Scaling Simultaneous Optimistic Optimization for High-Dimensional Non-Convex Functions with Low Effective Dimensions*

Hong Qian and **Yang Yu**
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210023, China
{qianh,yuy}@lamda.nju.edu.cn

Abstract

Simultaneous optimistic optimization (SOO) is a recently proposed global optimization method with a strong theoretical foundation. Previous studies have shown that SOO has a good performance in low-dimensional optimization problems, however, its performance is unsatisfactory when the dimensionality is high. This paper adapts random embedding to scaling SOO, resulting in the RESOO algorithm. We prove that the simple regret of RESOO depends only on the effective dimension of the problem, while that of SOO depends on the dimension of the solution space. Empirically, on some high-dimensional non-convex testing functions as well as hyper-parameter tuning tasks for multi-class support vector machines, RESOO shows significantly improved performance from SOO.

Introduction

Problem. Solving sophisticated optimizations is in the central problems of artificial intelligence. Let $f\colon \mathcal{X} \to \mathbb{R}$ be a function defined on a bounded region $\mathcal{X} \subseteq \mathbb{R}^D$, of which we assume that a global maximizer $\boldsymbol{x}^* \in \mathcal{X}$ always exists. An optimization problem can be formally written as

$$\boldsymbol{x}^* = \operatorname{argmax}_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}).$$

Without loss of generality, we assume $\mathcal{X} = [-1, 1]^D$ in this paper. We treat f as a *black-box* function that can only be evaluated point-wisely, i.e., we can only access $f(\boldsymbol{x})$ for any given solution $\boldsymbol{x} \in \mathcal{X}$. We assume that f is *deterministic*, i.e., every call of $f(\boldsymbol{x})$ returns the same value for the same $\boldsymbol{x}$. The performance of an optimization algorithm is evaluated by the *simple regret* (Bubeck, Munos, and Stoltz 2009), i.e., given n function evaluations, for maximization,

$$r_n = \max_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}) - f(\boldsymbol{x}(n)),$$

where $\boldsymbol{x}(n) \in \mathcal{X}$ is the solution with the highest function value found by the algorithm when the budget of n function evaluations is used up. The simple regret measures the difference between the true maximum of f and the best found by the algorithm. An algorithm is *no-regret* if it possesses the desirable asymptotic property $\lim_{n\to\infty} r_n = 0$.

*This research was supported by the NSFC (61375061, 61422304, 61223003), Foundation for the Author of National Excellent Doctoral Dissertation of China (201451).

Related Work. Methods with different principles have been proposed to address the black-box global optimization problems. Most of them can be roughly categorized into three kinds: meta-heuristic search, deterministic Lipschitz optimization methods, and Bayesian optimization methods. Meta-heuristic search algorithms are designed with inspired heuristics, such as evolutionary strategies (Hansen, Müller, and Koumoutsakos 2003) , which, however, are very weak in their theoretical foundations. Deterministic Lipschitz optimization methods require Lipschitz continuity assumption on f, either globally (Pintér 1996; Kearfott 1996; Strongin and Sergeyev 2000) or locally (Kleinberg, Slivkins, and Upfal 2008; Bubeck et al. 2011; Jones, Perttunen, and Stuckman 1993; Bubeck, Stoltz, and Yu 2011; Slivkins 2011; Munos 2011), which can have sound theoretical foundations. In addition, when a function evaluation is very expensive, Bayesian optimization methods (Brochu, Cora, and de Freitas 2010; Snoek, Larochelle, and Adams 2012) are particularly suitable, which are often theoretically supported under the assumption of Gaussian process priors.

We are interested in the deterministic Lipschitz optimization methods, and particularly the recently proposed *Simultaneous Optimistic Optimization* (SOO) algorithm (Munos 2011; 2014), since the *Local Lipschitz* continuity that SOO requires is intuitive, easy to be satisfied, and relatively easy to be verified. SOO incorporates an optimistic estimation of the function value with the branch-and-bound principle. It is worth mentioning that, although SOO assumes the *Local Lipschitz* continuity with respect to some semi-metric ℓ, fortunately ℓ does not need to be known. Because of these advantages, SOO has attracted attentions, such as the hybrid with Bayesian optimization to eliminate the optimization of acquisition functions (Wang et al. 2014). Also, variants of SOO have been proposed for, e.g., stochastic optimization (Valko, Carpentier, and Munos 2013) and parallel optimistic optimization (Grill, Valko, and Munos 2015).

However, previous studies have noticed that SOO may perform poorly in high-dimensional optimization problems (Preux, Munos, and Valko 2014; Derbel and Preux 2015). Meanwhile, it has been observed that in a wide range of high-dimensional optimization problems, such as hyper-parameter optimization in neural networks (Bergstra and Bengio 2012), intrinsically only a few low dimensions are effective. For these optimization problems with low effective

dimensions, random embedding and projection techniques are effective tools (Wang et al. 2013; Kaban, Bootkrajang, and Durrant 2013).

Our Contributions. This paper proposes RESOO that scales SOO to high-dimensional optimization problems via random embedding. RESOO performs the optimization by SOO in a low-dimensional solution space, where the function values of solutions are evaluated through the embedding into the original high-dimensional space. To perform the theoretical analysis of RESOO, we notice that the random embedding can probably preserve the local distance. By injecting this property into the Local Lipschitz continuity condition, we prove that the simple regret of RESOO depends only on the effective dimension of the problem. While the simple regret of SOO depends on the dimension of the solution space, RESOO is provably faster than SOO on the high-dimensional problems having low effective dimensions. Empirically, on both high-dimensional non-convex test function optimization and hyper-parameter optimization for multi-class SVM tasks, we show that RESOO performs better than SOO and random search.

The consequent sections introduce the SOO, describe the proposed RESOO, prove the regret bounds, present the empirical results, and finally conclude the paper.

