Journal of Artificial Intelligence Research 55 (2016) 361-387 Submitted 04/15; published 02/16

Bayesian Optimization in a Billion Dimensions via Random Embeddings

Ziyu Wang ZIYU.WANG@CS.OX.AC.UK
Department of Computer Science, University of Oxford

Frank Hutter FH@CS.UNI-FREIBURG.DE
Department of Computer Science, University of Freiburg

Masrour Zoghi M.ZOGHI@UVA.NL
Department of Computer Science, University of Amsterdam

David Matheson DAVIDM@CS.UBC.CA
Department of Computer Science, University of British Columbia

Nando de Freitas NANDO@CS.OX.AC.UK
Department of Computer Science, University of Oxford
Canadian Institute for Advanced Research

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, has important invariance properties, and applies to domains with both categorical and continuous variables. We present a thorough theoretical analysis of REMBO. Empirical results confirm that REMBO can effectively solve problems with billions of dimensions, provided the intrinsic dimensionality is low. They also show that REMBO achieves state-of-the-art performance in optimizing the 47 discrete parameters of a 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.

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

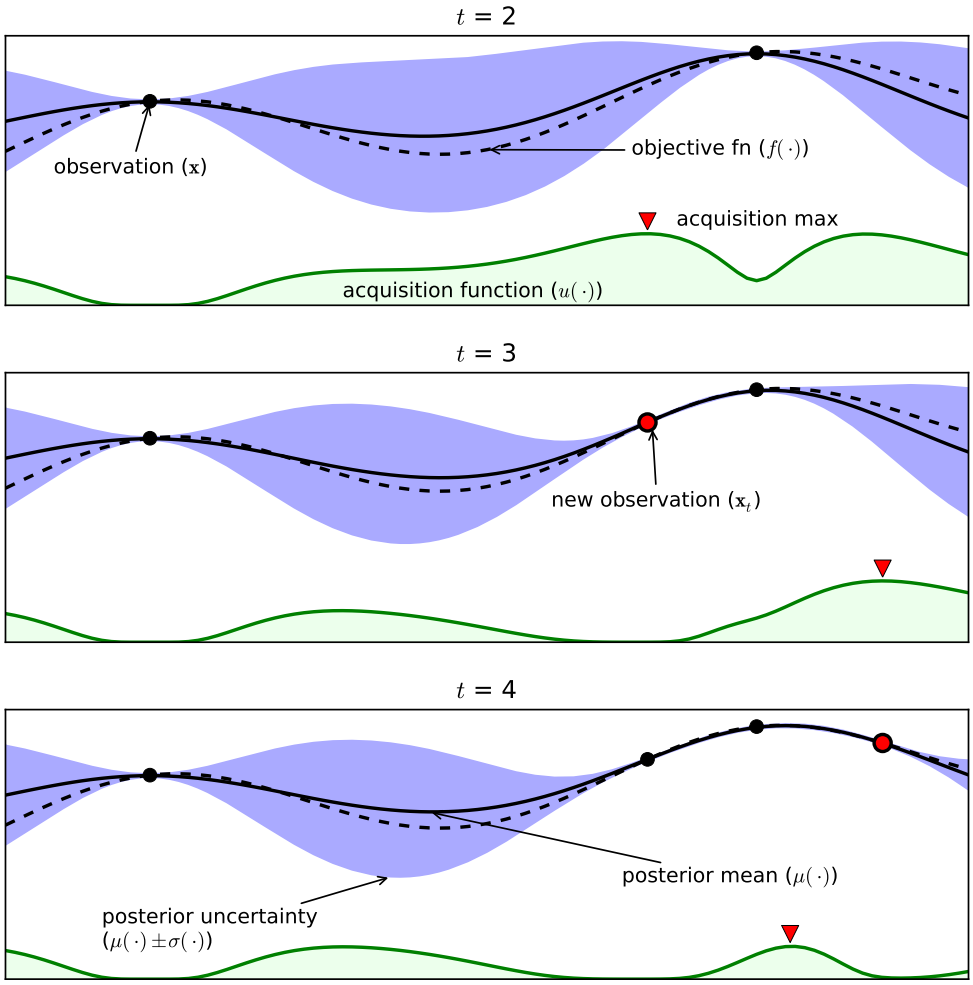

Figure 1: Three consecutive iterations of Bayesian optimization for a toy one-dimensional problem. The unknown objective function is approximated with at Gaussian process (GP) at each iteration. The figure shows the mean and confidence intervals for this process. It also shows the acquisition function in the lower green shaded plots. The acquisition is high where the GP predicts a high objective (exploitation) and where the prediction uncertainty is high (exploration). Note that the area on the far left remains under-sampled, as (despite having high uncertainty) it is correctly predicted to be unlikely to improve over the highest observation.

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 $\langle x, f(x) \rangle$. Step 1 uses a so-called *acquisition function* that quantifies the expected value of learning the value of $f(x)$ for each $x \in \mathcal{X}$. This procedure is illustrated in Figure 1.

The role of the acquisition function is to trade off exploration and exploitation; popular choices include Thompson sampling (Thompson, 1933; Hoffman, Shahriari, & de Freitas, 2014), probability of improvement (Jones, 2001), expected improvement (Močkus, 1994), upper-confidence-bounds (Srinivas, Krause, Kakade, & Seeger, 2010), and online portfolios of these (Hoffman, Brochu, & de Freitas, 2011). These are typically optimized by choosing points where the predictive mean is high (exploitation) and where the variance is large (exploration). Since they typically have an analytical expression that is easy to evaluate, they are much easier to optimize than the original objective function, using off-the-shelf numerical optimization algorithms.[1]

The term *Bayesian optimization* was coined several decades ago by Jonas Močkus (1982). A popular version of the method is known as *efficient global optimization* in the experimental design literature since the 1990s (Jones, Schonlau, & Welch, 1998). Often, the approximation of the objective function is obtained using Gaussian process (GP) priors. For this reason, the technique is also referred to as *GP bandits* (Srinivas et al., 2010). However, many other approximations of the objective have been proposed, including Parzen estimators (Bergstra, Bardenet, Bengio, & Kégl, 2011), Bayesian parametric models (Wang & de Freitas, 2011), treed GPs (Gramacy, Lee, & Macready, 2004) and random forests (Brochu, Cora, & de Freitas, 2009; Hutter, 2009; Hutter, Hoos, & Leyton-Brown, 2011). These may be more suitable than GPs when the number of iterations grows without bound, or when the objective function is believed to have discontinuities. We also note that often assumptions on the smoothness of the objective function are encoded without use of the Bayesian paradigm, while leading to similar algorithms and theoretical guarantees (see, for example, Bubeck, Munos, Stoltz, & Szepesvari, 2011, and the references therein). There is a rich literature on Bayesian optimization, and for further details we refer readers to more tutorial treatments (Brochu et al., 2009; Jones et al., 1998; Jones, 2001; Lizotte, Greiner, & Schuurmans, 2011; Močkus, 1994; Osborne, Garnett, & Roberts, 2009) and recent theoretical results (Srinivas et al., 2010; Bull, 2011; de Freitas, Smola, & Zoghi, 2012).

Bayesian optimization has been demonstrated to outperform other state-of-the-art black-box optimization techniques when function evaluations are expensive and the number of allowed function evaluations is therefore low (Hutter, Hoos, & Leyton-Brown, 2013). In recent years, it has found increasing use in the machine learning community (Rasmussen, 2003; Brochu, de Freitas, & Ghosh, 2007; Martinez-Cantin, de Freitas, Doucet, & Castellanos, 2007; Lizotte, Wang, Bowling, & Schuurmans, 2007; Frazier, Powell, & Dayanik, 2009; Azimi, Fern, & Fern, 2010; Hamze, Wang, & de Freitas, 2013; Azimi, Fern, & Fern, 2011; Hutter et al., 2011; Bergstra et al., 2011; Gramacy & Polson, 2011; Denil, Bazzani, Larochelle, & de Freitas, 2012; Mahendran, Wang, Hamze, & de Freitas, 2012; Azimi, Jalali, & Fern, 2012; Hennig & Schuler, 2012; Marchant & Ramos, 2012; Snoek, Larochelle, & Adams, 2012; Swersky, Snoek, & Adams, 2013; Thornton, Hutter, Hoos, & Leyton-Brown,

1. This optimization step can in fact be circumvented when using treed multi-scale optimistic optimization as recently demonstrated by Wang and de Freitas (2014). There also exist several more involved Bayesian non-linear experimental design approaches for constructing the acquisition function, where the utility to be optimized involves an entropy of an aspect of the posterior. This includes the work of Hennig and Schuler (2012) for finding maxima of functions, the works of Kueck, de Freitas, and Doucet (2006) and Kueck, Hoffman, Doucet, and de Freitas (2009) for learning functions, and the work of Hoffman, Kueck, de Freitas, and Doucet (2009) for estimating Markov decision processes. These works rely on expensive approximate inference methods for computing intractable integrals.

