Bayesian Optimization in a Billion Dimensions via Random Embeddings

Ziyu Wang ZIYUW@CS.UBC.CA
University of British Columbia

Masrour Zoghi M.ZOGHI@UVA.NL
University of Amsterdam

David Matheson DAVIDM@CS.UBC.CA
University of British Columbia

Frank Hutter HUTTER@CS.UBC.CA
University of British Columbia

Nando de Freitas NANDO@CS.UBC.CA
University of British Columbia

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, including regret bounds that only depend on the problem's intrinsic dimensionality. 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. 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; Bull, 2011; de Freitas et al., 2012).

Bayesian optimization has two ingredients. The first ingredient is a prior distribution that captures our beliefs about the behavior of the unknown objective function. The second ingredient is a risk function that describes the deviation from the global optimum. The expected risk is used to decide where to sample next. After observing a few samples of the objective, the prior is updated to produce a more informative posterior distribution over the space of objective functions

(see Figure 1 in Brochu et al., 2009). One problem with this maximum expected utility framework is that the risk is typically very hard to compute. This has led to the introduction of many sequential heuristics, known as acquisition functions, including Thompson sampling (Thompson, 1933), probability of improvement (Jones, 2001), expected improvement (Močkus, 1994) and upper-confidence-bounds (Srinivas et al., 2010). These acquisition functions trade-off exploration and exploitation. They are typically optimized by choosing points where the predictive mean is high (exploitation) and where the variance is large (exploration). Since the acquisition functions above 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. It is also possible to use dynamic portfolios of acquisition functions to improve the efficiency of the method (Hoffman et al., 2011).

The term Bayesian optimization seems to have been coined several decades ago by Jonas Močkus (1982). A popular version of the method has been known as efficient global optimization in the experimental design literature since the 1990s (Jones et al., 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 et al., 2011), Bayesian parametric models (Wang & de Freitas, 2011), treed GPs (Gramacy et al., 2004) and random forests (Brochu et al., 2009; Hutter, 2009). 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 et al., 2011, and the references therein).

In recent years, the machine learning community has increasingly used Bayesian optimization (Rasmussen, 2003; Brochu et al., 2007; Martinez-Cantin et al., 2007; Lizotte et al., 2007; Frazier et al., 2009; Azimi et al., 2010; Hamze et al., 2011; Azimi et al., 2011; Bergstra et al., 2011; Gramacy & Polson, 2011; Denil et al., 2012; Mahendran et al., 2012; Azimi et al., 2012; Hennig & Schuler, 2012; Marchant & Ramos, 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 by 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. 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. 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. However, their method is based on frequentist uncertainty estimates that can fail even for the optimization of very simple functions and lacks theoretical guarantees.

In the *linear* bandits case, Carpentier & Munos (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.

Many researchers have noted that for certain classes of problems most dimensions do not change the objective function significantly; examples include hyperparameter optimization for neural networks and deep belief networks (Bergstra & Bengio, 2012) and automatic configuration 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 & 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 1 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

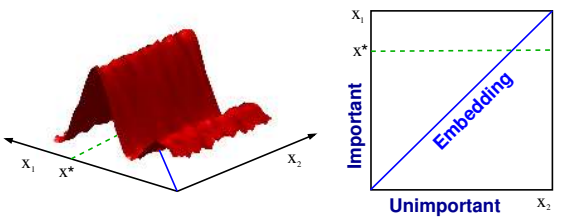

Figure 1. This function in D=2 dimesions only has d=1 *effective dimension*: the vertical axis indicated with the word important on the right hand side figure. Hence, the 1-dimensional embedding includes the 2-dimensional function's optimizer. It is more efficient to search for the optimum along the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

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. As we demonstrate in this paper, using random embeddings this simple idea largely scales to arbitrary D and d, allowing us to perform Bayesian optimization in a low-dimensional space to optimize a high-dimensional function with low intrinsic dimensionality. Importantly, this trick is not restricted to cases with axis-aligned intrinsic dimensions but applies to any d-dimensional linear subspace.

Following an explanation of GP-based Bayesian optimization (Section 2), we introduce the Random EMbedding Bayesian Optimization (REMBO) algorithm and state its theoretical properties, including regret bounds that only depend on the problem's intrinsic dimensionality (Section 3). Our experiments (Section 4) show that REMBO can solve problems of previously untenable high extrinsic dimensions. They also show that REMBO can achieve state-of-the-art performance when automatically tuning 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 & 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 (see Definition 4), 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; Vazquez & Bect, 2010; 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, such as DIRECT (Jones et al., 1993) and CMA-ES (Hansen & Ostermeier, 2001).

The Bayesian optimization procedure is shown in Algorithm 1.

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.

Algorithm 1 Bayesian Optimization

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

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. 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 practice, we do not typically perform optimization across all of $\mathbb{R}^D$, but rather across a compact subset $\mathcal{X} \subset \mathbb{R}^D$ (typically a box). 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 2. We still need to describe how REMBO chooses the bounded region $\mathcal{Y} \subset \mathbb{R}^d$, inside which it performs 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.

