RE-EXAMINING LINEAR EMBEDDINGS FOR HIGH-DIMENSIONAL BAYESIAN OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian optimization (BO) is a popular approach to optimize resource-intensive black-box functions. A significant challenge in BO is to scale to high-dimensional parameter spaces while retaining sample efficiency. A solution considered in previous literature is to embed the high-dimensional parameter space into a lower-dimensional manifold, often a random linear embedding. In this paper, we identify several crucial issues and misconceptions about the use of linear embeddings for BO. We thoroughly study and analyze the consequences of using linear embeddings and show that some of the design choices in current approaches adversely impact their performance. Based on this new theoretical understanding we propose ALEBO, a new algorithm for high-dimensional BO via linear embeddings that outperforms state-of-the-art methods on a range of problems.

1 INTRODUCTION

Bayesian optimization (BO) is a robust, sample-efficient technique for optimizing resource-intensive black-box functions (Mockus, 1989; Jones, 2001). BO has been successfully applied to diverse applications, ranging from automated machine learning (Snoek et al., 2012; Hutter et al., 2011) to robotics (Lizotte et al., 2007; Calandra et al., 2015; Rai et al., 2018). One of the most active topics of research in BO is how to extend current methods to higher-dimensional spaces. A common framework to tackle this problem is to consider a high-dimensional BO (HDBO) task as a standard BO problem in a low-dimensional embedding, where the embedding can be either a (typically random) linear projection or nonlinear, *e.g.*, via a multi-layer neural network (see Sec. 2 for a full review). An advantage of this framework is to explicitly decouple the problem of finding low-dimensional representations suitable for optimization from the actual optimization technique.

In this paper we study the use of linear embeddings for HDBO, and in particular we re-examine prior efforts to use random linear projections. Random projections are attractive for BO because, by the Johnson-Lindenstrauss lemma, they can be approximately distance-preserving (Johnson & Lindenstrauss, 1984) without requiring any data to learn the embedding. Random embeddings come with several strong theoretical guarantees, but have shown mixed empirical performance for HDBO.

The contributions of this paper are: **1)** We provide new results that identify why linear embeddings have performed poorly in HDBO. We show that existing approaches produce representations that cannot be well-modeled by a Gaussian process (GP), or representations that likely do not contain an optimum (Sec. 4). **2)** We construct a representation with better properties for BO (Sec. 5): we improve modelability by deriving a Mahalanobis kernel tailored for linear embeddings and adding polytope bounds to the embedding, and we show how to maintain a high probability that the embedding contains an optimum. **3)** We show that using this representation for BO outperforms a wide range of previous approaches for HDBO, on test functions up to $D = 1000$ (Sec. 6). These include the first results for HDBO with black-box constraints.

2 RELATED WORK

There are generally two approaches to extending BO into high dimensions. The first is to produce a low-dimensional embedding, do standard BO in this low-dimensional space, and then project up to the original space for function evaluations. The foundational work on embeddings for BO is

REMBO (Wang et al., 2016), which creates a linear embedding by generating a random projection matrix. Sec. 3 provides a thorough description of REMBO and several subsequent approaches based on random linear embeddings (Qian et al., 2016; Binois et al., 2018; Nayebi et al., 2019). If derivatives of f are available, the active subspace method can be used to recover a linear embedding (Constantine et al., 2014; Eriksson et al., 2018), or approximate gradients can be used (Djolonga et al., 2013). BO can also be done in nonlinear embeddings through VAEs (Gómez-Bombarelli et al., 2018; Lu et al., 2018; Moriconi et al., 2019). An attractive aspect of random embeddings is that they can be extremely sample-efficient, since the only model to be estimated is a low-dimensional GP.

The second approach to extend BO to high dimensions is to make use of surrogate models that better handle high dimensions, typically by imposing additional structure on the problem. Work along these lines include GPs with an additive kernel (Kandasamy et al., 2015; Wang et al., 2017; Gardner et al., 2017; Wang et al., 2018; Rolland et al., 2018; Mutný & Krause, 2018), cylindrical kernels (Oh et al., 2018), or deep neural network kernels (Antonova et al., 2017). Random forest is used as the surrogate model in SMAC (Hutter et al., 2011). These methods produce trade-offs between sample efficiency of the model and the ability to effectively optimize the acquisition function.

Here, we focus on the embedding approach and in particular the use of linear embeddings for HDBO. Without box bounds, REMBO comes with a strong guarantee: with probability 1, the embedding contains an optimum (Wang et al., 2016, Theorem 2). However, if function evaluations are limited to the box bounds, as is typical in BO problems, REMBO requires a collection of heuristics for which there are no longer guarantees on performance. While REMBO can perform well in some HDBO tasks, subsequent papers have found it can perform poorly even on tasks with a true low-dimensional linear subspace (*e.g.* Nayebi et al., 2019). In this paper, we analyze the properties of linear embeddings as they relate to BO, and show how to improve the representation of the function we seek to optimize.

3 Problem Framework and REMBO

In this section we define the problem framework and notation, and then describe BO via random linear projections (REMBO)—a promising method for HDBO—along with known challenges and follow-up work that has been proposed to address these issues.

Bayesian optimization We consider optimization problems of the form $\min_{\boldsymbol{x}\in\mathcal{B}} f(\boldsymbol{x})$ where f is a black-box function and $\mathcal{B}$ are box bounds. We assume gradients of f are unavailable. The box bounds on $\boldsymbol{x}$ specify the range of values that are reasonable or physically possible to evaluate. For instance, Gramacy et al. (2016) use BO for an environmental remediation problem in which each x_i represents the pumping rate of a particular pump, which has physical limitations. The problem may also include nonlinear constraints $c_j(\boldsymbol{x}) \leq 0$ where each c_j is itself a black-box function. BO is a form of sequential model-based optimization, where we construct a surrogate model for f and use that model to identify which parameters $\boldsymbol{x}$ should be evaluated next, according to an explore-exploit strategy. The surrogate model is typically a GP, $f \sim \mathcal{GP}(m(\cdot), k(\cdot,\cdot))$, with mean function $m(\cdot)$ and a kernel $k(\cdot,\cdot)$. Under the GP prior, the posterior for the value of $f(\boldsymbol{x})$ at any point in the space is a normal distribution with closed-form mean and variance. Using that posterior, we construct an acquisition function $\alpha(\boldsymbol{x})$ that specifies the value of a function evaluation at $\boldsymbol{x}$, such as Expected Improvement (EI) (Jones et al., 1998). We find $\boldsymbol{x}^* \in \arg\max_{\boldsymbol{x}\in\mathcal{B}} \alpha(\boldsymbol{x})$, and evaluate $f(\boldsymbol{x}^*)$.

The GP is useful for BO because it provides a well-calibrated posterior in closed form. With typical kernels and acquisition functions, $\alpha(\boldsymbol{x})$ is differentiable and can be effectively optimized. However, with typical kernels like the ARD RBF kernel, there are significant limitations. GPs are known to predict poorly in high dimensions, which for a GP is D larger than 15–20 (Wang et al., 2016; Li et al., 2016; Nayebi et al., 2019). This prevents BO from being a useful tool in high dimensions.

In HDBO, the objective $f : \mathbb{R}^D \to \mathbb{R}$ operates in a high-dimensional (D) space, which we call the *ambient space*. When using linear embeddings for HDBO, we assume there exists a low-dimensional linear subspace that captures all of the variation of f. Specifically, let $f_d : \mathbb{R}^d \to \mathbb{R}$, $d \ll D$, and let $\boldsymbol{T} \in \mathbb{R}^{d\times D}$ be a projection matrix from D down to d dimensions. The linear embedding assumption is that $f(\boldsymbol{x}) = f_d(\boldsymbol{T}\boldsymbol{x}) \;\; \forall \boldsymbol{x} \in \mathbb{R}^D$. $\boldsymbol{T}$ is unknown, and we only have access to f, not f_d. We assume without loss of generality that the box bounds are $\mathcal{B} = [-1, 1]^D$; the parameter space can always be scaled to these bounds.

REMBO: Bayesian optimization via random embedding REMBO generates a random projection matrix $\boldsymbol{A} \in \mathbb{R}^{D \times d_e}$ with each element drawn independently from $\mathcal{N}(0, 1)$ to specify a d_e-dimensional embedding. BO is done in the embedding to identify a point $\boldsymbol{y} \in \mathbb{R}^{d_e}$ to be evaluated, which is given objective value $f(\boldsymbol{Ay})$. The embedding dimension d_e should satisfy $d_e \geq d$ for the REMBO guarantee of containing an optimum to hold.