2013). Despite many success stories, the approach is restricted to problems of moderate dimension, typically up to about 10. 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. This is the goal of this paper.

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. As a result, there has been little progress on this challenging problem, with a few exceptions. Bergstra et al. (2011) introduced a non-standard Bayesian optimization method based on a tree of one-dimensional density estimators and applied it successfully to optimize the 238 parameters of a complex vision architecture (Bergstra, Yamins, & Cox, 2013). Hutter et al. (2011) used random forests models in Bayesian optimization to achieve state-of-the-art performance in optimizing up to 76 mixed discrete/continuous parameters of algorithms for solving hard combinatorial problems, and to successfully carry out combined model selection and hyperparameter optimization for the 768 parameters of the Auto-WEKA framework (Thornton et al., 2013). Eggensperger, Feurer, Hutter, Bergstra, Snoek, Hoos, and Leyton-Brown (2013) showed that these two methods indeed yielded the best performance for high-dimensional hyperparameter optimization (e.g., in deep belief networks). However, both are based on weak uncertainty estimates that can fail even for the optimization of very simple functions and lack theoretical guarantees.

In the *linear* bandits case, Carpentier and Munos (2012) recently proposed a compressed sensing strategy to attack problems with a high degree of sparsity. Also recently, Chen, Castro, and Krause (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 and colleagues provide empirical results only for synthetic examples (of up to 400 dimensions), but they provide key theoretical guarantees.

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 & Bengio, 2012), as well as other machine learning algorithms and various state-of-the-art algorithms for solving $\mathcal{NP}$-hard problems (Hutter, Hoos, & Leyton-Brown, 2014). 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 in a new Bayesian optimization variant based on random embeddings.

Figure 2 illustrates the idea behind random embeddings 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.

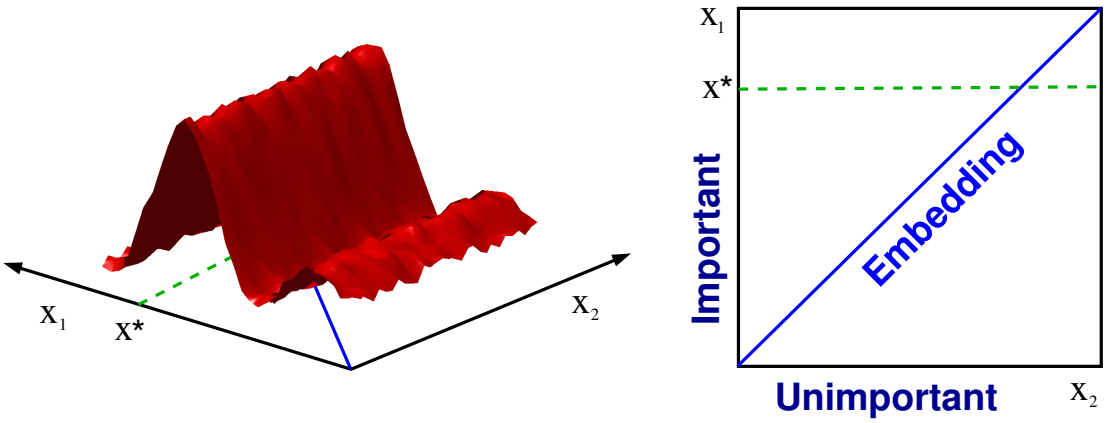

Figure 2: This function in D=2 dimesions only has d=1 *effective dimension*: the vertical axis indicated with the word important on the right hand side figure. Hence, the 1-dimensional embedding includes the 2-dimensional function's optimizer. It is more efficient to search for the optimum along the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

As we first demonstrated in a recent IJCAI conference paper (Wang, Zoghi, Hutter, Matheson, & de Freitas, 2013), random embeddings enable us to scale Bayesian optimization to arbitrary D provided the objective function has low intrinsic dimensionality. Importantly, the algorithm associated with this idea, which we called REMBO, is not restricted to cases with axis-aligned intrinsic dimensions but applies to any d-dimensional linear subspace. Djolonga, Krause, and Cevher (2013) recently proposed an adaptive, but more expensive, variant of REMBO with theoretical guarantees.

In this journal version of our work, we expand the presentation to provide more details throughout. In particular, we expand our description of the strategy for selecting the boundaries of the low-dimensional space and for setting the kernel length scale parameter; we show by means of an additional application (automatic configuration of random forest body-part classifiers) that the performance of our technique does not collapse when the problem does not have an obvious low effective dimensionality. Our experiments (Section 4) also show that REMBO can solve problems of previously untenable high extrinsic dimensions, and that REMBO can achieve state-of-the-art performance for optimizing the 47 discrete parameters of a popular mixed integer linear programming solver.

2. Bayesian Optimization

As mentioned in the introduction, 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 the book by 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 $\mathbf{x}_i \in \mathbb{R}^D$, 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 (see Definition 7 on page 371), 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 $\mathbf{x}_{1:t}$ 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:

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

For simplicity, we assume that $\mathbf{m}(\mathbf{x}_{1:t}) = \mathbf{0}$ and $m^* = 0$. 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 $\mathcal{D}_t = \{\mathbf{x}_{1:t}, \mathbf{f}_{1:t}\}$, and mean and variance

$$\begin{aligned} \mu(\mathbf{x}^*|\mathcal{D}_t) &= \mathbf{k}(\mathbf{x}^*, \mathbf{x}_{1:t})\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})^{-1}\mathbf{f}_{1:t} \\ \sigma(\mathbf{x}^*|\mathcal{D}_t) &= k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}(\mathbf{x}^*, \mathbf{x}_{1:t})\mathbf{K}(\mathbf{x}_{1:t}, \mathbf{x}_{1:t})^{-1}\mathbf{k}(\mathbf{x}_{1:t}, \mathbf{x}^*). \end{aligned}$$

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 (Močkus, 1982; Vazquez & Bect, 2010; Bull, 2011):

$$u(\mathbf{x}|\mathcal{D}_t) = \mathbb{E}(\max\{0, f_{t+1}(\mathbf{x}) - f(\mathbf{x}^+)\}|\mathcal{D}_t).$$

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, such as DIRECT (Jones, Perttunen, & Stuckman, 1993) and CMA-ES (Hansen & Ostermeier, 2001); it is only based on the GP model of the blackbox function f and does not require additional evaluations of f.

The Bayesian optimization procedure is shown in Algorithm 1.

Algorithm 1 Bayesian Optimization

1: Initialize $\mathcal{D}_0$ as $\emptyset$.
2: **for** $t = 1, 2, \ldots$ **do**
3: Find $\mathbf{x}_{t+1} \in \mathbb{R}^D$ by optimizing the acquisition function u: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x} \in \mathcal{X}} u(\mathbf{x} | \mathcal{D}_t)$.
4: Augment the data $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\mathbf{x}_{t+1}, f(\mathbf{x}_{t+1}))\}$.
5: Update the kernel hyper-parameters.
6: **end for**

3. Random Embedding for Bayesian Optimization

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 $d_e \leq D$, 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 $f(\mathbf{x}_\top + \mathbf{x}_\perp) = f(\mathbf{x}_\top)$, where $\mathcal{T}^\perp$ denotes the orthogonal complement of $\mathcal{T}$; and*
- *d_e is the smallest integer with this property.*

*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. Please refer to the appendix. □

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 Random EMbedding Bayesian Optimization (REMBO) algorithm (see Algorithm 2). REMBO first draws a random embedding (given by $\mathbf{A}$) and then performs Bayesian optimization in this embedded space.