Algorithm 2 REMBO: Bayesian Optimization with Random Embedding

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

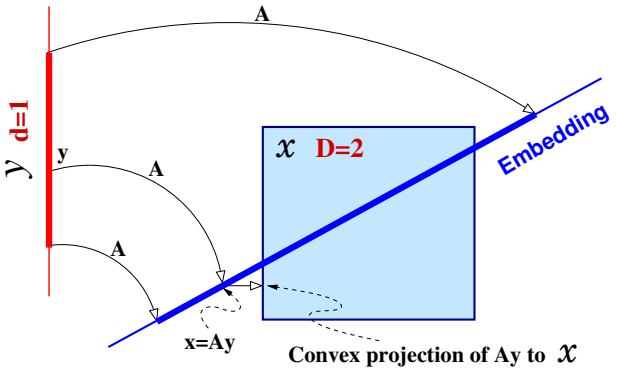

Figure 2. Embedding from $d = 1$ into $D = 2$. The box illustrates the 2D constrained space $\mathcal{X}$, while the thicker red line illustrates the 1D constrained space $\mathcal{Y}$. Note that if $\mathbf{A}\mathbf{y}$ is outside $\mathcal{X}$, it is projected onto $\mathcal{X}$. The set $\mathcal{Y}$ 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. 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 $\mathbf{x}^\star_\top \in \mathcal{X}$ 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 $[-\sqrt{d}, \sqrt{d}]^d$.

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 $\mathbf{x}$. We can do this by choosing d_e sub-components of the d-dimensional vector $\mathbf{y}$ while setting the remaining components 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 4. *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 in Definition 4. 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 projection operator for our box-constraint as above (see Figure 2).

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 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 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 $\{p_\mathcal{X}(\mathbf{A}\mathbf{y}) : \mathbf{y} \in \mathcal{Y}\}$.

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. We define our kernel for categorical variables as:

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

where $\mathbf{y}^{(1)}, \mathbf{y}^{(2)} \in \mathcal{Y} \subset \mathbb{R}^d$ and h 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 then map $\bar{x}_i$ to the corresponding discrete parameters by scaling and rounding. In our experiments, following Hutter (2009), we defined $h(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = |\{i : x^{(1)}_i \neq x^{(2)}_i\}|$ 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 Hamming distance between their mappings in the high-dimensional space.

3.3. Regret Bounds

When using the high-dimensional kernel k^D on $\{p_{\mathcal{X}}(\mathbf{A}\mathbf{y}) : \mathbf{y} \in \mathcal{Y}\} \subset \mathcal{X}$, we could easily apply previous theoretical results (Srinivas et al., 2010; Bull, 2011; de Freitas et al., 2012). However, this is not satisfying since the rates of convergence would still depend on D. If the low-dimensional embedding captures the optimizer, and since the search is conducted in $\{p_{\mathcal{X}}(\mathbf{A}\mathbf{y}) : \mathbf{y} \in \mathcal{Y}\}$ instead of $\mathcal{X}$, we should expect faster rates of convergence that only depend on the size of the embedding's dimensionality. The rest of this section shows that it is indeed possible to obtain rates of convergence that only depend on the embedding's dimensionality.

We begin our mathematical treatment with the definitions of *simple regret* and the *skew squared exponential (SSE)* kernel.

Definition 5. *Given a function $f : \mathcal{X} \to \mathbb{R}$ and a sequence of points $\{\mathbf{x}_t\}_{t=1}^{\infty} \subseteq \mathcal{X}$, the* simple regret *on the set $\mathcal{X}$ at time T is defined to be $r_f(T) = \sup_{\mathcal{X}} f - \max_{t=1}^{T} f(\mathbf{x}_t)$.*

Definition 6. *Given a symmetric, positive-definite matrix $\mathbf{\Lambda}$, we define the corresponding skew squared exponential kernel using the formula*

$$k_{\mathbf{\Lambda}}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = e^{-(\mathbf{x}^{(1)}-\mathbf{x}^{(2)})^{\top}\mathbf{\Lambda}^{-1}(\mathbf{x}^{(1)}-\mathbf{x}^{(2)})}.$$

Given $\mathbf{\Lambda}$, ℓ (as in the squared exponential kernel k_ℓ^d) and $\mathcal{X} \subseteq \mathbb{R}^d$, we denote the Reproducing Kernel Hilbert Spaces (RKHSs) corresponding to $k_{\mathbf{\Lambda}}$ and k_ℓ by $\mathcal{H}_{\mathbf{\Lambda}}(\mathcal{X})$ and $\mathcal{H}_\ell(\mathcal{X})$, respectively (Steinwart & Christmann, 2008, Definition 4.18). Moreover, given an arbitrary kernel k, we will denote its RKHS by $\mathcal{H}_k$.

Our main result below shows that the simple regret vanishes with rate $\mathcal{O}(t^{-\frac{1}{d}})$ with high probability when we use the squared exponential kernel. Note that we only make the assumption that the cost function restricted to $\mathcal{T}$ is governed by a skew symmetric kernel, a much weaker assumption than the standard assumption that the cost function is governed by an axis aligned kernel in D dimensions (see, e.g., Bull, 2011).

Theorem 7. *Let $\mathcal{X} \subset \mathbb{R}^D$ be a compact subset with non-empty interior that is convex and contains the origin and $f : \mathcal{X} \to \mathbb{R}$, a function with effective dimension d. Suppose that the restriction of f to its effective subspace $\mathcal{T}$, denoted $f|_{\mathcal{T}}$, is an element of the RKHS $\mathcal{H}_{\mathbf{\Lambda}}(\mathbb{R}^d)$ with $\mathbf{\Lambda}$ symmetric and positive definite and also satisfying $0 < r^2 < \lambda_{\min}(\mathbf{\Lambda}) \leq \lambda_{\max}(\mathbf{\Lambda}) < R^2$ for constants r and R.*

Let $\mathbf{A}$ be a $D \times d$ matrix, whose elements are drawn from the normal distribution $\frac{1}{\sqrt{d}}\mathcal{N}(0,1)$. Given any $\epsilon > 0$, we can choose a length-scale $\ell = \ell(\epsilon)$ such that running Expected Improvement with kernel k_ℓ on the restriction of f to the image of $\mathbf{A}$ inside $\mathcal{X}$ would have simple regret in $\mathcal{O}(t^{-\frac{1}{d}})$ with probability $1 - \epsilon$.

