Bayesian Optimization in High Dimensions via Random Embeddings

Bayesian optimization, machine learning

Abstract

Bayesian optimization techniques have been successfully applied to robotics, planning, sensor placement, recommendation, advertising, intelligent user interfaces and automatic algorithm configuration. Despite these successes, the approach is restricted to problems of moderate dimension, and several workshops on Bayesian optimization have identified its scaling to high-dimensions as one of the holy grails of the field. In this paper, we introduce a novel random embedding idea to attack this problem. The resulting Random EMbedding Bayesian Optimization (REMBO) algorithm is very simple and applies to domains with both categorical and continuous variables. The experiments demonstrate that REMBO can effectively solve high-dimensional problems, including automatic parameter configuration of a very popular mixed integer linear programming solver.

1 Introduction

Let $f : \mathcal{X} \to \mathbb{R}$ be a function on a compact subset $\mathcal{X} \subseteq \mathbb{R}^D$. We address the following global optimization problem

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}).$$

We are particularly interested in objective functions f that may satisfy one or more of the following criteria: they do not have a closed-form expression, are expensive to evaluate, do not have easily available derivatives, or are non-convex. We treat f as a *blackbox* function that only allows us to query its function value at arbitrary $x \in \mathcal{X}$. To address objectives of this challenging nature, we adopt the Bayesian optimization framework. There is a rich literature on Bayesian optimization, and we refer readers unfamiliar with the topic to more tutorial treatments [Brochu *et al.*, 2009; Jones *et al.*, 1998; Jones, 2001; Lizotte *et al.*, 2011; Močkus, 1994; Osborne *et al.*, 2009] and recent theoretical results [Srinivas *et al.*, 2010; de Freitas *et al.*, 2012].

In a nutshell, in order to optimize a blackbox function f, Bayesian optimization uses a prior distribution that captures our beliefs about the behavior of f, and updates this prior with sequentially acquired data. Specifically, it iterates the following phases: (1) use the prior to decide at which input $x \in \mathcal{X}$ to query f next; (2) evaluate $f(x)$; and (3) update the prior based on the new data $(x, f(x))$. Step 1 uses a so-called *acquisition function* that quantifies the expected value of learning the value of $f(x)$ for each $x \in \mathcal{X}$. Bayesian optimization methods differ in their choice of prior and their choice of this acquisition function.

In recent years, the artificial intelligence community has increasingly used Bayesian optimization; see for example [Martinez–Cantin *et al.*, 2009; Brochu *et al.*, 2009; Srinivas *et al.*, 2010; Hoffman *et al.*, 2011; Lizotte *et al.*, 2011; Azimi *et al.*, 2012]. Despite many success stories, the approach is restricted to problems of moderate dimension, typically up to about 10; see for example the excellent and very recent overview in [Snoek *et al.*, 2012]. Of course, for a great many problems this is all that is needed. However, to advance the state of the art, we need to scale the methodology to high-dimensional parameter spaces. It is difficult to scale Bayesian optimization to high dimensions. To ensure that a global optimum is found, we require good coverage of $\mathcal{X}$, but as the dimensionality increases, the number of evaluations needed to cover $\mathcal{X}$ increases exponentially.

For *linear* bandits, Carpentier et al [2012] recently proposed a compressed sensing strategy to attack problems with a high degree of sparsity. Also recently, Chen et al [2012] made significant progress by introducing a two stage strategy for optimization and variable selection of high-dimensional GPs. In the first stage, sequential likelihood ratio tests, with a couple of tuning parameters, are used to select the relevant dimensions. This, however, requires the relevant dimensions to be axis-aligned with an ARD kernel. Chen et al provide empirical results only for synthetic examples (of up to 400 dimensions), but they provide key theoretical guarantees. Hutter et al [2011] used Bayesian optimization with random forests to tune up to 76 parameters of algorithms for solving combinatorial problems. Their method uses frequentist uncertainty estimates, which are problematic when the number of points is small. We will provide a comparison to their method in our experiments.

Many researchers have noted that for certain classes of problems most dimensions do not change the objective function significantly; examples include hyper-parameter optimization for neural networks and deep belief networks [Bergstra and Bengio, 2012] and automatic configura-

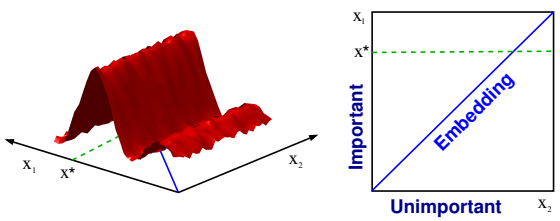

Figure 1: This function in D=2 dimensions only has d=1 *effective dimension*: the vertical axis indicated with the word important on the right hand side figure. Hence, the 1-dimensional embedding includes the 2-dimensional function's optimizer. It is more efficient to search for the optimum along the 1-dimensional random embedding than in the original 2-dimensional space.

tion of state-of-the-art algorithms for solving $\mathcal{NP}$-hard problems [Hutter, 2009]. That is to say these problems have *"low effective dimensionality"*. To take advantage of this property, [Bergstra and Bengio, 2012] proposed to simply use random search for optimization – the rationale being that points sampled uniformly at random in each dimension can densely cover each low-dimensional subspace. As such, random search can exploit low effective dimensionality *without knowing which dimensions are important*. In this paper, we exploit the same property, while still capitalizing on the strengths of Bayesian optimization. By combining randomization with Bayesian optimization, we are able to derive a new approach that outperforms each of the individual components.