In many practical optimization tasks, the goal is to optimize f over a compact subset $\mathcal{X} \subset \mathbb{R}^D$ (typically a box), and f can often not be evaluated outside of $\mathcal{X}$. Therefore, when REMBO selects a point $\mathbf{y}$ such that $\mathbf{A}\mathbf{y}$ is outside the box $\mathcal{X}$, it projects $\mathbf{A}\mathbf{y}$ onto $\mathcal{X}$ before evaluating f. That is, $g(\mathbf{y}) = f(p_\mathcal{X}(\mathbf{A}\mathbf{y}))$, where $p_\mathcal{X} : \mathbb{R}^D \to \mathbb{R}^D$ is the standard projection operator for our box-constraint: $p_\mathcal{X}(\mathbf{y}) = \arg\min_{\mathbf{z} \in \mathcal{X}} \|\mathbf{z} - \mathbf{y}\|_2$; see Figure 3. We still need to describe how REMBO chooses the bounded region $\mathcal{Y} \subset \mathbb{R}^d$, inside which it performs

Algorithm 2 REMBO: Bayesian Optimization with Random Embedding. Blue text denotes parts that are changed compared to standard Bayesian Optimization.

1: Generate a random matrix $\mathbf{A} \in \mathbb{R}^{D \times d}$
2: Choose the bounded region set $\mathcal{Y} \subset \mathbb{R}^d$
3: Initialize $\mathcal{D}_0$ as $\emptyset$.
4: **for** $t = 1, 2, \ldots$ **do**
5: Find $\mathbf{y}_{t+1} \in \mathbb{R}^d$ by optimizing the acquisition function u: $\mathbf{y}_{t+1} = \arg\max_{\mathbf{y} \in \mathcal{Y}} u(\mathbf{y} | \mathcal{D}_t)$.
6: Augment the data $\mathcal{D}_{t+1} = \mathcal{D}_t \cup \{(\mathbf{y}_{t+1}, f(\mathbf{A}\mathbf{y}_{t+1}))\}$.
7: Update the kernel hyper-parameters.
8: **end for**

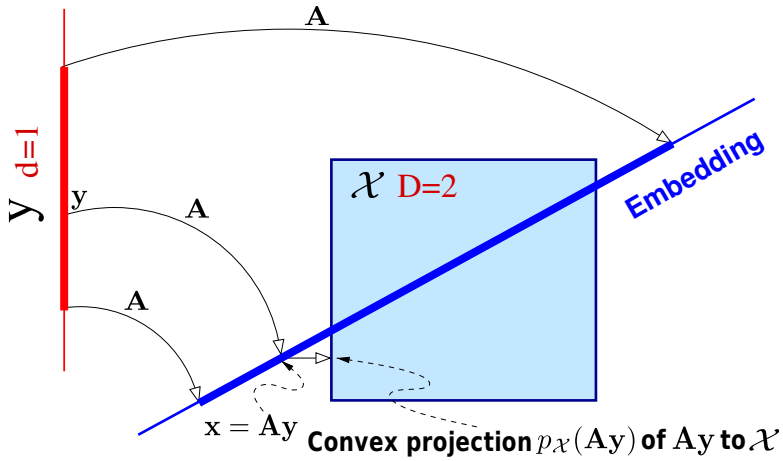

Figure 3: Embedding from $d = 1$ into $D = 2$. The box illustrates the 2D constrained space $\mathcal{X}$, while the thicker red line illustrates the 1D constrained space $\mathcal{Y}$. Note that if $\mathbf{A}\mathbf{y}$ is outside $\mathcal{X}$, it is projected onto $\mathcal{X}$. The set $\mathcal{Y}$ 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.

Bayesian optimization. This is important because REMBO's 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. 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.

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 $\mathbf{0}$. Suppose further that the effective subspace $\mathcal{T}$ of f is such that $\mathcal{T}$ is the span of d_e basis vectors, and 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. Please refer to the appendix. □

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

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

Hence, to choose $\mathcal{Y}$, we must ensure that the ball of radius d_e/ϵ, centred at the origin, lies inside $\mathcal{Y}$.

In practice, we have found that it is very unlikely that the optimizer falls on the corner of the box constraint, implying that $\|x_\top^\star\| < \sqrt{d_e}$. Thus setting $\mathcal{Y}$ too big may be unnecessarily wasteful. To improve our understanding of this effect, we developed a simulation study, in which we drew random Gaussian matrices, used them to map various potential optimizers $\mathbf{x}_\top^\star$ to their corresponding points $\mathbf{y}_\top^\star \in \mathcal{Y}$, and studied the norms of $\mathbf{y}_\top^\star$.

Assume for simplicity of presentation that $\mathcal{Y}$ is axis-aligned and d_e-dimensional (the argument applies when $d > d_e$). The section of the random matrix $\mathbf{A}$ that maps points in $\mathcal{Y}$ to $\mathcal{T}$ is a random Gaussian matrix of dimension $d_e \times d_e$. Let us call this section of the matrix $\mathbf{B}$. Since random Gaussian matrices are rotationally invariant in distribution, we have for any orthonormal matrix $\mathbf{O}$ and a random Gaussian matrix $\mathbf{B}$, $\mathbf{OB} \overset{d}{=} \mathbf{B}$. That is, $\mathbf{OB}$ and $\mathbf{B}$ are equal in distribution. Similarly, for $\mathbf{B}^{-1}$, $\mathbf{OB}^{-1} = \left(\mathbf{BO}^T\right)^{-1} \overset{d}{=} \mathbf{B}^{-1}$. Therefore, $\mathbf{B}^{-1}$ is also rotationally invariant. Hence, $\|\mathbf{B}^{-1}\mathbf{x}_\top\|_\infty \overset{d}{=} \|\mathbf{B}^{-1}\mathbf{x}'_\top\|_\infty$ as long as $\|\mathbf{x}_\top\|_2 = \|\mathbf{x}'_\top\|_2$. Following this equivalence for the supremum norm of projected vectors, it suffices to choose a point with the largest norm in $[-1,1]^{d_e}$ in our simulations. We chose $\mathbf{x}_\top = [1,1,\cdots,1]$.

We conducted simulations for several embedding dimensions, $d_e \in \{1,2,\cdots,50\}$, by drawing 10000 random Gaussian matrices and computing $\|\mathbf{B}^{-1}\mathbf{x}\|_\infty$. We found that with empirical probability above $1-\epsilon$ (for decreasing values of ϵ), it was the case that

$$\|\mathbf{B}^{-1}\mathbf{x}\|_\infty < \frac{1}{\epsilon}\max\{\log(d_e), 1\}.$$

These simulations indicate that we could set $\mathcal{Y} = \left[-\frac{1}{\epsilon}\max\{\log(d_e),1\}, \frac{1}{\epsilon}\max\{\log(d_e),1\}\right]^{d_e}$. We did this in our experiments and in particular chose $\epsilon = \log(d)/\sqrt{d}$, so that $\mathcal{Y}$ was $[-\sqrt{d}, \sqrt{d}]^d$. Note that Theorem 3 is not useful for this choice, which suggests that there is room to improve this aspect of our theory.

Some careful readers may wonder about the effect of the extrinsic dimensionality D. In the following theorem, we show that given the same intrinsic dimensions, the extrinsic dimensionality does not have an effect at all; in other words, REMBO is invariant to the addition of unimportant dimensions.

Theorem 4 (Invariance to addition of unimportant dimensions)**.** *Let $f : \mathbb{R}^{d_e} \to \mathbb{R}$ and for any $D \in \mathbb{N}$, $D \geq d_e$, define $f_D : \mathbb{R}^D \to \mathbb{R}$ such that f_D adds $D - d_e$ truly unimportant dimensions to f: $f_D(\mathbf{z}) = f(\mathbf{z}_{1:d_e})$. Let $\mathbf{A}_1 \in \mathbb{R}^{D_1 \times d}$ and $\mathbf{A}_0 \in \mathbb{R}^{(D_2 - D_1)\times d}$ be random Gaussian matrices with $D_2 \geq D_1 \geq d$ and let $\mathbf{A}_2 = \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_0\end{bmatrix}$. Then, REMBO run using the*

same dimension $d \geq d_e$ and bounded region $\mathcal{Y}$ yields exactly the same function values when run with $\mathbf{A}_1$ on f_{D_1} as when run with $\mathbf{A}_2$ on f_{D_2}.