This theorem does not follow directly from the results of Bull (2011), since the kernel is not aligned with the axes, both in the high-dimensional space and the lower dimensional embedding. Thus, even given the true hyper-parameter the aforementioned paper will not entail a convergence result.

Please refer to the appendix for the proof of this theorem. The general idea of the proof is as follows. If we have a squared exponential kernel k_ℓ, with a smaller length scale than a given kernel k, then an element f of the RKHS of k is also an element of the RKHS of k_ℓ. So, when running expected improvement, one can safely use k_ℓ instead of k as the kernel and still get a regret bound. Most of the proof is dedicated to finding a length scale ℓ that fits "underneath" our kernel, so we can replace our kernel with k_ℓ, to which we can apply the results of Bull (2011).

The above theorem requires the embedded dimension and the effective dimension to coincide, but due to Proposition 1 in (de Freitas et al., 2012), we strongly believe that the analysis in (Bull, 2011) can be modified to allow for situations in which some of the ARD parameters are zero, which is precisely what is preventing us from extending this result to the case where $d > d_e$.

3.4. 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. 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 stan-

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 $\sqrt{\sigma(\mathbf{x}_{t+1})} < t_\sigma$ **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 $i \mod 20 = 0$ **or** $C = 5$ **then**
    if $C = 5$ **then**
      $U = \max\{0.9\ell, L\}$
      $C = 0$
    end if
    Learning the hyper-parameter by optimizing the log marginal likelihood by using DIRECT: $\ell = \arg\max_{l\in[L,U]} \log p(\mathbf{f}_{1:t+1}|\mathbf{x}_{1:t+1}, l)$
  end if
end for

dard 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.

The motivation of this algorithm is to rather err on the side of having too small a length scale.[1] 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. So, 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 $t_\sigma = 0.002$.

[1] A similar idea is exploited in the proof of Theorem 7.

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, in the appendix 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.

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 3.4. For each optimization of the acquisition function, this version runs both DIRECT (Jones et al., 1993) and CMA-ES (Hansen & Ostermeier, 2001) and uses the result of the better of the two. The code for REMBO, as well as all data used in our experiments will be made publicly available in the near future.

Some of our experiments required substantial computational resources, with the computational expense of each experiment depending mostly on the cost of evaluating the respective blackbox 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 4.3 required 4-5 hours per run, and optimizing the random forest classifier in Appendix D required 4-5 days per run. In total, we used over half a year of CPU time for the experiments in this paper.

In each experiment, 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.

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.

4.2. Bayesian Optimization in a Billion Dimensions

In this section, we add empirical evidence to our theoretical finding from Section 3 that REMBO's performance is independent of the extrinsic dimensionality D when using the low-dimensional kernel $k_\ell^d(\mathbf{y}^1, \mathbf{y}^2)$ from Definition 4. Specifically, using synthetic data, we show that when using that kernel REMBO has no problem scaling up to as many as 1 billion dimensions. We also study REMBO's invariance properties and empirically evaluate our two strategies for increasing its success probability.

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 $f(\mathbf{x}_{1:D}) = b(x_i, x_j)$, where b is the Branin function (see Lizotte, 2008, for its exact formula), and where dimensions i and j are selected once using a random permutation of $1, \ldots, D$. 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 first study the effectiveness of the two techniques for increasing REMBO's success probability that we proposed in Section 3.1. To empirically study the effectiveness of using internal dimensionalities $d > d_e$, and of interleaving multiple independent runs, k, we ran REMBO with all combinations of three different values of d and k. The results in Table 1 demonstrate that both techniques helped improve REMBO's performance, with interleaved runs being the more effective strategy. We note that in 13/50 REMBO runs, the global optimum was indeed not contained in the box $\mathcal{Y}$ that 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. Here we evaluated all REMBO versions 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. The results show that performing multiple independent runs nevertheless substantially improved REMBO's performance. Using a larger d was also effective in increasing the probability of the optimizer falling into REMBO's box $\mathcal{Y}$ but at the same time slowed down REMBO's convergence (such that interleaving several short runs lost its effectiveness). We conclude that using several interleaved runs of REMBO with small $d \geq d_e$ performs best.

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.

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

Finally, 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 (right) shows that REMBO's performance is not affected by this rotation.

4.3. 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 demon-

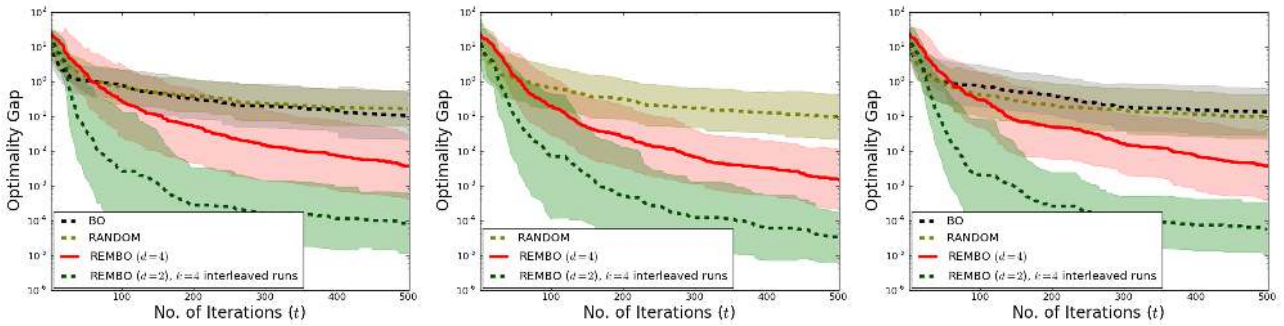

Figure 3. Comparison of random search (RANDOM), standard Bayesian optimization (BO), and REMBO. Left: $D = 25$ extrinsic dimensions; Middle: $D = 1\,000\,000\,000$ extrinsic dimensions; Right: $D = 25$, with a rotated objective function. For each method, we plot means and 1/4 standard deviation confidence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

strated 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 & 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., 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[2], 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 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 overfitting). 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 problems, REMBO appears to effectively exploit the low effective dimensionality of at least this particular optimization problem.