Figure 1 illustrates our approach in a nutshell. Assume we know that a given $D = 2$ dimensional black-box function $f(x_1, x_2)$ only has $d = 1$ important dimensions, but we do not know which of the two dimensions is the important one. We can then perform optimization in the embedded 1-dimensional subspace defined by $x_1 = x_2$ since this is guaranteed to include the optimum. This idea enables us to perform Bayesian optimization in a low-dimensional space to optimize a high-dimensional function with low intrinsic dimensionality. Importantly, it is not restricted to cases with axis-aligned intrinsic dimensions.

2 Bayesian Optimization

Bayesian optimization has two ingredients that need to be specified: The prior and the acquisition function. In this work, we adopt GP priors. We review GPs very briefly and refer the interested reader to [Rasmussen and Williams, 2006]. A GP is a distribution over functions specified by its mean function $m(\cdot)$ and covariance $k(\cdot, \cdot)$. More specifically, given a set of points $\mathbf{x}_{1:t}$, with $\mathbf{x}_i \subseteq \mathbb{R}^D$, we have

$$\mathbf{f}(\mathbf{x}_{1:t}) \sim \mathcal{N}(\mathbf{m}(\mathbf{x}_{1:t}), \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})),$$

where $\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ serves as the covariance matrix. A common choice of k is the squared exponential function, but many other choices are possible depending on our degree of belief about the smoothness of the objective function.

An advantage of using GPs lies in their analytical tractability. In particular, given observations $\mathbf{x}_{1:n}$ with corresponding values $\mathbf{f}_{1:t}$, where $f_i = f(\mathbf{x}_i)$, and a new point $\mathbf{x}^*$, the joint distribution is given by:

$$\begin{bmatrix} \mathbf{f}_{1:t} \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{m}(\mathbf{x}_{1:t}), \begin{bmatrix} \mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t}) & \mathbf{k}(\mathbf{x}_{1:t}, \mathbf{x}^*) \\ \mathbf{k}(\mathbf{x}^*, \mathbf{x}_{1:t}) & k(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix}\right).$$

For simplicity, we assume that $\mathbf{m}(\mathbf{x}_{1:t}) = \mathbf{0}$. Using the Sherman-Morrison-Woodbury formula, one can easily arrive at the posterior predictive distribution:

$$f^*|\mathcal{D}_t, \mathbf{x}^* \sim \mathcal{N}(\mu(\mathbf{x}^*|\mathcal{D}_t), \sigma(\mathbf{x}^*|\mathcal{D}_t)),$$

with data $\mathcal{D}_t = \{\mathbf{x}_{1:t}, \mathbf{f}_{1:t}\}$, mean $\mu(\mathbf{x}^*|\mathcal{D}_t) = \mathbf{k}(\mathbf{x}^*, \mathbf{x}_{1:t})\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})^{-1}\mathbf{f}_{1:t}$ and variance $\sigma(\mathbf{x}^*|\mathcal{D}_t) = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, \mathbf{x}_{1:t})\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})^{-1}\mathbf{k}(\mathbf{x}_{1:t}, \mathbf{x}^*)$. That is, we can compute the posterior predictive mean $\mu(\cdot)$ and variance $\sigma(\cdot)$ exactly for any point $\mathbf{x}^*$.

At each iteration of Bayesian optimization, one has to re-compute the predictive mean and variance. These two quantities are used to construct the second ingredient of Bayesian optimization: The acquisition function. In this work, we report results for the expected improvement acquisition function $u(\mathbf{x}|\mathcal{D}_t) = \mathbb{E}(\max\{0, f_{t+1}(\mathbf{x}) - f(\mathbf{x}^+)\}|\mathcal{D}_t)$ [Močkus, 1982; Bull, 2011]. In this definition, $\mathbf{x}^+ = \arg\max_{\mathbf{x}\in\{\mathbf{x}_{1:t}\}} f(\mathbf{x})$ is the element with the best objective value in the first t steps of the optimization process. The next query is: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} u(\mathbf{x}|\mathcal{D}_t)$. Note that this utility favors the selection of points with high variance (points in regions not well explored) and points with high mean value (points worth exploiting). We also experimented with the UCB acquisition function [Srinivas *et al.*, 2010; de Freitas *et al.*, 2012] and found it to yield similar results. The optimization of the closed-form acquisition function can be carried out by off-the-shelf numerical optimization procedures. The Bayesian optimization procedure is shown in Algorithm 1.

Algorithm 1 Bayesian Optimization

1: **for** $t = 1, 2, \ldots$ **do**
2: Find $\mathbf{x}_{t+1} \in \mathbb{R}^D$ by optimizing the acquisition function u: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} u(\mathbf{x}|\mathcal{D}_t)$.
3: Augment the data $\mathcal{D}_{t+1} = \{\mathcal{D}_t, (\mathbf{x}_{t+1}, f(\mathbf{x}_{t+1}))\}$
4: **end for**

3 REMBO

Before introducing our new algorithm and its theoretical properties, we need to define what we mean by effective dimensionality formally.

Definition 1. *A function $f : \mathbb{R}^D \to \mathbb{R}$ is said to have **effective dimensionality** d_e, with $d_e < D$, if there exists a linear subspace $\mathcal{T}$ of dimension d_e such that for all $\mathbf{x}_\top \in \mathcal{T} \subset \mathbb{R}^D$ and $\mathbf{x}_\perp \in \mathcal{T}^\perp \subset \mathbb{R}^D$, we have $f(\mathbf{x}) = f(\mathbf{x}_\top + \mathbf{x}_\perp) = f(\mathbf{x}_\top)$, where $\mathcal{T}^\perp$ denotes the orthogonal complement of $\mathcal{T}$. We call $\mathcal{T}$ the **effective subspace** of f and $\mathcal{T}^\perp$ the **constant subspace**.*

This definition simply states that the function does not change along the coordinates $\mathbf{x}_\perp$, and this is why we refer to $\mathcal{T}^\perp$ as the constant subspace. Given this definition, the following theorem shows that problems of low effective dimensionality can be solved via random embedding.