Simultaneous Optimistic Optimization

Simultaneous Optimistic Optimization (SOO) combines the branch-and-bound principle with the optimistic optimization idea from multi-armed bandit for black-box optimization in continuous domain (Munos 2011; 2014). It is implemented via resorting to a hierarchical partitioning of the solution space $\mathcal{X}$ by building a K-ary tree. Let (h, i) denote a node of the tree indexed by its depth h ($h \geq 0$) and index i ($0 \leq i \leq K^h - 1$), and $X_{h,i} \subseteq \mathcal{X}$ denote the corresponding cell. The root node $(0, 0)$ (cell $X_{0,0}$) of the tree represents the whole solution space $\mathcal{X}$. Each node (h, i) possesses K children nodes $\{(h+1, i_k)\}_{1 \leq k \leq K}$, and the children partition the corresponding parent cell $X_{h,i}$ into cells $\{X_{h+1,i_k}\}_{1 \leq k \leq K}$. For each node (h, i), a specific solution $\boldsymbol{x}_{h,i} \in X_{h,i}$, which is always the center of cell $X_{h,i}$, is evaluated. The corresponding function value $f(\boldsymbol{x}_{h,i})$ represents the quality of node (h, i).

Let t and t' denote the number of node expansions and the number of function evaluations, respectively. SOO first initializes the tree $\mathcal{T}_0 = \{(0, 0)\}$ that only contains the root node. Then, it incrementally builds a tree $\mathcal{T}_t$ for $t = 1, \ldots, n/K$, where n is the number of function evaluation budget and we assume that n/K is a positive integer for simplicity. Let $\mathcal{L}_t$ denote the set of leaves in $\mathcal{T}_t$. SOO expands several leaves simultaneously. When a node is expanded, its children nodes are evaluated. At each round, SOO expands at most one leaf per depth, and a leaf is expanded if and only if it has the highest function value among all leaves of the same or lower depths. When function value budget is used up, i.e., $t' = n$, SOO returns the solution $\boldsymbol{x}(n)$ with the highest function value among all nodes of the current tree. In addition, we should note that SOO regards $h_{\max}(t)$, which restricts the maximum depth of the tree after t node expansions, as a parameter of the algorithm. The purpose of introducing $h_{\max}(t)$ is to make a trade-off between deep exploitation and broad exploration.

SOO has a solid theoretical guarantee on the convergence rate of its simple regret. The theoretical guarantee relies on the assumption that there exists a positive constant L and a semi-metric ℓ (ℓ may not satisfy the triangle inequality) such that $f(\boldsymbol{x}^*) - f(\boldsymbol{x}) \leq L \cdot \ell(\boldsymbol{x}^*, \boldsymbol{x})$ for all $\boldsymbol{x}$, where $\boldsymbol{x}^*$ is a maximizer of f. This assumption characterizes the local smoothness of f around the maximizer $\boldsymbol{x}^*$. Although SOO assumes the existence of such ℓ, it does not require the explicit knowledge of ℓ (true local smoothness) when implementing SOO. Thus, the assumption is not unrealistic.

However, it has been observed that the performance of SOO in high-dimensional optimization problems is unsatisfactory. We could intuitively imagine that SOO grows a tree to partition the solution space into grids. In a high-dimensional solution space, partitioning grids is of low efficiency, and thus SOO may mainly focus on exploration only. Furthermore, the inefficiency of SOO for high-dimensional problems could be reflected from the simple regret bound, which we will elaborate in the Theoretical Study section.

RESOO

For some high-dimensional functions, the function value is affected by a few effective dimensions. For this kind of problems, we adopt the concept of effective dimension which has been formally defined in (Wang et al. 2013).

DEFINITION 1 (Effective Dimension)
*A function $f\colon \mathbb{R}^D \to \mathbb{R}$ is said to have **effective dimension** d_e with $d_e < D$, if there exists a linear subspace $\mathcal{V} \subseteq \mathbb{R}^D$ with dimension d_e such that for all $\boldsymbol{x} \in \mathbb{R}^D$, we have $f(\boldsymbol{x}) = f(\boldsymbol{x}_e + \boldsymbol{x}_c) = f(\boldsymbol{x}_e)$, where $\boldsymbol{x}_e \in \mathcal{V} \subseteq \mathbb{R}^D$, $\boldsymbol{x}_c \in \mathcal{V}^\perp \subseteq \mathbb{R}^D$ and $\mathcal{V}^\perp$ denotes the orthogonal complement of $\mathcal{V}$. We call $\mathcal{V}$ the **effective subspace** of f and $\mathcal{V}^\perp$ the **constant subspace**.*

Intuitively, the definition means that f only changes along the subspace $\mathcal{V}$ (effective subspace), i.e., the function value does not vary along $\mathcal{V}^\perp$ (constant subspace). Given this definition, the following Lemma (Wang et al. 2013) indicates that random embedding can be effective for problems with low effective dimension. To apply random embedding, we only need to know an upper bound of effective dimension $d \geq d_e$ instead of getting d_e exactly. We omit the proof since it can be found in (Wang et al. 2013). Let $\mathcal{N}$ denote the Gaussian distribution with zero mean and $1/D$ variance.

LEMMA 1
Given a function $f\colon \mathbb{R}^D \to \mathbb{R}$ with effective dimension d_e, and a random matrix $\boldsymbol{A} \in \mathbb{R}^{D \times d}$ with independent entries sampled from $\mathcal{N}$ where $d \geq d_e$, then, with probability 1, for any $\boldsymbol{x} \in \mathbb{R}^D$, there exists a $\boldsymbol{y} \in \mathbb{R}^d$ such that $f(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{y})$.

Lemma 1 implies that, given a random embedding matrix $\boldsymbol{A} \in \mathbb{R}^{D \times d}$, for any maximizer $\boldsymbol{x}^* \in \mathbb{R}^D$, there exists $\boldsymbol{y}^* \in \mathbb{R}^d$ such that $f(\boldsymbol{A}\boldsymbol{y}^*) = f(\boldsymbol{x}^*)$. Thus, we can optimize the lower-dimensional function $g(\boldsymbol{y}) = f(\boldsymbol{A}\boldsymbol{y})$ rather than optimizing the original high-dimensional $f(\boldsymbol{x})$.

To perform SOO in low dimension, we need firstly choose a low-dimensional bounded rectangle region (low-

dimensional solution space) $\mathcal{Y} \subseteq \mathbb{R}^d$. The principle of choosing $\mathcal{Y}$ is that the maximizer of original problem is contained in the low-dimensional solution space $\mathcal{Y}$ with high probability. The following Lemma indicates that, under this principle, we can find an appropriate $\mathcal{Y} \subseteq \mathbb{R}^d$ that only relies on d, i.e., the upper bound of the effective dimension d_e. This Lemma is a slight modification of Theorem 3 in (Wang et al. 2013).