The main challenges for using REMBO come when dealing with box bounds in the ambient space. We may select a point $\boldsymbol{y}$ in the embedding to be evaluated and find that its projection to the ambient space, $\boldsymbol{Ay}$, falls outside $\mathcal{B}$. The first challenge this poses is a theoretical challenge: $\mathbb{R}^{d_e}$ is guaranteed to contain an optimum, but that optimum is *not* guaranteed to project up to $\mathcal{B}$. When function evaluations are restricted to the box bounds, the embedding may not contain an optimum—it is not difficult to construct examples of this. The second challenge posed by box bounds is the practical challenge of how function evaluations should be done for points that project outside $\mathcal{B}$. Here REMBO introduces three heuristics. First, the embedding is given box bounds $[-\sqrt{d_e}, \sqrt{d_e}]^{d_e}$. BO will only select points within those bounds to be projected up and evaluated. Second, if a point $\boldsymbol{y}$ in the embedding projects up outside $\mathcal{B}$, then it is clipped to $\mathcal{B}$. Let $p_{\mathcal{B}} : \mathbb{R}^D \to \mathbb{R}^D$ be the L^2 projection that maps $\boldsymbol{x}$ to its nearest point in $\mathcal{B}$. A point $\boldsymbol{y}$ in the embedding is given objective value $f(p_{\mathcal{B}}(\boldsymbol{Ay}))$, which can always be evaluated. Note that clipping to $\mathcal{B}$ renders the projection of $\boldsymbol{y}$ to the ambient space a nonlinear transformation whenever $\boldsymbol{Ay} \notin \mathcal{B}$. Third, the optimization is done with k=4 separate projections, to improve the chances of generating an embedding that contains an optimum inside $[-\sqrt{d_e}, \sqrt{d_e}]^{d_e}$. Since these embeddings are independent, no data can be shared across them.

Extensions of REMBO Binois et al. (2015) consider the issue of non-injectivity, where the L^2 projection causes many points in the embedding to map to the same vertex of $\mathcal{B}$. They define a warped kernel that reduces non-injectivity. Binois et al. (2018) consider the issue of setting bounds on the embedding. They define a projection matrix $\boldsymbol{B} \in \mathbb{R}^{d \times D}$ that maps from the ambient space down to the embedding, and replace the L^2 projection with a projection γ that maps $\boldsymbol{y}$ to the closest point in $\mathcal{B}$ that satisfies $\boldsymbol{Bx} = \boldsymbol{y}$. The γ projection resolves the core challenge of REMBO related to setting bounds in the embedding: we can restrict the optimization in the embedding to points for which $\exists \boldsymbol{x} \in \mathcal{B}$ s.t. $\boldsymbol{Bx} = \boldsymbol{y}$, and so heuristic box bounds in the embedding are no longer required. The γ projection projects to the same points on the facets of $\mathcal{B}$ as the L^2 projection.

Binois (2015) studies different choices for the projection matrix and shows that BO performance can be improved for small d by sampling each row of $\boldsymbol{A}$ from the unit hypersphere $\mathbb{S}^{d_e-1}$. If $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{d_e})$, then $\frac{\boldsymbol{z}}{||\boldsymbol{z}||}$ is a random sample from $\mathbb{S}^{d_e-1}$, so this amounts to normalizing the rows of the usual REMBO projection matrix.

HeSBO (Nayebi et al., 2019) is a recent extension of REMBO that avoids clipping to $\mathcal{B}$ by changing the projection matrix $\boldsymbol{A}$. In $d_e = 1$, it is easy to see that the projection matrix $\boldsymbol{A} = \boldsymbol{1}$, which sets every $x_i = y$, is optimal. With this projection we can set bounds of $[-1, 1]$ on the embedding and there is no need for L^2 projections because every point in the embedding will map to a point in $\mathcal{B}$. HeSBO extends this to $d_e > 1$ by setting each row of $\boldsymbol{A}$ to have a single non-zero element, which is randomly set to ± 1. The column with the non-zero value is chosen uniformly at random. Thus, each parameter in the ambient space is mapped directly to a parameter in the embedding, $x_i = \pm y_j$, where j is sampled uniformly from $\{1, \ldots, d_e\}$ and $\pm$ is chosen uniformly at random. The embedding is given box bounds of $[-1, 1]^{d_e}$.

4 Challenges with Linear Embeddings

Heuristics for handling box bounds when utilizing linear embeddings introduce several issues that impact HDBO performance. We highlight one recent observation from Binois et al. (2018), along with two novel observations about how existing methods can make it difficult to learn high-dimensional surrogates.

Most points in the embedding map to the facets of $\mathcal{B}$. Fig. 1 shows the probability that an interior point in the embedding projects up to the interior of $\mathcal{B}$. This is measured empirically by sampling $\boldsymbol{y}$ uniformly at random from $[-\sqrt{d_e}, \sqrt{d_e}]^{d_e}$, sampling $\boldsymbol{A}$ with $\mathcal{N}(0, 1)$ entries, and then

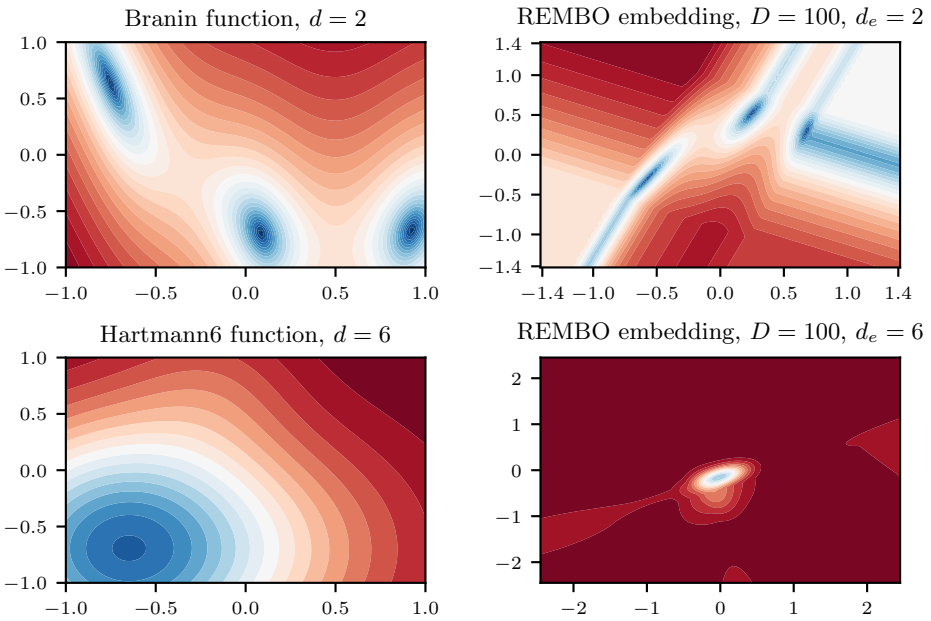

Figure 2: A visualization of REMBO embeddings for two test functions. *(Top left)* The Branin function, d=2, extended to D=100. *(Top right)* A REMBO embedding of the D=100 Branin function. *(Bottom left)* A center slice of the d=6 Hartmann6 function, similarly extended to D=100. *(Bottom right)* The same slice of a REMBO embedding of that function. The embedding produces distortions in the function that render it difficult to model.

checking if $\boldsymbol{Ay} \in \mathcal{B}$. Even for small D, with $d_e > 2$ practically all of the volume in the embedding projects up outside the box bounds, and is thus clipped to a facet of $\mathcal{B}$.

This is an issue because it means the optimization will be done primarily on the facets of $\mathcal{B}$ and not in the interior, which will likely not even be reached in a typical BO random initialization. We will see below that the function behaves very differently on points projected to the facets. The problem cannot be resolved by simply shrinking the box bounds in the embedding. Binois et al. (2018) provide an excellent study of the issue of setting bounds in the embedding and show that with the REMBO strategy there is no good way to do this. The projection of $\mathcal{B}$ down to the embedding produces a geometric object called a zonotope, a star-shaped object with up to $2\sum_{i=0}^{d-1}\binom{D-1}{i}$ vertices (Ferrez et al., 2005). Shrinking box bounds in the embedding cuts off the vertices of the zonotope and increases the chance of not containing an optimum.

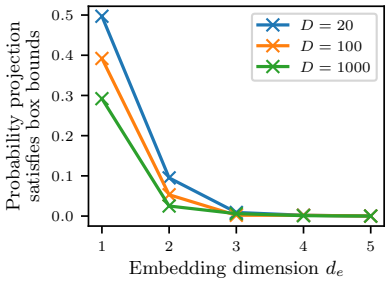

Figure 1: The probability that a randomly selected point in the REMBO embedding satisfies the ambient box bounds after being projected up. For $d_e > 2$, nearly all points in the embedding map outside the box bounds.

Projection to the facets of $\mathcal{B}$ produces a nonlinear distortion in the function. The function value at any point in the embedding is measured as $f(p_{\mathcal{B}}(\boldsymbol{Ay}))$ where, as described above, most points are being mapped to a facet by $p_{\mathcal{B}}$. This has a powerful, detrimental effect on the ability to model f in the embedding. Fig. 2 provides visualizations of an actual REMBO embedding for two classic test functions: the Branin (d=2) and Hartmann6 (d=6) functions, both extended to D=100 by adding unused variables. The REMBO embedding for the Branin function contains all three optima, however there is visible distortion to the function caused by the the clipping to $\mathcal{B}$. The embedding for the Hartmann6 function is even more heavily distorted.

Even if the function is well-modeled by a GP in the true low-dimensional space, the distortion produced by the REMBO projection transforms it into one on the embedding that is not appropriate for a GP. This can happen for any embedding strategy that cannot guarantee all points in the embedding project into $\mathcal{B}$. The distortion induced by mapping to the facet depends on the relative angles of the facet and the true embedding. Projection to a facet essentially induces a non-stationarity in the kernel: each of the $2D$ facets sits at different angles to the true subspace, and so the change in the rate of function variance will differ for each. To correct for the non-stationarity, we would have to estimate the true subspace $\boldsymbol{T}$, which with $d \times D$ entries is not feasible for D large.