Theorem 2. *Assume we are given a function $f : \mathbb{R}^D \to \mathbb{R}$ with effective dimensionality d_e and a random matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$ with independent entries sampled according to $\mathcal{N}(0,1)$ and $d \geq d_e$. Then, with probability 1, for any $\mathbf{x} \in \mathbb{R}^D$, there exists a $\mathbf{y} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = f(\mathbf{A}\mathbf{y})$.*

Proof. Since f has effective dimensionality d_e, there exists an effective subspace $\mathcal{T} \subset \mathbb{R}^D$, such that $\text{rank}(\mathcal{T}) = d_e$. Furthermore, any $\mathbf{x} \in \mathbb{R}^D$ decomposes as $\mathbf{x} = \mathbf{x}_\top + \mathbf{x}_\perp$, where $\mathbf{x}_\top \in \mathcal{T}$ and $\mathbf{x}_\perp \in \mathcal{T}^\perp$. Hence, $f(\mathbf{x}) = f(\mathbf{x}_\top + \mathbf{x}_\perp) = f(\mathbf{x}_\top)$. Therefore, without loss of generality, it will suffice to show that for all $\mathbf{x}_\top \in \mathcal{T}$, there exists a $\mathbf{y} \in \mathbb{R}^d$ such that $f(\mathbf{x}_\top) = f(\mathbf{A}\mathbf{y})$.

Let $\mathbf{\Phi} \in \mathbb{R}^{D\times d_e}$ be a matrix, whose columns form an orthonormal basis for $\mathcal{T}$. Hence, for each $\mathbf{x}_\top \in \mathcal{T}$, there exists a $\mathbf{c} \in \mathbb{R}^{d_e}$ such that $\mathbf{x}_\top = \mathbf{\Phi}\mathbf{c}$. Let us for now assume that $\mathbf{\Phi}^T\mathbf{A}$ has rank d_e. If $\mathbf{\Phi}^T\mathbf{A}$ has rank d_e, there exists a $\mathbf{y}$ such that $(\mathbf{\Phi}^T\mathbf{A})\mathbf{y} = \mathbf{c}$. The orthogonal projection of $\mathbf{A}\mathbf{y}$ onto $\mathcal{T}$ is given by $\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{A}\mathbf{y} = \mathbf{\Phi}\mathbf{c} = \mathbf{x}_\top$. Thus $\mathbf{A}\mathbf{y} = \mathbf{x}_\top + \mathbf{x}'$ for some $\mathbf{x}' \in \mathcal{T}^\perp$ since $\mathbf{x}_\top$ is the projection $\mathbf{A}\mathbf{y}$ onto $\mathcal{T}$. Consequently, $f(\mathbf{A}\mathbf{y}) = f(\mathbf{x}_\top + \mathbf{x}') = f(\mathbf{x}_\top)$.

It remains to show that, with probability one, the matrix $\mathbf{\Phi}^T\mathbf{A}$ has rank d_e. Let $\mathbf{A}_e \in \mathbb{R}^{D\times d_e}$ be a submatrix of $\mathbf{A}$ consisting of any d_e columns of $\mathbf{A}$, which are *i.i.d.* samples distributed according to $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Then, $\mathbf{\Phi}^T\mathbf{a}_i$ are *i.i.d.* samples from $\mathcal{N}(\mathbf{0}, \mathbf{\Phi}^T\mathbf{\Phi}) = \mathcal{N}(\mathbf{0}_{d_e}, \mathbf{I}_{d_e\times d_e})$, and so we have $\mathbf{\Phi}^T\mathbf{A}_e$, when considered as an element of $\mathbb{R}^{d_e^2}$, is a sample from $\mathcal{N}(\mathbf{0}_{d_e^2}, \mathbf{I}_{d_e^2\times d_e^2})$. On the other hand, the set of singular matrices in $\mathbb{R}^{d_e^2}$ has Lebesgue measure zero, since it is the zero set of a polynomial (i.e. the determinant function) and polynomial functions are Lebesgue measurable. Moreover, the Normal distribution is absolutely continuous with respect to the Lebesgue measure, so our matrix $\mathbf{\Phi}^T\mathbf{A}_e$ is almost surely non-singular, which means that it has rank d_e and so the same is true of $\mathbf{\Phi}^T\mathbf{A}$, whose columns contain the columns of $\mathbf{\Phi}^T\mathbf{A}_e$. □

Theorem 2 says that given any $\mathbf{x} \in \mathbb{R}^D$ and a random matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$, with probability 1, there is a point $\mathbf{y} \in \mathbb{R}^d$ such that $f(\mathbf{x}) = f(\mathbf{A}\mathbf{y})$. This implies that for any optimizer $\mathbf{x}^\star \in \mathbb{R}^D$, there is a point $\mathbf{y}^\star \in \mathbb{R}^d$ with $f(\mathbf{x}^\star) = f(\mathbf{A}\mathbf{y}^\star)$. Therefore, instead of optimizing in the high dimensional space, we can optimize the function $g(\mathbf{y}) = f(\mathbf{A}\mathbf{y})$ in the lower dimensional space. This observation gives rise to our new algorithm Bayesian Optimization with Random Embedding (REMBO), described in Algorithm 2. REMBO first draws a random embedding (given by $\mathbf{A}$) and then performs Bayesian optimization in this embedded space.

An important detail is how REMBO chooses the bounded region $\mathcal{Y}$, inside which it performs Bayesian optimization. This is important because its effectiveness depends on the size of $\mathcal{Y}$. Locating the optimum within $\mathcal{Y}$ is easier if $\mathcal{Y}$ is small, but if we set $\mathcal{Y}$ too small it may not actually contain the global optimizer (see Figure 2). In the following theorem, we show that we can choose $\mathcal{Y}$ in a way that only depends on the effective dimensionality d_e such that the optimizer of the original problem is contained in the low dimensional space with constant probability. (If $\mathbf{A}\mathbf{y}$ is outside the box $\mathcal{X}$, it is projected onto $\mathcal{X}$.)