LEMMA 2

Given a function $f\colon \mathcal{X} \to \mathbb{R}$ with effective dimension $d_e \leq d$, where $\mathcal{X} = [-1,1]^D$. Denote $\boldsymbol{x}^* \in \mathcal{X}$ as a maximizer. Assume that the effective subspace $\mathcal{V}$ of f is a span of d_e basis vectors. Let $\boldsymbol{x}_e^* \in \mathcal{V} \cap \mathcal{X}$ be a maximizer of f inside $\mathcal{V}$. If $\boldsymbol{A}$ is a $D \times d$ random matrix with independent entries sampled from $\mathcal{N}$, then, for any $\eta \in (0,1)$, there exists a maximizer $\boldsymbol{y}^* \in \mathcal{Y} = [-d/\eta, d/\eta]^d \subseteq \mathbb{R}^d$ such that $f(\boldsymbol{A}\boldsymbol{y}^*) = f(\boldsymbol{x}_e^*)$ with probability at least $1-\eta$.

Proof. By Theorem 3 in (Wang et al. 2013), it can be derived directly that, with probability at least $1-\eta$, there exists a maximizer $\boldsymbol{y}^* \in \mathbb{R}^d$ such that $f(\boldsymbol{A}\boldsymbol{y}^*) = f(\boldsymbol{x}_e^*)$ and $\|\boldsymbol{y}^*\|_2 \leq \|\boldsymbol{x}_e^*\|_2 \cdot \sqrt{d_e}/\eta$. Note that $\mathcal{X} = [-1,1]^D$, $\boldsymbol{x}_e^* \in \mathcal{V} \cap \mathcal{X}$ is a maximizer of f inside $\mathcal{V}$ and $\mathcal{V}$ is the span of d_e basis vectors, we have

$$\|\boldsymbol{y}^*\|_2 \leq \|\boldsymbol{x}_e^*\|_2 \cdot \sqrt{d_e}/\eta \leq d_e/\eta.$$

Therefore, by $d \geq d_e$ and $\|\boldsymbol{y}^*\|_\infty \leq \|\boldsymbol{y}^*\|_2$, we deduce that, with probability at least $1-\eta$,

$$\|\boldsymbol{y}^*\|_\infty \leq \|\boldsymbol{y}^*\|_2 \leq d_e/\eta \leq d/\eta.$$

At last, letting $\mathcal{Y} = [-d/\eta, d/\eta]^d$ proves the Lemma. ■

To implement SOO in the low-dimension solution space $\mathcal{Y} = [-d/\eta, d/\eta]^d \subseteq \mathbb{R}^d$, an important issue is that there may exist $\boldsymbol{y}' \in \mathcal{Y}$ such that $\boldsymbol{A}\boldsymbol{y}' \notin \mathcal{X}$. The fact results in that f can not be evaluated at point $\boldsymbol{A}\boldsymbol{y}'$. We overcome this problem by Euclidean projection, i.e., $\boldsymbol{A}\boldsymbol{y}'$ is projected to $\mathcal{X}$, when it is outside $\mathcal{X}$, by $P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}) = \operatorname{argmin}_{\boldsymbol{x}\in\mathcal{X}} \|\boldsymbol{x} - \boldsymbol{A}\boldsymbol{y}\|_2$ for any $\boldsymbol{y} \in \mathcal{Y}$. Let $[\boldsymbol{x}]_i$ denote the i-th coordinate of $\boldsymbol{x}$. Since $\mathcal{X} = [-1,1]^D$, the Euclidean projection is:

$$[P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y})]_i = \begin{cases} -1, & \text{if } [\boldsymbol{A}\boldsymbol{y}]_i < -1; \\ [\boldsymbol{A}\boldsymbol{y}]_i, & \text{if } -1 \leq [\boldsymbol{A}\boldsymbol{y}]_i \leq 1; \\ 1, & \text{otherwise.} \end{cases} \quad (1)$$

Hence, we employ SOO to optimize the low-dimensional function $g(\boldsymbol{y}) = f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}))$ for $\boldsymbol{y} \in \mathcal{Y} = [-d/\eta, d/\eta]^d$ via random embedding. Another issue is that $\mathcal{Y}$ contains the maximizer only with a probability. One approach to handle the issue is to restart SOO $M \geq 1$ times each with an independently sampled random embedding matrix. Thus, the probability of success for M times independent random embedding is at least $1-\eta^M$, which converges to 1 exponentially with respect to M. In addition, note that the total function evaluation budget is n, thus the function evaluation budget for each random embedding SOO should be n/M and we assume w.l.o.g. that n/M is a positive integer.

Summing up the discussions above gives the proposed simultaneous optimistic optimization with random embedding (RESOO), which is shown in Algorithm 1. For exploration-exploitation trade-off, we choose $h_{\max}(t) = \sqrt{t}$. Note that RESOO is restarted M times independently each with n/M budget.

Algorithm 1 Simultaneous Optimistic Optimization with Random Embedding (RESOO)

Input:

Low-dimensional solution space $\mathcal{Y} = [-d/\eta, d/\eta]^d$;
Branching factor K;
Maximum depth function $h_{\max}(t) = \sqrt{t}$;
Number of running algorithm independently M.