Proof. We only need to show that for each $\mathbf{y} \in \mathbb{R}^d$, we have $f_{D_1}(\mathbf{A}_1\mathbf{y}) = f_{D_2}(\mathbf{A}_2\mathbf{y})$ since this step of REMBO (line 6 of Algorithm 2) is the only one that differs between the two algorithm runs. When this function evaluation step yields the same results for every $\mathbf{y} \in \mathbb{R}^d$, then the two REMBO runs behave identically since the algorithm is otherwise identical and deterministic after the selection of $\mathbf{A}$ in Step 1. Since $\mathbf{A}_2 = \begin{bmatrix}\mathbf{A}_1 \\ \mathbf{A}_0\end{bmatrix}$, we have $\mathbf{A}_2\mathbf{y} = \begin{bmatrix}\mathbf{A}_1\mathbf{y} \\ \mathbf{A}_0\mathbf{y}\end{bmatrix}$. Since $D_2 \geq D_1 \geq d_e$, the first d_e entries of this $D_2 \times 1$ vector $\mathbf{A}_2\mathbf{y}$ are the first d_e entries of $\mathbf{A}_1\mathbf{y}$. We thus have $f_{D_1}(\mathbf{A}_1\mathbf{y}) = f([\mathbf{A}_1\mathbf{y}]_{1:d_e}) = f([\mathbf{A}_2\mathbf{y}]_{1:d_e}) = f_{D_2}(\mathbf{A}_2\mathbf{y})$. □

Finally, we show that REMBO is also invariant to rotations in the sense that given different rotation matrices, running REMBO would result in the same distributions of observed function values. The argument is made concise in the following results.

Lemma 5. *Consider function $f : \mathbb{R}^D \to \mathbb{R}$. Let $f_{\mathbf{R}} : \mathbb{R}^D \to \mathbb{R}$ be such that $f_{\mathbf{R}}(\mathbf{x}) = f(\mathbf{R}\mathbf{x})$ for some orthonormal matrix $\mathbf{R} \in \mathbb{R}^{D\times D}$. Then, REMBO run in bounded region $\mathcal{Y}$ yields exactly the same sequence of function values when run with $\mathbf{A}$ on f as when run with $\mathbf{R}^{-1}\mathbf{A}$ on $f_{\mathbf{R}}$ for a matrix $\mathbf{A} \in \mathbb{R}^{D\times d}$.*

Proof. REMBO uses f and $\mathbf{A}$ (resp. $f_{\mathbf{R}}$ and $\mathbf{R}^{-1}\mathbf{A}$) only in one spot (in line 6). Thus, the proof is trivial by showing that $f(\mathbf{A}\mathbf{y}_{t+1}) = f_{\mathbf{R}}(\mathbf{R}^{-1}\mathbf{A}\mathbf{y}_{t+1})$ through simple algebra:

$$f_{\mathbf{R}}(\mathbf{R}^{-1}\mathbf{A}\mathbf{y}_{t+1}) = f(\mathbf{R}\mathbf{R}^{-1}\mathbf{A}\mathbf{y}_{t+1}) = f(\mathbf{A}\mathbf{y}_{t+1}).$$

□

Theorem 6 (Invariance to rotations)**.** *Consider function $f : \mathbb{R}^D \to \mathbb{R}$. Let $f_{\mathbf{R}} : \mathbb{R}^D \to \mathbb{R}$ be such that $f_{\mathbf{R}}(\mathbf{x}) = f(\mathbf{R}\mathbf{x})$ for some orthonormal matrix $\mathbf{R} \in \mathbb{R}^{D\times D}$. Then, given random Gaussian matrices $\mathbf{A}_1 \in \mathbb{R}^{D\times d}$ and $\mathbf{A}_2 \in \mathbb{R}^{D\times d}$, REMBO run in bounded region $\mathcal{Y}$ yields in distribution the same sequence of function values when run with $\mathbf{A}_1$ on f as when run with $\mathbf{A}_2$ on $f_{\mathbf{R}}$.*

Proof. Since $\mathbf{R}$ is orthonormal, we have $\mathbf{R}^{-1}\mathbf{A}_1 \overset{d}{=} \mathbf{A}_2$. Therefore, REMBO run in bounded region $\mathcal{Y}$ yields in distribution the same sequence of function values when run with $\mathbf{R}^{-1}\mathbf{A}_1$ on $f_{\mathbf{R}}$ as when run with $\mathbf{A}_2$ on $f_{\mathbf{R}}$. We have also by Lemma 5 that REMBO run in bounded region $\mathcal{Y}$ yields exactly the same sequence of function values when run with $\mathbf{A}_1$ on f as when run with $\mathbf{R}^{-1}\mathbf{A}_1$ on $f_{\mathbf{R}}$. The conclusion follows from combining the previous arguments. □

3.1 Increasing the Success Rate of REMBO

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 compute clusters.

Another way of increasing REMBO's success rate is to increase the dimensionality d it uses internally. When $d > d_e$, with probability 1 we have $\binom{d}{d_e}$ different embeddings of dimensionality d_e. That is, we only need to select d_e columns of $\mathbf{A} \in \mathbb{R}^{D \times d}$ to represent the d_e relevant dimensions of $\mathbf{x}$. The algorithm can achieve this by setting the remaining $d - d_e$ sub-components of the d-dimensional vector $\mathbf{y}$ to zero. Informally, since we have more embeddings, it is more likely that one of these will include the optimizer. In our experiments, we will assess the merits and shortcomings of these two strategies.

3.2 Choice of Kernel

Since REMBO uses GP-based Bayesian optimization to search in the region $\mathcal{Y} \subset \mathbb{R}^d$, we need to define a kernel between two points $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \mathcal{Y}$. We begin with the standard definition of the squared exponential kernel:

Definition 7. *Let $K_{SE}(\mathbf{y}) = \exp(-\|\mathbf{y}\|^2/2)$. Given a length scale $\ell > 0$, we define the corresponding squared exponential kernel as*

$$k_\ell^d(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = K_{SE}\left(\frac{\mathbf{y}^{(1)} - \mathbf{y}^{(2)}}{\ell}\right)$$

It is possible to work with two variants of this kernel. First, we can use $k_\ell^d(\mathbf{y}^1, \mathbf{y}^2)$ as in Definition 7. We refer to this kernel as the low-dimensional kernel. We can also adopt an implicitly defined high-dimensional kernel on $\mathcal{X}$:

$$k_\ell^D(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = K_{SE}\left(\frac{p_\mathcal{X}(\mathbf{A}\mathbf{y}^{(1)}) - p_\mathcal{X}(\mathbf{A}\mathbf{y}^{(2)})}{\ell}\right),$$

where $p_\mathcal{X} : \mathbb{R}^D \to \mathbb{R}^D$ is the projection operator for our box-constraint as above (see Figure 3).

Note that when using this 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.

Both kernel choices have strengths and weaknesses. The low-dimensional kernel has the benefit of only requiring the construction of a GP in the space of intrinsic dimensionality d, whereas the high-dimensional kernel requires the GP to be constructed in a space of extrinsic dimensionality D. However, the low-dimensional kernel may waste time exploring in the region of the embedding outside of $\mathcal{X}$ (see Figure 2) because two points far apart in this region may be projected via $p_\mathcal{X}$ to nearby points on the boundary of $\mathcal{X}$. The high-dimensional kernel is not affected by this problem because the search is conducted directly on $\{p_\mathcal{X}(\mathbf{A}\mathbf{y}) : \mathbf{y} \in \mathcal{Y}\}$ with distances calculated in $\mathcal{X}$ and not in $\mathcal{Y}$.