As baselines for judging our performance in configuring `lpsolve`, we used the configuration procedures ParamILS (Hutter et al., 2009) and SMAC (Hutter et al., 2011). ParamILS and SMAC have been specifically designed for the configuration of algorithms with many discrete parameters and yield state-of-the-art performance for this task.

As Figure 4.3 (top) shows, ParamILS and SMAC indeed outperformed random search and BO. However, remarkably, our vanilla REMBO method performed even 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, as shown by Hutter et al. (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 4.3 (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

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. Our resulting REMBO algorithm is coor-

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

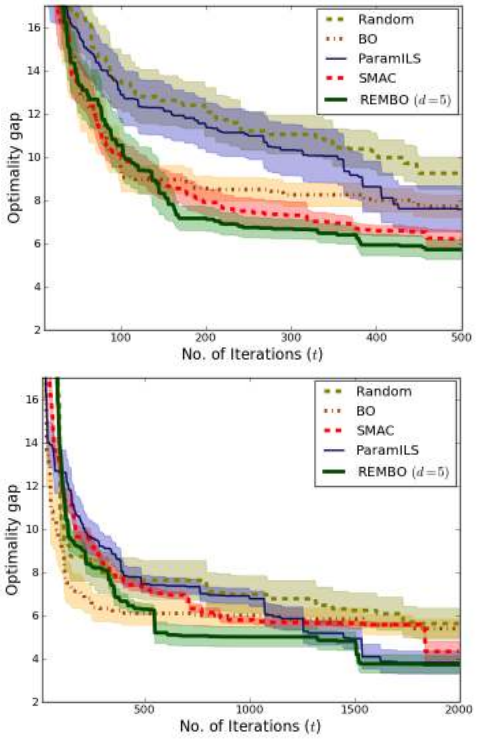

Figure 4. Performance of various methods for configuration of `lpsolve`; we show the optimality gap `lpsolve` achieved with the configurations found by the various methods (lower is better). Top: a single run of each method; Bottom: performance with $k = 4$ interleaved runs.

dinate independent and has provable regret bounds that are independent of the extrinsic dimensionality D. Moreover, 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 previously untenable extrinsic dimensionalities of up to 1 billion. 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.

References

Azimi, J., Fern, A., and Fern, X. Batch bayesian optimization via simulation matching. NIPS, 2010.

Azimi, J., Fern, A., and Fern, X.Z. Budgeted optimization with concurrent stochastic-duration experiments. NIPS, 2011.

Azimi, J., Jalali, A., and Fern, X.Z. Hybrid batch bayesian optimization. In *ICML*, 2012.

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

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

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

Brochu, E., de Freitas, N., and Ghosh, A. Active preference learning with discrete choice data. In *NIPS*, pp. 409–416, 2007.

Brochu, E., Cora, V. M., and de Freitas, N. 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.

Bubeck, S., Munos, R., Stoltz, G., and Szepesvari, C. X-armed bandits. *JMLR*, 12:1655–1695, 2011.

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

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

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

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

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

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

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

Gramacy, R. B., Lee, H. K. H., and Macready, W. G. Parameter space exploration with Gaussian process trees. In *ICML*, pp. 45–52, 2004.

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

Hamze, F., Wang, Z., and de Freitas, N. Self-avoiding

random dynamics on integer complex systems. Technical Report arXiv:1111.5379v2, 2011.

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

Hennig, P. and Schuler, C.J. Entropy search for information-efficient global optimization. *The Journal of Machine Learning Research*, 98888:1809–1837, 2012.

Hoffman, M., Brochu, E., and de Freitas, N. Portfolio allocation for Bayesian optimization. In *UAI*, pp. 327–336, 2011.

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

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

Hutter, F., Hoos, H. H., Leyton-Brown, K., and Stützle, T. ParamILS: an automatic algorithm configuration framework. *JAIR*, 36:267–306, 2009.

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

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

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Parallel algorithm configuration. In *LION*, pp. 55–70, 2012.

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

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

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

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

Lizotte, D., Wang, T., Bowling, M., and Schuurmans, D. Automatic gait optimization with Gaussian process regression. In *IJCAI*, 2007.

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

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

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

Martinez-Cantin, R., de Freitas, N., Doucet, A., and Castellanos, J. A. Active policy learning for robot planning and exploration under uncertainty. 2007.

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

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

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

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

Rasmussen, C. E. Gaussian processes to speed up hybrid Monte Carlo for expensive Bayesian integrals. 2003.

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

Rudelson, M. and Vershynin, R. Non-asymptotic theory of random matrices: Extreme singular values. In *Int. Congress of Mathematicians*, pp. 1576–1599, 2010.

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

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. Real-time human pose recognition in parts from single depth images. In *CVPR*, pp. 1297–1304, 2011.

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

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

Steinwart, I. and Christmann, A. *Support Vector Machines*. Springer, 2008.