Procedure:

Initialize $S = \emptyset$.
for $k = 1$ to M **do**
  Sample a random matrix $\boldsymbol{A} \in \mathbb{R}^{D\times d}$ with $\boldsymbol{A}_{i,j} \sim \mathcal{N}$.
  Initialize the tree $\mathcal{T}_0 = \{(0,0)\}$, and evaluate $f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}_{0,0}))$, set $t = 0$, $t' = 1$.
  while $t' < n/M$ **do**
    Set $v_{\max} = -\infty$.
    for $h = 0$ to $\min\{\text{depth}(\mathcal{T}_t), h_{\max}(t)\}$ **do**
      Select $(h,i) = \operatorname{argmax}_{(h,j)\in\mathcal{L}_t} f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}_{h,j}))$ among all leaves $(h,j) \in \mathcal{L}_t$ with depth h.
      if $f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}_{h,i})) \geq v_{\max}$ **then**
        Node expansion: add the K children $\{(h+1, i_k)\}_{1\leq k\leq K}$ of (h,i) to $\mathcal{T}_t$.
        Function evaluation: evaluate these K children $f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}_{h+1,i_k}))$, where $1 \leq k \leq K$.
        Set $v_{\max} = f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}_{h,i}))$, $t = t+1$ and $t' = t' + K$.
        if $t' \geq n/M$ **then**
          Find $\boldsymbol{x}(n) = \operatorname{argmax}_{(h,i)\in\mathcal{T}_t} P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}_{h,i})$, let $S = S \cup \{\boldsymbol{x}(n)\}$, and **break**.
        end if
      end if
    end for
  end while
end for
return $\boldsymbol{x}(n) = \operatorname{argmax}_{\boldsymbol{x}\in S} f(\boldsymbol{x})$.

Theoretical Study

In this section, we will prove the simple regret bound of RESOO, and compare it with that of SOO. For the purpose of self-contained and the convenient of comparison, we first state the assumptions made in (Munos 2011) when analyzing SOO and the simple regret bound of SOO.

Let $\ell(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^\alpha$ with $\alpha > 0$ denote the semi-metric, which means that ℓ may not satisfy the triangle inequality. Assumption 1 is on the local smoothness with respect to ℓ of f around any global maximizer $\boldsymbol{x}^*$. The local smoothness is characterized by locally one-sided Lipschitz continuity. Although this assumption is a special case of Assumption 2 in (Munos 2011), it does not affect the generality of this assumption and the corresponding regret bound.

ASSUMPTION 1 (Locally One-sided Lipschitz Continuity)

There exists $L > 0$ such that, for all $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{x}^* \in \mathcal{X}^*$, $f(\boldsymbol{x}^*) - f(\boldsymbol{x}) \leq L \cdot \ell(\boldsymbol{x}^*, \boldsymbol{x})$, where $\mathcal{X}^*$ is the set containing all the global maximizers of f.

Assumption 1 implies that the function does not decrease too fast around any global maximizer. Assumptions 2 and 3,

which are as same as Assumptions 3 and 4 in (Munos 2011), are about the property of hierarchical partitioning with respect to ℓ for the solution space.

ASSUMPTION 2 (Bounded Diameters)
There exists a decreasing sequence $\delta(h) > 0$, such that for any depth $h \geq 0$ and any cell $X_{h,i}$ of depth h, we have $\sup_{\boldsymbol{x} \in X_{h,i}} \ell(\boldsymbol{x}_{h,i}, \boldsymbol{x}) \leq \delta(h)$, where $\delta(h) = C\gamma^h$ for some $C > 0$ and $0 < \gamma < 1$.

ASSUMPTION 3 (Well-shaped Cells)
There exists $\nu > 0$ such that for any depth $h \geq 0$, any cell $X_{h,i}$ contains an ℓ-ball of radius $\nu\delta(h)$ centered in $X_{h,i}$.

To satisfy Assumptions 2 and 3, C and γ would have to take different values that rely on the values of α and the dimension of solution space. In fact, Assumptions 2 and 3 can be satisfied when $\mathcal{X} = [-1, 1]^D$ and the split is done along the largest dimension of a cell, which is the case in following of the paper. Since $\ell(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^\alpha$, for high-dimensional solution space $\mathcal{X}$, we can derive that $\sup_{\boldsymbol{x} \in X_{h,i}} \ell(\boldsymbol{x}_{h,i}, \boldsymbol{x}) \in \mathcal{O}(D^{\alpha/2} K^{-\alpha h/D})$. In addition to the above assumptions, the regret bound of SOO is closely related to the near-optimality dimension defined in (Munos 2011). Let $\mathcal{X}_\varepsilon = \{\boldsymbol{x} \in \mathcal{X} \mid f(\boldsymbol{x}) \geq f(\boldsymbol{x}^*) - \varepsilon\}$, which is the set of ε-optimal solutions. We state the definition below.

DEFINITION 2 (Near-optimality Dimension)
The near-optimality dimension is the smallest $d_{no} \geq 0$ such that there exists $\widetilde{C} > 0$, for all $\varepsilon > 0$, the maximal number of disjoint ℓ-balls with radius $\nu\varepsilon$ and center in $\mathcal{X}_\varepsilon$ that $\mathcal{X}_\varepsilon$ can be packed by is less than $\widetilde{C}\varepsilon^{-d_{no}}$.

Obviously, the near-optimality dimension d_{no} depends on the dimension of solution space D. Let $\varphi(z)$ be a non-negative and monotonically non-descending function with respect to z, for $\mathcal{X} = [-1, 1]^D$, $d_{no} = \varphi(D)$. Now, we introduce the simple regret bound of SOO described in Theorem 1, and derive it from Corollary 2 in (Munos 2011).

THEOREM 1
Under Assumption 1, 2 and 3, letting $h_{\max}(t) = \sqrt{t}$, if the near-optimality dimension $d_{no} > 0$, then, for large enough n, the simple regret r_n of SOO is upper bounded:

$$r_n \leq L \cdot C^{\frac{d_{no}+1}{d_{no}}} \Big(\frac{\widetilde{C}}{1-\gamma^{d_{no}}}\Big)^{\frac{1}{d_{no}}} \Big(\frac{n}{K}\Big)^{-\frac{1}{2d_{no}}} \in \mathcal{O}\Big(D^{\frac{\alpha(\varphi(D)+1)}{2\varphi(D)}} n^{-\frac{1}{2\varphi(D)}}\Big);$$

If the near-optimality dimension $d_{no} = 0$, then the simple regret r_n of SOO is upper bounded:

$$r_n \leq L \cdot C \cdot \gamma^{\sqrt{n/K} \min\{1/\widetilde{C}, 1\} - 1} \in \mathcal{O}\Big(D^{\frac{\alpha}{2}} \widetilde{\gamma}^{\sqrt{n}}\Big),$$

where $\widetilde{\gamma} = \gamma^{\min\{1/\widetilde{C}, 1\}/\sqrt{K}} \in (0, 1)$.