The idea behind using low-dimensional embeddings for HDBO is that it enables the use of standard BO techniques on the embedding. However, from these results we see that for the REMBO projection with box bounds we cannot expect to successfully model the function on the embedding with a regular GP. The problem is especially acute for $d_e > 2$ where, as shown in Fig. 1, nearly all points in the embedding map to one of $2D$ facets.

Linear projections do not preserve product kernels. Although less visible than that produced by the projection to the facets, there is also distortion to interior points just from the linear projection $\boldsymbol{A}$. The ARD kernels typically used in GP modeling are product kernels that decompose the covariance into the covariance across each dimension. Inside the embedding, moving along a single dimension will move across all dimensions of the true subspace, at rates depending on the angles between the embedding and the true subspace. A product kernel in the true subspace will not produce a product kernel in the embedding; we will see this more explicitly in Sec. 5.1.

Linear embeddings can have a low probability of containing an optimum. HeSBO avoids the primary two challenges of BO in linear embeddings: all interior points in the embedding map to interior points of $\mathcal{B}$, and there is no longer any need for the L^2 projection and thus no distortion from projecting to facets. However, for $d_e > 1$ there is no guarantee that the embedding will contain an optimum, and in fact the probability of containing an optimum can be quite low. Consider the example of an axis-aligned true subspace: f operates only on some set of d elements of $\boldsymbol{x}$, which we denote $\mathcal{I} = \{i_1, \ldots, i_d\}$. For $d = 2$ and $d_e \geq 2$, there are three possible embeddings: x_{i_1} and x_{i_2} map to different features in the embedding, $x_{i_1} = x_{i_2}$, or $x_{i_1} = -x_{i_2}$. These three embeddings are visualized in Appendix A.1. In the first case the embedding successfully captures the entire true subspace and we can expect the optimization to be successful. However, in the other two cases the embedding is only able to reach the diagonals of the true subspace, which, unless f happens to have an optimum on the diagonal, will not reach the optimal value. Under a uniform prior on the location of optima, we can compute analytically the probability that the HeSBO embedding contains an optimum (see Appendix A.1). The probability is independent of D, but is low for even moderate values of d. For instance, with $d = 6$, $d_e = 20$ gives only a 44% chance of recovering an optimum.

Relative to REMBO, HeSBO improves the ability to effectively model and optimize in the embedding, but reduces the likelihood of the embedding containing an optimum. Empirically, this trade-off leads to HeSBO having better BO performance than REMBO. Like HeSBO, here we wish to eliminate the L^2 projection and thus improve our ability to model and optimize in the embedding. We will show that this can be done while maintaining a much higher chance of the embedding containing an optimum, which will further improve BO performance.

5 Learning and Optimizing in Linear Embeddings

We now show how to overcome the embedding issues described in Sec. 4. Similarly to Binois et al. (2018), we define the embedding via a matrix $\boldsymbol{B} \in \mathbb{R}^{d_e \times D}$ that projects from the ambient space down to the embedding, and $f_B(\boldsymbol{y}) = f(\boldsymbol{B}^\dagger \boldsymbol{y})$ as the function evaluated on the embedding. The new techniques we develop here are applicable to any linear embedding, not just random embeddings.

5.1 A Kernel for Learning in a Linear Embedding

As discussed in Sec. 4, a product kernel over dimensions of the true subspace (ARD) does not translate to a product kernel over dimensions in the embedding. However, stationarity in the true subspace does imply stationarity in the embedding, and this result gives the appropriate kernel structure.

Proposition 1. *Suppose the function on the true subspace is drawn from a GP with an ARD RBF kernel: $f_d \sim \mathcal{GP}(m(\cdot), k_{RBF}(\cdot,\cdot))$. For any pair of points in the embedding $\boldsymbol{y}$ and $\boldsymbol{y}'$,*

$$Cov[f_B(\boldsymbol{y}), f_B(\boldsymbol{y}')] = \sigma^2 \exp\left(-(\boldsymbol{y}-\boldsymbol{y}')^\top \boldsymbol{\Gamma}(\boldsymbol{y}-\boldsymbol{y}')\right) ,$$

where σ^2 is the kernel variance of f_d, and $\boldsymbol{\Gamma} \in \mathbb{R}^{d_e \times d_e}$ is symmetric and positive definite.

Proof. To determine the covariance in function values of points in the embedding, we first project up to the ambient space and then project down to the true subspace:

$$f_B(\boldsymbol{y}) = f(\boldsymbol{B}^\dagger \boldsymbol{y}) = f_d(\boldsymbol{T}\boldsymbol{B}^\dagger \boldsymbol{y}).$$

Then,

$$\begin{aligned}\mathrm{Cov}[f_B(\boldsymbol{y}), f_B(\boldsymbol{y}')] &= \mathrm{Cov}[f_d(\boldsymbol{T}\boldsymbol{B}^\dagger \boldsymbol{y}), f_d(\boldsymbol{T}\boldsymbol{B}^\dagger \boldsymbol{y})] \\ &= \sigma^2 \exp\left(-(\boldsymbol{T}\boldsymbol{B}^\dagger \boldsymbol{y} - \boldsymbol{T}\boldsymbol{B}^\dagger \boldsymbol{y}')^\top \boldsymbol{D}(\boldsymbol{T}\boldsymbol{B}^\dagger \boldsymbol{y} - \boldsymbol{T}\boldsymbol{B}^\dagger \boldsymbol{y}')\right) ,\end{aligned}$$

where $\boldsymbol{D} = \mathrm{diag}\left(\left[\frac{1}{2\ell_1^2}, \ldots, \frac{1}{2\ell_d^2}\right]\right)$. Let $\boldsymbol{\Gamma} = (\boldsymbol{T}\boldsymbol{B}^\dagger)^\top \boldsymbol{D}(\boldsymbol{T}\boldsymbol{B}^\dagger)$. Because $\boldsymbol{D}$ is positive definite, it follows that $\boldsymbol{\Gamma}$ is symmetric and positive definite. □

This kernel replaces the ARD Euclidean distance with a Mahalanobis distance, and so we refer to it as the Mahalanobis kernel. Similar kernels have been used for GP regression in other settings (Vivarelli & Williams, 1999; Snelson & Ghahramani, 2006). This result shows that the impact of the linear projection on the kernel can be correctly handled by fitting a $\frac{d_e(d_e+1)}{2}$-parameter distance metric rather than the typical d_e-parameter ARD metric. In Appendix A.2, we show that this kernel produces significantly better predictions in linear embeddings than an ARD kernel.

5.2 Avoiding Nonlinear Projections

The most significant distortions seen in Fig. 2 result from clipping projected points to $\mathcal{B}$. We can avoid this by constraining the optimization in the embedding to points that do not project up outside the bounds, that is, $\boldsymbol{B}^\dagger \boldsymbol{y} \in \mathcal{B}$. Let $\alpha(\boldsymbol{y})$ be the acquisition function evaluated in the embedding that we wish to optimize. We select the next point to evaluate by solving

$$\max_{\boldsymbol{y} \in \mathbb{R}^{d_e}} \alpha(\boldsymbol{y}) \quad \text{subject to} \quad -\mathbf{1} \leq \boldsymbol{B}^\dagger \boldsymbol{y} \leq \mathbf{1}. \tag{1}$$

Note that there are no box bounds on the embedding. The constraints added to the acquisition function optimization form a polytope, which can be efficiently handled with off-the-shelf optimization tools. Appendix A.3 provides visualizations of the embedding subject to these constraints.

5.3 The Probability the Embedding Contains an Optimum

Restricting the embedding with the constraints in (1) eliminates distortions from clipping to $\mathcal{B}$, but it also reduces the volume of the embedding and thus reduces the probability that the embedding contains an optimum. To understand the performance of BO in the linear embedding, it is critical to understand this probability, which we denote P_{opt}.

P_{opt} depends on where the optima are in the ambient space—for instance, an optimum at $\mathbf{0}$ will always be contained in the embedding. Suppose the true subspace has an optimum at $\boldsymbol{z}^*$, and let $\mathcal{O}(\boldsymbol{T}, \boldsymbol{z}^*) = \{\boldsymbol{x} : \boldsymbol{T}\boldsymbol{x} = \boldsymbol{z}^*\}$ be the set of optima in the ambient space. Let $\mathcal{E}(\boldsymbol{B}) = \{\boldsymbol{x} : \boldsymbol{B}^\dagger \boldsymbol{B}\boldsymbol{x} = \boldsymbol{x}\}$ be the points that can be evaluated from the embedding. If we have a prior over the locations of optima (that is, over $\boldsymbol{T}$ and $\boldsymbol{z}^*$) then we can compute P_{opt} as:

$$P_{\text{opt}} = \mathbb{E}_{\boldsymbol{B},\boldsymbol{T},\boldsymbol{z}^*}\left[\mathbf{1}_{\mathcal{O}(\boldsymbol{T},\boldsymbol{z}^*)\cap\mathcal{E}(\boldsymbol{B})\cap\mathcal{B}\neq\varnothing}\right]. \tag{2}$$

Importantly, $\mathcal{O}(\boldsymbol{T}, \boldsymbol{z}^*)$, $\mathcal{E}(\boldsymbol{B})$, and $\mathcal{B}$ are all polyhedra, so their intersection can be tested by solving a linear program (see Appendix A.4). The expectation can be estimated with Monte Carlo sampling from the prior over $\boldsymbol{T}$ and $\boldsymbol{z}^*$ and from the chosen generating distribution of $\boldsymbol{B}$.

