{\nabla_{\mathbf{x}}\mathbb{E}_{p(f(\mathbf{x})|\mathbf{x},\mathcal{D})}\left[f(\mathbf{x})\right]}{\sqrt{Var_{p(f(\mathbf{x})|\mathbf{x},\mathcal{D})}\left[f(\mathbf{x})\right]}}\right]$. Since the KLD method only calculates approximate derivatives around the chosen points in $\mathcal{D}$ that are always unevenly distributed, it is a biased estimator, while our importance score estimation is unbiased.

Each time the algorithm fits GP to the existing query-output pairs, the marginal log likelihood (MLL) of GP is maximized by updating parameters Θ and σ_0. VS-BO takes negative MLL as the loss and uses its value as the stopping criteria of the stepwise-forward selection. More specifically, VS-BO sequentially selects variables according to the important score, and when a new variable is added, the algorithm will fit GP again by only using those chosen variables and records a new final loss (line 6 in Algorithm 2). If the new loss is nearly identical to the previous loss (the loss of fitted GP when the new variable is not included), then the selection step stops (line 9 in Algorithm 2) and all those already chosen variables are important variables.

Consider the squared exponential kernel, a common kernel choice for GP, which is given by

$$k\left(\mathbf{x}, \mathbf{x}', \Theta = \{\rho_{1:D}^2, \alpha_0^2\}\right) = \alpha_0^2 \exp\left(-\tfrac{1}{2}\textstyle\sum_{i=1}^{D}\rho_i^2(\mathbf{x}_i - \mathbf{x}_i')^2\right) \tag{6}$$

where ρ_i^2 is the inverse squared length scale of the i-th variable. On the one hand, when only a small subset of variables in $\mathbf{x}$ are important, the variable selection is similar to adding a L_0 regularization for GP fitting step. Denote ρ as $\rho = [\rho_1^2, \dots, \rho_D^2]$, then we can see that the variable selection step chooses a subset of variables and specifies $\rho_i^2 = 0$ when i-th variable is in $\mathbf{x}_{nipt}$, leading $\|\rho\|_0$ to be small. Therefore, fitting GP by only using variables in $\mathbf{x}_{ipt}$ is similar to learning the kernel function with sparse parameters. On the other

hand, when in the worst case every variable is equally important, $\mathbf{x}_{ipt}$ is likely to contain nearly all the variables in $\mathbf{x}$, and in that case VS-BO degenerates to vanilla BO.

In our experiments, we use a variant version, called VS-momentum, to do the variable selection more robustly. Intuitively, queries obtained after one variable selection step can give extra information on the accuracy of this variable selection. If empirically a new maximizer is found, then this variable selection step is likely to have found real important variables, hence most of these variables should be kept at the next variable selection step. Otherwise, most should be removed and new variables need to be added. The details of VS-momentum are described in section A of the appendix.

3.2 Sampling for unimportant variables

We propose a method based on Covariance Matrix Adaptation Evolution Strategy (CMA-ES) to obtain the new value of unimportant variables for each iteration. CMA-ES is an evolutionary algorithm for numerically optimizing a function. For each generation k, the algorithm samples new offsprings from a multivariate Gaussian distribution $\mathcal{N}\left(m^{(k-1)}, (\sigma^{(k-1)})^2\right)$ and updates $m^{(k-1)}$ and $(\sigma^{(k-1)})^2$ based on these new samples and their corresponding function values. Details of this algorithm can be seen in Hansen [2016].

Using the same approach as CMA-ES, VS-BO uses the initialized data $\{(\mathbf{x}^i, y^i)\}_{i=1}^{N_{init}}$ to initialize the multivariate Gaussian distribution $p(\mathbf{x} \mid \mathcal{D})$ (line 5 in Algorithm 1), and for every N_{vs} iterations, it updates the distribution based on new query-output pairs (line 9 in Algorithm 1). Because of the property of Gaussian distribution, the conditional distribution $p(\mathbf{x}_{nipt} \mid \mathbf{x}_{ipt}, \mathcal{D})$ is easily derived and is also a multivariate Gaussian distribution. Therefore, $\mathbf{x}_{nipt}^t$ can be sampled from the Gaussian distribution $p(\mathbf{x}_{nipt} \mid \mathbf{x}_{ipt}^t, \mathcal{D})$ (line 13 in Algorithm 1) when $\mathbf{x}_{ipt}^t$ is obtained.