Proof. By Corollary 2 in (Munos 2011), we directly have that, $r_n \leq L \cdot C^{\frac{d_{no}+1}{d_{no}}} \big(\frac{\widetilde{C}}{1-\gamma^{d_{no}}}\big)^{\frac{1}{d_{no}}} \big(\frac{n}{K}\big)^{-\frac{1}{2d_{no}}}$ if $d_{no} > 0$, and $r_n \leq L \cdot C \cdot \gamma^{\sqrt{n/K} \min\{1/\widetilde{C}, 1\} - 1}$ if $d_{no} = 0$.

Since $\mathcal{X} = [-1, 1]^D$, the split is done along the largest dimension of a cell and $\ell(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|_2^\alpha$, it is easy to verify that $\sup_{\boldsymbol{x} \in X_{h,i}} \ell(\boldsymbol{x}_{h,i}, \boldsymbol{x}) \in \mathcal{O}(D^{\alpha/2} K^{-\alpha h/D})$ for high-dimensional solution space $\mathcal{X}$. Therefore, letting $\delta(h) = C\gamma^h \in \Theta(D^{\alpha/2} K^{-\alpha h/D})$ will meet Assumption 2. Note that $d_{no} = \varphi(D)$ for $\mathcal{X} = [-1, 1]^D$ and $\varphi(z)$ is a non-negative and monotonically non-descending function, we have $d_{no} \geq \varphi(1)$. Thus, if $d_{no} > 0$, we have $(\widetilde{C}(1 - \gamma^{d_{no}})^{-1})^{1/d_{no}} \in \mathcal{O}(1)$ and $K^{1/2d_{no}} \in \mathcal{O}(1)$.

Hence, if $d_{no} > 0$, then $r_n \in \mathcal{O}\big(D^{\frac{\alpha(\varphi(D)+1)}{2\varphi(D)}} n^{-\frac{1}{2\varphi(D)}}\big)$. If $d_{no} = 0$, then $r_n \in \mathcal{O}\big(D^{\frac{\alpha}{2}} \widetilde{\gamma}^{\sqrt{n}}\big)$. ■

In the low-dimensional solution space $\mathcal{Y} = [-d/\eta, d/\eta]^d$, the split is also done along the largest dimension of a cell. Thus, we can verify directly that $g(\boldsymbol{y}) = f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}))$ also satisfies Assumption 2 and 3 in the low-dimensional solution space $\mathcal{Y}$. Note that the random embedding in RESOO can preserve the local distance, which is stated formally by the Johnson-Lindenstrauss lemma (Achlioptas 2003; Vempala 2004) below. By injecting this property into the local Lipschitz continuity assumption, we prove that the simple regret of RESOO relies only on the upper bound of effective dimension of the problem, which is shown in Theorem 2.

LEMMA 3
A set of m points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m$ in $\mathbb{R}^d$ can be embedded (or projected) to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ in $\mathbb{R}^D$ such that all pairwise distances are preserved, i.e.,

$$(1 - \epsilon)\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2 \leq \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \leq (1 + \epsilon)\|\boldsymbol{y}_i - \boldsymbol{y}_j\|_2^2,$$

if $D > \frac{9 \ln m}{\epsilon^2 - \epsilon^3}$ and $0 < \epsilon \leq 1/2$, where $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{y}$, $\boldsymbol{A} \in \mathbb{R}^{D \times d}$ and $\boldsymbol{A}_{i,j}$ are sampled i.i.d. from $\mathcal{N}(0, 1/D)$.

THEOREM 2
Under Assumption 1, 2, 3, given $0 < \epsilon \leq 1/2$, if $D \in \Omega(\epsilon^{-2})$, $h_{\max}(t) = \sqrt{t}$, and the near-optimality dimension $d_{no} > 0$, then, for large enough n, with probability at least $1 - \eta^M$, the simple regret r_n of RESOO is upper bounded:

$$r_n \leq (1 + \epsilon)^{\frac{\alpha}{2}} L \cdot C^{\frac{d_{no}+1}{d_{no}}} \Big(\frac{\widetilde{C}}{1-\gamma^{d_{no}}}\Big)^{\frac{1}{d_{no}}} \Big(\frac{n}{MK}\Big)^{-\frac{1}{2d_{no}}} \in \mathcal{O}\Big((\eta^{-1} d^3)^{\frac{\alpha(\varphi(d)+1)}{2\varphi(d)}} n^{-\frac{1}{2\varphi(d)}}\Big);$$

If the near-optimality dimension $d_{no} = 0$, then with probability at least $1 - \eta^M$, the simple regret r_n of RESOO is upper bounded:

$$r_n \leq (1 + \epsilon)^{\frac{\alpha}{2}} L \cdot C \cdot \gamma^{\sqrt{n/MK} \min\{1/\widetilde{C}, 1\} - 1} \in \mathcal{O}\Big((\eta^{-1} d^3)^{\frac{\alpha}{2}} \widetilde{\gamma}^{\sqrt{n}}\Big),$$

where $\widetilde{\gamma} = \gamma^{\min\{1/\widetilde{C}, 1\}/\sqrt{MK}} \in (0, 1)$.

Proof. Applying Theorem 1 to prove this theorem, we only need to check whether Assumption 1 holds in low-dimensional solution space $\mathcal{Y}$ or not. Let $g(\boldsymbol{y}) = f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}))$ for $\boldsymbol{y} \in \mathcal{Y} = [-d/\eta, d/\eta]^d$. By Lemma 2 and the procedure of RESOO, we have that for RESOO, with probability at least $1 - \eta^M$, $g(\boldsymbol{y}^*) - g(\boldsymbol{y}) = f(\boldsymbol{A}\boldsymbol{y}^*) - f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}))$, where $\boldsymbol{y}^*$ is any global maximizer in $\mathcal{Y}$.