Algorithm 2 REMBO: Bayesian Optimization with Random Embedding

1: Generate a random matrix $\mathbf{A}$
2: Choose the set $\mathcal{Y}$
3: **for** $t = 1, 2, \ldots$ **do**
4: Find $\mathbf{y}_{t+1} \in \mathbb{R}^d$ by optimizing the acquisition function u: $\mathbf{y}_{t+1} = \arg\max_{\mathbf{y}\in\mathcal{Y}} u(\mathbf{y}|\mathcal{D}_t)$.
5: Augment the data $\mathcal{D}_{t+1} = \{\mathcal{D}_t, (\mathbf{y}_{t+1}, f(\mathbf{A}\mathbf{y}_{t+1})\}$
6: Update the kernel hyper-parameters.
7: **end for**

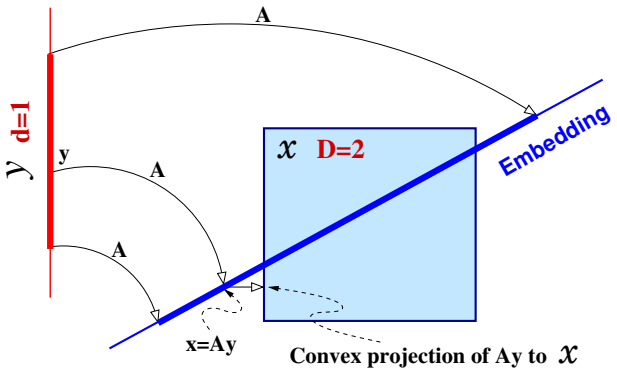

Figure 2: Embedding from $d = 1$ into $D = 2$. The box illustrates the 2D constrained space $\mathcal{X}$, while the thicker red line illustrates the 1D constrained space $\mathcal{Y}$. Note that if $\mathbf{A}\mathbf{y}$ is outside $\mathcal{X}$, it is projected onto $\mathcal{X}$. The set $\mathcal{Y}$ must be chosen large enough so that the projection of its image, $\mathbf{A}\mathcal{Y}$, onto the effective subspace (vertical axis in this diagram) covers the vertical side of the box.

Theorem 3. *Suppose we want to optimize a function $f : \mathbb{R}^D \to \mathbb{R}$ with effective dimension $d_e \leq d$ subject to the box constraint $\mathcal{X} \subset \mathbb{R}^D$, where $\mathcal{X}$ is centered around $\mathbf{0}$. Let us denote one of the optimizers by $\mathbf{x}^\star$. Suppose further that the effective subspace $\mathcal{T}$ of f is such that $\mathcal{T}$ is the span of d_e basis vectors. Let $\mathbf{x}^\star_\top \in \mathcal{T} \cap \mathcal{X}$ be an optimizer of f inside $\mathcal{T}$. If $\mathbf{A}$ is a $D \times d$ random matrix with independent standard Gaussian entries, there exists an optimizer $\mathbf{y}^\star \in \mathbb{R}^d$ such that $f(\mathbf{A}\mathbf{y}^\star) = f(\mathbf{x}^\star_\top)$ and $\|\mathbf{y}^\star\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2$ with probability at least $1 - \epsilon$.*

Proof. Since $\mathcal{X}$ is a box constraint, by projecting $\mathbf{x}^\star$ to $\mathcal{T}$ we get $\mathbf{x}^\star_\top \in \mathcal{T} \cap \mathcal{X}$. Also, since $\mathbf{x}^\star = \mathbf{x}^\star_\top + \mathbf{x}_\perp$ for some $\mathbf{x}_\perp \in \mathcal{T}^\perp$, we have $f(\mathbf{x}^\star) = f(\mathbf{x}^\star_\top)$. Hence, $\mathbf{x}^\star_\top$ is an optimizer. By using the same argument as appeared in Proposition 1, it is easy to see that with probability 1 $\forall \mathbf{x} \in \mathcal{T}\ \exists \mathbf{y} \in \mathbb{R}^d$ such that $\mathbf{A}\mathbf{y} = \mathbf{x} + \mathbf{x}_\perp$ where $\mathbf{x}_\perp \in \mathcal{T}^\perp$. Let $\mathbf{\Phi}$ be the matrix whose columns form a standard basis for $\mathcal{T}$. Without loss of generality, we can assume that $\mathbf{\Phi} = [\mathbf{I}_{d_e}\ \mathbf{0}]^T$. Then, as shown in Proposition 2, there exists a $\mathbf{y}^\star \in \mathbb{R}^d$ such that $\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{A}\mathbf{y}^\star = \mathbf{x}^\star_\top$. Note that for each column of $\mathbf{A}$, we have

$$\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{a}_i \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix}\mathbf{I}_{d_e} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\right).$$

Therefore $\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{A}\mathbf{y}^\star = \mathbf{x}^\star_\top$ is equivalent to $\mathbf{B}\mathbf{y}^\star = \bar{\mathbf{x}}^\star_\top$ where $\mathbf{B} \in \mathbb{R}^{d_e \times d_e}$ is a random matrix with independent standard Gaussian entries and $\bar{\mathbf{x}}^\star_\top$ is the vector that contains the first d_e entries of $\mathbf{x}^\star_\top$ (the rest are 0's). By Theorem 3.4 of [Sankar *et al.*, 2003], we have

$$\mathbb{P}\left[\|\mathbf{B}^{-1}\|_2 \geq \frac{\sqrt{d_e}}{\epsilon}\right] \leq \epsilon.$$