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

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

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

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 $\mathbf{\Phi}^T\mathbf{A}_e$. □

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 $\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{A}\mathbf{y}^\star = \mathbf{x}^\star_\top$ 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 $1 - \epsilon$, $\|\mathbf{y}^\star\| \leq \|\mathbf{B}^{-1}\|_2 \|\bar{\mathbf{x}}^\star_\top\|_2 = \|\mathbf{B}^{-1}\|_2 \|\mathbf{x}^\star_\top\|_2 \leq \frac{\sqrt{d_e}}{\epsilon}\|\mathbf{x}^\star_\top\|_2$. □

C. Proof of Theorem 7

Before embarking on the proof of Theorem 7, we introduce some definitions and state a few preliminary results, which we quote from (Steinwart & Christmann, 2008) and (Bull, 2011) to facilitate the reading of this exposition.

Definition 8. *Given a map $\pi : \mathcal{S} \to \mathcal{T}$ between any two sets $\mathcal{S}$ and $\mathcal{T}$, and any map $f : \underbrace{\mathcal{T} \times \cdots \times \mathcal{T}}_{n\text{-times}} \to \mathbb{R}$, with $n \geq 1$, we define the* pull-back *of f under π as follows:*

$$\pi^* f(s_1, \ldots, s_n) := f(\pi s_1, \ldots, \pi s_n).$$

That is one evaluates the pull-back $\pi^* f$ on points in $\mathcal{S}$ by first "pushing them forward" onto $\mathcal{T}$ and then using f to get a number.

Also, if the map π is given by a matrix $\mathbf{A}$, we will use the notation $\mathbf{A}^* f$ for the pull-back of f under the linear map induced by $\mathbf{A}$. Moreover, given a matrix $\mathbf{A}$ and a set $\mathcal{S}$ in its target space, we will denote by $\mathbf{A}^{-1}(\mathcal{S})$ the set of all points that are mapped into $\mathcal{S}$ by $\mathbf{A}$.

Lemma 9 (Lemma 4.6 in (Steinwart & Christmann, 2008))**.** *Let k_1 be a kernel on $\mathcal{X}_1$ and k_2 be a kernel on $\mathcal{X}_2$. Then $k_1 \cdot k_2$ is a kernel on $\mathcal{X}_1 \times \mathcal{X}_2$.*

Remark 10. *In the proof of the above lemma in (Steinwart & Christmann, 2008), it is argued that if $\mathcal{H}_i$ is the RKHS of k_i for $i = 1, 2$, then $\mathcal{H}_1 \widehat{\otimes} \mathcal{H}_2$ is the RKHS of $k_1 \cdot k_2$, where $\widehat{\otimes}$ is the tensor product of Hilbert spaces and the elements of $\mathcal{H}_1 \widehat{\otimes} \mathcal{H}_2$ have the form $h(x_1, x_2) = \sum_i f_i(x_1) g_i(x_2)$, where the functions $f_i : \mathcal{X}_1 \to \mathbb{R}$ are elements of $\mathcal{H}_1$ and the functions $g_i : \mathcal{X}_2 \to \mathbb{R}$ are elements of $\mathcal{H}_2$.*

Corollary 11 (Corollary 4.43 in (Steinwart & Christmann, 2008))**.** *Given $\mathcal{X} \subseteq \mathbb{R}^d$ with non-empty interior, and $r > 0$, then we have an isomorphism of Hilbert spaces $\mathcal{H}_r(\mathcal{X}) \cong \mathcal{H}_r(\mathbb{R}^d)$, through an extension operator $I_{\mathcal{X}}$ (whose definition is omitted, since it is not needed here).*

Proposition 12 (The second part of Proposition 4.46 in (Steinwart & Christmann, 2008))**.** *Given $0 < \ell < U$, for all non-empty $\mathcal{X} \subseteq \mathbb{R}^d$, the RKHS $\mathcal{H}_U(\mathcal{X})$ can be mapped under the identity map into $\mathcal{H}_\ell(\mathcal{X})$ and we have the following bound:*

$$\|\mathrm{id} : \mathcal{H}_U \to \mathcal{H}_\ell\| \leq \left(\frac{U}{\ell}\right)^{\frac{d}{2}},$$

where the norm in the equation is the operator norm, i.e. $\sup_{f \in \mathcal{H}_U} \frac{\|f\|_{\mathcal{H}_\ell}}{\|f\|_{\mathcal{H}_U}}$.

Remark 13. *Note that here, the map* id *signifies the following: the element $f \in \mathcal{H}_U$ corresponds to a real-valued function on $\mathcal{X}$, which we will also denote by f, so one can pose the question whether or not this function is an element of $\mathcal{H}_\ell$ as well, and the existence of the map* $\mathrm{id} : \mathcal{H}_U \to \mathcal{H}_\ell$ *implies that f is indeed an element of $\mathcal{H}_\ell$. Equivalently, $\mathcal{H}_U \subset \mathcal{H}_\ell$ and*

$$\|f\|_{\mathcal{H}_\ell} \leq \left(\frac{U}{\ell}\right)^{\frac{d}{2}} \|f\|_{\mathcal{H}_U}.$$

In the proof of our theorem below, we will extend this result for squared exponential kernels to skewed squared exponential kernels.

Proposition 14 (Theorem 2 in (Bull, 2011), paraphrased for our particular setting)**.** *Given a squared exponential kernel k_ℓ on a compact subset $\mathcal{Y} \subset \mathbb{R}^d$ and a function $f \in \mathcal{H}_\ell(\mathcal{Y})$, then applying Expected Improvement to f results in simple regret that diminishes according to $\mathcal{O}\big(t^{-\frac{1}{d}}\big)$, with the constants worsening as the norm $\|f\|_{\mathcal{H}_\ell(\mathcal{Y})}$ increases.*

Proof of Theorem 7. Let $\mathbf{\Pi} : \mathcal{X} \twoheadrightarrow \mathcal{T}$ denote the (*unknown*) orthogonal projection onto the effective subspace of f; we will also denote the corresponding matrix by $\mathbf{\Pi}$. (Please refer to the right hand side of Figure 1 for an illustration of a 2-dimensional space, a 1-dimensional embedding (slanted line) and a 1-dimensional effective space (vertical axis).)

