BOCK : Bayesian Optimization with Cylindrical Kernels

ChangYong Oh [1] **Efstratios Gavves** [1] **Max Welling** [1 2]

Abstract

A major challenge in Bayesian Optimization is the *boundary issue* (Swersky, 2017) where an algorithm spends too many evaluations near the boundary of its search space. In this paper we propose BOCK, Bayesian Optimization with Cylindrical Kernels, whose basic idea is to transform the ball geometry of the search space using a cylindrical transformation. Because of the transformed geometry, the Gaussian Process-based surrogate model spends less budget searching near the boundary, while concentrating its efforts relatively more near the center of the search region, where we expect the solution to be located. We evaluate BOCK extensively, showing that it is not only more accurate and efficient, but it also scales successfully to problems with a dimensionality as high as 500. We show that the better accuracy and scalability of BOCK even allows optimizing modestly sized neural network layers, as well as neural network hyperparameters.

1. Introduction

When we talk about stars and galaxies we use *parsecs* to describe structures, yet when we discuss the world around us we use *meters*. In other words, the natural lengthscale scale with which we describe the world increases with distance away from us. We believe this same idea is useful when performing optimization in high dimensional spaces.

In Bayesian Optimization (or other forms of hyperparameter optimization) we define a cube or a ball and search for the solution inside that volume. The origin of that sphere is special in the sense that this represents the part of space with the highest probability if finding the solution. Moreover, in high dimensions, when we move outwards, the amount of volume contained in an annulus with width δR, $A(\mathbf{c}; R - \delta R, R) = \{\mathbf{x} | R - \delta R < \|\mathbf{x} - \mathbf{c}\| < R\}$, grows exponentially with distance R. As such, if we would spend an equal amount of time searching each volume element δV, we would spend all our time at the boundary of our search region. This effective attraction to the places with more volume is the equivalent of an "entropic force" in physics, and in the case of optimization is highly undesirable, since we expect the solution at a small radius R.

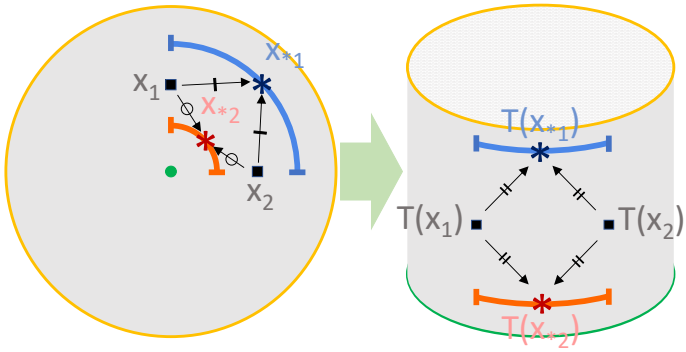

Figure 1. Many of the problems in Bayesian Optimization relate to the *boundary issue* (too much volume near the boundary (Swersky, 2017)), because of the Euclidean geometry of the search space ball. Because of the boundary issue, we spend much of the evaluation budget in a particular region of the search space, the boundaries, which contradicts our *prior assumption* that the solution most likely lies close to the origin. We propose BOCK, whose basic idea is to apply a cylindrical transformation of the search space that expands the volume near the ball center while contracting the volume near the boundaries.

In this paper we, therefore, reformulate Bayesian Optimization in a transformed space, where a ball, $B(\mathbf{x}; R) = \{\mathbf{x} | \|\mathbf{x} - \mathbf{c}\| \leq R\}$, is mapped to a cylinder, $C(p, q; \mathbf{c}, L = \{(r, \mathbf{a}) | r \in [p, q], \|\mathbf{d} - \mathbf{c}\| = L\}$ (see Figure 1). In this way, every annulus of width δR contains an equal amount of volume for every radius R, and the entropic force pulling the optimizer to the boundary disappears. We call our method BOCK, for *Bayesian Optimization with Cylindrical Kernel*. We find that our algorithm is able to successfully handle much higher dimensional problems than standard Bayesian optimizers. As a result, we manage to not only optimize modestly sized neural network layers (up to 500 dimensions in our experiments), obtaining solutions competitive to SGD training, but also hyper-optimize stochastic depth Resnets (Huang et al., 2016).

[1]QUvA Lab, Informatic Institute, University of Amsterdam, Amsterdam, Netherlands [2]Canadian Institute for Advanced Research, Toronto, Canada. Correspondence to: ChangYong Oh <C.Oh@uva.nl>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

Algorithm 2 Bayesian Optimization pipeline.

Input: surrogate model $\mathcal{M}$, acquisition function α, search space X, initial training data $\mathcal{D}_{init}$, function f
Output: optimum $\mathbf{x}_{opt} \in X$ of f
Initialize $\mathcal{D} = \mathcal{D}_{init}$
while evaluation budget available **do**
  Set $\mu(\cdot|\mathcal{D}), \sigma^2(\cdot|\mathcal{D}) \leftarrow \mathcal{M}|\mathcal{D}$ *// Surrogate function returns predictive mean function and predictive variance function by fitting $\mathcal{M}$ to $\mathcal{D}$*
  Maximize $\widehat{\mathbf{x}} = \arg\max_{\mathbf{x} \in X} \alpha(\mu(\mathbf{x}\,|\mathcal{D}), \sigma^2(\mathbf{x}\,|\mathcal{D}))$ *// Acquisition function suggests next evaluation by maximization*
  Evaluate $\hat{y} = f(\widehat{\mathbf{x}})$ *// Evaluate the score of the point selected by the acquisition function*
  Set $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\widehat{\mathbf{x}}, \hat{y})\}$ *// Update the training dataset by including the newly evaluated pair $(\widehat{\mathbf{x}}, \hat{y})$*
end while