Compared to BO, it is much faster to update the evolutionary algorithm and obtain new queries, although these queries are less precise than those from BO. VS-BO takes advantage of the strength of these two methods by using them on different variables. Important variables are crucial to the function value, therefore VS-BO uses the framework of BO on them to obtain precise queries. Unimportant variables do not effect the function value too much so there is no need to spend large time budget to search for extremely precise values. Hence, they are determined by CMA-ES to reduce runtime. In addition, when the variable selection step is inaccurate, VS-BO degenerates to an algorithm that is similar to CMA-ES rather than random sampling, therefore this sampling strategy can help improve the robustness of the performance of the whole algorithm.

4 Computational complexity analysis

From the theoretical perspective, we prove that running BO by only using those important variables is able to decrease the runtime of both the step of fitting the GP and maximizing the acquisition function. Specifically, we have the following theorem:

Theorem 1. *Suppose the cardinality of $\mathbf{x}_{ipt}$ is p and the Quasi-Newton method (QN) is used for both fitting the GP and maximizing the acquisition function. Under the choice of commonly used kernel functions and acquisition functions, if only variables in $\mathbf{x}_{ipt}$ are used for fitting the GP and maximizing the acquisition function, then the complexity of each step of QN is $\mathcal{O}(p^2 + pn^2 + n^3)$ for fitting the GP and $\mathcal{O}(p^2 + pn + n^2)$ for maximizing the acquisition function, where n is the number of queries that are already obtained.*

The proof is in section B of the appendix. Note that the method for fitting the GP and maximizing the acquisition function under the framework of BoTorch is limited-memory BFGS, which is indeed a QN method. Since the complexity is related to the square of p, selecting a small subset of variables (so that p is small) can decrease the runtime of BO. Figure 4 in the appendix empirically shows that compared to vanilla BO, VS-BO can both reduce the runtime of fitting a GP and optimizing the acquisition function, especially when n is not small.

5 Experiments

We compare VS-BO to a broad selection of existing methods: vanilla BO, REMBO and its variant REMBO Interleave, HeSBO and ALEBO, LineBO and SAASBO. The details of implementations of these methods as well as hyperparameter settings are described in section C of the appendix. Note that in order to decide the direction of the one-dimensional line, for each iteration LineBO needs to evaluate the black box function multiple times with multiple queries, making the comparison between this method and other BO methods unfair. In our experiments, LineBO will evaluate the black box function with 10 different queries for each iteration.

5.1 Synthetic problems

We use the Hartmann6 ($d_e = 6$), Styblinski-Tang4 ($d_e = 4$) and Branin ($d_e = 2$) functions as test functions. Previous high-dimensional BO work extends these functions to high dimension by adding unrelated variables, while in our work we present a harder test setting that has not been tried before by adding both unrelated and unimportant (but not totally unrelated) variables. For example, in the Hartmann6 case with the standard Hartmann6 function $f_{Hartmann6}(\mathbf{x}_{[1:6]})$ we first construct a new function $F_{hm6}(\mathbf{x})$ by adding variables with a range of importance, $F_{hm6}(\mathbf{x}) = f_{Hartmann6}(\mathbf{x}_{[1:6]}) + 0.1 f_{Hartmann6}(\mathbf{x}_{[7:12]}) + 0.01 f_{Hartmann6}(\mathbf{x}_{[13:18]})$, and we further extend it to $D = 50$ by adding unrelated variables; see section D for full details. The dimension of the effective subspace of F_{hm6} is 18, while the dimension of important variables is only 6. We hope that VS-BO can find those important variables successfully. For each embedding-based method we evalualte both $d = 4$ and $d = 6$.

Figures 1 and Figure 5 in the appendix show performance of VS-BO and other BO methods on these three synthetic functions. When the iteration budget is fixed (Figure 1a, Figure 5a), the value on average found by VS-BO after 200 iterations is the best or comparable to the best in all three cases. When the wall clock time or CPU time budget for