For our analysis here, we give $\boldsymbol{T}$ a uniform prior over axis-aligned subspaces as described in Sec. 4, and we give z^* a uniform prior in that subspace. Under these uniform priors, we can evaluate (2) to

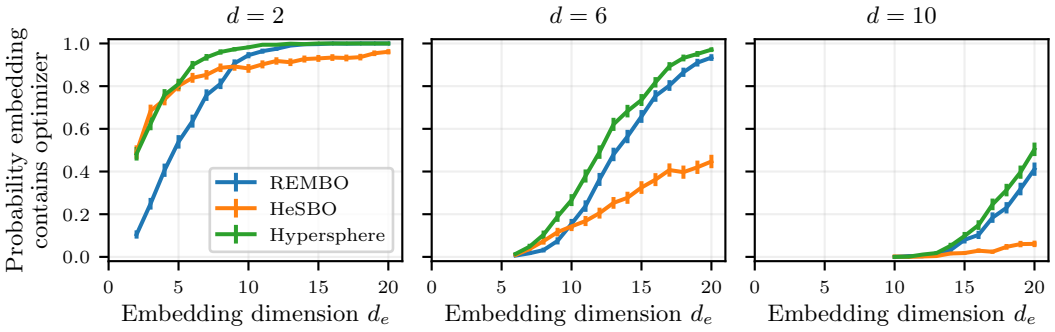

Figure 3: Probability the embedding contains an optimum (P_{opt}) when restricted to the constraints of (Eq. equation 1), under a uniform prior for the location of the optima and $D = 100$, for three embedding strategies. Setting $d_e > d$ rapidly increases P_{opt}, and high probabilities can achieved with reasonable values of d_e. Hypersphere sampling produces the best embedding, particularly for d small.

compute P_{opt} as a function of $\boldsymbol{B}$, D, d, and d_e. Fig. 3 shows these probabilities for $D = 100$ as a function of d and d_e, for three strategies for generating the projection matrix: the REMBO strategy of $\mathcal{N}(0, 1)$, the HeSBO projection matrix, and the unit hypersphere sampling described in Sec. 4. Increasing d_e above d rapidly improves the probability of containing an optimum. For $d = 6$, with $d_e = 6$ the probability is nearly 0, while increasing d_e to 12 is sufficient to raise it to 0.5 and with $d_e = 20$ it is nearly 1. Across all values of d and d_e, hypersphere sampling produces the embedding with the best chance of containing an optimum. Appendix A.4 shows P_{opt} for more values of D and d. By using hypersphere sampling and selecting $d_e > d$, we can maintain a high P_{opt} while still avoiding clipping to $\mathcal{B}$.

5.4 A New Method for BO with Linear Embeddings: ALEBO

We combine the results and insight gained into a new method for HDBO, which we call adaptive linear embedding BO (ALEBO), since the kernel metric and embedding bounds are adapted with the choice of $\boldsymbol{B}$. The approach is given in algorithm form in Appendix A.5, along with implementation details. In short, GP modeling in the embedding is done using the Mahalanobis kernel of Proposition 1. Uncertainty from the hyperparameters $\boldsymbol{\Gamma}$ is propagated into the posterior by sampling from a Laplace approximation posterior. The acquisition function is optimized over the polytope in (1). A high probability of the embedding containing an optimum is maintained by using hypersphere sampling for the projection matrix, and selecting $d_e > d$. Code is available at `github.com/anonymized-for-review`.

6 Benchmark Experiments

We evaluate the performance of ALEBO on synthetic HDBO tasks, and compare its performance to a broad selection of HDBO methods. We include in these benchmarks: REMBO and HeSBO; additive kernel methods Add-GP-UCB (Kandasamy et al., 2015) and Ensemble BO (EBO) (Wang et al., 2018); SMAC, which uses a random forest model; CMA-ES, an evolutionary strategy (Hansen et al., 2003); and quasirandom search (Sobol). For ALEBO we took $d_e = 2d$ for these experiments.

Fig. 4 shows BO performance on the Hartmann6 D=100 problem, averaged across 50 runs, each using an independently sampled projection matrix. Starting from about 75 iterations, ALEBO achieved the best optimization performance. For the other linear embedding approaches, we see that HeSBO does improve significantly over REMBO and was able to quickly reach values below -2, but then stalled. Relative to other methods, ALEBO had low variance in the final best-value, which is important in real applications where one can typically only run one optimization run.

Benchmark experiments were also run using the Branin problem and the Gramacy problem, which includes two black-box constraints that must also be modeled. The Branin optimization was done for d_e ranging from 2 to 8 and D from 50 to 1000, and we found that BO performance was not too

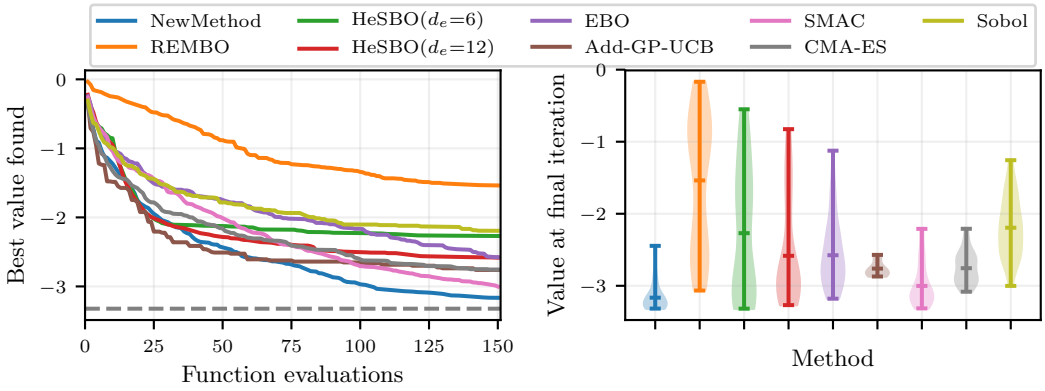

Figure 4: BO performance on the Hartmann6 problem ($d = 6$) extended to $D = 100$, for 9 methods. *(Left)* Best value by each iteration, averaged over 50 runs. *(Right)* Distribution of the best value at the final iteration. Starting at iteration 75, ALEBO achieved the best average performance with low variance in the final performance across runs.

sensitive to d_e or D. These results, along with more details of the benchmark methods and analysis of the results, are given in Appendix A.7.

7 Discussion

Our work highlights the importance of two basic requirements for an embedding to be useful for optimization that are often not examined critically by the literature: 1) the function must be well-modeled on the embedding; and 2) the embedding should contain an optimum. To the first point, we showed how polytope constraints on the embedding eliminate boundary distortions, and we derived a Mahalanobis kernel appropriate for GP modeling in a linear embedding. These two contributions allow effective modeling in the embedding space. To the second point, we developed an approach for computing the probability that the embedding contains an optimum, which we then used to construct embeddings with a higher chance of containing an optimum, via hypersphere sampling and selecting d_e larger than d.

These same two considerations are important for any embedding, not just linear. For instance, when constructing a VAE for BO, it will be equally important to ensure the function remains well-modeled on the embedding, and that bounds are not handled in a way that adds distortion. We must also ensure that the VAE embedding captures enough of the ambient space to have a high probability of containing an optimum. With linear embeddings we were able to derive analytical quantities for answering these questions—more work in this area will be needed for nonlinear embeddings.

Given D and d, we can solve (2) to determine the probability of containing an optimum for any d_e, and thus select d_e based on a desired target probability. We showed on test problems that BO performance was not too sensitive to the exact choice of d_e. In reality, of course, we do not know d, in which case selecting an appropriate embedding dimension remains an important open question.

References

Rika Antonova, Akshara Rai, and Christopher G Atkeson. Deep kernels for optimizing locomotion controllers. In *1st Conference on Robot Learning*, CoRL, pp. 47–56, 2017.

Mickaël Binois. *Uncertainty quantification on Pareto fronts and high-dimensional strategies in Bayesian optimization, with applications in multi-objective automotive design*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne, 2015.

Mickaël Binois, David Ginsbourger, and Olivier Roustant. A warped kernel improving robustness in Bayesian optimization via random embeddings. In *Proceedings of the International Conference on Learning and Intelligent Optimization*, LION, pp. 281–286, 2015.

Mickaël Binois, David Ginsbourger, and Olivier Roustant. On the choice of the low-dimensional domain for global optimization via random embeddings. *arXiv preprint arXiv:1704.05318*, 2018.

Roberto Calandra, André Seyfarth, Jan Peters, and Marc P. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence*, 76(1):5–23, 2015.

Paul G. Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to Kriging surfaces. *SIAM Journal on Scientific Computing*, 36:A1500–A1524, 2014.

Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional Gaussian process bandits. In *Advances in Neural Information Processing Systems 26*, NIPS, pp. 1025–1033, 2013.

David Eriksson, Kun Dong, Eric Hans Lee, David Bindel, and Andrew Gordon Wilson. Scaling Gaussian process regression with derivatives. In *Advances in Neural Information Processing Systems 31*, NIPS, pp. 6867–6877, 2018.

Jean-Albert Ferrez, Kornei Fukuda, and Th. M. Liebling. Solving the fixed rank convex quadratic maximization in binary variables by a parallel zonotope construction algorithm. *European Journal of Operational Research*, 166(1):35–50, 2005.

Jacob Gardner, Chuan Guo, Kilian Q. Weinberger, Roman Garnett, and Roger Grosse. Discovering and exploiting additive structure for Bayesian optimization. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, AISTATS, pp. 1311–1319, 2017.

Jacob R. Gardner, Matt J. Kusner, Zhixiang Xu, Kilian Q. Weinberger, and John P. Cunningham. Bayesian optimization with inequality constraints. In *Proceedings of the 31st International Conference on Machine Learning*, ICML, 2014.

Rafael Gómez-Bombarelli, Jennifer N. Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018.

Robert B. Gramacy, Genetha A. Gray, Sébastien Le Digabel, Herbert K. H. Lee, Pritam Ranjan, Garth Wells, and Stefan M. Wild. Modeling an augmented Lagrangian for blackbox constrained optimization. *Technometrics*, 58(1):1–11, 2016.

Nikolaus Hansen, Sibylle D. Mller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation*, 11(1):1–18, 2003.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, LION, pp. 507–523, 2011.

William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary mathematics*, 26(189–206):1, 1984.

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

Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional Bayesian optimisation and bandits via additive models. In *Proceedings of the 32nd International Conference on Machine Learning*, ICML, pp. 295–304, 2015.

Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. Constrained Bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

Chun-Liang Li, Kirthevasan Kandasamy, Barnabás Póczos, and Jeff Schneider. High dimensional Bayesian optimization via restricted projection pursuit models. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, AISTATS, pp. 884–892, 2016.

Daniel J. Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic gait optimization with Gaussian process regression. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI, pp. 944–949, 2007.

Xiaoyu Lu, Javier González, Zhenwen Dai, and Neil Lawrence. Structured variationally auto-encoded optimization. In *Proceedings of the 35th International Conference on Machine Learning*, ICML, pp. 3267–3275, 2018.

Jonas Mockus. *Bayesian approach to global optimization: theory and applications*. Mathematics and its applications: Soviet series. Kluwer Academic, 1989.

Riccardo Moriconi, K. S. Sesh Kumar, and Marc P. Deisenroth. High-dimensional Bayesian optimization with manifold Gaussian processes. *arXiv preprint arXiv:1902.10675*, 2019.

Mojmír Mutný and Andreas Krause. Efficient high dimensional Bayesian optimization with additivity and quadrature Fourier features. In *Advances in Neural Information Processing Systems 31*, NIPS, pp. 9005–9016, 2018.

Amin Nayebi, Alexander Munteanu, and Matthias Poloczek. A framework for Bayesian optimization in embedded subspaces. In *Proceedings of the 36th International Conference on Machine Learning*, ICML, pp. 4752–4761, 2019.

ChangYong Oh, Efstratios Gavves, and Max Welling. BOCK : Bayesian optimization with cylindrical kernels. In *Proceedings of the 35th International Conference on Machine Learning*, ICML, pp. 3868–3877, 2018.

Hong Qian, Yi-Qi. Hu, and Yang Yu. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI, 2016.

Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher G. Atkeson. Bayesian optimization using domain knowledge on the ATRIAS biped. In *Proceedings of the IEEE International Conference on Robotics and Automation*, ICRA, pp. 1771–1778, 2018.

Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional Bayesian optimization via additive models with overlapping groups. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, AISTATS, pp. 298–307, 2018.

Edward Snelson and Zoubin Ghahramani. Variable noise and dimensionality reduction for sparse Gaussian processes. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, UAI, pp. 461–468, 2006.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, NIPS, pp. 2951–2959, 2012.

Francesco Vivarelli and Christopher K. I. Williams. Discovering hidden features with Gaussian processes regression. In *Advances in Neural Information Processing Systems 11*, pp. 613–619, 1999.

Zi Wang, Chengtao Li, Stefanie Jegelka, and Pushmeet Kohli. Batched high-dimensional Bayesian optimization via structural kernel learning. In *Proceedings of the 34th International Conference on Machine Learning*, ICML, pp. 3656–3664, 2017.

Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale Bayesian optimization in high-dimensional spaces. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, AISTATS, 2018.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

A APPENDIX

This appendix contains a number of additional results and analyses to supplement the main text.

A.1 HESBO EMBEDDINGS

We consider HeSBO embeddings in the case of a random axis-aligned true subspace, and a uniform prior on the location of the optimum within that subspace. As explained in Sec 4, with $d = 2$ and this prior, regardless of d_e or D there are three possible embeddings: (1) Each of the active parameters are captured by a parameter in the embedding; (2) the embedding is constrained to the diagonal $x_{i_1} = x_{i_2}$; or (3) the embedding is constrained to the diagonal $x_{i_1} = -x_{i_2}$. Fig. 5 shows these three embeddings for the Branin problem from the top row of Fig. 2.

Within the first embedding, the optimal value of 0.398 can be reached. Within the second, the best value is 0.925 and within the third it is 17.18. Under a uniform prior on location of the optimum within a random axis-aligned true subspace, it is easy to compute the probability that the HeSBO embedding contains an optimum:

$$P_{\text{opt}}(d_e) = \frac{d_e!}{(d_e - d)! d_e^d}. \tag{3}$$

For $d = 2$, this is exactly the probability of the first embedding shown in Fig. 5. This probability increases with d_e, and is exactly the probability shown in Fig. 3.

A.2 THE MAHALANOBIS KERNEL

When fitting the Mahalanobis kernel derived in Proposition 1, we use an approximate Bayesian treatment of $\mathbf{\Gamma}$ to improve model performance while still maintaining tractability. We propagate uncertainty in $\mathbf{\Gamma}$ into the GP posterior by first constructing a posterior for $\mathbf{\Gamma}$ using a Laplace approximation with a diagonal Hessian, and then drawing m samples from that posterior. The marginal posterior for $f(\boldsymbol{y})$ can then be approximated as:

$$p(f(\boldsymbol{y})) \approx \frac{1}{m} \sum_{i=1}^{m} p(f(\boldsymbol{y}) | \mathbf{\Gamma}^i).$$

Because of the GP prior, each conditional posterior $p(f(\boldsymbol{y})|\mathbf{\Gamma}^i)$ is a normal distribution with known mean μ_i and variance σ_i^2. Thus the posterior $p(f(\boldsymbol{y}))$ is a mixture of Gaussians, which we can approximate using moment matching:

$$p(f(\boldsymbol{y})) \approx \mathcal{N}\left(\frac{1}{m} \sum_{i=1}^{m} \mu_i, \frac{1}{m} \sum_{i=1}^{m} \sigma_i^2 + \text{Var}_i[\mu_i] \right).$$

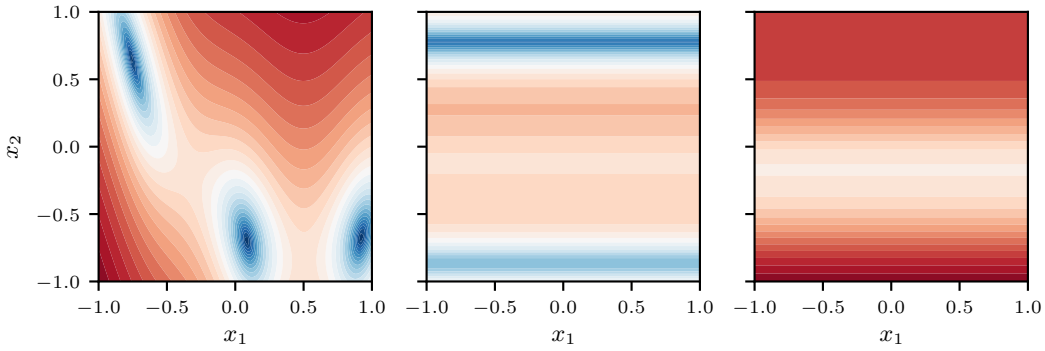

Figure 5: Three possible HeSBO embeddings of the $d = 2$ Branin function. *(Left)* The first embedding fully captures the function, and thus captures all three optima. *(Middle)* The second is restricted to the subspace $x_1 = -x_2$. This subspace does not contain an optimum, but comes fairly close. *(Right)* The third embedding is restricted to the subspace $x_1 = x_2$ and does not come close to any optima.

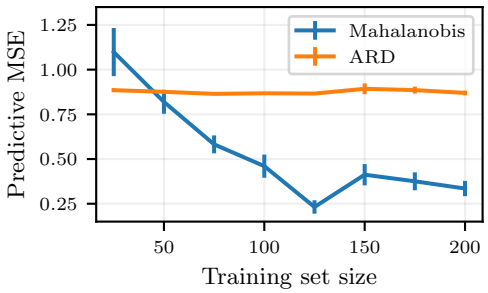

Figure 6: Empirical learning curves on the Hartmann6 $D = 100$ problem, with a $d_e = 12$ embedding restricted to the polytope of (1). Figure shows test MSE averaged across repeated trials, and error bars show 2 standard errors. An ARD kernel is not able to make accurate predictions in the embedding, while the Mahalanobis kernel is.