Since f satisfies Assumption 1 in the high-dimensional solution space $\mathcal{X}$ with respect to ℓ, we have $g(\boldsymbol{y}^*) - g(\boldsymbol{y}) =$

$f(\boldsymbol{A}\boldsymbol{y}^*) - f(P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y})) \leq L \cdot \ell(\boldsymbol{A}\boldsymbol{y}^*, P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y}))$. According to the definition of Euclidean projection (cf. formula (1)), it can be verified directly that $\|\boldsymbol{A}\boldsymbol{y}^* - P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y})\|_2 \leq \|\boldsymbol{A}\boldsymbol{y}^* - \boldsymbol{A}\boldsymbol{y}\|_2$ and thus $\ell(\boldsymbol{A}\boldsymbol{y}^*, P_{\mathcal{X}}(\boldsymbol{A}\boldsymbol{y})) \leq \ell(\boldsymbol{A}\boldsymbol{y}^*, \boldsymbol{A}\boldsymbol{y})$.

Combining the discussions above with the Johnson-Lindenstrauss lemma, we deduce that, with probability at least $1 - \eta^M$,

$$g(\boldsymbol{y}^*) - g(\boldsymbol{y}) \leq (1+\epsilon)^{\frac{\alpha}{2}} L \cdot \ell(\boldsymbol{y}^*, \boldsymbol{y}), \quad \forall \boldsymbol{y} \in \mathcal{Y},$$

where $\boldsymbol{y}^*$ is any global maximizer in $\mathcal{Y}$. That is to say, Assumption 1 holds in low-dimensional solution space $\mathcal{Y}$ with probability at least $1 - \eta^M$. Therefore, we can invoke Theorem 1 here to derive the simple regret bound of RESOO.

At last, for low-dimensional solution space $\mathcal{Y}$, we have that $\sup_{\boldsymbol{y} \in Y_{h,i}} \ell(\boldsymbol{y}_{h,i}, \boldsymbol{y}) \in \mathcal{O}(\eta^{-\alpha/2} d^{3\alpha/2} K^{-\alpha h/d})$ and $d_{no} = \varphi(d)$ where φ is a non-negative and monotonically non-descending function, which proves the theorem. ■

It is worthwhile to point out that, when $d_{no} = 0$, which is often the case as discussed in (Munos 2011), and $M = 1$, the convergence rate of simple regret for RESOO is approximately $(\eta D/d^3)^{\frac{\alpha}{2}}$ times faster than that of SOO, when d is much smaller than D. This indicates that RESOO is more efficient for high-dimensional optimization problems with low effective dimensions, and thus RESOO possesses the better scalability than SOO.

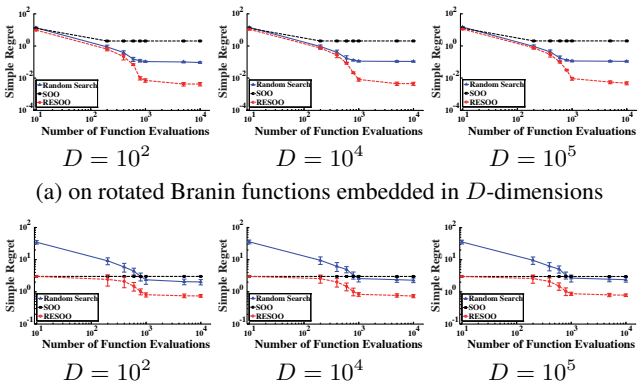
Empirical Study

In this section, we verify the scalability and effectiveness of RESOO empirically.In our experiment, for SOO and RESOO, the split is done along the largest dimension of a cell and the branching factor K is set to be 3.

Testing Functions

Two popular non-convex optimization test functions, Branin function (Lizotte 2008) and Rosenbrock function (Picheny, Wagner, and Ginsbourger 2013), are used. Branin function has $d_e = 2$ effective dimensions and Rosenbrock function has $d_e = 4$ effective dimensions. They are embedded in a D-dimensional space $\mathcal{X}$ with a large D. The embedding is done by, firstly, adding additional $D-2$ dimensions for Branin and $D-4$ dimensions for Rosenbrock, but the additional dimensions do not affect the value of the functions at all. Secondly, the embedded functions are rotated via a random rotation matrix. We set $\mathcal{X} = [-1, 1]^D$ and it is easy to be implemented by rescaling. We compare RESOO with SOO and random search. Random search, which uniformly and randomly sample points in each dimension, has been applied to optimize some difficult problems and shows acceptable performance (Bergstra and Bengio 2012). The simple regret value is adopted here to measure the performance of optimization algorithms.

We test the algorithms using the function evaluation budget $n \in \{10^1, 2\times10^2, 4\times10^2, 6\times10^2, 8\times10^2, 10^3, 5\times10^3, 10^4\}$, and the dimension of solution space $D = 10^2, 10^4, 10^5$ for Branin function and Rosenbrock functions. For RESOO maximizing Branin, let $d = 4 > d_e = 2$, and for RESOO maximizing Rosenbrock, let $d = 7 > d_e = 4$. We set $M = 2$ and $\eta = 1/3$, which means that we run RESOO twice independently (each of them only have $n/2$ budget) in low-dimensional solution space $\mathcal{Y} = [-3d, 3d]^d$. Random search and RESOO are repeated 30 times independently to report their mean performance.



$D = 10^2$ $D = 10^4$ $D = 10^5$

(a) on rotated Branin functions embedded in D-dimensions

(b) on rotated Rosenbrock functions embedded in D-dimensions

Figure 1: The convergence performance of RESOO.

Table 1: The simple regret (mean±standard deviation) of RESOO with different parameters d and M.

d	$M=1$	$M=2$	$M=4$	$M=10$
	1000-dimensional Branin function			
1	3.816±1.940	2.797±0.997	1.710±0.276	3.320±0.627
2	0.004±0.002	0.002±0.001	0.001±0.001	0.107±0.092
4	0.003±0.001	0.075±0.010	0.093±0.020	0.236±0.101
10	0.191±0.174	0.130±0.071	0.118±0.022	0.592±0.294
	1000-dimensional Rosenbrock function			
2	2.986±0.782	2.925±0.502	2.894±0.239	2.992±0.172
4	1.353±0.201	1.213±0.172	1.556±0.107	2.691±0.105
7	2.486±1.178	1.472±0.482	2.116±0.141	2.812±0.092
20	1.726±0.112	1.877±0.102	2.687±0.073	2.973±0.065

The convergence results are depicted in Figure 1. We can observe that, firstly, in every sub-figure, the simple regret curve of RESOO is the lowest, showing that RESOO has the best convergence rate; Secondly, comparing sub-figures with different dimensions, the convergence rate of RESOO is almost not effected, which verifies the theoretical result that the simple regret of RESOO relies on the upper bound of effective dimension d instead of D; Finally, we notice that SOO may even be worse than random search (on Rosenbrock functions), while RESOO shows to be consistently better than both SOO and random search. We hypothesize that this advantage might be due to the randomness of the random embedding and restarts.