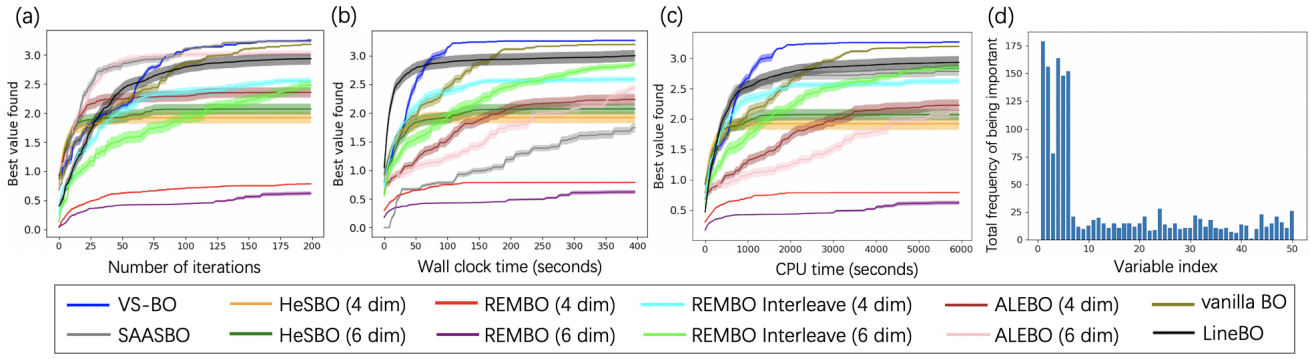

Figure 1: (a,b,c) Performance of BO methods on Hartmann6 test functions. For each test function, we perform 20 independent runs for each method except SAASBO which is very time consuming, we perform 10 runs instead. We plot the mean and 1/8 standard deviation of the best maximum value found by (a) iterations, (b) wall clock time or (c) CPU time. (d) The total frequency of being chosen as important for each variable. Note that the first 6 variables on Hartmann6 case are the most important in reality.

BO is fixed (Figure 1b, Figure 1c, Figure 5b, Figure 5c), VS-BO can find a large function value with high computational efficiency. Figures 1d and 5d show the frequency of being chosen as important for each variable in steps of variable selection of VS-BO. Since we perform 20 runs of VS-BO on each test function, each run has 200 iterations and important variables are re-selected every 20 iterations, the maximum frequency each variable can be chosen as important is 200. Most of the time VS-BO can accurately find all the real important variables and meanwhile control false positive selection of important variables. Vanilla BO under the framework of BoTorch can also achieve good performance for the fixed iteration budget, however, it is computationally inefficient. For embedding-based methods, the results reflect some of their shortcomings. First, the performance of these methods are more variable than VS-BO; for example, HeSBO with $d = 6$ performs very well in the Styblinski-Tang4 case but not in the others; Second, embedding-based methods are sensitive to the choice of the embedding dimension d: they perform especially badly when d is smaller than the dimension of important variables (see results of the Hartmann6 case) and may still perform poorly even when d is larger (such as ALEBO with $d = 6$ in the Styblinski-Tang4 case), while VS-BO can automatically learn the dimension. One advantage of embedding-based methods is that they may have a better performance than VS-BO within a very limited iteration budget (for example 50 iterations), which is expected since a number of data points are needed for VS-BO to make the variable selection accurate. Under the fixed iteration budget, SAASBO performs very well except on Styblinski-Tang4 case. However, SAASBO needs a significantly higher time budget for each iteration compared to other methods. Although evaluating the function 10 times for each iteration, LineBO is still significantly worse than any other methods (Figure 6) except on Hartmann6 case.

Spagnol [2019] introduces several simple heuristics to sample unimportant variables and shows that a mix strategy is the best, i.e. for each iteration, the algorithm samples values of unimportant variables from the uniform distribution with probability 0.5 and uses the values from the previous query that have the maximal function value with probability 0.5. To compare the mix strategy with our sampling strategy, i.e. sampling from CMA-ES related posterior, we replace our strategy with the mix strategy in VS-BO, creating a variant method called VSBO mix. All the other parts in VSBO mix are the same as those in VS-BO. We compare VS-BO with VSBO mix on these three synthetic functions, and Figure 7 in the appendix show that in general our sampling strategy is clearly better than the mix strategy.

5.2 Real-world problems

We compare VS-BO with other methods on two real-world problems. First, VS-BO is tested on the rover trajectory optimization problem presented in Wang [2017], a problem with a 60-dimensional input domain. Second, it is tested on the vehicle design problem MOPTA08 [Jones, 2008], a problem with 124 dimensions. On these two problems, we evaluate both $d = 6$ and $d = 10$ for each embedding-based method, except we omit ALEBO with $d = 10$ since it is very time consuming. The detailed settings of these two problems are described in section C of the appendix.

