Trading Convergence Rate with Computational Budget in High Dimensional Bayesian Optimization

Hung Tran-The, Sunil Gupta, Santu Rana, Svetha Venkatesh
Applied Artificial Intelligence Institute,Deakin University, Geelong, Australia
{hung.tranthe, sunil.gupta, santu.rana, svetha.venkatesh}@deakin.edu.au

Abstract

Scaling Bayesian optimisation (BO) to high-dimensional search spaces is a active and open research problems particularly when no assumptions are made on function structure. The main reason is that at each iteration, BO requires to find global maximisation of acquisition function, which itself is a non-convex optimization problem in the original search space. With growing dimensions, the computational budget for this maximisation gets increasingly short leading to inaccurate solution of the maximisation. This inaccuracy adversely affects both the convergence and the efficiency of BO. We propose a novel approach where the acquisition function only requires maximisation on a discrete set of low dimensional subspaces embedded in the original high-dimensional search space. Our method is free of any low dimensional structure assumption on the function unlike many recent high-dimensional BO methods. Optimising acquisition function in low dimensional subspaces allows our method to obtain accurate solutions within limited computational budget. We show that in spite of this convenience, our algorithm remains convergent. In particular, cumulative regret of our algorithm only grows sub-linearly with the number of iterations. More importantly, as evident from our regret bounds, our algorithm provides a way to trade the convergence rate with the number of subspaces used in the optimisation. Finally, when the number of subspaces is "sufficiently large", our algorithm's cumulative regret is at most $\mathcal{O}^*(\sqrt{T\gamma_T})$ as opposed to $\mathcal{O}^*(\sqrt{DT\gamma_T})$ for the GP-UCB of Srinivas et al. (2012), reducing a crucial factor $\sqrt{D}$ where D being the dimensional number of input space. We perform empirical experiments to evaluate our method extensively, showing that its sample efficiency is better than the existing methods for many optimisation problems involving dimensions up to 5000.

Introduction

Bayesian optimization (BO) offers an efficient solution to find the global optimum of expensive black-box functions, a problem that is all pervasive in real-world experimental design applications. However, the scalability of BO is particularly compromised in high dimensions (> 15 dimensions).

The main difficulty that a BO algorithm faces in high dimensions is that at each iteration, it needs to find the global maximum of a surrogate function called the *acquisition function* in order to suggest the next function evaluation point. The acquisition function balances two conflicting requirements: evaluating the function at a location where the function may peak as indicated by knowledge collected so far (exploitation) and evaluating the function at a location to reduce our uncertaiity about the function (exploration). The acquisition function itself is a non-convex optimisation problem in the original search space. With growing dimensions, any fixed computational budget for this optimisation becomes quickly insufficient leading to inaccurate solutions. This inaccuracy adversely affects both the convergence and the efficiency of the BO algorithm.

In order to make the problem scalable, most of the current methods make restrictive structural assumptions such as the function having an effective low-dimensional subspace (Wang et al. 2013; Djolonga, Krause, and Cevher 2013; Garnett, Osborne, and Hennig 2014; Eriksson et al. 2018; Nayebi, Munteanu, and Poloczek 2019; Zhang, Li, and Su 2019), or being decomposable in subsets of dimensions (Kandasamy 2015; Li 2016; Rolland et al. 2018; Mutný and Krause 2018; Hoang et al. 2018). Through these assumptions, the acquisition function becomes easier to optimise and the global optimum can be found. However, such assumptions are rather strong and since we are dealing with unknown function, we have no way of knowing if such assumptions hold in reality.

Without these assumptions, the high-dimensional BO problem is more challenging. There have been some recent attempts to develop scalable BO methods. For example, (Rana et al. 2017) use an elastic Gaussian process model to reduce the "flatness" of the acquistion function in high dimensions and thus improve the solution. Although this method helps obtains improved optimum of the acquistion function, it still does not provide any way to scale BO to high dimensions. (Oh, Gavves, and Welling 2018) devise a method under the assumption that the solution does not lie at the boundary of the search space and thus drive the BO search towards the interior of the search space using polar co-ordinate transformations. Thus, the convergence analysis of (Oh, Gavves, and Welling 2018) depends on whether their assumptions holds. Some other methods are

based on subspaces (Qian, Hu, and Yu 2016; Li et al. 2017; Kirschner et al. 2019), which are more amenable to convergence analysis. Among them, LineBO (Kirschner et al. 2019) is the first to provide a complete analysis. In LineBO, the maximisation of the acquisition function is performed on a one-dimensional random subspace. Although this solution is computationally effective, the regret bound of LineBO does not scale well with the search space dimension and gets increasingly worse compared to the regret of the GP-UCB (Srinivas et al. 2012). *Thus we still need a method that is both computationally efficient and offers optimal sample efficiency in high dimensions in a flexible manner*.

We propose a novel algorithm for high-dimensional BO without making any restrictive assumptions on the function structure and show that our algorithm attains better regret than previous methods. The key insight of our method is that instead of maximizing the acquisition function on the full search space that is computationally prohibitive in high dimensions, we will maximize it on a restricted space that consist of multiple low-dimensional subspaces. This approach has several benefits. If the acquisition function is maximised in low-dimensional subspaces then computations involved in finding the solution of acquisition function optimisation are practically feasible while multiple subspaces can still cover the search space well if they are chosen in a principled manner. Further, this method allows us to theoretically derive a cumulative regret as a function of the number of subspaces. The crucial advantage is that we can *trade between the computations and the convergence rate* of the algorithm while still maintaining efficient convergence (i.e. a sublinear growth of cumulative regret). Our contributions in this paper are:

- A novel BO algorithm based on multiple random subspaces that offers the flexibility to trade between the computations and the convergence rate. To do this, we propose to decompose the original search space into multiple lower-dimensional spaces via a randomisation technique.
- We derive an upper bound on the cumulative regret for our algorithm and theoretically show that it has a sublinear growth rate and further, larger the number of subspaces, tighter the cumulative regret. Further, when the number of subspaces is large enough, the cumulative regret is at most of order $\mathcal{O}^*(\sqrt{T\gamma_T})$ as opposed to the regret bound of GP-UCB $\mathcal{O}^*(\sqrt{DT\gamma_T})$. The regret bound of our algorithm is tighter by a factor of $\sqrt{D}$. In some situations (detailed in the paper), this improvement is possible with fewer computations than GP-UCB - a double advantage.
- We also study a special case when the objective function has an effective low dimensional subspace as assumed by many previous methods e.g. REMBO (Wang et al. 2013). We show that our algorithm automatically benefits from this low dimensional structure with the tighter bound.
- We extensively evaluate our method using a variety of optimisation tasks including optimisation of several benchmark functions as well as learning the parameters of a moderately sized neural network and a classification model based on non-convex ramp loss. We compare our method with several existing high dimensional BO algorithms and demonstrate that given equal computational budget, the sample efficiency of our method is consistently better than that of the existing methods.