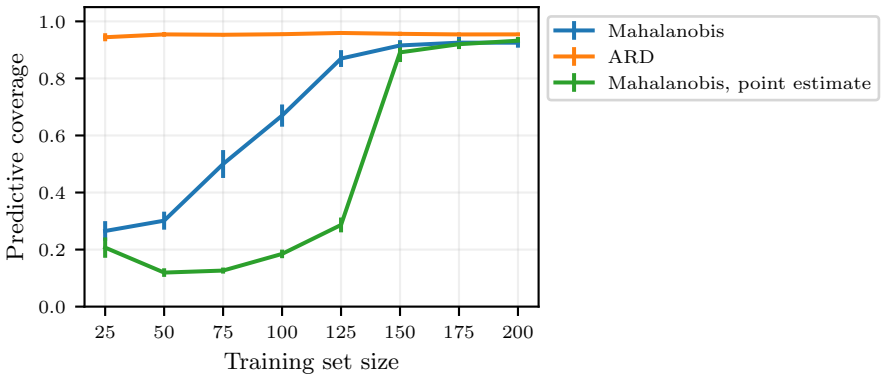

Figure 7: For the same simulation as Fig. 6, coverage on test set predictions, averaged over random training sets with error bars showing two standard errors. Nominal coverage is 95%. Uncertainty intervals are over-conservative, but are much better with approximate posterior sampling of $\mathbf{\Gamma}$ (Mahalanobis) than with using a point estimate for $\mathbf{\Gamma}$ (Mahalanobis, point estimate).

The Mahalanobis kernel significantly improves modeling performance inside the linear embedding, particularly for larger values of d. Here we show this using data from the Hartmann6 $D = 100$ function from Fig. 2 to construct empirical learning curves. We used hypersphere sampling to produce a projection matrix $\boldsymbol{B}$ with $d_e = 12$ (the value used in the BO benchmarks for this problem in Sec. 6). We generated random training and test sets from within the embedding (that is, within the polytope given by (1)) using rejection sampling. The test set was kept fixed, but across trials different training sets with varying sizes were generated. In each trial, we fit a GP to the training set and used it to make predictions on the test set, and computed predictive mean squared error (MSE). These were averaged across 25 trials for each training set size to produce the learning curves in Fig. 6.

Fig. 6 shows learning curves for GPs with two kernels: the Mahalanobis kernel given in Proposition 1, and a regular ARD kernel. Despite only being in $d_e = 12$ dimensions, the ARD kernel was unable to learn in the embedding, whereas the Mahalanobis was. This shows the importance of having the correct kernel for the embedding.

Handling uncertainty in $\mathbf{\Gamma}$ is critical for getting reasonable posterior uncertainty. Fig. 7 shows this using the same simulation as Fig. 6, this time measuring coverage on the test set predictions. The figure shows that the Bayesian treatment of $\mathbf{\Gamma}$ led to significantly better coverage for smaller training set sizes. In BO exploration is driven by model uncertainty, so well-calibrated uncertainty intervals are especially important.

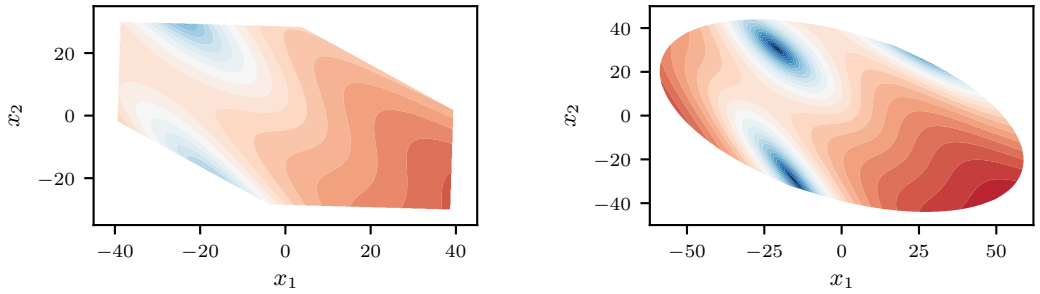

Figure 8: *(Left)* An embedding from a $\mathcal{N}(0, 1)$ projection matrix on the same Branin $D = 100$ problem from Fig. 2 subject to constraints of (1). *(Right)* The embedding from the same projection matrix after normalizing the columns to produce unit circle samples. Sampling from the unit circle increases the probability that an optimum will fall within the embedding.

A.3 Polytope Bounds on the Embedding

Rather than using projections to the box bounds $\mathcal{B}$, we specify polytope constraints in (1). Fig. 8 illustrates the embedding with these constraints for the same Branin $D = 100$ problem from the top row of Fig. 2. The embedding in the left figure was created with the REMBO strategy of sampling each entry from $\mathcal{N}(0, 1)$. For the embedding in the right figure, that same projection matrix had each column normalized. This converts the projection matrix to be a sample from the unit circle, as described in Sec. 4.

The $\mathcal{N}(0, 1)$ embedding does not contain any optima within the polytope bounds. Converting that projection matrix to a hypersphere sample rounds out the vertices of the polytope and expands the space to capture two of the optima. Consistent with Fig. 3, we see that hypersphere sampling significantly improves the chances of the embedding containing an optimum.

A.4 Evaluating the Probability the Embedding Contains an Optimum

As in other parts of the paper, we consider a uniform prior on the location of the optimum within a random axis-aligned subspace. A random true projection matrix $\boldsymbol{T}$ is sampled by selecting d columns at random and setting each to one of the d-dimensional unit vectors. $\boldsymbol{z}^*$ is then sampled uniformly at random from $[-1, 1]^d$. $\boldsymbol{B}$ is sampled according to the desired strategy, which in our experiments was REMBO, HeSBO, or hypersphere. Given these three quantities, we can evaluate whether or not $\boldsymbol{B}$ contains an optimum subject to the constraints of (1) by solving the following linear program:

$$
\begin{aligned}
&\text{maximize } \mathbf{0}^\top \boldsymbol{x} \\
&\text{subject to } \boldsymbol{T}\boldsymbol{x} = \boldsymbol{z}^*, \\
&\qquad\qquad (\boldsymbol{B}^\dagger \boldsymbol{B} - \boldsymbol{I})\boldsymbol{x} = \mathbf{0}, \\
&\qquad\qquad \boldsymbol{x} \geq -\mathbf{1}, \\
&\qquad\qquad \boldsymbol{x} \leq \mathbf{1}.
\end{aligned}
$$

If this problem is feasible, then the embedding produced by $\boldsymbol{B}$ contains an optimum. If it is infeasible, then it does not. Solving this over many draws of $\boldsymbol{T}$, $\boldsymbol{z}^*$, and $\boldsymbol{B}$ produces an estimate of P_{opt} under that prior for the location of optima. Here we used a uniform prior, but this linear program can be taken to compute P_{opt} under any prior.

Fig. 9 shows P_{opt} for a wide range of values of d and D, for hypersphere sampling. Across this wide range we see that for many values of d we can achieve high values of P_{opt} with reasonable values of d_e.

A.5 The ALEBO Algorithm

The steps of the The ALEBO method are given in Algorithm 1.

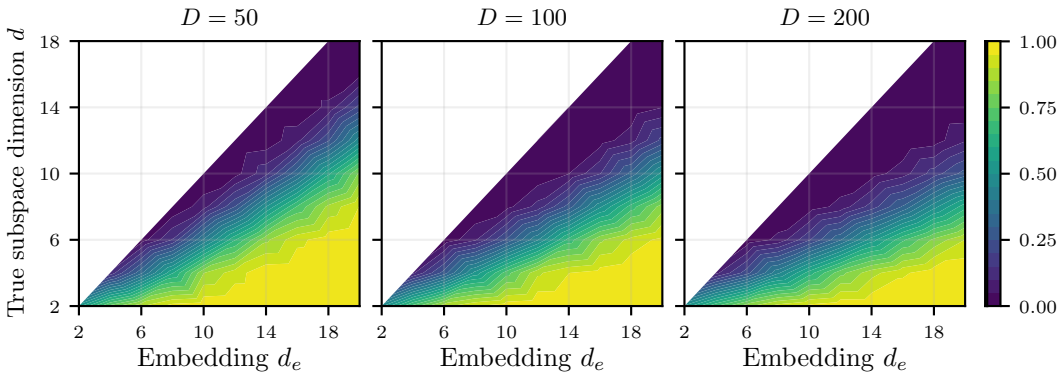

Figure 9: P_{opt} for hypersphere sampling, as estimated in Fig. 3 but here for a wider range of values of d and D. Contour color indicates P_{opt}. Doubling D decreases P_{opt} for d and d_e fixed, however even at $D = 200$, high values of P_{opt} with reasonable values of d_e can be had for many values of d.

Algorithm 1: ALEBO method for high-dimensional BO in a linear embedding.