Recall from the theorem statement that $f|_{\mathcal{T}}$ is assumed to be an element of the RKHS $\mathcal{H}_\Lambda$, and that we have $f = \mathbf{\Pi}^* f|_{\mathcal{T}}$, i.e. f is obtained from "stretching $f|_{\mathcal{T}}$ open" along the orthogonal subspace of $\mathcal{T}$. From this, we can conclude that f is an element of $\mathcal{H}_{k^D}$, with $k^D := \mathbf{\Pi}^* k_\Lambda$, where k_Λ is the kernel on the effective subspace $\mathcal{T}$.

Now, given the embedding $\mathbb{R}^d \hookrightarrow \mathbb{R}^D$ defined by the matrix $\mathbf{A}$, the pull-back function $\mathbf{A}^* f$ is an element of the RKHS $\mathcal{H}_{\mathbf{A}^* k^D}$: Henceforth, we will use the notation

$$k^d := \mathbf{A}^* k^D = \mathbf{A}^* \mathbf{\Pi}^* k_\Lambda = (\mathbf{\Pi}\mathbf{A})^* k_\Lambda.$$

In the remainder of this proof, we replace k^d with a squared exponential kernel k_ℓ that is "thinner" than k^d and so $\mathbf{A}^* f$ is also an element of the RKHS of k_ℓ. By showing that this is true, REMBO (which uses k_ℓ) has enough approximation power. Moreover, the statement of Proposition 14 applies.

Since k^D is constant along $\mathcal{T}^\perp$, we get that k^d is in turn constant along $\mathbf{A}^{-1}\left(\mathcal{T}^\perp\right)$, and since $\mathbf{A}$ is randomly chosen, the linear subspace $\mathbf{A}^{-1}\left(\mathcal{T}^\perp\right)$ will almost surely be zero dimensional. Let us introduce the notation $\mathbf{\Pi}$ for the $d \times D$ matrix that projects vectors in $\mathbb{R}^D$ onto the effective subspace $\mathcal{T}$.

We almost surely have that the $d \times d$ matrix $\mathbf{\Pi A}$ is non-singular, since the space of singular matrices has measure 0, given the fact that it is the zero-set of a polynomial, namely the determinant. Therefore, the kernel k^d, which has the form

$$k^d(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}) = e^{(\mathbf{y}^{(1)}-\mathbf{y}^{(2)})^\top \mathbf{A}^\top \mathbf{\Pi}^\top \mathbf{\Lambda}^{-1} \mathbf{\Pi A} (\mathbf{y}^{(1)}-\mathbf{y}^{(2)})}$$

is also SSE, since $\mathbf{A}^\top \mathbf{\Pi}^\top \mathbf{\Lambda}^{-1} \mathbf{\Pi A}$ is symmetric and positive definite, simply because of the symmetry and positive definiteness of $\mathbf{\Lambda}^{-1}$: given $\mathbf{y} \neq 0$, we have

$$\mathbf{y}^\top \mathbf{A}^\top \mathbf{\Pi}^\top \mathbf{\Lambda}^{-1} \mathbf{\Pi A} \mathbf{y} = \widetilde{\mathbf{y}}^\top \mathbf{\Lambda}^{-1} \widetilde{\mathbf{y}} > 0,$$

where $\widetilde{\mathbf{y}} := \mathbf{\Pi A}\mathbf{y} \neq 0$ since $\mathbf{\Pi A}$ is invertible. In what follows, we will use the notation $\mathbf{\Lambda}_d := \left(\mathbf{A}^\top \mathbf{\Pi}^\top \mathbf{\Lambda}^{-1} \mathbf{\Pi A}\right)^{-1}$.

Since $\mathbf{\Pi}$ is an orthogonal projection matrix, it has an SVD decomposition $\mathbf{\Pi} = \mathbf{USV}$ consisting of an orthogonal $d \times d$ matrix $\mathbf{U}$, an orthogonal $D \times D$ matrix $\mathbf{V}$ and a $d \times D$ matrix $\mathbf{S}$ that has the following form:

$$\mathbf{S} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \end{bmatrix}$$

. Now, given a fixed orthogonal matrix $\mathbf{O}$ and a random Gaussian vector $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{D\times D})$, due to the rotational symmetry of the normal distribution, the vector $\mathbf{Ov}$ is also a sample from $\mathcal{N}(\mathbf{0}, \mathbf{I}_{D\times D})$. Therefore, given $\mathbf{O}$ as above, and a random Gaussian matrix Γ, then $\mathbf{O}\Gamma$ is also a random Gaussian matrix with the same distribution of entries. Moreover, given $\mathbf{S}$ as above, $\mathbf{S}\Gamma$ is a $d \times D$ random Gaussian matrix, since multiplying any matrix by $\mathbf{S}$ on the left simply extracts the first d rows of the matrix.