Preliminaries

We consider the global maximization problem of the form

$$\mathbf{x}^* = \text{argmax}_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{1}$$

in a compact search space $\mathcal{X} = [-1, 1]^D$. In this paper we are especially concerned about problems with high values of D. We consider functions f that are blackbox and expensive to evaluate, and our goal is to find the optimum in a minimal number of samples. We further assume we only have access to noisy evaluations of f in the form $u = f(\mathbf{x}) + \epsilon$ where the noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is i.i.d. Gaussian.

Bayesian Optimization

BO offers a principal framework to approach the global optimisation problem. The standard BO routine consists of two key steps: estimating the black-box function from function evaluation data and maximizing an acquisition function to suggest next function evaluation point balancing exploration and exploitation. Gaussian process (GP) (Rasmussen and Williams 2005) is a popular choice for the first step due to its tractability for posterior and predictive distributions: $f(\mathbf{x}) = \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$ are the mean and the covariance (or kernel) functions. Popular covariance functions include linear kernel, squared exponential (SE) kernel, Matern kernel etc. The predictive mean and variance of Gaussian process is a Gaussian distribution. Given a set of observations $\mathcal{D}_{1:t} = \{\mathbf{x}_i, u_i\}_{i=1}^t$, the predictive distribution can be derived as: $P(f_{t+1}|\mathcal{D}_{1:t}, \mathbf{x}) = \mathcal{N}(\mu_{t+1}(\mathbf{x}), \sigma^2_{t+1}(\mathbf{x}))$, where $\mu_{t+1}(\mathbf{x}) = \mathbf{k}^T[\mathbf{K}+\sigma^2\mathbf{I}]^{-1}\mathbf{u}+ m(\mathbf{x})$ and $\sigma^2_{t+1}(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}^T[\mathbf{K} + \sigma^2\mathbf{I}]^{-1}\mathbf{k}$. In the above expression we define $\mathbf{k} = [k(\mathbf{x}, \mathbf{x}_1), ..., k(\mathbf{x}, \mathbf{x}_t)]$, $\mathbf{K} = [k(\mathbf{x}_i, \mathbf{x}_j)]_{1\leq i,j\leq t}$ and $\mathbf{u} = [u_1, \ldots, u_t]$.

The acquisition functions are designed to trade off between exploration of the search space and exploitation of current promising region. Some examples of acquisition functions include Expected Improvement (EI) (Mockus 1974) and GP-UCB (Srinivas et al. 2012). A GP-UCB acquisition function at iteration $t + 1$ is defined as

$$a_{t+1}(\mathbf{x}) = \mu_t(\mathbf{x}) + \sqrt{\beta_{t+1}}\sigma_t(\mathbf{x}) \tag{2}$$

where β_{t+1} is a parameter to balance exploration and exploitation. There are guidelines (Srinivas et al. 2012) for setting β_{t+1} to achieve sublinear regret.

Proposed Method

To solve the Bayesian optimization problem in high dimensions, the main difficulty is the prohibitive computational burden when maximising the acquisition function which requires solving a non-convex optimization problem in the same search space. Working with a small computational budget usually directly affects the quality of the point suggested by the acquisition step and consequently, many BO

algorithms that require finding exactly the maximisers of the acquisition function (e.g. EI, GP-UCB) perform poorly in high dimensions. We propose a novel approach for this problem as follows. Instead of maximising the acquisition function on the whole space, we perform it only on a discrete set of low-dimensional subspaces generated by random sampling. This approach brings several benefits. First, it solves the computation challenge effectively. Second, it provides a way to trade between the computations and the sample efficiency for convergence. Finally, our analysis suggests that when the number of subspaces are sufficiently high, our approach can simultaneously offer both lower computational requirement and better sample efficiency.

Before introducing our method, we formally define the low dimensional subspaces (see Definition 1) that will be used in our method.

Subspace Description

Lemma 1. *Given a $d \in \{1, ..., D-1\}$, for any $x \in [-1,1]^D$, there exists a $y \in [-1,1]^d$ and a $z \in [-1,1]^{D-d}$ such that*

$$\mathbf{x} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z}$$

where matrix $\mathbf{A} = [\mathbf{0}^{d\times(D-d)}, \mathbf{I}^{d\times d}]^T$ and matrix $\mathbf{B} = [\mathbf{I}^{(D-d)\times(D-d)}, \mathbf{0}^{(D-d)\times d}]^T$.

Proof. Given any $\mathbf{x} \in \mathcal{X}$, we denote the first $D-d$ elements of vector $\mathbf{x}$ by $[\mathbf{x}]_{1:D-d}$ and the last d elements of vector $\mathbf{x}$ by $[\mathbf{x}]_{D-d+1:D}$. We set $\mathbf{y} = [\mathbf{x}]_{D-d+1:D} \in [-1,1]^d$ and $z = [\mathbf{x}]_{1:D-d} \in [-1,1]^{D-d}$. Thus, we have $\mathbf{x} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z}$ with $\mathbf{y} \in [-1,1]^d$ and $\mathbf{z} \in [-1,1]^{D-d}$. □

The main idea is that we split the dimensions of any point x in D dimensional space into two groups: the first $D-d$ dimensions correspond to z and the last d dimensions correspond to y. Next, we define a set of subspaces based on this idea in the following form:

Definition 1 (Embedding Subspace)**.** Given a $d \in \{1, ..., D-1\}$ and a vector $\mathbf{z} \in \mathcal{Z} = [-1,1]^{D-d}$. We define an embedding subspace $\mathcal{S}(\mathbf{A}, \mathbf{z})$ as

$$\mathcal{S}(\mathbf{A}, \mathbf{z}) = \{\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z} | \mathbf{y} \in \mathcal{Y} = [-1,1]^d\} \quad (3)$$

where matrix $\mathbf{A} = [\mathbf{0}^{d\times(D-d)}, \mathbf{I}^{d\times d}]^T$ and matrix $\mathbf{B} = [\mathbf{I}^{(D-d)\times(D-d)}, \mathbf{0}^{(D-d)\times d}]^T$.