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

Algorithm 3 Bayesian Optimization with Hyper-parameter Optimization.

input Threshold t_σ.
input Upper and lower bounds $U > L > 0$ for hyper-parameter.
input Initial length scale hyper-parameter $\ell \in [L, U]$.
Initialize $C = 0$
for $t = 1, 2, \ldots$ **do**
  Find $\mathbf{x}_{t+1}$ by optimizing the acquisition function u: $\mathbf{x}_{t+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} u(\mathbf{x}|\mathcal{D}_t)$.
  if $\sqrt{\sigma^2(\mathbf{x}_{t+1})} < t_\sigma$ **then**
    $C = C + 1$
  else
    $C = 0$
  end if
  Augment the data $\mathcal{D}_{t+1} = \{\mathcal{D}_t, (\mathbf{x}_{t+1}, f(\mathbf{x}_{t+1}))\}$
  if $t \mod 20 = 0$ **or** $C = 5$ **then**
    if $C = 5$ **then**
      $U = \max\{0.9\ell, L\}$
      $C = 0$
    end if
    Learn the hyper-parameter by optimizing the log marginal likelihood by using DIRECT and CMA-ES: $\ell = \arg\max_{l\in[L,U]} \log p(\mathbf{f}_{1:t+1}|\mathbf{x}_{1:t+1}, l)$
  end if
end for

problems that contain discrete choices. We define our kernel for categorical variables as:

$$k_\lambda^D(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = \exp\left(-\frac{\lambda}{2} h(s(\mathbf{A}\mathbf{y}^{(1)}), s(\mathbf{A}\mathbf{y}^{(2)}))^2\right),$$

where $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \mathcal{Y} \subset \mathbb{R}^d$, the function s maps continuous d-dimensional vectors to discrete D-dimensional vectors, and h defines the distance between two discrete vectors. In more detail, $s(\mathbf{x})$ first uses $p_\mathcal{X}$ to project $\mathbf{x}$ to $\bar{\mathbf{x}} \in [-1, 1]^D$. For each dimension $\bar{x}_i$ of $\bar{\mathbf{x}}$, s then maps $\bar{x}_i$ to a discrete value by scaling and rounding. In our experiments, following Hutter (2009), we defined $h(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = |\{i : x_i^{(1)} \neq x_i^{(2)}\}|$ 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 Hamming distance between their mappings in the high-dimensional space.

3.3 Hyper-parameter Optimization

For Bayesian optimization (and therefore REMBO), it is difficult to manually estimate the true length scale hyper-parameter of a problem at hand. To avoid any manual steps and to achieve robust performance across diverse sets of objective functions, in this paper we adopted an adaptive hyper-parameter optimization scheme. The length scale of GPs is often set by maximizing marginal likelihood (Rasmussen & Williams, 2006; Jones et al., 1998). However, as demonstrated by Bull (2011), this approach, when implemented naively, may not guarantee convergence. This is not only true of approaches that maximize the marginal

likelihood, but also of approaches that rely on Monte Carlo sampling from the posterior distribution (Brochu, Brochu, & de Freitas, 2010; Snoek et al., 2012) when the number of data is very small, unless the prior is very informative.

Here, we propose to optimize the length scale parameter ℓ by maximizing the marginal likelihood subject to an upper bound U which is decreased when the algorithm starts exploiting too much. Full details are given in Algorithm 3. We say that the algorithm is exploiting when the standard deviation at the maximizer of the acquisition function $\sqrt{\sigma(\mathbf{x}_{t+1})}$ is less than some threshold t_σ for 5 consecutive iterations. Intuitively, this means that the algorithm did not emphasize exploration (searching in new parts of the space, where the predictive uncertainty is high) for 5 consecutive iterations. When this criterion is met, the algorithm decreases its upper bound U multiplicatively and re-optimizes the hyper-parameter subject to the new bound. Even when the criterion is not met the hyper-parameter is re-optimized every 20 iterations. For each optimization of the acquisition function, the algorithm runs both DIRECT (Jones et al., 1993) and CMA-ES (Hansen & Ostermeier, 2001) and uses the result of the best of the two options. The astute reader may wonder about the difficulty of optimizing the acquisition functions. For REMBO, however, we have not found the optimization of the acquisition function to be a problem since we only need to optimize it in the low-dimensional space and our acquisition function evaluations are cheap, allowing us tens of thousands of evaluations in seconds that (empirically) suffice to cover the low-dimensional space well.

The motivation of this algorithm is to rather err on the side of having too small a length scale: given a squared exponential kernel k_ℓ, with a smaller length scale than another kernel k, one can show that any function f in the RKHS characterized by k is also an element of the RKHS characterized by k_ℓ. Thus, when running expected improvement, one can safely use k_ℓ instead of k as the kernel of the GP and still preserve convergence (Bull, 2011). We argue that (with a small enough lower bound L) the algorithm would eventually reduce the upper bound enough to allow convergence. Also, the algorithm would not explore indefinitely as L is required to be positive. In our experiments, we set the initial constraint $[L, U]$ to be $[0.01, 50]$ and set $t_\sigma = 0.002$.

We want to stress the fact that the above argument is only known to hold for a class of kernels over continuous domains (e.g. squared exponential and Matérn class kernels). Although we believe that a similar argument could be made for integer and categorical kernels, rigorous arguments concerning convergence under these kernels remain a challenge in Bayesian optimization.

4. Experiments

We now study REMBO empirically. We first use synthetic functions of small intrinsic dimensionality $d_e = 2$ but extrinsic dimension D up to 1 billion to demonstrate REMBO's independence of D. Then, we apply REMBO to automatically optimize the 47 parameters of a widely-used mixed integer linear programming solver and demonstrate that it achieves state-of-the-art performance. However, we also warn against the blind application of REMBO. To illustrate this, we study REMBO's performance for tuning the 14 parameters of a random forest body part classifier used by Kinect. In this application, all the $D = 14$ parameters appear to be important, and while REMBO (based on $d = 3$) finds

reasonable solutions (better than random search and comparable to what domain experts achieve), standard Bayesian optimization can outperform REMBO (and the domain experts) in such moderate-dimensional spaces. More optimistically, this random forest tuning application shows that REMBO does not fail catastrophically when it is not clear that the optimization problem has low effective dimensionality.

4.1 Experimental Setup

For all our experiments, we used a single robust version of REMBO that automatically sets its GP's length scale parameter as described in Section 3.3. The code for REMBO, as well as all data used in our experiments is publicly available at `https://github.com/ziyuw/rembo`.

Some of our experiments required substantial computational resources, with the computational expense of each experiment depending mostly on the cost of evaluating the respective black-box function. While the synthetic experiments in Section 4.2 only required minutes for each run of each method, optimizing the mixed integer programming solver in Section 4.4 required 4-5 hours per run, and optimizing the random forest classifier in Section 4.5 required 4-5 days per run. In total, we used over half a year of CPU time for the experiments in this paper. In the first two experiments, we study the effect of our two methods for increasing REMBO's success rate (see Section 3.1) by running different numbers of independent REMBO runs with different settings of its internal dimensionality d.

4.2 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

$$g(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$$

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.

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

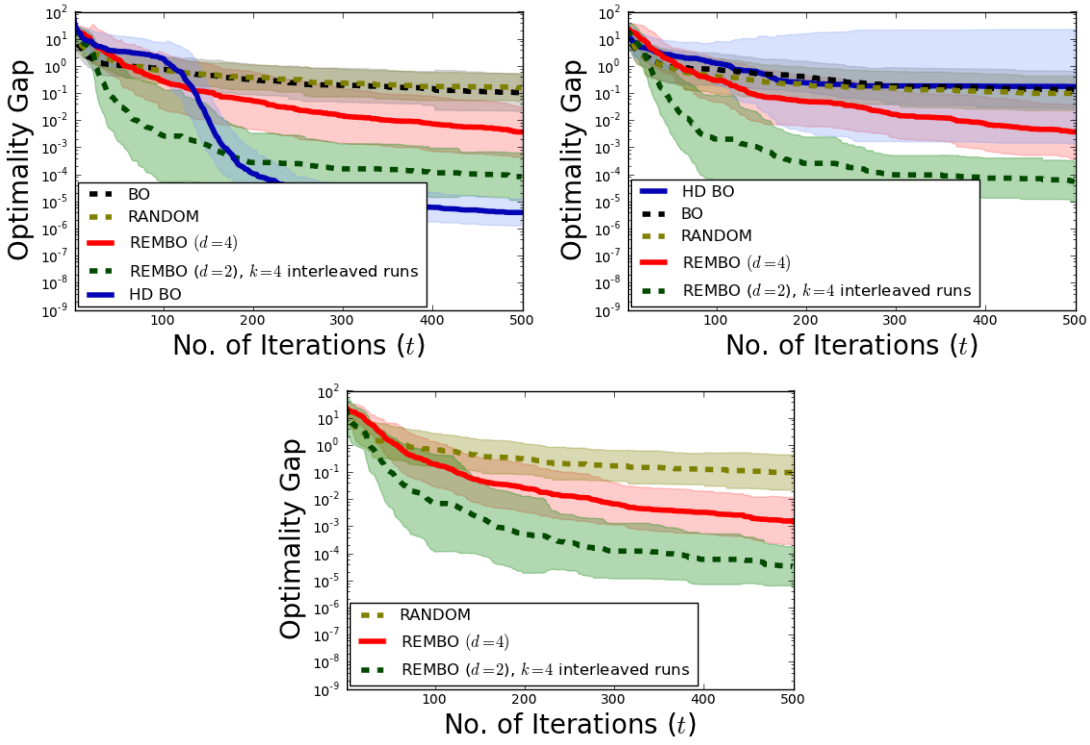

Figure 4: Comparison of random search (RANDOM), Bayesian optimization (BO), method by Chen et al. (2012) (HD BO), and REMBO. Left: $D = 25$ extrinsic dimensions; Right: $D = 25$, with a rotated objective function; Bottom: $D = 10^9$ extrinsic dimensions. We plot means and 1/4 standard deviation confidence intervals of the optimality gap across 50 trials.

box $\mathcal{Y}$ but at the same time slows down REMBO's convergence (such that interleaving several short runs loses 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 4, 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 $f(\mathbf{R}\mathbf{x})$. Figure 4 (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

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.