Given this, if we fix an orthogonal decomposition $\mathbf{\Lambda}^{-1} = \mathbf{P}^\top \mathbf{D}^{-1} \mathbf{P}$, where $\mathbf{P}$ is orthogonal and $\mathbf{D}$ is a diagonal matrix with the eigenvalues of $\mathbf{\Lambda}$ along the diagonal, we can conclude that

$$\mathbf{G} := \mathbf{P\Pi A} = \mathbf{PUSVA}$$

is a random Gaussian matrix, and so the matrix $\mathbf{\Lambda}_d$ can be decomposed into random Gaussian and diagonal matrices as follows:

$$\mathbf{\Lambda}_d^{-1} = \mathbf{G}^\top \mathbf{D}^{-1} \mathbf{G}.$$

Let $s_{\min}$ and $s_{\max}$ denote the smallest and the largest singular values of a matrix. With this notation in hand, we point out the following two facts about concentration of singular values:

I. Since for any pair of matrices $\mathbf{A}$ and $\mathbf{B}$, we have $s_{\max}(\mathbf{AB}) \leq s_{\max}(\mathbf{A}) s_{\max}(\mathbf{B})$, we get

$$\frac{1}{\lambda_{\min}(\mathbf{\Lambda}_d)} = \lambda_{\max}(\mathbf{\Lambda}_d^{-1}) \leq s_{\max}(\mathbf{G})^2 s_{\max}(\mathbf{D}^{-1}) \leq \frac{s_{\max}(\mathbf{G})^2}{r^2}$$

and since $\mathbf{G}$ is a random matrix with Gaussian entries, we have (cf. Equation 2.3 in (Rudelson & Vershynin, 2010))

$$P\left(s_{\max}(\mathbf{G}) < 2\sqrt{d} + t\right) \leq 1 - 2e^{-t^2/2},$$

and so with probability $1 - \frac{\epsilon}{2}$, we have

$$s_{\max}(\mathbf{G}) < 2\sqrt{d} + \sqrt{2 \ln \frac{4}{\epsilon}}.$$

Therefore, with probability $1 - \frac{\epsilon}{2}$, we have

$$\lambda_{\min}(\mathbf{\Lambda}_d) > \left(\frac{r}{2\sqrt{d} + \sqrt{2 \ln \frac{4}{\epsilon}}} \right)^2. \tag{1}$$

Henceforth, we will use the notation

$$\ell = \ell(\epsilon) := \frac{r}{2\sqrt{d} + \sqrt{2 \ln \frac{4}{\epsilon}}}$$

II. On the other hand, we have

$$\frac{1}{\lambda_{\max}(\mathbf{\Lambda}_d)} = \lambda_{\min}(\mathbf{\Lambda}_d^{-1}) \geq s_{\min}(\mathbf{G})^2 s_{\min}(\mathbf{D}^{-1}) \geq \frac{s_{\min}(\mathbf{G})^2}{R^2}$$

together with the following probabilistic bound on $s_{\min}(\mathbf{G})$ (cf. Equation 3.2 in (Rudelson & Vershynin, 2010)):

$$P\left(s_{\min}(\mathbf{G}) > \frac{\delta}{\sqrt{d}} \right) > 1 - \delta.$$

So, with probability $1 - \frac{\epsilon}{2}$, we have

$$s_{\min}(\mathbf{G}) > \frac{\epsilon}{2\sqrt{d}},$$

and so

$$\lambda_{\max}(\mathbf{\Lambda}_d) < \frac{4dR^2}{\epsilon^2} \tag{2}$$

holds with probability $1 - \frac{\epsilon}{2}$.

In what follows, we will use the notation:

$$U = U(\epsilon) := \frac{2R\sqrt{d}}{\epsilon}$$

Now, with these estimates in hand, we can go ahead and show that the following bound holds with probability $1 - \epsilon$:

$$\|\mathbf{A}^* f\|_{\mathcal{H}_\ell(\mathbf{A}^{-1}(\mathcal{X}))} \leq \left(\frac{U(\epsilon)}{\ell(\epsilon)} \right)^{\frac{d}{2}} \|f|_{\mathcal{T}}\|_{\mathcal{H}_{\mathbf{\Lambda}}(\mathcal{T})} \tag{3}$$

This claim follows from the following sequence of facts:

A. Since the transformation $\mathbf{\Pi A}$ is invertible, we have that the map $(\mathbf{\Pi A})^* : \mathcal{H}_{\mathbf{\Lambda}}(\mathbb{R}^d) \to \mathcal{H}_{\mathbf{\Lambda}_d}(\mathbb{R}^d)$ (recall that $k_{\mathbf{\Lambda}_d} = k_{(\mathbf{\Pi A})^* \mathbf{\Lambda}}$) that sends $g \in H_{\mathbf{\Lambda}}$ to $(\mathbf{\Pi A})^* g$ is an isomorphism of Hilbert spaces and so

$$\|f|_{\mathcal{T}}\|_{\mathcal{H}_{\mathbf{\Lambda}}(\mathbb{R}^d)} = \|\mathbf{A}^* f\|_{\mathcal{H}_{\mathbf{\Lambda}_d}(\mathbb{R}^d)} \tag{4}$$

since we have $\mathbf{A}^* f = \mathbf{A}^* \left(\mathbf{\Pi}^* f|_{\mathcal{T}} \right) = (\mathbf{\Pi A})^* f|_{\mathcal{T}}$.

B. If we denote the eigenvalues of $\mathbf{\Lambda}_d$ by $\lambda_1 < \cdots < \lambda_d$, then, first of all, by Corollary 11, we have

$$\mathcal{H}_{\mathbf{\Lambda}_d}\left(\mathbf{A}^{-1}(\mathcal{X}) \right) \cong \mathcal{H}_{\mathbf{\Lambda}_d}(\mathbb{R}^d)$$

and also by Remark 10, we have the isomorphisms

$$\begin{aligned} \mathcal{H}_{\mathbf{\Lambda}_d}(\mathbb{R}^d) &\cong \mathcal{H}_{\sqrt{\lambda_1}}(\mathbb{R}) \,\hat{\otimes}\, \cdots \,\hat{\otimes}\, \mathcal{H}_{\sqrt{\lambda_d}}(\mathbb{R}) \\ \mathcal{H}_\ell(\mathbb{R}^d) &\cong \mathcal{H}_\ell(\mathbb{R}) \quad \,\hat{\otimes}\, \cdots \,\hat{\otimes}\, \mathcal{H}_\ell(\mathbb{R}), \end{aligned}$$

where $\hat{\otimes}$ denotes the tensor product of Hilbert spaces.