The Proposed Algorithm

Our algorithm (presented in Algorithm 1) closely follows the standard BO algorithm. The main difference lies in the acquisition step that guides the selection of the next function evaluation point. We refer to our algorithm as **MS-UCB**.

Lemma 1 implies that there exists a $\mathbf{y}^* \in [-1,1]^d$ and $\mathbf{z}^* \in [-1,1]^{D-d}$ such that

$$\mathbf{x}^* = \mathbf{A}\mathbf{y}^* + \mathbf{B}\mathbf{z}^* \quad (4)$$

Therefore, we can preserve any optimum point $\mathbf{x}^*$ in the original space via $\mathbf{y}^*, \mathbf{z}^*$ in two lower-dimensional spaces (via dimension splitting). This observation gives rise to our new idea that instead of optimising f on the original space, we can perform it in the lower dimensional spaces $\mathcal{Y}$ and $\mathcal{Z}$:

$$\mathbf{y}^*, \mathbf{z}^* = \text{argmax}_{\mathbf{y}\in\mathcal{Y}, \mathbf{z}\in\mathcal{Z}} f(\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z}) \quad (5)$$

where we denote the range $[-1,1]^d$ by $\mathcal{Y}$ and the range $[-1,1]^{D-d}$ by $\mathcal{Z}$. Note that this is different from (Wang et al. 2013; Djolonga, Krause, and Cevher 2013) where they assume an effective subspace on f. In that case, using a low-dimensional subspace can solve the BO problem. In our problem without any restrictive assumption on f, we need to use two low-dimensional subspaces to preserve an optimum. As a result, when Eq (5) is established, maximising the acquisition function a_t will be performed in spaces $\mathcal{Y}$ and $\mathcal{Z}$ as

$$\mathbf{y}_t, \mathbf{z}_t = \text{argmax}_{\mathbf{y}\in\mathcal{Y}, \mathbf{z}\in\mathcal{Z}} a_t(\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z}) \quad (6)$$

In high dimensions where we may set $d << D$, the $(D-d)$-dimensional space $\mathcal{Z}$ would itself be a high-dimensional space. Therefore, maximising the acquisition function in this space is still computationally expensive. To deal with this problem, we work with a finite set of samples $\mathbf{z} \in \mathcal{Z}$ and maximise the acquisition function only on this finite set instead of the whole space $\mathcal{Z}$. Let Z_t be the set of all $\mathbf{z}$ generated up to iteration t. With this modification, our acquisition step becomes:

$$\mathbf{y}_t, \mathbf{z}_t = \text{argmax}_{\mathbf{y}\in\mathcal{Y}, \mathbf{z}\in\mathcal{Z}_t} a_t(\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z}) \quad (7)$$

Once $\mathbf{z}$ is sampled, a subspace $\mathcal{S}(\mathbf{A}, \mathbf{z})$ as in Definition 1 is generated. Let $\mathcal{X}_t \triangleq \{\mathcal{S}(\mathbf{A}, \mathbf{z}^i) \mid \mathbf{z}^i \in Z_t\}$. The Eq (7) can then be re-written as

$$\mathbf{x}_t = \text{argmax}_{\mathbf{x}\in\mathcal{X}_t} a_t(\mathbf{x}) \quad (8)$$

By Definition 1, we can see that for any subspace $\mathcal{S}(\mathbf{A}, \mathbf{z}^i)$, $\mathcal{S}(\mathbf{A}, \mathbf{z}^i) \subset \mathcal{X}$. Thus, the suggested point x_t is always within $\mathcal{X}$. This is a benefit of our dimension splitting based subspace projection as opposed to complex corrections required for previous methods ((Wang et al. 2013; Qian, Hu, and Yu 2016)). We note that for our method $a_t(\mathbf{x}) = \mu_{t-1}(\mathbf{x}) + \sqrt{\beta_t}\sigma_{t-1}(\mathbf{x})$ where $\beta_t = 2log(\frac{\pi^2 t^2}{\delta}) + 2dlog(2bd\sqrt{log(\frac{6Da}{\delta})t^2})$. The β_t of GP-UCB depends linearly on D whereas the dependence of our β_t on D is only $\sqrt{\log D}$.

A simple way to maximise a_t on $\mathcal{X}_t$ is to maximise a_t on each subspace $\mathcal{S}(\mathbf{A}, \mathbf{z}_t^i)$ separately and thus the returned value $\mathbf{x}_t$ is the maximizer on the all subspaces at iteration t. For a small d, $\mathcal{S}(\mathbf{A}, \mathbf{z}^i)$ is a low-dimensional subspace of $\mathcal{X}$. Thus, maximizing the acquisition function on such a subspace is computationally cheaper.

Importantly, we can show that maximising a_t on a discrete set of subspaces can result in low regret by proposing a strategy to choose set Z_t. At iteration t, we uniformly randomly draw N_t samples of z, $\{\mathbf{z}_t^i \in \mathcal{Z} \mid i = 1, ..., N_t\}$ and then construct Z_t as $Z_t = Z_{t-1} \cup \{\mathbf{z}_t^1, ..., \mathbf{z}_t^{N_t}\}$. We choose $N_t = N_0 t^\alpha$, where $N_0, \alpha \in \mathbb{N}$, $N_0 \geq 1$ and $\alpha \geq 0$.