Thus, with probability at least $1-\epsilon$, $\|\mathbf{y}^\star\| \leq \|\mathbf{B}^{-1}\|_2\|\bar{\mathbf{x}}^\star_\top\|_2 = \|\mathbf{B}^{-1}\|_2\|\mathbf{x}^\star_\top\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2$. □

Theorem 3 says that if the set $\mathcal{X}$ in the original space is a box constraint, then there exists an optimizer $\mathbf{x}^\star_\top \in \mathcal{X}$ that is d_e-sparse such that with probability at least $1-\epsilon$, $\|\mathbf{y}^\star\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2$ where $f(\mathbf{A}\mathbf{y}^\star) = f(\mathbf{x}^\star_\top)$. If the box constraint is $\mathcal{X} = [-1,1]^D$ (which is always achievable through rescaling), we have with probability at least $1-\epsilon$ that

$$\|\mathbf{y}^\star\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\sqrt{d_e}.$$

Hence, to choose $\mathcal{Y}$, we just have to make sure that the ball of radius d_e/ϵ satisfies $(\mathbf{0}, \frac{d_e}{\epsilon}) \subseteq \mathcal{Y}$. In most practical scenarios, we found that the optimizer does not fall on the boundary which implies that $\|\mathbf{x}^\star_\top\|_2 < d_e$. Thus setting $\mathcal{Y}$ too big may be unnecessarily wasteful. In all our experiments we set $\mathcal{Y}$ to be $[-\sqrt{d}, \sqrt{d}]^d$.

Theorem 3 only guarantees that $\mathcal{Y}$ contains the optimum with probability at least $1-\epsilon$; with probability $\delta \leq \epsilon$ the optimizer lies outside of $\mathcal{Y}$. There are several ways to guard against this problem. One is to simply run REMBO multiple times with different independently drawn random embeddings. Since the probability of failure with each embedding is δ, the probability of the optimizer not being included in the considered space of k independently drawn embeddings is δ^k. Thus, the failure probability vanishes exponentially quickly in the number of REMBO runs, k. Note also that these independent runs can be trivially parallelized to harness the power of modern multi-core machines and large clusters.

3.1 Choice of Kernel

We begin our discussion on kernels with the definition of the squared exponential kernel between two points, $\mathbf{y}^{(1)}, \mathbf{y}^{(2)}$, on $\mathcal{Y} \subseteq \mathbb{R}^d$. Given a length scale $\ell > 0$, we define the corresponding squared exponential kernel as

$$k^d_\ell(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \exp\left(-\frac{\|\mathbf{y}^{(1)} - \mathbf{y}^{(2)}\|^2}{2\ell^2}\right).$$

It is possible to work with two variants of this kernel. First, we can use $k^d_\ell(\mathbf{y}^1, \mathbf{y}^2)$ as above. We refer to this kernel as the low-dimensional kernel. We can also adopt an implicitly defined high-dimensional kernel on $\mathcal{X}$:

$$k^D_\ell(\mathbf{y}^{(1)}, \mathbf{y}^2) = \exp\left(-\frac{\|p_\mathcal{X}(\mathbf{A}\mathbf{y}^{(1)}) - p_\mathcal{X}(\mathbf{A}\mathbf{y}^{(2)})\|^2}{2\ell^2}\right),$$

where $p_\mathcal{X} : \mathbb{R}^D \to \mathbb{R}^D$ is the standard projection operator for our box-constraint: $p_\mathcal{X}(\mathbf{x}) = \arg\min_{\mathbf{z}\in\mathcal{X}}\|\mathbf{z} - \mathbf{x}\|_2$; see Figure 2.

Note that when using the high-dimensional kernel, we are fitting the GP in D dimensions. However, the search space is no longer the box $\mathcal{X}$, but it is instead given by the much smaller subspace $\{p_\mathcal{X}(\mathbf{A}\mathbf{y}) : \mathbf{y} \in \mathcal{Y}\}$. *Importantly, in practice it is easier to maximize the acquisition function in this subspace.* Our experiment on automatic algorithm configuration will show that indeed optimizing the acquisition function in low-dimensions leads to significant gains of REMBO over standard Bayesian optimization.

The low-dimensional kernel has the benefit of only having to construct a GP in the space of intrinsic dimensionality d, whereas the high-dimensional kernel has to construct the GP in a space of extrinsic dimensionality D. However, the choice of kernel also depends on whether our variables are continuous, integer or categorical. The categorical case is important because we often encounter optimization problems that contain discrete choices (in fact, we focus on this case in our application in the experiments). We define our kernel for categorical variables as:

$$k^D_\lambda(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \exp\left(-\frac{\lambda}{2}g(s(\mathbf{A}\mathbf{y}^{(1)}), s(\mathbf{A}\mathbf{y}^{(2)}))^2\right),$$

where $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \mathcal{Y} \subset \mathbb{R}^d$ and g defines the distance between 2 vectors. The function s maps continuous vectors to discrete vectors. In more detail, $s(\mathbf{x})$ first projects $\mathbf{x}$ to $[-1,1]^D$ to generate $\bar{\mathbf{x}}$. For each dimension $\bar{x}_i$ of $\bar{\mathbf{x}}$, s maps $\bar{x}_i$ to the corresponding discrete parameters by scaling and rounding. In our experiments, following [Hutter, 2009], we defined $g(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = |\{i : x^{(1)}_i \neq x^{(2)}_i\}|$ so as not to impose an artificial ordering between the values of categorical parameters. In essence, we measure the distance between two points in the low-dimensional space as the distance between their mappings in the high-dimensional space.