(see Figure 4, right). To the best of our knowledge, the only other existing method that can be run in such high dimensionality is random search.

For reference, we also evaluated the method of Chen et al. (2012) for these functions, confirming that it does not handle rotation gracefully: while it performed best in the non-rotated case for $D = 25$, it performed worst in the rotated case. It could not be used efficiently for more than $D = 1,000$. Based on a Mann-Whitney U test with Bonferroni multiple-test correction, all performance differences were statistically significant, except Random vs. standard BO. Finally, comparing REMBO to the method of Chen et al. (2012), we also note that REMBO is much simpler to implement and that its results are very reliable (with interleaved runs).

4.3 Synthetic Discrete Experiment

In this section, we test the high-dimensional kernel with a synthetic experiment. Specifically, we again optimize the Branin function, but restrict its domain to 225 discrete points on a regular grid. As above, we added 23 additional irrelevant dimensions to make the problem 25-dimensional in total.

We used a small fixed budget of 100 function evaluations for all algorithms involved as the problem would require no more than 225 evaluations to be solved completely. We used $k = 4$ interleaved runs for REMBO. We again compare REMBO to random search and standard BO. For REMBO, we use the high-dimensional kernel to handle the discrete nature of the problem. The result of the comparison is summarized in Figure 5. Standard BO again suffered from the high extrinsic dimensionality and performed slightly worse than random search. REMBO, on the other hand, performed well in this setting.

4.4 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

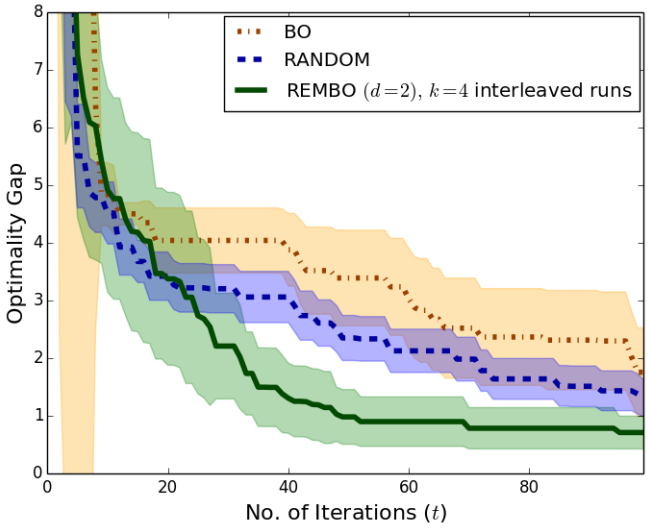

Figure 5: Comparison of random search (RANDOM), Bayesian optimization (BO), and REMBO. $D = 25$ extrinsic dimensions. We plot means and 1/4 standard deviation confidence intervals of the optimality gap across 50 trials.

automated fashion (Močkus, Močkus, & Močkus, 1999; Hutter, Hoos, Leyton-Brown, & Stützle, 2009; Hutter, Hoos, & Leyton-Brown, 2010; Vallati, Fawcett, Gerevini, Hoos, & Saetti, 2011; Bergstra et al., 2011; Wang & de Freitas, 2011). These successes have led to a paradigm shift in algorithm development towards the active design of highly parameterized frameworks that can be automatically customized to particular problem domains using optimization (Hoos, 2012; Bergstra et al., 2013; Thornton et al., 2013). The resulting algorithm configuration problems have been shown to have low dimensionality (Hutter et al., 2014), and 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 (Berkelaar, Eikland, & Notebaert, 2016) , a popular mixed integer programming (MIP) 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 MIP encoding of a wildlife corridor problem from computational sustainability (Gomes, van Hoeve, & Sabharwal, 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 3.2. While we have not proven any theoretical guarantees for discrete optimization

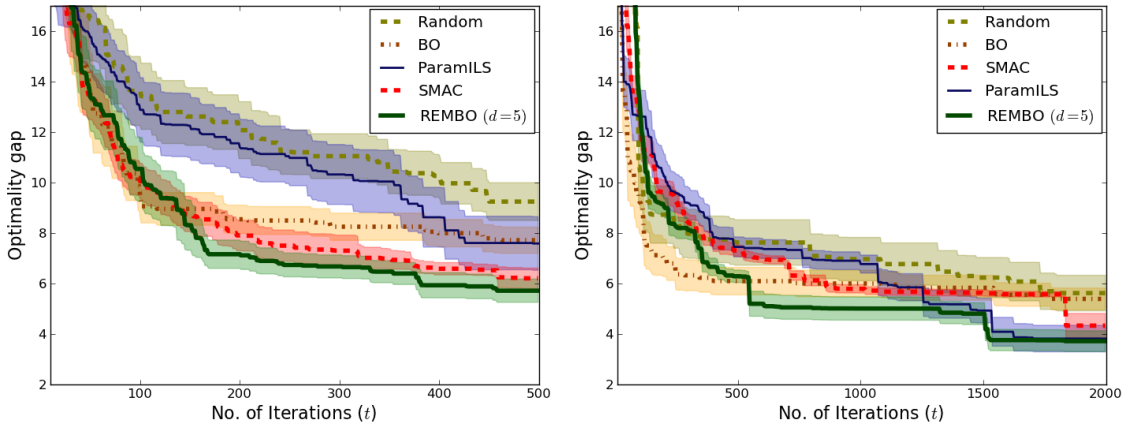

Figure 6: Performance of various methods for configuration of `lpsolve`; we show the optimality gap `lpsolve` achieved with the configurations found by the various methods (lower is better). Left: a single run of each method; Right: performance with $k = 4$ interleaved runs.

problems, REMBO appears to effectively exploit the low effective dimensionality of at least this particular optimization problem.

Figure 6 (left) compares BO, REMBO, and the baseline random search against ParamILS (Hutter et al., 2009) and SMAC (Hutter et al., 2011). ParamILS and SMAC were specifically designed for the configuration of algorithms with many discrete parameters and define the current state of the art for this problem. Nevertheless, here SMAC and our vanilla REMBO method performed best. Based on a Mann-Whitney U test with Bonferroni multiple-test correction, they both yielded statistically significantly better results than both Random and standard BO; no other performance differences were significant. The figure only shows REMBO with $d = 5$ to avoid clutter, but we did not optimize this parameter; the only other value we tried ($d = 3$) resulted in indistinguishable .

As in the synthetic experiment, REMBO's performance could be further improved by using multiple interleaved runs. However, as shown by Hutter, Hoos, and Leyton-Brown (2012), multiple independent runs can also improve the performance of SMAC and especially ParamILS. Thus, to be fair, we re-evaluated all approaches using interleaved runs. Figure 6 (right) shows that ParamILS and REMBO benefitted most from interleaving $k = 4$ runs. However, the statistical test results did not change, still showing that SMAC and REMBO outperformed Random and BO, with no other significant performance differences.

4.5 Automatic Configuration of Random Forest Kinect Body Part Classifier

We now evaluate REMBO's performance for optimizing the 14 parameters of a random forest body part classifier. This classifier closely follows the proprietary system used in the Microsoft Kinect (Shotton, Fitzgibbon, Cook, Sharp, Finocchio, Moore, Kipman, & Blake, 2011) and is available at `https://github.com/david-matheson/rftk`.

Figure 7: Left: ground truth depth, ground truth body parts and predicted body parts; Right: features specified by offsets u and v.