Algorithm 1 MS-UCB Algorithm

Input: Input space $\mathcal{X} = [-1, 1]^D$; a low dimension $1 \leq d < D$, $Z_0 = \emptyset$, the parameters N_0 and α.

1: Sample initial points to construct $\mathcal{D}_0$.
2: Build a Gaussian process using $\mathcal{D}_0$.
3: **for** $t = 1, 2, ..., T$ **do**
4: Sample uniformly at random N_t values of $\mathbf{z}_t^i \in [-1, 1]^{D-d}$, where $1 \leq i \leq N_t = N_0 t^\alpha$.
5: Update $Z_t = Z_{t-1} \cup \{\mathbf{z}_t^1, ..., \mathbf{z}_t^{N_t}\}$.
6: Maximise acquisition function to obtain $\mathbf{x}_t$ by following Eq (8).
7: Sample $u_t = f(\mathbf{x}_t) + \epsilon_t$.
8: Augment the data $\mathcal{D}_t = \{\mathcal{D}_{t-1}, (\mathbf{x}_t, u_t)\}$.
9: Update the Gaussian process using $\mathcal{D}_t$.
10: **end for**

Convergence Analysis

In this section, we analyse the convergence of our proposed MS-UCB algorithm. For this, we use the regret, which tells us how much better we could have done in iteration t had we known $\mathbf{x}^*$, formally $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$. In many applications, such as recommender systems, robotic control, etc, we care about the quality of the points chosen at every iteration t, and thus a natural quantity to consider is the cumulative regret that is defined as $R_T = \sum_{1 \leq t \leq T} r_t$, the sum of regrets incurred over a horizon of T iterations. If we can show that the cumulative regret is sublinear for a given algorithm, then $lim_{T \to \infty} R_T/T = 0$, meaning the algorithm efficiently converges to the optimum.

To derive a cumulative regret $R_t = \sum_{t=1}^{T} r_t$, we will seek to bound $r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t)$ for any t. At each iteration t, we denote by $\mathcal{S}_t^0 \triangleq \{\mathbf{A}\mathbf{y}^* + \mathbf{B}\mathbf{z}_i\}_{\mathbf{z}_i \in Z_t}$. Let $\mathbf{z}_t^* \triangleq \text{argmin}_{\mathbf{z} \in \mathcal{Z}_t} ||\mathbf{z} - \mathbf{z}^*||_1$ and $f_{\mathcal{S}_t^0}^{max} \triangleq f(\mathbf{A}\mathbf{y}^* + \mathbf{B}\mathbf{z}_t^*)$. To obtain a bound on r_t, we write it as

$$r_t = f(\mathbf{x}^*) - f(\mathbf{x}_t) \quad (9)$$

$$= \underbrace{f(\mathbf{x}^*) - f_{\mathcal{S}_t^0}^{max}}_{\text{Term 1}} + \underbrace{f_{\mathcal{S}_t^0}^{max}}_{\text{Term 2}} - \underbrace{f(\mathbf{x}_t)}_{\text{Term 3}} \quad (10)$$

Term 1 will be bounded from the result of Lemma 4, Term 2 will be bounded from the result of Lemma 3 and Term 3 will bounded from the result of Lemma 2. Before proceeding to the proofs, we need to make the following assumption.

Assumption 1 (Gradients of GP Sample Paths (Ghosal and Roy 2006)). *Let $f \sim \mathcal{GP}(\mathbf{0}, k)$ and k is a stationary kernel. The partial derivatives of f satisfies the following condition. There exist constants $a, b > 0$ such that*

$$\mathbb{P}[sup_{\mathbf{x} \in \mathcal{X}} |\frac{\partial f}{\partial x_i}| > L] \leq ae^{-(L/b)^2}$$

for all $L > 0$ and for all $i \in \{1, 2, ..., D\}$

Similar to Lemma 5.5 of (Srinivas et al. 2012), we have the following bound on the actual function observations.

Lemma 2 (Bounding Term 3). *Pick a $\delta \in (0, 1)$ and set $\beta_t^0 = 2log(\pi^2 t^2/(6\delta))$. Then we have*

$$f(\mathbf{x}_t) \geq \mu_{t-1}(\mathbf{x}_t) - \sqrt{\beta_t^0}\sigma_{t-1}(\mathbf{x}_t) \quad (11)$$

holds with probability $\geq 1 - \delta$.

Lemma 3 (Bounding Term 2). *Pick a $\delta \in (0, 1)$ and set $\beta_t^1 = 2log(\frac{\pi^2 t^2}{3\delta}) + 2dlog(2bd\sqrt{log(\frac{2Da}{\delta})}t^2)$. Then, under Assumption 1 there exists a $\mathbf{x}' \in \mathcal{S}(\mathbf{A}, \mathbf{z}_t^*)$ such that*

$$f_{\mathcal{S}_t^0}^{max} \leq \mu_{t-1}(\mathbf{x}') + \sqrt{\beta_t^1}\sigma_{t-1}(\mathbf{x}') + \frac{1}{t^2} \quad (12)$$

holds with probability $\geq 1 - \delta$.

Lemma 4 (Bounding Term 1). *Pick a $\delta \in (0, 1)$ and set $v_0 = 2b\sqrt{log(\frac{2Da}{\delta})}(\Gamma(D - d + 1))^{\frac{1}{D-d}}$, where $\Gamma(D - d + 1) = (D - d)!$. With probability at least $1 - \delta$, we have*

$$f(\mathbf{x}^*) - f_{\mathcal{S}_t^0}^{max} \leq v_0(\frac{1}{|Z_t|}log(\frac{2}{\delta}))^{\frac{1}{D-d}}. \quad (13)$$

All proofs are provided in details in Supplementary Material. Different from confidence bound techniques that are used to bound Term 2 and Term 3, we use the randomisation techniques on sampled set Z_t and a result from (Wang 2005) to bound Term 1. Combining the results from Lemmas 2, 3 and 4 we obtain a bound on r_t and then sum it over iteration 1 to T to obtain the following theorem providing an upper bound on the cumulative regret R_T. The notation $\mathcal{O}^*$ is a variant of $\mathcal{O}$, where log factors are suppressed.