Data: D, d_e, n_{init}, n_{BO}.
Result: Approximate optimizer $\boldsymbol{x}^*$.

Generate a random projection matrix $\boldsymbol{B}$ by sampling D points from the hypersphere $\mathbb{S}^{d_e-1}$.
Generate n_{init} random points $\boldsymbol{y}^i$ in the embedding using rejection sampling to satisfy polytope (1).
Let $\mathcal{D} = \{(\boldsymbol{y}^i, f(\boldsymbol{B}^\dagger \boldsymbol{y}^i)\}_{i=1}^{n_{\text{init}}}$ be the initial data.
for $j = 1, \ldots, n_{BO}$ **do**
    Fit a GP by maximizing marginal log-likelihood of $\mathcal{D}$, with the Mahalanobis kernel.
    Draw posterior samples of $\boldsymbol{\Gamma}$ using a Laplace approximation. Marginalize over the posterior with moment matching.
    Use the GP to find $\boldsymbol{y}^j$ that maximizes the acquisition function according to (1).
    Update $\mathcal{D}$ with $(\boldsymbol{y}^j, f(\boldsymbol{B}^\dagger \boldsymbol{y}^j))$
return $\boldsymbol{B}^\dagger \boldsymbol{y}^*$, *for the best point* $\boldsymbol{y}^*$.

In Line 1 the embedding is specified by generating a random projection matrix. We use hypersphere sampling, which gave the best P_{opt} in Fig. 3 among strategies tried here. This could be replaced with a different strategy for constructing a projection matrix should one be more appropriate for a particular setting.

A.6 Handling Black-Box Constraints in High-Dimensional Bayesian Optimization

In many applications of BO, in addition to the black-box objective f there are black-box constraints c_j and we seek to solve the optimization problem

$$\begin{aligned} &\text{minimize } f(\boldsymbol{x}) \\ &\text{subject to } c_j(\boldsymbol{x}) \leq 0, \quad j = 1, \ldots, J, \\ &\qquad \boldsymbol{x} \in \mathcal{B}. \end{aligned}$$

In most settings the constraint functions c_j are evaluated simultaneously with the objective f. Constraints are typically handled in BO by fitting a separate GP to each outcome (that is, to f and to each c_j). The acquisition function is then modified to consider not only the objective value but also whether the constraints are likely to be satisfied (*e.g.*, Gardner et al., 2014).

The extension of BO in an embedding to constrained BO is straightforward, so long as the same embedding is used for every outcome. A separate GP (in our case, using the Mahalanobis kernel) is fit to data from each outcome. Because the embedding is shared, predictions can be made for all of the outcomes at any point in the embedding. This allows us to evaluate and optimize an acquisition function for constrained BO in the embedding. Once a point is selected, it is projected up to the ambient space and evaluated on f and each c_j as usual. Random projections are especially well-suited for constrained BO because there is no harm in requiring the same projection for all outcomes, since it is a random projection anyway.

A.7 Additional Experimental Results

Here we provide results from two additional problems (Branin and Gramacy), and provide a study of the sensitivity of BO performance to d_e and D. We also provide implementation details for the experiments.

A.7.1 Method Implementations and Experiment Setup

The linear embedding methods (REMBO, HeSBO, and ALEBO) were all implemented using BoTorch, a framework for BO in PyTorch, and so used the same acquisition functions and the same tooling for optimizing the acquisition function. EI was the acquisition function for the Hartmann6 and Branin benchmarks, and NEI (Letham et al., 2019) was used to handle the constraints in the Gramacy problem. ALEBO and HeSBO were given a quasirandom initialization of 10 points from a scrambled Sobol sequence. REMBO was given a Sobol initialization of 2 points for each of its 4 projections used within a run.

The remaining methods used reference implementations from their authors with default settings for the package: EBO[1], Add-GP-UCB [2], SMAC[3], and CMA-ES[4]. EBO requires an estimate of the best function value, and for each problem was given the true best function value. SMAC and CMA-ES require an initial point, and were given the point at the center of the ambient space box bounds.

For all three problems, the optimization was repeated 50 times for each method, with the exception of Add-GP-UCB which was significantly slower than other methods (each Hartmann6 optimization took $\sim$17 hours) and so was repeated only 10 times. The function evaluations for all problems were noiseless, so the stochasticity throughout the run and in the final value all comes from stochasticity in the methods themselves. For linear embedding methods the main sources of stochasticity are in generating the random projection matrix and in the quasirandom initialization.

[1] `github.com/zi-w/Ensemble-Bayesian-Optimization`

[2] `github.com/dragonfly/dragonfly`, with option `acq="add_ucb"`

[3] `github.com/automl/SMAC3`, `SMAC4AC` mode

[4] `github.com/CMA-ES/pycma`

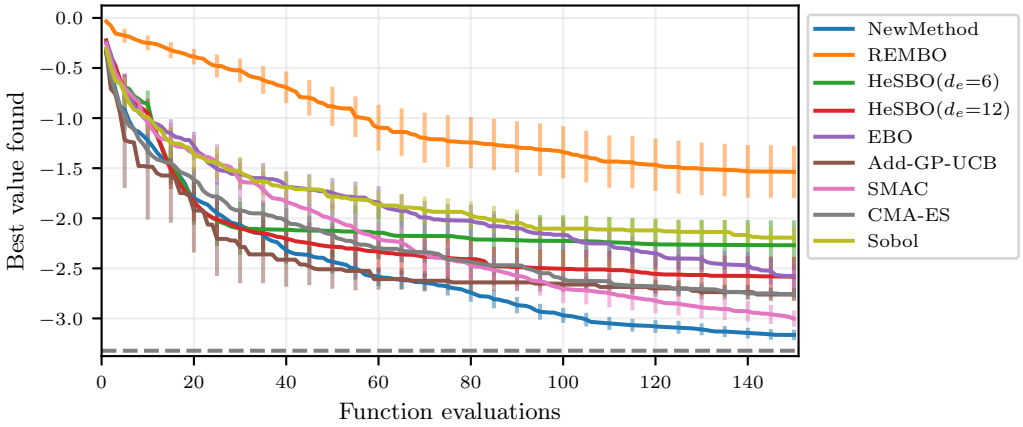

Figure 10: The same results from Fig. 4, with error bars showing two standard errors. Starting at about iteration 75, ALEBO significantly outperformed other approaches for HDBO.

Nayebi et al. (2019) also used the Hartmann6 and Branin functions with axis-aligned embeddings to evaluate HeSBO. They used $d_e = 4$ for Branin and $d_e = 6$ for Hartmann6. In our comparisons here we used those same choices, but for Hartmann6 additionally evaluated HeSBO with $d_e = 12$ to provide a more direct comparison to ALEBO, for which we used $d_e = 12$.

A.7.2 Hartmann6 Benchmark Problem

Fig. 10 shows the same results from Fig. 4, except with standard errors for the values across optimization iterations. REMBO performed worse than quasirandom on this problem, despite there being a true linear subspace that satisfies the REMBO assumptions. The source of the poor performance is the poor representation of the function on the embedding illustrated in Fig. 2. The remaining methods all performed better than quasirandom. CMA-ES was competitive with all of the methods except SMAC and ALEBO, which is somewhat surprising since it is not designed to have the same degree of sample efficiency as BO methods. HeSBO and Add-GP-UCB both did very well early on, but then got stuck and did not progress significantly after about iteration 50. SMAC continued to improve throughout the entire optimization, as did ALEBO, which achieved the best performance.

Fig. 11 shows the average time required to generate a candidate (fit the model and optimize the acquisition function) per iteration of the optimization. Even with the additional parameters in the Mahalanobis kernel and the added linear constraints to the acquisition function optimization, BO with ALEBO required similar time as other methods for high-dimensional BO. The average of 25s per iteration is short relative to the function evaluation time of typical resource-intensive BO applications.

A.7.3 Branin Benchmark Problem

BO results for the Branin problem are shown in Fig. 12. The results are qualitatively similar to those seen with the Hartmann6 function, except here REMBO outperformed quasirandom, and in fact outperformed HeSBO as well, and matched CMA-ES. The additive kernel methods and SMAC all performed similarly, and, starting from around iteration 20, ALEBO performed the best. The distribution of final iterations shows that in one iteration the ALEBO embedding did not contain an optimum and so achieved a final value near 10. However, across all 50 runs nearly all achieved a value very close to the optimum, leading to the best average performance.