In addition, we investigate how the estimated upper bound d and the number of restarts M affect the performance of RESOO. It should be noticed that when M is as large as the budget n, RESOO will degenerate to random search. We test on 1000-dimensional Branin function with $d_e = 2$ and 1000-dimensional Rosenbrock function with $d_e = 4$, and set $\eta = 1/3$, the budget of function evaluations $n = 600$. We run RESOO independently M times (each of them only have $600/M$ budget) in the low-dimensional solution space $\mathcal{Y} = [-3d, 3d]^d$. RESOO is repeated 30 times. The simple regret of RESOO under different parameters is shown in Table 1.

Table 2: Testing accuracies of the tuned multi-class SVM. An entry is bolded if the mean value is the highest in the row. RESOO is performed with two different estimates of effective dimension, i.e., d_1, d_2 with $d_1 < d_2$.

	Shared hyper-parameter ($D = 1$)			Separated hyper-parameters ($D = 0.5 \cdot \#class(\#class - 1)$)			
Dataset (#class)	Random Search	Grid Search	SOO	Random Search	SOO	RESOO (d_1)	RESOO (d_2)
MNIST (10)	93.75%±0.12%	94.58%	94.61%	94.13%±0.18%	94.33%	**94.81**%±0.05%	94.75%±0.04%
PenDigits (10)	95.55%±0.06%	95.56%	95.65%	95.72%±0.10%	95.58%	**95.92**%±0.06%	95.86%±0.05%
Vowel (11)	82.51%±0.17%	82.29%	82.73%	82.79%±0.21%	82.30%	**83.40**%±0.03%	83.16%±0.06%
Letter (26)	84.38%±0.19%	85.12%	85.22%	85.08%±0.23%	85.06%	**85.47**%±0.04%	85.38%±0.05%

It can be observed that, as we expected, a better estimation of the effective dimension, i.e., a smaller d, leads to a better performance, as long as $d \geq d_e$. But when $d < d_e$, RESOO has a very poor performance. This is intuitive since $d < d_e$ violates the theoretical assumption. For the parameter M, since RESOO succeeds with probability at least $1 - (1/3)^M$, if M is too small, the success probability is insufficient, e.g, when $M = 1, d = 2$ for Branin function and $M = 1, d = 7$ for Rosenbrock function. Meanwhile, if M is too large, the budget for each run of RESOO will be insufficient, which also results in unsatisfied performance. Therefore, a moderate M is needed to balance the success probability and the budget for RESOO.

Hyper-parameter Tuning for Multi-class SVM

We apply RESOO to tune the hyper-parameters in a multi-class SVM. Given a labeled training set $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$, the classification task in machine learning is to train a classifier from $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$ to predict the label of unseen data. In this section, we only consider the linear classifier $h(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + b$, thus we need to learn parameters $\boldsymbol{w}, b$ from $\{\boldsymbol{x}_i, y_i\}_{i=1}^m$. One of the famous linear classifiers is support vector machines (SVM) (Vapnik 1998) with linear kernel. If $y \in \{-1, +1\}$, SVM is formulated as $\min_{\boldsymbol{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \lambda \sum_{i=1}^m \xi_i$ subject to $y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i \wedge \xi_i \geq 0$ for $i = 1, \ldots, m$, where $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_m)^\top$. We should notice that there is a hyper-parameter $\lambda > 0$ in SVM which determines the trade-off between margin-maximization and the slack penalty minimization. If we are faced with multi-class classification, one common approach is to apply one-vs-one strategy to reduce the multi-class classification problem to a set of binary classification problems. Let N denote the number of class, then we need to train $\binom{N}{2} = \frac{N(N-1)}{2}$ binary classifiers, and thus there are $\Theta(N^2)$ number of hyper-parameters in all the classifiers. We often treat all the hyper-parameters λ as the same (one hyper-parameter) to reduce the cost of hyper-parameter tuning, assuming that the balance of every two classes is the same. However, this assumption hardly holds. Therefore, we try to optimize the $\binom{N}{2}$ hyper-parameters in multi-class SVM. Also, noticing that some classes may share the same hyper-parameters, this optimization problem can have a low effective dimension.

We compare RESOO with SOO, random search which is also applied to optimize the hyper-parameters in neural networks and deep belief networks (Bergstra and Bengio 2012), and grid search. The range of hyper-parameter λ of each binary classifier is set as $[10^{-3}, 10^2]$. For SOO and random search, we both perform them under the settings of same hyper-parameter value for different classifiers ($\mathcal{X} = [10^{-3}, 10^2]$) and different hyper-parameter values for different classifiers ($\mathcal{X} = [10^{-3}, 10^2]^{N(N-1)/2}$), respectively. For grid search, which selects n (budget of function evaluations) grid points over $[10^{-3}, 10^2]$ uniformly as the value of hyper-parameter, we only perform it under the setting of shared hyper-parameter value. We test on the data sets *MNIST* (LeCun et al. 1998) and *PenDigits*, *Vowel*, *Letter* (Blake, Keogh, and Merz 1998). They contains $10, 10, 11, 26$ classes respectively, and thus there are $45, 45, 55, 325$ hyper-parameters when the hyper-parameters are considered separately. All the features in each dataset are normalized into $[-1, 1]$ or $[0, 1]$. For each dataset, we separate it into training set, validation set and testing set. The hyper-parameter is tuned on the validation set, and the higher accuracy on the validation set the better the hyper-parameters. That is to say, we evaluate the quality of hyper-parameters with the accuracy of multi-class classifier on the validation set. The multi-class classifier with the best hyper-parameters that each method finds will be tested on the testing set. The accuracy on the testing set of each method on each dataset is reported in Table 2. We set the budget of function evaluations $n \approx 2D$. Specifically, for *MNIST*, *PenDigits* and *Vowel*, $n = 100$ and for *Letter*, $n = 600$. For RESOO, we choose two upper bounds of effective dimension $d_1 = 15, d_2 = 30$ for *MNIST*, *PenDigits* and *Vowel*, and $d_1 = 50, d_2 = 100$ for *Letter*. We set $\mathcal{Y} = [0, 10d_i/\eta]^{d_i}$ and $M = 2$ for RESOO, where $\eta = 1/3$ and $i = 1, 2$. Each randomized methods are repeated 30 times independently.