Theorem 1. *Pick a $\delta \in (0, 1)$. Then, the cumulative regret of the proposed MS-UCB algorithm is bounded as*

- $R_T \leq \mathcal{O}^*(\sqrt{dT\gamma_T} + D)$ *if* $\alpha \geq D - d - 1$.
- $R_T \leq \mathcal{O}^*(\sqrt{dT\gamma_T} + \frac{D^2 - Dd}{D - d - \alpha - 1} T^{1 - \frac{\alpha+1}{D-d}})$ *if* $0 \leq \alpha < D - d - 1$.

with probability greater than $1-\delta$, where γ_T is the maximum information gain about the function f from any set of observations of size T. α is related to the number of subspaces chosen as $N_t = N_0 t^\alpha$.

Next, we show that if the objective function has a low dimensional effective subspace meaning when there are directions in which the function is constant, our algorithm can benefit automatically from this structure. The following Theorem provides a regret bound for this scenario.

Theorem 2 (For functions having an effective subspace). *Pick a $\delta \in (0, 1)$. If there exists a linear subspace $\mathcal{T} \subset \mathbb{R}^D$ with d_e dimensions such that $\forall \mathbf{x} \in \mathbb{R}^D$, $f(\mathbf{x}) = f(\mathbf{x}_\top)$ where $\mathbf{x}_\top \in \mathcal{T}$ is the orthogonal projection of x onto $\mathcal{T}$, then*

- $R_T \leq \mathcal{O}^*(\sqrt{dT\gamma_T} + d_e)$ *if* $\alpha \geq d_e - 1$.
- $R_T \leq \mathcal{O}^*(\sqrt{dT\gamma_T} + \frac{d_e^2}{d_e - \alpha - 1} T^{1 - (\alpha+1)/d_e})$ *if* $0 \leq \alpha < d_e - 1$.

with probability greater than $1 - \delta$, where γ_T is the maximum information gain about the function f from any set of observations of size T.

Discussion

There are two important parameters in our method, (1) the dimension d that is used for acquisition function maximisation and (2) parameter α that is related to the number of subspaces on which the acquisition function is maximised. In this section, we provide a discussion on these parameters. We also analyse the computational cost involved in maximising the acquisition function comparing it with that of GP-UCB.

On Dimension d. The proposed MS-UCB algorithm is applied for $1 \leq d < D$, We consider two extreme cases where $d = 0$ or $d = D$.

- For $d = 0$, our Definition 1 is still valid however, in this case, the space $\mathcal{Y}$ is degenerated to 0. It follows that $\mathcal{S}(A, \mathbf{z}) = \{\mathbf{z}\}$ that means that the subspace becomes a unique point. The acquisition step becomes finding the maximiser of the acquisition function of a set of sampled points. We obtain the same result as in Theorem 1 for $d = 0$ with a slight modification of $\beta_t = 4log(\pi^2 t^2/2\delta)$ for Algorithm 1.
- For $d = D$, the space $\mathcal{Z}$ is degenerated into 0 and space $\mathcal{Y}$ becomes space $\mathcal{X}$. In this case, the idea of using sampling on space $\mathcal{Z}$ is not being utilised and the algorithm reduces to the standard GP-UCB algorithm working directly on the original high-dimensional space $\mathcal{X}$.

On the Number of Subspaces. In Theorem 1, the parameter N_0 does not affect the convergence rate of R_T. We thus only discuss the different cases of α. When $\alpha = 0$, Theorem 1 says that $R_T \leq \mathcal{O}^*(\sqrt{dT\gamma_T} + \frac{D^2-Dd}{D-d-1}T^{1-1/(D-d)})$. Although the regret growth now has an additional term $T^{1-1/(D-d)}$, but even in this case our algorithm has a sublinear cumulative regret. As expected, larger the value of α, tighter the cumulative regret until $\alpha \geq D - d - 1$, after which the cumulative regret no longer depends on α as the term $T^{1-(\alpha+1)/(D-d)}$ gets dominated by $\sqrt{dT\gamma_T}$. In this case, $R_T \leq \mathcal{O}^*(\sqrt{dT\gamma_T})$ for a large enough T. Comparison with regret bound of GP-UCB $\mathcal{O}^*(\sqrt{DT\gamma_T})$, our algorithm's bound is tighter by the factor $\sqrt{D}$. *To our best knowledge, ours is the first algorithm that obtains a cumulative regret bound better than GP-UCB's for any D being the dimensional number of input space under the Assumption 1.*

On the Computation Complexity. As mentioned in (Kandasamy 2015), in any grid or branch and bound methods, maximising a function to within ζ accuracy, requires $\mathcal{O}(\zeta^{-D})$ calls to a_t. Since we need to solve $|Z_t|$ d dimensional optimization problems for acquisition functions, it requires only $\mathcal{O}(|Z_t|\zeta^{-d})$ calls.

In our algorithm, we consider $N_t = N_0 t^\alpha$, and thus $|Z_t| = \sum_{j=1}^{t} N_0 j^\alpha$, which can be bounded by $\frac{(t+1)^{\alpha+1}-1}{\alpha+1}$ using the results from (Chlebus 2009). The largest computation is at iteration T where $|Z_T| = \sum_{j=1}^{T} N_0 j^\alpha < N_0(T+1)^{\alpha+1}/(\alpha+1)$. To have reduced computations in maximising the acquisition function, we should set α so that $(N_0(T+1)^{\alpha+1}/(\alpha+1))\zeta^{-d} < \zeta^{-D}$. If we choose $\zeta = \frac{1}{(T+1)^2}$ and $N_0 = 1$ then by choosing $\alpha < 2(D-d)-1$, the condition is satisfied. Thus, combining with Theorem 1, we can say, if there exists

$$D - d - 1 \leq \alpha < 2(D-d) - 1$$

then our algorithm can obtain both a cumulative regret $\mathcal{O}^*(\sqrt{dT\gamma_T})$ that is tighter than GP-UCB's and a computational cost that is cheaper than GP-UCB's when maximising the acquisition function with $\frac{1}{(T+1)^2}$- accuracy.