We begin by describing some details of the dataset and classifier in order to build intuition for the objective function and the parameters being optimized. The data we used consists of pairs of depth images and ground truth body part labels. Specifically, we used 1 500 pairs of 320x240 resolution depth and body part images, each of which was synthesized from a random pose of the CMU mocap dataset. Depth, ground truth body parts and predicted body parts (as predicted by the classifier described below) are visualized for one pose in Figure 7 (left). There are 19 body parts plus one background class. For each of these 20 possible labels, the training data contained 25 000 pixels, randomly selected from 500 training images. Both validation and test data contained *all* pixels in the 500 validation and test images, respectively.

The random forest classifier is applied to one pixel P at a time. At each node of each of its decision trees, it computes the depth difference between two pixels described by offsets from P and compares this to a threshold. At training time, many possible pairs of offsets are generated at random, and the pair yielding highest information gain for the training data points is selected. Figure 7 (right) visualizes a potential feature for the pixel in the green box: it computes the depth difference between the pixels in the red box and the white box, specified by respective offsets u and v. At training time, u and v are drawn from two independent 2-dimensional Gaussian distributions, each of which is parameterized by its two mean parameters μ_1 and μ_2 and three covariance terms Σ_{11}, Σ_{12}, and Σ_{22} ($\Sigma_{21} = \Sigma_{12}$ because of symmetry). These constitute 10 of the parameters that need to be optimized, with range [-50,50] for the mean components and [1, 200] for the covariance terms. Low covariance terms yield local features, while high terms yield global features. Next to these ten parameters, the random forest classifier has four other standard parameters, outlined in Table 2. It is well known in computer vision that many of the parameters described here are important. Much research has been devoted to identifying their best values, but results are dataset specific, without definitive general answers.

The objective in optimizing these RF classifier parameters is to find a parameter setting that learns the best classifier in a given time budget of five minutes. To enable competitive performance in this short amount of time, at each node of the tree only a random subset of

Table 2: Parameter ranges for random forest classifier. For the purpose of optimization, the maximum tree depth and the number of potential offsets were transformed to log space.

Parameter	**Range**
Max. tree depth	[1 60]
Min. No. samples for non leaf nodes	[1 100]
No. potential offsets to evaluate	[1 5000]
Bootstrap for per tree sampling	[T F]

data points is considered. Also note that the above parameters do not include the number of trees T in the random forest; since performance improves monotonically in T, we created as many trees as possible in the time budget. Trees are constructed depth first and returned in their current state when the time budget is exceeded. Using a fixed budget results in a subtle optimization problem because of the complex interactions between the various parameters (maximum depth, number of potential offsets, number of trees and accuracy).

It is unclear a priori whether a low-dimensional subspace of these 14 interacting parameters exists that captures the classification accuracy of the resulting random forests. We performed large-scale computational experiments with REMBO, random search, and standard Bayesian optimization (BO) to study this question. In this experiment, we used the high-dimensional kernel for REMBO to avoid the potential over-exploration problems of the low-dimensional kernel described in Section 3.2. We believed that $D = 14$ dimensions would be small enough to avoid inefficiencies in fitting the GP in D dimensions. This belief was confirmed by the observation that standard BO (which operates in $D = 14$ dimensions) performed well for this problem.

Figure 8 (left) shows the results that can be obtained by a single run of random search, BO, and REMBO. Remarkably, REMBO clearly outperformed random search, even based on as few as $d = 3$ dimensions.[2] However, since the extrinsic dimensionality was "only" a moderate $D = 14$, standard Bayesian optimization performed well, and since it was not limited to a low-dimensional subspace it outperformed REMBO. Nevertheless, several REMBO runs actually performed very well, comparably with the best runs of BO. Consequently, when running $k = 4$ interleaved runs of each method, REMBO performed almost as well as BO, matching its performance up to about 450 function evaluations (see Figure 8, right).

We conclude that the parameter space of this RF classifier does not appear to have a clear low effective dimensionality; since the extrinsic dimensionality is only moderate, this leads REMBO to perform somewhat worse than standard Bayesian optimization, but it is still possible to achieve reasonable performance based on as little as $d = 3$ dimensions.

2. Due to the large computational expense of this experiment (in total over half a year of CPU time), we only performed conclusive experiments with $d = 3$; preliminary runs of REMBO with $d = 4$ performed somewhat worse than those with $d = 3$ for a budget of 200 function evaluations, but were still improving at that point.

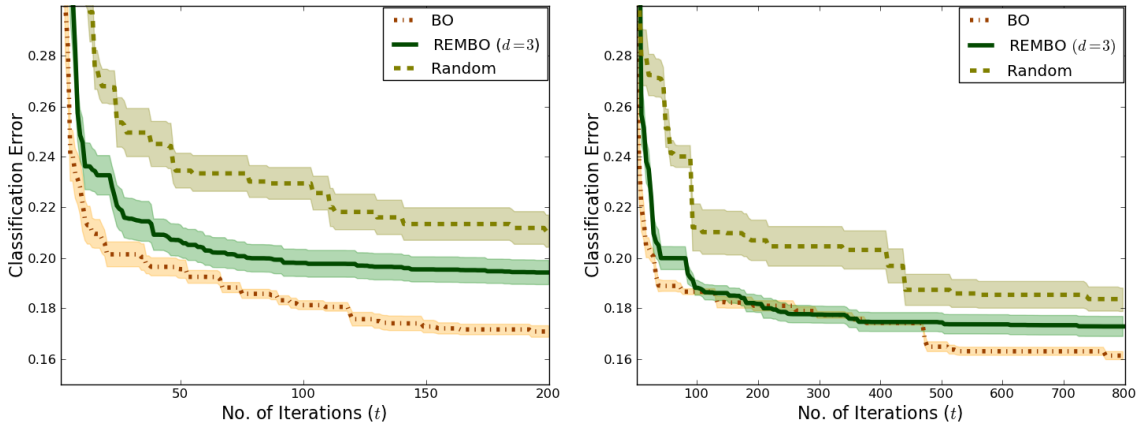

Figure 8: Performance of various methods for optimizing RF parameters for body part classification. For all methods, we show RF accuracy (mean $\pm$ 1/4 standard deviation across 10 runs) for all 2.2 million non background pixels in the 500-pose validation set, using the RF parameters identified by the method. The results on the test set were within 1% of the results on the validation set. Left: performance with a single run of each method; Right: performance with $k = 4$ interleaved runs.

5. Conclusion

We have demonstrated that it is possible to use random embeddings in Bayesian optimization to optimize functions of extremely high extrinsic dimensionality D provided that they have low intrinsic dimensionality d_e. Moreover, our resulting REMBO algorithm is coordinate independent and it only requires a simple modification of the original Bayesian optimization algorithm; namely multiplication by a random matrix. We proved REMBO's independence of D theoretically and empirically validated it by optimizing low-dimensional functions embedded in previously untenable extrinsic dimensionalities of up to 1 billion. We also theoretically and empirically showed REMBO's rotational invariance. Finally, we demonstrated that REMBO achieves state-of-the-art performance for optimizing the 47 discrete parameters of a popular mixed integer 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.

We note that the central idea of our work – using an otherwise unmodified optimization procedure in a randomly embedded space – in principle could be applied to arbitrary optimization procedures. Evaluating the effciency of this technique for other procedures is an interesting topic for future work.

Acknowledgements

We thank Christof Schötz for proofreading a draft of this article. Frank Hutter gratefully acknowledges funding by the German Research Foundation (DFG) under Emmy Noether grant HU 1900/2-1.

Appendix A. Proof of Theorem 2

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 $\mathbf{\Phi}^T\mathbf{A}_e$. □

Appendix B. Proof of Theorem 3

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

$$\mathbf{\Phi} = \begin{bmatrix} \mathbf{I}_{d_e} \\ \mathbf{0} \end{bmatrix}$$

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

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

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

References

Azimi, J., Fern, A., & Fern, X. (2010). Batch Bayesian optimization via simulation matching. In *Advances in Neural Information Processing Systems*, pp. 109–117.

Azimi, J., Fern, A., & Fern, X. (2011). Budgeted optimization with concurrent stochastic-duration experiments. In *Advances in Neural Information Processing Systems*, pp. 1098–1106.