Figure 5. Left: ground truth depth, ground truth body parts and predicted body parts; Right: features specified by offsets u and v.

C. Using Equations (1) and (2) together with Proposition 12, we can conclude that

$$\begin{aligned}
\|\mathbf{A}^* f\|_{\mathcal{H}_\ell(\mathbb{R}^d)} &= \left\| \sum_i g_i^1 \otimes \cdots \otimes g_i^d \right\|_{\mathcal{H}_\ell(\mathbb{R})\hat{\otimes}\cdots\hat{\otimes}\mathcal{H}_\ell(\mathbb{R})} \\
&\leq \left(\frac{\sqrt{\lambda_1}}{\ell}\right)^{\frac{1}{2}} \cdots \left(\frac{\sqrt{\lambda_d}}{\ell}\right)^{\frac{1}{2}} \left\| \sum_i g_i^1 \otimes \cdots \otimes g_i^d \right\|_{\mathcal{H}_{\lambda_1}(\mathbb{R})\hat{\otimes}\cdots\hat{\otimes}\mathcal{H}_{\lambda_d}(\mathbb{R})} \\
&\leq \left(\frac{U}{\ell}\right)^{\frac{d}{2}} \left\| \sum_i g_i^1 \otimes \cdots \otimes g_i^d \right\|_{\mathcal{H}_{\lambda_1}(\mathbb{R})\hat{\otimes}\cdots\hat{\otimes}\mathcal{H}_{\lambda_d}(\mathbb{R})} \\
&= \left(\frac{U}{\ell}\right)^{\frac{d}{2}} \|\mathbf{A}^* f\|_{\mathcal{H}_{\mathbf{\Lambda}_d}(\mathbb{R}^d)}, \qquad (5)
\end{aligned}$$

where the two inequalities are true with probability $1 - \frac{\epsilon}{2}$ each, and so they both hold with probability $1 - \epsilon$.

Composing the Inequality (5) with Equality (4) gives us the bound claimed in Inequality (3).

Now that we know that the $\mathcal{H}_\ell(\mathbb{R}^d)$ norm of $\mathbf{A}^* f$ is finite, we can apply the Expected Improvement algorithm to it on the set $\mathbf{A}^{-1}(\mathcal{X})$ with kernel k_ℓ, instead of the unknown kernel k_{Λ_d}, and then Proposition 14 tells us that the simple regret would be in $\mathcal{O}\left(t^{-\frac{1}{d}}\right)$. □

D. Automatic Configuration of Random Forest Kinect Body Part Classifier

We now present an additional experiment evaluating 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 et al., 2011) and will be publicly released in the near future.

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 5 (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 5 (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.

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]

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 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 6 (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.[3] 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 6, 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.

This experiment also shows that automatic configuration techniques can reveal scientific facts about the problem; for example how to choose the depth of trees in RFs. For this reason, we feel it is important to advance to the machine learning community the following message about methodology. For any specific dataset, if researchers

[3]Due to the very 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.

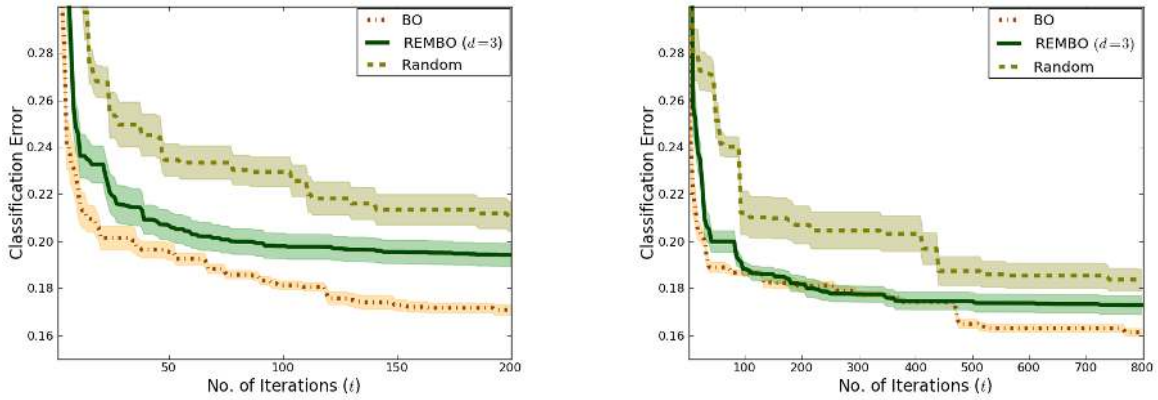

Figure 6. Performance of various methods for optimizing RF parameters for body part classification. For all methods, we show RF accuracy (mean ± 1/4 standard deviation across 10 runs) for all 2.2 million non background pixels in the 500-pose validation set, using the RF parameters identified by the method. 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.

were to release the obtained objective function evaluations, other researchers could use these values to expedite their experiments and gain greater knowledge about the problem domain. For example, our experiments with RFs took many days with powerful clusters of computers. By releasing not only the code but the samples of the objective function, other researchers could build on this data and ultimately learn a model for the objective function in this domain and, therefore, understand how the random forests parameters and design choices interact and affect performance. Every experiment should count.