Our demonstration of REMBO, in the domain of algorithm configuration, will use the high-dimensional kernel because the parameters in need of tuning are categorical. However, to provide the reader with a taste of the potential gains that the low dimensional kernel could offer in continuous spaces, we will use a synthetic optimization problem. This synthetic example will also allow us to easily discuss different properties of the algorithm, including rotational invariance and dependency on d.

Finally, when using the high-dimensional kernel, the regret bounds of [Srinivas *et al.*, 2010; Bull, 2011; de Freitas *et al.*, 2012] apply. For the low dimensional case, we have derived regret bounds that only depend on the intrinsic dimensionality. For lack of space, we will present these in a longer technical report version of this paper.

4 Experiments

For all our experiments, we used a single robust version of REMBO that automatically sets its GP's length scale parameter using a variant of maximum likelihood (the precise description and software will be released after the review process). For each optimization of the acquisition function, this version runs both DIRECT [Jones *et al.*, 1993] and CMA-ES [Hansen and Ostermeier, 2001] and uses the result of the better of the two. Again, the code for REMBO, as well as all

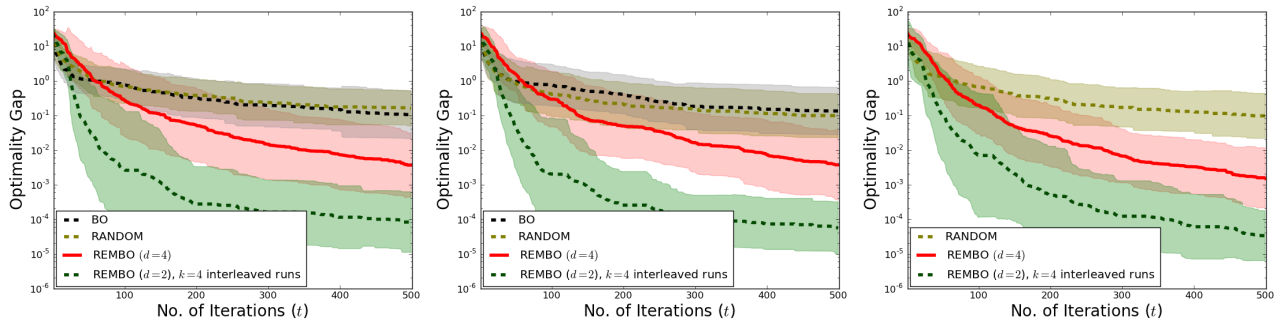

Figure 3: Comparison of random search (RANDOM), standard Bayesian optimization (BO), and REMBO. Left: $D = 25$ extrinsic dimensions; Middle: $D = 25$, with a rotated objective function; Right: $D = 1\,000\,000\,000$ extrinsic dimensions. For each method, we plot means and $1/4$ standard deviation confidence intervals of the optimality gap across 50 trials.

data used in our experiments will be made publicly available after the double-blind review.

4.1 Bayesian Optimization in a Billion Dimensions

The experiments in this section employ a standard $d_e = 2$-dimensional benchmark function for Bayesian optimization, embedded in a D-dimensional space. That is, we add $D - 2$ additional dimensions which do not affect the function at all. More precisely, the function whose optimum we seek is $f(\mathbf{x}_{1:D}) = g(x_i, x_j)$, where g is the Branin function (for its exact formula, see [Lizotte, 2008]) and where i and j are selected once using a random permutation. To measure the performance of each optimization method, we used the *optimality gap*: the difference of the best function value it found and the optimal function value.

k	$d = 2$	$d = 4$	$d = 6$
10	0.0022 ± 0.0035	0.1553 ± 0.1601	0.4865 ± 0.4769
5	0.0004 ± 0.0011	0.0908 ± 0.1252	0.2586 ± 0.3702
4	0.0001 ± 0.0003	0.0654 ± 0.0877	0.3379 ± 0.3170
2	0.1514 ± 0.9154	0.0309 ± 0.0687	0.1643 ± 0.1877
1	0.7406 ± 1.8996	0.0143 ± 0.0406	0.1137 ± 0.1202

Table 1: Optimality gap for $d_e = 2$-dimensional Branin function embedded in $D = 25$ dimensions, for REMBO variants using a total of 500 function evaluations. The variants differed in the internal dimensionality d and in the number of interleaved runs k (each such run was only allowed $500/k$ function evaluations). We show mean and standard deviations of the optimality gap achieved after 500 function evaluations.

We evaluate REMBO using a fixed budget of 500 function evaluations that is spread across multiple interleaved runs — for example, when using $k = 4$ interleaved REMBO runs, each of them was only allowed 125 function evaluations. We study the choices of k and d by considering several combinations of these values. The results in Table 1 demonstrate that interleaved runs helped improve REMBO's performance. We note that in 13/50 REMBO runs, the global optimum was indeed not contained in the box $\mathcal{Y}$ REMBO searched with $d = 2$; this is the reason for the poor mean performance of REMBO with $d = 2$ and $k = 1$. However, the remaining 37 runs performed very well, and REMBO thus performed well when using multiple interleaved runs: with a failure rate of 13/50=0.26 per independent run, the failure rate using $k = 4$ interleaved runs is only $0.26^4 \approx 0.005$. One could easily achieve an arbitrarily small failure rate by using many independent parallel runs. Using a larger d is also effective in increasing the probability of the optimizer falling into REMBO's box $\mathcal{Y}$ but at the same time slows down REMBO's convergence (such that interleaving several short runs looses its effectiveness).

Next, we compared REMBO to standard Bayesian optimization (BO) and to random search, for an extrinsic dimensionality of $D = 25$. Standard BO is well known to perform well in low dimensions, but to degrade above a tipping point of about 15-20 dimensions. Our results for $D = 25$ (see Figure 3, left) confirm that BO performed rather poorly just above this critical dimensionality (merely tying with random search). REMBO, on the other hand, still performed very well in 25 dimensions.