Figures 2a,b and 8a,b in the appendix show the performance of VS-BO and other BO methods on these two problems. When the iteration budget is fixed (Figure 2a and 8a), SAASBO has the best performance, while Figure 2b and 8b show that VS-BO is more computationally efficient than the other methods. Figure 2c and 8c in the appendix shows the frequency of being chosen as important for each variable when VS-BO is used. Since there is no ground truth of important variables in these real-world cases, we use a sampling experiment to test whether those more frequently-chosen variables are more important. Specifically, we sample the first 5 variables that have been chosen most frequently by a Sobol sequence and fix the values of other variables with the values in the best query we have found (the query having the highest function value). We then calculate the function values of this set of samples. Likewise, we also sample the first 5 variables that have been chosen least frequently and evaluate the functions. Figure 2d and 8d show that the variance of function values from the first set of samples is significantly higher

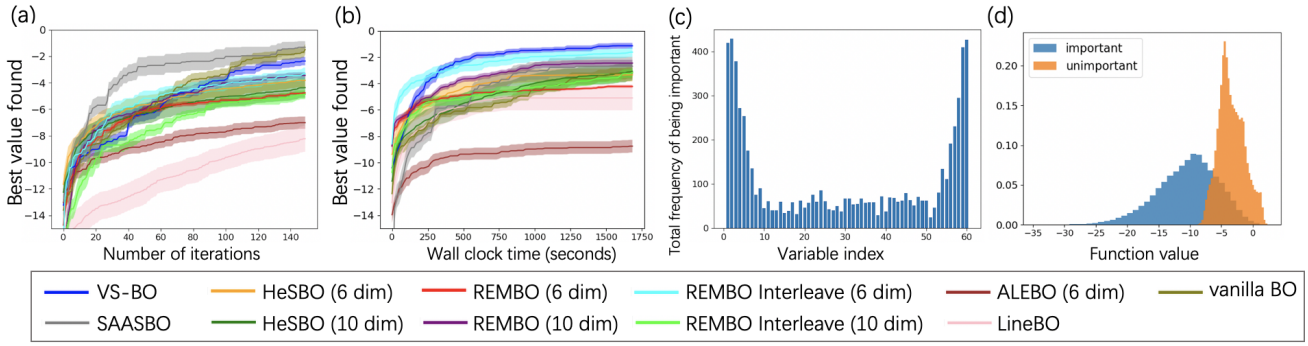

Figure 2: (a,b) Performance of BO methods on the rover trajectory problem. For each test function, we do 20 independent runs for each method except SAASBO which is very time consuming, we do 10 runs instead. We plot the mean and 1/4 standard deviation of the best maximum value found by (a) iterations and (b) wall clock time. (c) The total frequency of being chosen as important for each variable. (d) The distribution of function values when sampling the first 5 variables that have been chosen most frequently (important) or the first 5 variables that have been chosen least frequently (unimportant) with all the other variables fixed.

than that from the second, especially on the MOPTA08 problem, indicating that those frequently selected variables indeed have more significant effect on the function value.

6 Conclusion

We propose a new method, VS-BO, for high-dimensional BO that is based on the assumption that variables of the input can be divided to two categories: important and unimportant. Our method can assign variables into these two categories with no need for pre-specifying any crucial hyper-parameter and uses different strategies to decide values of the variables in different categories to reduce runtime. The good performance of our method on synthetic and real-world problems further verify the rationality of the assumption. We show the computational efficiency of our method both theoretically and empirically. Further, important variables found by VS-BO can help improve the interpretability of the model and increase our understanding of the black-box function.

References

[Auer, 2002] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[Bergstra *et al.*, 2013] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123, 2013.

[Berkenkamp *et al.*, 2016] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *arXiv preprint arXiv:1602.04450*, 2016.

[Brochu *et al.*, 2010] Eric Brochu, Vlad M Cora, and Nando 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.

[Calandra *et al.*, 2016] Roberto Calandra, André Seyfarth, Jan Peters, and Marc Peter Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2016.

[Derksen and Keselman, 1992] Shelley Derksen and Harvey J Keselman. Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables. *British Journal of Mathematical and Statistical Psychology*, 45(2):265–282, 1992.

[Djolonga *et al.*, 2013] Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional Gaussian process bandits. In *Advances in Neural Information Processing Systems*, pages 1025–1033, 2013.

[Eriksson and Jankowiak, 2021] David Eriksson and Martin Jankowiak. High-Dimensional Bayesian Optimization with Sparse Axis-Aligned Subspaces. *arXiv preprint arXiv:2103.00349*, 2021.