In Table 2, by comparing the two columns of random search, it can be observed that the separated hyper-parameters can lead to higher accuracy. Note that this higher accuracy is free since the optimization budge is the same. Then we observe that, although SOO has a better performance than grid search in the low dimension, it has a worse performance for optimizing the separated hyper-parameters, as it does not scale well. RESOO shows the best performance on all the four data sets, verifying its ability in high-dimensional problems. We also observe that a better estimate of the effective dimension ($d_1 < d_2$) leads to the better performance.

Conclusion

This paper aims at addressing the scalability of SOO, which is a remarkable theoretical-grounded black-box optimization method. We adapt the random embedding technique to scaling SOO, and propose the RESOO algorithm with better

scalability. We prove that the simple regret of RESOO depends only on the effective dimensions, even if the problem has a high dimensionality. Empirically, we show that RESOO performs better than SOO and random search on high-dimensional non-convex testing functions. We also apply RESOO to tune the hyper-parameters for multi-class SVM, leading the best accuracy comparing with the shared hyper-parameter multi-class SVM and separated hyper-parameter multi-class SVM optimized by random search and SOO.

Although the performance of SOO might be limited comparing with other methods such as Bayesian optimization algorithms, the studies with SOO are useful as the techniques can be transferred into other methods, e.g., in (Wang et al. 2014). We will also study the random embedding technique with other methods, e.g., (Yu, Qian, and Hu 2016). This paper only touches the case when the effective dimension is low. A more important and challenging issue is to optimize high-dimensional functions with high effective dimensions.

References

Achlioptas, D. 2003. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Computer and System Sciences* 66(4):671–687.

Bergstra, J., and Bengio, Y. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13:281–305.

Blake, C. L.; Keogh, E.; and Merz, C. J. 1998. UCI Repository of machine learning databases. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Brochu, E.; Cora, V. M.; and de Freitas, N. 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *CoRR* abs/1012.2599.

Bubeck, S.; Munos, R.; Stoltz, G.; and Szepesvári, C. 2011. *X*-armed bandits. *Journal of Machine Learning Research* 12:1655–1695.

Bubeck, S.; Munos, R.; and Stoltz, G. 2009. Pure exploration in multi-armed bandits problems. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, 23–37.

Bubeck, S.; Stoltz, G.; and Yu, J. Y. 2011. Lipschitz bandits without the lipschitz constant. In *Proceedings of the 22nd International Conference on Algorithmic Learning Theory*, 144–158.

Derbel, B., and Preux, P. 2015. Simultaneous optimistic optimization on the noiseless bbob testbed. In *Proceedings of the 2015 IEEE Congress on Evolutionary Computation*, 2010–2017.

Grill, J.-B.; Valko, M.; and Munos, R. 2015. Black-box optimization of noisy functions with unknown smoothness. In *Advances in Neural Information Processing Systems 29*.

Hansen, N.; Müller, S. D.; and Koumoutsakos, P. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation* 11(1):1–18.

Jones, D. R.; Perttunen, C. D.; and Stuckman, B. E. 1993. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications* 79(1):157–181.

Kaban, A.; Bootkrajang, J.; and Durrant, R. J. 2013. Towards large scale continuous EDA: a random matrix theory perspective. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, 383–390.

Kearfott, R. B. 1996. *Rigorous global search: continuous problems*. Nonconvex Optimization and Its Applications. Dordrecht, Boston, London: Kluwer Academic Publishers.

Kleinberg, R.; Slivkins, A.; and Upfal, E. 2008. Multi-armed bandits in metric spaces. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, 681–690.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lizotte, D. 2008. *Practical Bayesian Optimization*. Ph.D. Dissertation, University of Alberta, Canada.

Munos, R. 2011. Optimistic optimization of a deterministic function without the knowledge of its smoothness. In *Advances in Neural Information Processing Systems 24*, 783–791.

Munos, R. 2014. From bandits to Monte-Carlo Tree Search: The optimistic principle applied to optimization and planning. *Foundations and Trends in Machine Learning* 7(1):1–130.

Picheny, V.; Wagner, T.; and Ginsbourger, D. 2013. A benchmark of kriging-based infill criteria for noisy optimization. *Structural and Multidisciplinary Optimization* 48(3):607–626.

Pintér, J. D. 1996. *Global Optimization in Action, Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications*. Nonconvex Optimization and Its Applications. Dordrecht, Boston, London: Kluwer Academic Publishers.

Preux, P.; Munos, R.; and Valko, M. 2014. Bandits attack function optimization. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation*, 2245–2252.

Slivkins, A. 2011. Multi-armed bandits on implicit metric spaces. In *Advances in Neural Information Processing Systems 24*, 1602–1610.

Snoek, J.; Larochelle, H.; and Adams, R. P. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, 2960–2968.

Strongin, R. G., and Sergeyev, Y. D. 2000. *Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms*. Nonconvex Optimization and Its Applications. Dordrecht, Boston, London: Kluwer Academic Publishers.

Valko, M.; Carpentier, A.; and Munos, R. 2013. Stochastic simultaneous optimistic optimization. In *Proceedings of the 30th International Conference on Machine Learning*, 19–27.

Vapnik, V. 1998. *Statistical learning theory*. Wiley.

Vempala, S. S. 2004. *The Random Projection Method*, volume 65. Providence, Rhode Island: American Mathematical Society.

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 23rd International Joint Conference on Artificial Intelligence*, 1778–1784.

Wang, Z.; Shakibi, B.; Jin, L.; and de Freitas, N. 2014. Bayesian multi-scale optimistic optimization. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 1005–1014.

Yu, Y.; Qian, H.; and Hu, Y.-Q. 2016. Derivative-free optimization via classification. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI'16)*.