The poor performance of HeSBO on this problem can be attributed entirely to the embedding not containing an optimum. Recall that for this problem there are exactly three possible HeSBO embeddings, which are shown in Fig. 5. As explained in Appendix A.1, the first embedding contains the optimum of 0.398, while the best value in the other embeddings are 0.925 and 17.18. Thus, if the BO were able to find the true optimum within each embedding with the budget of 50 function

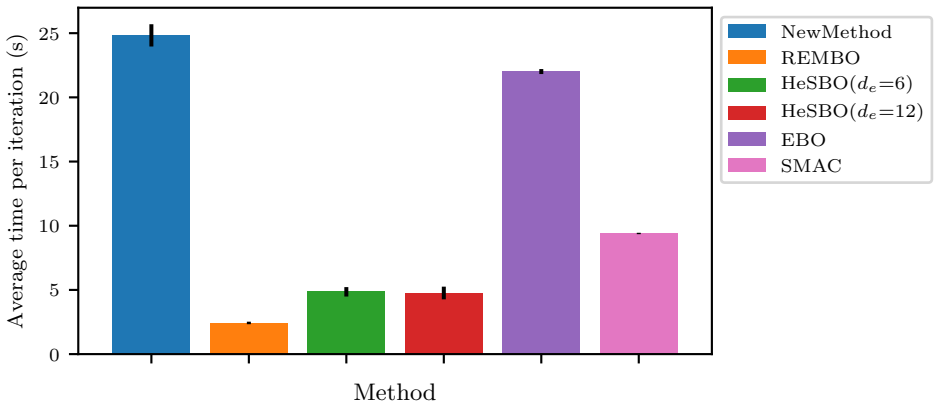

Figure 11: Running times for each method on the Hartmann6 benchmark test of Fig. 4, as average time per iteration across all 150 iterations and across the 50 optimization runs. Error bar shows two standard errors across runs. Not shown are: CMA-ES and Sobol, which had running times of less than 0.1s per iteration, and Add-GP-UCB, which required 400s per iteration.

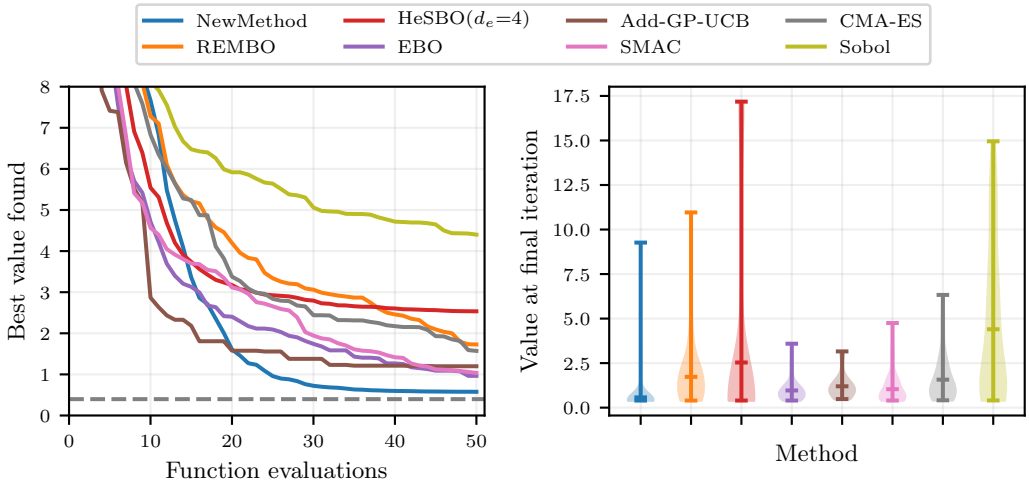

Figure 12: BO performance on the Branin problem ($d = 2$) extended to $D = 100$ with unused parameters. ALEBO showed the best performance after 20 iterations. HeSBO performed poorly because of the third embedding in Fig. 5 which has an optimal value of 17.18, and a 12.5% chance of being selected.

evaluations given in this experiment, the expected best value found by HeSBO would be:

$$0.398P_{\text{opt}} + 0.925\left(\frac{1-P_{\text{opt}}}{2}\right) + 17.18\left(\frac{1-P_{\text{opt}}}{2}\right).$$

This is the best average performance one can hope to achieve using the HeSBO embedding on this problem. Using (3) we can compute P_{opt} for $d_e = 4$ as 0.75, and it follows that the HeSBO expected best value is 2.56. This is nearly exactly the average best-value shown in Fig. 12. The poor performance of HeSBO is thus not related to BO, but comes entirely from the 12.5% chance of generating an embedding whose optimal value is 17.18. The presence of these embeddings can be clearly seen in the distribution of final best values in the figure.

A.7.4 Gramacy Benchmark Problem

The Gramacy problem is a toy-problem from Gramacy et al. (2016) with $d = 2$ that includes two black-box constraints. The problem has 3 local optima and a single global optimum. For the linear

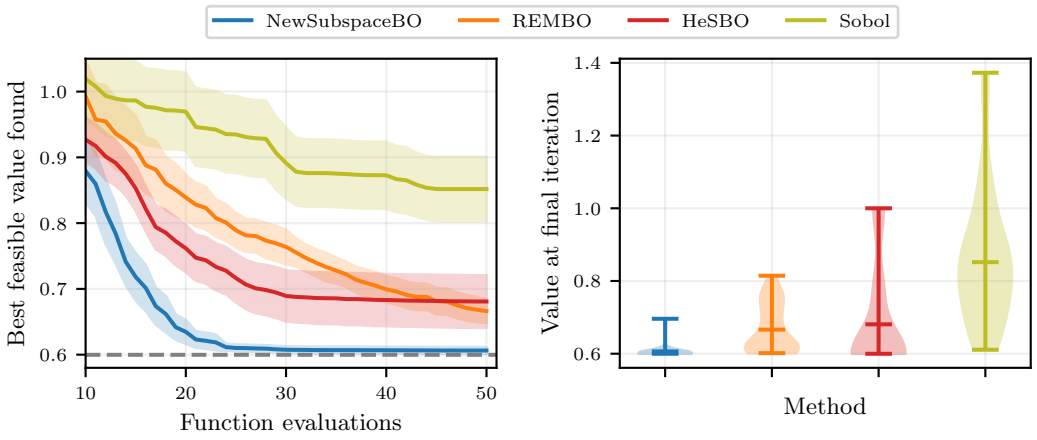

Figure 13: BO performance on the Gramacy problem, which includes black-box constraints. *(Left)* Best value reached that satisfied the constraints, averaged over 50 runs with two standard errors shaded. *(Right)* the distribution of best feasible value found across the entire optimization. ALEBO can effectively optimize with black-box constraints.

embedding methods, constrained optimization was done as described in Appendix A.6. ALEBO and HeSBO both used $d_e = 4$. The other optimization methods (EBO, Add-GP-UCB, SMAC, and CMA-ES) do not support black-box constraints and so they are not included in this comparison.

Optimization results are shown in Fig. 13, starting at iteration 10 where all methods had evaluated at least one feasible point in each of the 50 optimization runs. The results are qualitatively similar to those seen on the other problems. All BO methods outperformed Sobol. HeSBO did well initially but average performance stalled due to the 25% chance of the embedding missing an optimum, and by iteration 50 REMBO performance had caught up. ALEBO performed the best and reached values very close to optimal after only 25 iterations.

A.7.5 Performance as a Function of Embedding and Ambient Dimensions

We study sensitivity of BO performance to the embedding dimension d_e and the ambient dimension D using the Branin function. To test dependence on d_e, for $D = 100$ we ran 50 optimization runs for each of $d_e \in \{2, 3, 4, 5, 6, 7, 8\}$. To test dependence on D, for $d_e = 4$ we ran 50 optimization runs with ALEBO for each of $D \in \{50, 100, 200, 500, 1000\}$. Note that the $d_e = 4$ and $D = 100$ case in each of these is exactly the optimization problem of Fig. 12. The results are shown in Fig. 14.

For $d_e = d$, optimization performance was poor. From Fig. 3 we know this is because there is a low probability of the embedding containing an optimizer. Increasing d_e increases that probability, but also increases the dimensionality of the embedding and thus reduces the sample efficiency of the BO in the embedding. This trade-off can be clearly seen in the figure, where performance improved until $d_e = 5$ and then started to degrade. Fortunately, the degradation is not substantial: even at $d_e = 8$ ALEBO significantly outperformed all of the comparison methods in Fig. 12.

Optimization performance is similarly stable when varying the ambient dimension D and only degrades slightly from $D = 50$ to $D = 1000$. Even at $D = 1000$, ALEBO had better performance than the comparison methods had on $D = 100$.

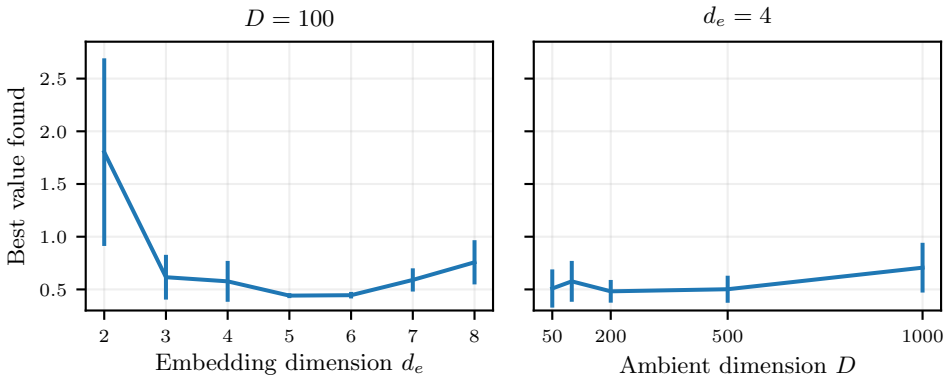

Figure 14: Final best value for the Branin problem (Fig. 12) for ALEBO *(Left)* for ambient dimension $D = 100$ as a function of embedding dimension d_e, and *(Right)* for $d_e = 4$ as a function of D. BO performance was consistently good for a wide range of values of d_e and D.