One important advantage of REMBO is that — in contrast to the approach of [Chen *et al.*, 2012] — it does not require the effective dimension to be coordinate aligned. To demonstrate this fact empirically, we rotated the embedded Branin function by an orthogonal rotation matrix $\mathbf{R} \in \mathbb{R}^{D \times D}$. That is, we replaced $f(\mathbf{x})$ by $f(\mathbf{Rx})$. Figure 3 (middle) shows that REMBO's performance is not affected by this rotation. Finally, since REMBO is independent of the extrinsic dimensionality D as long as the intrinsic dimensionality d_e is small, it performed just as well in $D = 1\,000\,000\,000$ dimensions (see Figure 3, right). To the best of our knowledge, the only other existing method that can be run in such high dimensionality is random search.

4.2 Automatic Configuration of a Mixed Integer Linear Programming Solver

State-of-the-art algorithms for solving hard computational problems tend to parameterize several design choices in order to allow a customization of the algorithm to new problem domains. Automated methods for algorithm configuration have recently demonstrated that substantial performance gains of state-of-the-art algorithms can be achieved in a fully automated fashion [Močkus *et al.*, 1999; Hutter *et al.*, 2010; Bergstra *et al.*, 2011; Wang and de Freitas, 2011]. These successes have led to a paradigm shift in algorithm devel-

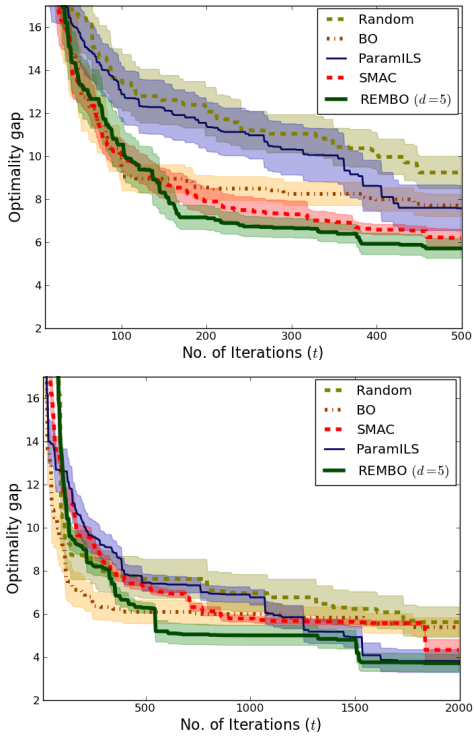

Figure 4: Performance of various methods for configuration of `lpsolve`; we show the optimality gap `lpsolve` achieved with the configurations found by the various methods (lower is better). Top: a single run of each method; Bottom: performance with $k = 4$ interleaved runs. For each method, we plot means and $1/4$ standard deviation confidence intervals.

opment towards the active design of highly parameterized frameworks that can be automatically customized to particular problem domains using optimization [Hoos, 2012; Bergstra *et al.*, 2012].

It has long been suspected that the resulting algorithm configuration problems have low dimensionality [Hutter, 2009]. Here, we demonstrate that REMBO can exploit this low dimensionality even in the discrete spaces typically encountered in algorithm configuration. We use a configuration problem obtained from [Hutter *et al.*, 2010], aiming to configure the 40 binary and 7 categorical parameters of lpsolve[1], a popular mixed integer linear programming solver that has been downloaded over 40 000 times in the last year. The objective is to minimize the optimality gap `lpsolve` can obtain in a time limit of five seconds for a mixed integer programming (MIP) encoding of a wildlife corridor problem from computational sustainability [Gomes *et al.*, 2008]. Algorithm configuration usually aims to improve performance for a representative set of problem instances, and effective methods need to solve two orthogonal problems: searching the parameter space effectively and deciding how many instances to use in each evaluation (to trade off computational overhead and over-fitting). Our contribution is for the first of these problems; to focus on how effectively the different methods search the parameter space, we only consider configuration on a single problem instance.

Due to the discrete nature of this optimization problem, we could only apply REMBO using the high-dimensional kernel for categorical variables $k_\lambda^D(\mathbf{y}^{(1)}, \mathbf{y}^{(2)})$ described in Section 3.1. While we have not proven any theoretical guarantees for discrete optimization problems, REMBO appears to effectively exploit the low effective dimensionality of at least this particular optimization problem.

Figure 4.2 (top) compares BO, REMBO, and the baseline random search against the configuration procedures ParamILS (which was used for all configuration experiments in [Hutter *et al.*, 2010]) and SMAC [Hutter *et al.*, 2011]. ParamILS and SMAC have been specifically designed for the configuration of algorithms with many discrete parameters and define the current state of the art for this problem. As the figure shows, SMAC outperformed both BO and random search by a comfortable margin. However, our vanilla REMBO method performed slightly better. While the figure only shows REMBO with $d = 5$ to avoid clutter, we by no means optimized this parameter; the only other value we tried was $d = 3$, which resulted in indistinguishable performance.

As in the synthetic experiment, REMBO's performance could be further improved by using multiple interleaved runs. However, it is known that multiple independent runs can also improve the performance of the other procedures, especially ParamILS [Hutter *et al.*, 2012]. Thus, to be fair, we re-evaluated all approaches using interleaved runs. Figure 4.2 (bottom) shows that when using $k = 4$ interleaved runs of 500 evaluations each, REMBO and ParamILS performed best, with a slight advantage for REMBO early on in the search.