Experiments

To evaluate the performance of our MS-UCB, we have conducted a set of experiments involving optimization of four benchmark functions and two real applications. We compare our approach against six baselines: (1) Standard GP-UCB (Srinivas et al. 2012), (2) DropoutUCB (Li et al. 2017), (3) LineBO (Kirschner et al. 2019) which restricts the search space to a one-dimensional subspace, (4) SRE (Qian, Hu, and Yu 2016) which uses sequential random embeddings several times sequentially, (5) REMBO (Wang et al. 2013), and (6) HeSBO (Nayebi, Munteanu, and Poloczek 2019) which use hashing-enhanced embedded subspaces. Among these baselines, the first three baselines do not make assumptions on the structure of the objective function, SRE assumes a tiny effect for some of the dimensions i.e. ϵ-bounded while REMBO and HeSBO assume a low effective dimensional structure of the function.

For all experiments, we scale the search space of objective functions to convert into $[-1, 1]^D$. We implemented our proposed MS-UCB, LineBO, DropoutUCB and SRE in Python 3 using GPy. For all other algorithms we used the authors' reference implementations. For Gaussian process, we used Matern kernel and estimated the kernel hyper-parameters automatically from data. Each algorithm was randomly initialized with 20 points. To maximise the acquisition function, we used LBFGS-B algorithm with $10 \times D$ random starts.

Optimization of Benchmark Functions

In this section, we test the algorithms on several optimization benchmark functions: Ackley, Levy, Hyper-Elippsoid and Camelback functions. For the first three functions we assume full dimensionality while for the two-dimensional Camelback function, we simulate a scenario so that the function has a low dimensional effective subspace. For this, we augment the Camelback function with auxiliary dimensions. The Ackley function is widely used for testing optimization algorithms. it is characterized by a nearly flat outer region while the Hyper-Elippsoid function is used to demonstrate that our algorithms can effectively work for functions with interacting variables. We evaluate the progress of each algorithm using the log distance to the true optimum, that is, $\log_{10}(f(x^*) - f(x_t))$ where $f(x_t)$ is the function value sampled at iteration t. For each test function, we repeat the experiments 30 times. We plot the mean and a confidence bound of one standard deviation across all the runs.

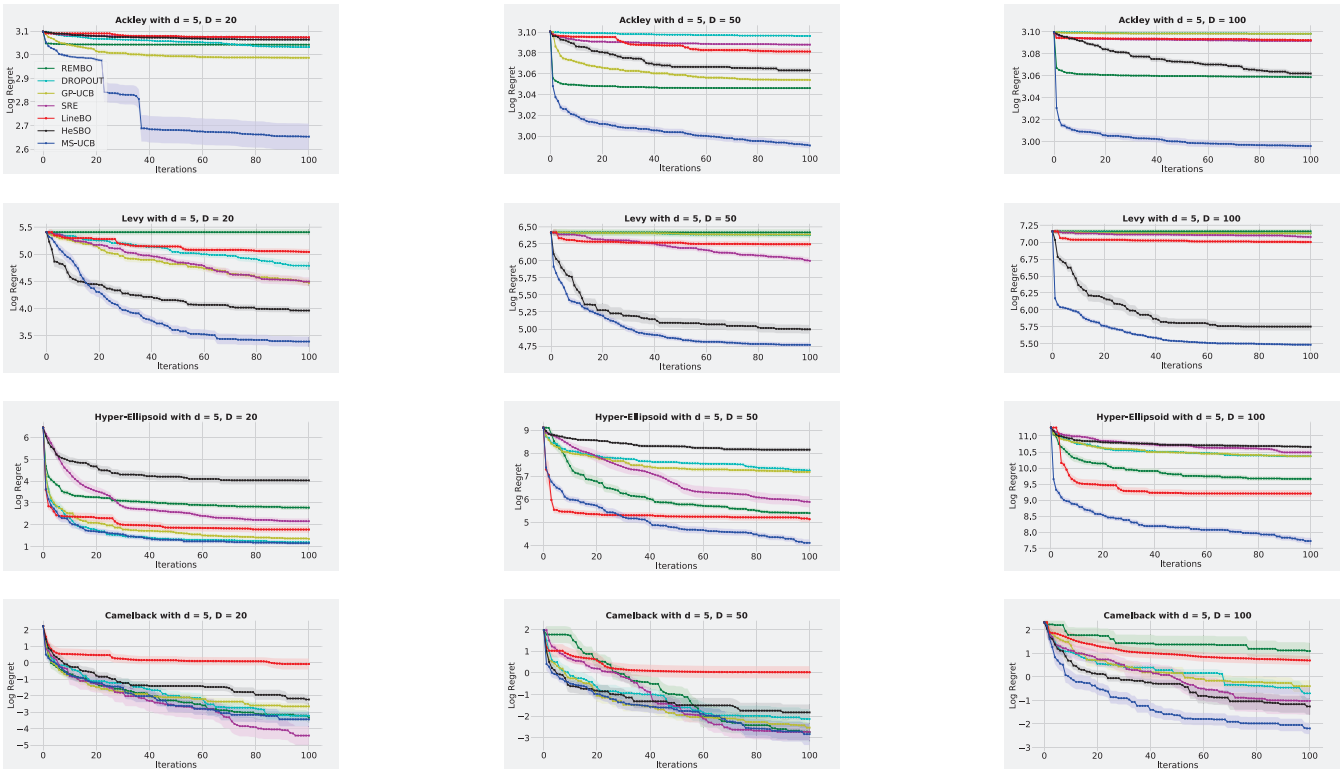
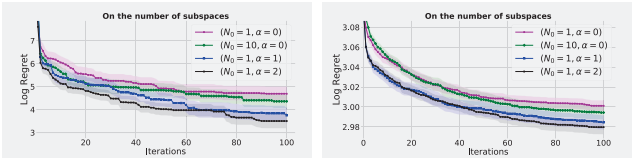


Figure 1: Comparison of baselines and the proposed MS-UCB method on four standard functions for 20, 50 and 100 input dimensions. For all cases, we set $d = 5$ except LineBO and GP-UCB. The y-axis presents log distance to the true optimum (Smaller value is better).