Azimi, J., Jalali, A., & Fern, X. (2012). Hybrid batch Bayesian optimization. In *International Conference on Machine Learning*.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pp. 2546–2554.

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

Bergstra, J., Yamins, D., & Cox, D. D. (2013). Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, pp. 115–123.

Berkelaar, M., Eikland, K., & Notebaert, P. (2016). *lpsolve : Open source (Mixed-Integer) Linear Programming system*. `http://lpsolve.sourceforge.net/`.

Brochu, E., Brochu, T., & de Freitas, N. (2010). A Bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 103–112.

Brochu, E., Cora, V. M., & de Freitas, N. (2009). A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. Tech. rep. UBC TR-2009-23 and arXiv:1012.2599v1, Dept. of Computer Science, University of British Columbia.

Brochu, E., de Freitas, N., & Ghosh, A. (2007). Active preference learning with discrete choice data. In *Advances in Neural Information Processing Systems*, pp. 409–416.

Bubeck, S., Munos, R., Stoltz, G., & Szepesvari, C. (2011). X-armed bandits. *Journal of Machine Learning Research*, *12*, 1655–1695.

Bull, A. D. (2011). Convergence rates of efficient global optimization algorithms. *Journal of Machine Learning Research*, *12*, 2879–2904.

Carpentier, A., & Munos, R. (2012). Bandit theory meets compressed sensing for high dimensional stochastic linear bandit. In *Artificial Intelligence and Statistics*, pp. 190–198.

Chen, B., Castro, R., & Krause, A. (2012). Joint optimization and variable selection of high-dimensional Gaussian processes. In *International Conference on Machine Learning*.

de Freitas, N., Smola, A., & Zoghi, M. (2012). Exponential regret bounds for Gaussian process bandits with deterministic observations. In *International Conference on Machine Learning*.

Denil, M., Bazzani, L., Larochelle, H., & de Freitas, N. (2012). Learning where to attend with deep architectures for image tracking. *Neural Computation*, *24*(8), 2151–2184.

Djolonga, J., Krause, A., & Cevher, V. (2013). High dimensional Gaussian process bandits. In *Advances in Neural Information Processing Systems*, pp. 1025–1033.

Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., & Leyton-Brown, K. (2013). Towards an empirical foundation for assessing Bayesian optimization of hyperparameters. In *NIPS Workshop on Bayesian Optimization in Theory and Practice*.

Frazier, P., Powell, W., & Dayanik, S. (2009). The knowledge-gradient policy for correlated normal beliefs. *INFORMS journal on Computing*, *21*(4), 599–613.

Gomes, C. P., van Hoeve, W., & Sabharwal, A. (2008). Connections in networks: A hybrid approach. In *International Conference on Integration of Artificial Intelligence and Operations Research*, Vol. 5015, pp. 303–307.

Gramacy, R. B., Lee, H. K. H., & Macready, W. G. (2004). Parameter space exploration with Gaussian process trees. In *International Conference on Machine Learning*, pp. 45–52.

Gramacy, R., & Polson, N. (2011). Particle learning of gaussian process models for sequential design and optimization. *Journal of Computational and Graphical Statistics*, *20*(1), 102–118.

Hamze, F., Wang, Z., & de Freitas, N. (2013). Self-avoiding random dynamics on integer complex systems. *ACM Transactions on Modelling and Computer Simulation*, *23*(1), 9:1–9:25.

Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, *9*(2), 159–195.

Hennig, P., & Schuler, C. (2012). Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, *98888*, 1809–1837.

Hoffman, M., Brochu, E., & de Freitas, N. (2011). Portfolio allocation for Bayesian optimization. In *Uncertainty in Artificial Intelligence*, pp. 327–336.

Hoffman, M., Kueck, H., de Freitas, N., & Doucet, A. (2009). New inference strategies for solving Markov decision processes using reversible jump MCMC. In *Uncertainty in Artificial Intelligence*, pp. 223–231.

Hoffman, M., Shahriari, B., & de Freitas, N. (2014). On correlation and budget constraints in model-based bandit optimization with application to automatic machine learning. In *Artificial Intelligence and Statistics*.

Hoos, H. H. (2012). Programming by optimization. *Communications of the ACM*, *55*(2), 70–80.

Hutter, F. (2009). *Automated Configuration of Algorithms for Solving Hard Computational Problems*. Ph.D. thesis, University of British Columbia, Vancouver, Canada.

Hutter, F., Hoos, H., & Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. In *International Conference on Machine Learning*.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2010). Automated configuration of mixed integer programming solvers. In *Conference on Integration of Artificial Intelligence and Operations Research*, pp. 186–202.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pp. 507–523.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2012). Parallel algorithm configuration. In *Learning and Intelligent Optimization*, pp. 55–70.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). An evaluation of sequential model-based optimization for expensive blackbox functions. In *Proceedings of GECCO-13 Workshop on Blackbox Optimization Benchmarking (BBOB'13)*.

Hutter, F., Hoos, H. H., Leyton-Brown, K., & Stützle, T. (2009). ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, *36*, 267–306.

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

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

Jones, D., Schonlau, M., & Welch, W. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, *13*(4), 455–492.

Kueck, H., de Freitas, N., & Doucet, A. (2006). SMC samplers for Bayesian optimal nonlinear design. In *IEEE Nonlinear Statistical Signal Processing Workshop*, pp. 99–102.

Kueck, H., Hoffman, M., Doucet, A., & de Freitas, N. (2009). Inference and learning for active sensing, experimental design and control. In *Pattern Recognition and Image Analysis*, Vol. 5524, pp. 1–10.

Lizotte, D., Greiner, R., & Schuurmans, D. (2011). An experimental methodology for response surface optimization methods. *Journal of Global Optimization*, *53*(4), 1–38.

Lizotte, D., Wang, T., Bowling, M., & Schuurmans, D. (2007). Automatic gait optimization with Gaussian process regression. In *International Joint Conference on Artificial Intelligence*, pp. 944–949.

Mahendran, N., Wang, Z., Hamze, F., & de Freitas, N. (2012). Adaptive MCMC with Bayesian optimization. *Journal of Machine Learning Research - Proceedings Track*, *22*, 751–760.

Marchant, R., & Ramos, F. (2012). Bayesian optimisation for intelligent environmental monitoring. In *NIPS workshop on Bayesian Optimization and Decision Making*.

Martinez-Cantin, R., de Freitas, N., Doucet, A., & Castellanos, J. A. (2007). Active policy learning for robot planning and exploration under uncertainty. In *Robotics, Science and Systems*.

Močkus, J. (1982). The Bayesian approach to global optimization. In *Systems Modeling and Optimization*, Vol. 38, pp. 473–481. Springer.

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

Močkus, J., Močkus, A., & Močkus, L. (1999). *Bayesian approach for randomization of heuristic algorithms of discrete programming*. American Math. Society.

Osborne, M. A., Garnett, R., & Roberts, S. J. (2009). Gaussian processes for global optimisation. In *Learning and Intelligent Optimization*, pp. 1–15.

Rasmussen, C. E. (2003). Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals. In *Bayesian Statistics 7*.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. The MIT Press.

Sankar, A., Spielman, D., & Teng, S. (2003). Smoothed analysis of the condition numbers and growth factors of matrices. Tech. rep. Arxiv preprint cs/0310022, MIT.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., & Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *IEEE Computer Vision and Pattern Recognition*, pp. 1297–1304.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2960–2968.

Srinivas, N., Krause, A., Kakade, S. M., & Seeger, M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, pp. 1015–1022.

Swersky, K., Snoek, J., & Adams, R. P. (2013). Multi-task Bayesian optimization. In *Advances in Neural Information Processing Systems*, pp. 2004–2012.

Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, *25*(3/4), 285–294.

Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 847–855.

Vallati, M., Fawcett, C., Gerevini, A. E., Hoos, H. H., & Saetti, A. (2011). Generating fast domain-optimized planners by automatically configuring a generic parameterised planner. In *ICAPS Planning and Learning Workshop*.

Vazquez, E., & Bect, J. (2010). Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and Inference*, *140*, 3088–3095.

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

Wang, Z., & de Freitas, N. (2014). Bayesian multiscale optimistic optimization. In *Artificial Intelligence and Statistics*.

Wang, Z., Zoghi, M., Hutter, F., Matheson, D., & de Freitas, N. (2013). Bayesian optimization in high dimensions via random embeddings. In *International Joint Conference on Artificial Intelligence*, pp. 1778–1784.