[Frazier, 2018] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

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

[Griffiths and Hernández-Lobato, 2017] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained Bayesian optimization for automatic chemical design. *arXiv preprint arXiv:1709.05501*, 2017.

[Gupta *et al.*, 2020] Sunil Gupta, Santu Rana, Svetha Venkatesh, et al. Trading Convergence Rate with Computational Budget in High Dimensional Bayesian Optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2425–2432, 2020.

[Hansen, 2016] Nikolaus Hansen. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[Hutter *et al.*, 2010] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Automated configuration of mixed

integer programming solvers. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming*, pages 186–202. Springer, 2010.

[Hutter *et al.*, 2014] Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *International Conference on Machine Learning*, pages 754–762. PMLR, 2014.

[Jones, 2008] Donald R Jones. Large-scale multi-disciplinary mass optimization in the auto industry. In *MOPTA 2008 Conference (20 August 2008)*, 2008.

[Kandasamy *et al.*, 2015] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pages 295–304, 2015.

[Kirschner *et al.*, 2019] Johannes Kirschner, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In *International Conference on Machine Learning*, pages 3429–3438. PMLR, 2019.

[Klein *et al.*, 2017] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*, pages 528–536. PMLR, 2017.

[Letham *et al.*, 2020] Ben Letham, Roberto Calandra, Akshara Rai, and Eytan Bakshy. Re-examining linear embeddings for high-dimensional Bayesian optimization. *Advances in Neural Information Processing Systems*, 33, 2020.

[Li *et al.*, 2016] Chun-Liang Li, Kirthevasan Kandasamy, Barnabás Póczos, and Jeff Schneider. High dimensional Bayesian optimization via restricted projection pursuit models. In *Artificial Intelligence and Statistics*, pages 884–892, 2016.

[Marco *et al.*, 2017] Alonso Marco, Felix Berkenkamp, Philipp Hennig, Angela P Schoellig, Andreas Krause, Stefan Schaal, and Sebastian Trimpe. Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with Bayesian optimization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1557–1563. IEEE, 2017.

[Metzen, 2016] Jan Hendrik Metzen. Minimum regret search for single- and multi-task optimization. *arXiv preprint arXiv:1602.01064*, 2016.

[Močkus, 1975] Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer, 1975.

[Moriconi *et al.*, 2019] Riccardo Moriconi, Marc P Deisenroth, and KS Kumar. High-dimensional Bayesian optimization using low-dimensional feature spaces. *arXiv preprint arXiv:1902.10675*, 2019.

[Nayebi *et al.*, 2019] Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian optimization in embedded subspaces. In *International Conference on Machine Learning*, pages 4752–4761. PMLR, 2019.

[Paananen *et al.*, 2019] Topi Paananen, Juho Piironen, Michael Riis Andersen, and Aki Vehtari. Variable selection for gaussian processes via sensitivity analysis of the posterior predictive distribution. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1743–1752, 2019.

[Rana *et al.*, 2017] Santu Rana, Cheng Li, Sunil Gupta, Vu Nguyen, and Svetha Venkatesh. High dimensional Bayesian optimization with elastic Gaussian process. In *International conference on machine learning*, pages 2883–2891. PMLR, 2017.

[Rolland *et al.*, 2018] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional Bayesian optimization via additive models with overlapping groups. *arXiv preprint arXiv:1802.07028*, 2018.

[Snoek *et al.*, 2012] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, 2012.

[Spagnol *et al.*, 2019] Adrien Spagnol, Rodolphe Le Riche, and Sébastien Da Veiga. Bayesian optimization in effective dimensions via kernel-based sensitivity indices. In *International Conference on Applications of Statistics and Probability in Civil Engineering*, 2019.

[Ulmasov *et al.*, 2016] Doniyor Ulmasov, Caroline Baroukh, Benoit Chachuat, Marc Peter Deisenroth, and Ruth Misener. Bayesian optimization with dimension scheduling: Application to biological systems. In *Computer Aided Chemical Engineering*, volume 38, pages 1051–1056. Elsevier, 2016.

[Wang *et al.*, 2016] 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.

[Wang *et al.*, 2017] Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional Bayesian optimization via structural kernel learning. In *International Conference on Machine Learning*, pages 3656–3664. PMLR, 2017.

[Williams and Rasmussen, 2006] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.