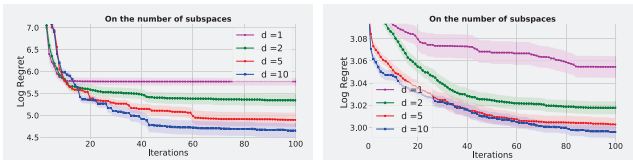
On Scalability: We perform experiments to empirically assess the scalability of our proposed MS-USB method and the baselines and report the results in Figure 1. We choose $d = 5$ for all methods except LineBO for which $d = 1$ is a requirement and the GP-UCB which works directly in original D-dimensional space. We use $N_0 = 1, \alpha = 0$ as parameters for our method. Recall that even with $\alpha = 0$, our method only has a sublinear cumulative regret growth. We study the cases with different input dimensions: $(D = 20, 50, 100)$. We can see that GP-UCB performs poorly in most cases on all test functions. The poor performance of GP-UCB is partly due to inaccurate solution of acquisition function optimisation in high dimensions. On the other hand, our method does better than all the baselines scaling well with the dimensions. Our method overcomes the difficulty by optimizing the acquisition function only on a set of d-dimensional subspaces, and thus with a limited computation budget, the acquisition function optimisation is performed more accurately. Dropout and SRE that do not provide a vanishing regret scale poorly with high dimensions. LineBO performs poorly except for the Hyper-Ellipsoid function. For the Camelback function with just two effective dimensions, our method is still competitive to other methods that are designed to exploit such structure. Our method achieves a slightly better accuracy than SRE when $D = 50$ and the best one when $D = 100$. Thus our proposed MS-UCB outperforms baselines and scales well in high dimensions.



(a) Hyper-Ellipsoid function (b) Ackley function

Figure 2: Log Regret vs iterations for varying number of subspaces (N_0 and α).

On the number of subspaces: The number of subspaces per iteration is a control parameter in our method. It is set via N_0 and in particular α. In Theorem 1, we show that N_0 does not affect much the regret. We study the effect of these parameters creating four MS-UCB variants: $(N_0 = 1, \alpha = 0)$, $(N_0 = 10, \alpha = 0)$, $(N_0 = 1, \alpha = 1)$ and finally $(N_0 = 1, \alpha = 2)$. We fix $D = 100$ and $d = 5$. Figure 2 shows a comparison of these variants for Ackley function and Hyper-Ellipsoid function. It indicates if we increase N_0 or α (extending the search space), the regret decreases. For the case where $\alpha = 2$, we need to generate more



(a) Hyper-Ellipsoid function (b) Ackley function

Figure 3: Log Regret vs iterations for varying dimension d.

subspaces at each iteration (at a quadratic rate: t^2). This demands more computational budget to maximise the acquisition function. It may be the reason why compared to case $(N_0 = 1, \alpha = 0)$, the regret of case $(N_0 = 1, \alpha = 2)$ improved only slightly.

On subspace dimension d: Dimension d represents the dimension of the subspace. It directly affects the computational requirement of acquisition function maximization, and thus the BO optimization performance. As this computational requirement grows exponentially with increasing d, it is often computationally hard to find the global maximum in more than 10 dimensions. Thus we only study the value of d up to 10, *i.e.* we study cases: $d = 1, 2, 5, 10$ for problems in $D = 100$. We set $N_0 = 1, \alpha = 0$ in our method. Figure 3 shows the performance of our method for different cases of d for 100 dimensional Ackley function and Hyper-Ellipsoid function. It clearly indicates there is a faster convergence rate for a larger d, though at a higher computational cost.

Learning Parameters of Machine Learning Models

Neural Network Parameter Search: We evaluate our algorithm for learning the parameters of a neural network model as proposed by (Nayebi, Munteanu, and Poloczek 2019). Here we are given a neural network with one hidden layer having h nodes. The goal is to learn the weights between the hidden layer and the outputs in order to minimize the loss on the MNIST data set ((LeCun and Cortes 2010)). We denote these weights by W_2. For all experiments, W_2 is optimized by Bayesian optimization while the other weights and biases (we denote by W_1) are optimized by the Adam algorithm. We refer to (Oh, Gavves, and Welling 2018) for more details. We try two cases: $h = 10$ and $h = 50$. Since the network has 10 outputs (for 10 digits of MNIST), the two cases lead to optimisation in dimensions $10 \times 10 = 100$ and $50 \times 10 = 500$ respectively. Figure 5 shows the validation loss for REMBO, LineBO, SRE, HeSBO and our proposed MS-UCB. As seen from the figure, MS-UCB clearly outperforms the baselines for both cases.

Learning Classification Model with Ramp Loss: We also test our algorithm to optimize the parameters of a classification model with a nonconvex Ramp loss, following (Qian, Hu, and Yu 2016). The task is to find a vector w and a scalar b to minimize $f(w, b) = \frac{1}{2}||w||_2^2 + C\sum_{l=1}^{L} R_s(y_l(w_T v_l + b))$, where v_l are the training instances and $y_l \in \{-1, +1\}$ are the corresponding labels.

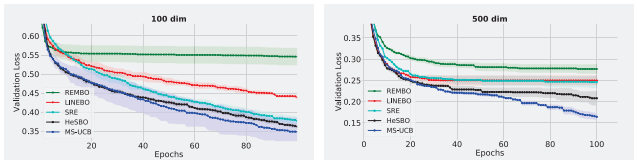


Figure 4: The neural network benchmark with target dimension $d = 10, N_0 = 1, \alpha = 1$ for two case where $W_2 = 10 \times 10$ and $W_2 = 50 \times 10$. For all experiments, W_2 is optimized by Bayesian optimization while other weights and biases are optimized by Adam algorithm.

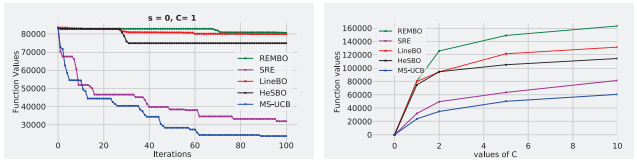


Figure 5: Left panel shows the progress of function values vs time for $s = 0$ and $C = 1$. Right panel shows the comparison of achieved loss function values against the hyper-parameter C of the Ramp loss.