5 Conclusion

The paper has shown that it is possible to use random embeddings in Bayesian optimization to optimize functions of high extrinsic dimensionality D provided that they have low intrinsic dimensionality d_e. The new algorithm, REMBO, is extremely simple because it only requires a simple modification of the original Bayesian optimization algorithm; namely multiplication by a random matrix. We confirmed REMBO's independence of D empirically by optimizing low-dimensional functions embedded in high dimensions. Finally, we demonstrated that REMBO achieves excellent performance for optimizing the 47 discrete parameters of a popular mixed integer linear programming solver, thereby providing further evidence for the observation (already put forward by Bergstra, Hutter and colleagues) that, for many problems of great practical interest, the number of important dimensions indeed appears to be much lower than their extrinsic dimensionality. Of course, we do not know how many practical optimization problems fall within the class of problems where REMBO applies. With time and the release of the code, we will find more evidence. For the time being, the success achieved in the examples presented in this paper (including the automatic configuration of a very popular solver) is very encouraging.

[1] http://lpsolve.sourceforge.net/

References

[Azimi *et al.*, 2012] J. Azimi, A. Jalali, and X.Z. Fern. Hybrid batch Bayesian optimization. In *ICML*, 2012.

[Bergstra and Bengio, 2012] J. Bergstra and Y. Bengio. Random search for hyper-parameter optimization. *JMLR*, 13:281–305, 2012.

[Bergstra *et al.*, 2011] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *NIPS*, pages 2546–2554, 2011.

[Bergstra *et al.*, 2012] J. Bergstra, D. Yamins, and D. D. Cox. Making a science of model search. *CoRR*, abs/1209.5111, 2012.

[Brochu *et al.*, 2009] E. Brochu, V. M. Cora, and N. de Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Technical Report UBC TR-2009-23 and arXiv:1012.2599v1, Dept. of Computer Science, University of British Columbia, 2009.

[Bull, 2011] A. D. Bull. Convergence rates of efficient global optimization algorithms. *JMLR*, 12:2879–2904, 2011.

[Carpentier and Munos, 2012] A. Carpentier and R. Munos. Bandit theory meets compressed sensing for high dimensional stochastic linear bandit. In *AIStats*, pages 190–198, 2012.

[Chen *et al.*, 2012] B. Chen, R.M. Castro, and A. Krause. Joint optimization and variable selection of high-dimensional Gaussian processes. In *ICML*, 2012.

[de Freitas *et al.*, 2012] N. de Freitas, A. Smola, and M. Zoghi. Exponential regret bounds for Gaussian process bandits with deterministic observations. In *ICML*, 2012.

[Gomes *et al.*, 2008] C. P. Gomes, W.J. van Hoeve, and A. Sabharwal. Connections in networks: A hybrid approach. In *CPAIOR*, volume 5015, pages 303–307, 2008.

[Hansen and Ostermeier, 2001] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[Hoffman *et al.*, 2011] M. Hoffman, E. Brochu, and N. de Freitas. Portfolio allocation for Bayesian optimization. In *UAI*, pages 327–336, 2011.

[Hoos, 2012] H. H. Hoos. Programming by optimization. *Commun. ACM*, 55(2):70–80, 2012.

[Hutter *et al.*, 2010] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Automated configuration of mixed integer programming solvers. In *CPAIOR*, pages 186–202, 2010.

[Hutter *et al.*, 2011] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *LION*, pages 507–523, 2011.

[Hutter *et al.*, 2012] F. Hutter, H. H. Hoos, and K. Leyton-Brown. Parallel algorithm configuration. In *LION*, pages 55–70, 2012.

[Hutter, 2009] F. Hutter. *Automated Configuration of Algorithms for Solving Hard Computational Problems*. PhD thesis, University of British Columbia, Vancouver, Canada, 2009.

[Jones *et al.*, 1993] David R Jones, C D Perttunen, and B E Stuckman. Lipschitzian optimization without the Lipschitz constant. *J. of Optimization Theory and Applications*, 79(1):157–181, 1993.

[Jones *et al.*, 1998] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *J. of Global optimization*, 13(4):455–492, 1998.

[Jones, 2001] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21(4):345–383, 2001.

[Lizotte *et al.*, 2011] D. Lizotte, R. Greiner, and D. Schuurmans. An experimental methodology for response surface optimization methods. *J. of Global Optimization*, pages 1–38, 2011.

[Lizotte, 2008] D. Lizotte. *Practical Bayesian Optimization*. PhD thesis, University of Alberta, Canada, 2008.

[Martinez–Cantin *et al.*, 2009] R. Martinez–Cantin, N. de Freitas, E. Brochu, J. Castellanos, and A. Doucet. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27(2):93–103, 2009.

[Močkus *et al.*, 1999] J. Močkus, A. Močkus, and L. Močkus. *Bayesian approach for randomization of heuristic algorithms of discrete programming*. American Math. Society, 1999.

[Močkus, 1982] J. Močkus. The Bayesian approach to global optimization. In *Systems Modeling and Optimization*, volume 38, pages 473–481. Springer, 1982.

[Močkus, 1994] J. Močkus. Application of Bayesian approach to numerical methods of global and stochastic optimization. *J. of Global Optimization*, 4(4):347–365, 1994.

[Osborne *et al.*, 2009] M. A. Osborne, R. Garnett, and S. J. Roberts. Gaussian processes for global optimisation. In *LION*, 2009.

[Rasmussen and Williams, 2006] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[Sankar *et al.*, 2003] A. Sankar, D.A. Spielman, and S.H. Teng. Smoothed analysis of the condition numbers and growth factors of matrices. *Arxiv preprint cs/0310022*, 2003.

[Snoek *et al.*, 2012] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *NIPS*, 2012.

[Srinivas *et al.*, 2010] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *ICML*, 2010.

[Wang and de Freitas, 2011] Z. Wang and N. de Freitas. Predictive adaptation of hybrid Monte Carlo with Bayesian parametric bandits. In *NIPS Deep Learning and Unsupervised Feature Learning Workshop*, 2011.