$Rs(u) = H_1(u) - H_s(u)$ with $s < 1$ where $H_s(u) = \max(0, s - u)$. We use the *Gisette* dataset from the UCI repository (Newman and Merz 1998) with dimension $D = 5000$. We study the effectiveness of all algorithms fixing the hyper-parameters to $s = 0$ and $C \in \{1, 2, 5, 10\}$. We set $d = 10$ for REMBO, SRE, HeSBO and our method. For SRE, we set the number of sequential random embeddings $m = 5$ as suggested in (Qian, Hu, and Yu 2016). For our method, we consider the variant MS-UCB-1,1). As shown in Figure 5, our method has consistently the best performance across different settings of C, followed by SRE. REMBO, LineBO, HESBO perform poorly for this application. This shows the effectiveness of our method.

Conclusion

We propose a scalable Bayesian optimisation to optimise expensive blackbox functions in high dimensions. Unlike many previous existing methods, our algorithm does not make any additional assumption about the structure of the function (e.g. low effective dimension and additivity). In our method the acquisition function only requires maximisation on a discrete set of low dimensional subspaces embedded in the original high-dimensional search space and thus does not have high computational requirements for maximising acquisition functions. By varying the number of low dimensional subspaces, our algorithm has a flexibility to trade the optimisation convergence rate with the computational budget. This feature is important for many practical applications. We analyse our algorithm theoretically and show that irrespective of the number of subspaces, our algorithm always has a sublinear growth rate for cumulative regret. Further, we provide a regime for the number of subspaces where

our algorithm has both tighter regret bound as well as lower computational requirement compared to the GP-UCB algorithm of (Srinivas et al. 2012). We perform experiments for many optimisation problems in high dimensions and show that the sample efficiency of our algorithm is better than the existing methods given the same computational budget for optimising acquisition function.

Acknowledgments

This research was partially funded by the Australian Government through the Australian Research Council (ARC). Prof Venkatesh is the recipient of an ARC Australian Laureate Fellowship (FL170100006).

References

Chlebus, E. 2009. An approximate formula for a partial sum of the divergent p-series. *Appl. Math. Lett.* 22:732–737.

Djolonga, J.; Krause, A.; and Cevher, V. 2013. High-dimensional gaussian process bandits. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, 1025–1033. USA: Curran Associates Inc.

Eriksson, D.; Dong, K.; Lee, E. H.; Bindel, D.; and Wilson, A. G. 2018. Scaling gaussian process regression with derivatives. In *Advances in Neural Information Processing Systems*, 6868–6878.

Garnett, R.; Osborne, M. A.; and Hennig, P. 2014. Active learning of linear embeddings for gaussian processes. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, 230–239. Arlington, Virginia, United States: AUAI Press.

Ghosal, S., and Roy, A. 2006. Posterior consistency of gaussian process prior for nonparametric binary regression. *Ann. Statist.* 34(5):2413–2429.

Hoang, T. N.; Hoang, Q. M.; Ouyang, R.; and Low, K. H. 2018. Decentralized high-dimensional bayesian optimization with factor graphs. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 3231–3238.

Kandasamy. 2015. High dimensional bayesian optimisation and bandits via additive models. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, 295–304. JMLR.org.

Kirschner, J.; Mutny, M.; Hiller, N.; Ischebeck, R.; and Krause, A. 2019. Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces. In Chaudhuri, K., and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 3429–3438. Long Beach, California, USA: PMLR.

LeCun, Y., and Cortes, C. 2010. MNIST handwritten digit database.

Li, C.; Gupta, S.; Rana, S.; Nguyen, V.; Venkatesh, S.; and Shilton, A. 2017. High dimensional bayesian optimization using dropout. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 2096–2102. AAAI Press.

Li, C.-L. 2016. High dimensional bayesian optimization via restricted projection pursuit models. In Gretton, A., and Robert, C. C., eds., *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, 884–892. Cadiz, Spain: PMLR.

Mockus, J. 1974. On bayesian methods for seeking the extremum. In *Proceedings of the IFIP Technical Conference*, 400–404. London, UK, UK: Springer-Verlag.

Mutný, M., and Krause, A. 2018. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, 9019–9030. USA: Curran Associates Inc.

Nayebi, A.; Munteanu, A.; and Poloczek, M. 2019. A framework for Bayesian optimization in embedded subspaces. In Chaudhuri, K., and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 4752–4761. Long Beach, California, USA: PMLR.

Newman, C. B. D., and Merz, C. 1998. UCI repository of machine learning databases.

Oh, C.; Gavves, E.; and Welling, M. 2018. BOCK : Bayesian optimization with cylindrical kernels. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, 3865–3874. PMLR.

Qian, H.; Hu, Y.-Q.; and Yu, Y. 2016. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 1946–1952. AAAI Press.

Rana, S.; Li, C.; Gupta, S.; Nguyen, V.; and Venkatesh, S. 2017. High dimensional bayesian optimization with elastic gaussian process. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, 2883–2891. JMLR.org.

Rasmussen, C. E., and Williams, C. K. I. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

Rolland, P.; Scarlett, J.; Bogunovic, I.; and Cevher, V. 2018. High-dimensional bayesian optimization via additive models with overlapping groups. In Storkey, A., and Perez-Cruz, F., eds., *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84 of *Proceedings of Machine Learning Research*, 298–307. Playa Blanca, Lanzarote, Canary Islands: PMLR.

Srinivas, N.; Krause, A.; Kakade, S. M.; and Seeger, M. W. 2012. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Trans. Inf. Theor.* 58(5):3250–3265.

Wang, Z.; Zoghi, M.; Hutter, F.; Matheson, D.; and De Freitas, N. 2013. Bayesian optimization in high dimensions via random embeddings. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, 1778–1784. AAAI Press.

Wang, X. 2005. Volumes of generalized unit balls. *Mathematics Magazine* 78(5):390 – 395.

Zhang, M.; Li, H.; and Su, S. W. 2019. High dimensional bayesian optimization via supervised dimension reduction. *CoRR* abs/1907.08953.