2. Background

2.1. Bayesian Optimization

Bayesian optimization aims at finding the global optimum of black-box functions, namely

$$\mathbf{x}_{opt} = \arg\min_{\mathbf{x}} f(\mathbf{x}) \tag{1}$$

The general pipeline of Bayesian Optimization is given in Alg. 1. Prior to starting, a search space must be defined, where the optimum $f(\mathbf{x}_{opt})$ will be searched for. Given this search space, the initial training dataset must be set, typically by naive guessing where the solution might lie or by informed expert knowledge of the problem. Having completed these two steps, Bayesian Optimization proceeds in an iterative fashion. At each round, in the absence of any other information regarding the nature of $f(\mathbf{x})$ a surrogate model attempts to approximate the behavior of $f(\mathbf{x})$ based on the so far observed points $(\mathbf{x}_i, y_i), y_i = f(\mathbf{x}_i)$. The surrogate function is then followed by an acquisition function that suggests the next most interesting point $\mathbf{x}_{i+1}$ that should be evaluated. The pair $(\mathbf{x}_i, y_i)$ is added to the training dataset, $\mathcal{D} = \mathcal{D} \cup (\mathbf{x}_i, y_i)$, and the process repeats until the optimization budget is depleted.

The first design choice of the Bayesian Optimization pipeline is the surrogate model. The task of the surrogate model is to model probabilistically the behavior of $f(\cdot)$ in the x-space in terms of *(a)* a predictive mean $\mu(\mathbf{x}_* \,|\, \mathcal{D})$ that approximates the value of $f(x)$ at any point $\mathbf{x}_*$, and *(b)* a predictive variance that represents the uncertainty of the surrogate model in this prediction. Any model that can provide a predictive mean and variance can be used as a surrogate model, including random forests (Hutter et al., 2011), tree-based models (Bergstra et al., 2011) and neural networks (Snoek et al., 2015; Springenberg et al., 2016). Among other things, Gaussian Processes not only provide enough flexibility it terms of kernel design but also allow for principled and tractable quantification of uncertainty (Rasmussen & Williams, 2006). Therefore, we choose Gaussian Processes as our surrogate model. The predictive mean and the predictive variance of Gaussian processes are given as below

$$\mu(\mathbf{x}_* \,|\, \mathcal{D}) = K_{*\mathcal{D}}(K_{\mathcal{D}\mathcal{D}} + \sigma^2 I)^{-1}\mathbf{y} \tag{2}$$

$$\sigma^2(\mathbf{x}_* \,|\, \mathcal{D}) = K_{**} - K_{*\mathcal{D}}(K_{\mathcal{D}\mathcal{D}} + \sigma^2_{obs} I)^{-1} K_{\mathcal{D}*} \tag{3}$$

where $K_{**} = K(\mathbf{x}_*, \mathbf{x}_*)$, $K_{*\mathcal{D}}$ is a row vector whose ith entry is $K(\mathbf{x}_*, \mathbf{x}_i)$, $K_{\mathcal{D}*} = (K_{*\mathcal{D}})^T$, $[K_{\mathcal{D}\mathcal{D}}]_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$, σ^2_{obs} is the variance of observational noise and $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_i$ is the dataset of observations so far.

The second design choice of the Bayesian Optimization pipeline is the acquisition function. The predictive mean and the predictive variance from the surrogate model is input to the acquisition function that quantifies the significance of every point in $\mathbf{x}$ as a next evaluation point. While different acquisition functions have been explored in the literature (Thompson, 1933; Kushner, 1964; Močkus, 1975; Srinivas et al., 2009; Hennig & Schuler, 2012; Hernández-Lobato et al., 2014), they all share the following property: they return high scores at regions of either high predictive variance (high but uncertain reward), or low predictive mean (modest but certain reward).

Last, the third design choice of the Bayesian Optimization pipeline, often overlooked, is the search space. In (Snoek et al., 2014) the kernel of the surrogate model is defined on a warped search space, thus allowing for a more flexible modeling of $f(\mathbf{x})$ by the surrogate function. As the search space defines where optimal solutions are to be sought for, the search space definition is a means of infusing prior knowledge into the Bayesian Optimization. Usually, a search space is set so that the expected optimum is close to the center.

2.2. High-dimensional Bayesian Optimization

Even with its successes in many applications, several theoretical as well as practical issues (Shahriari et al., 2016) still exist when employing Bayesian Optimization to real world problems. Among others, many Bayesian optimization algorithms are restricted in practice to problems of moderate dimensions. In high dimensional problems, one suffers from the curse of dimensionality. To overcome the curse of dimensionality, several works make structural assumptions, such as low effective dimensionality (Wang et al., 2016; Bergstra & Bengio, 2012) or additive structure (Kandasamy et al., 2015; Wang et al., 2017).

Because of the way Gaussian Processes quantify uncertainty, the curse of dimensionality is a serious challenge for

Gaussian Processes-based Bayesian Optimization in high dimensions. Since in high dimensions data points typically lie mostly on the boundary, and anyways far away from each other, the predictive variance tends to be higher in the regions near the boundary. Thus, the acquisition function is somewhat biased to choose evaluations near the boundary, hence, biasing Bayesian Optimization towards solution near the boundary and away from the center, contradicting with the prior assumption. This is the *boundary issue*(Swersky, 2017).

2.3. Contributions

Different from the majority of the Bayesian Optimization methods that rely on a Euclidean geometry of the search space implicitly or explicitly(Hutter et al., 2011; Bergstra et al., 2011; Snoek et al., 2012; 2014; 2015; Swersky et al., 2013; Wang et al., 2017), the proposed BOCK applies a cylindrical geometric transformation on it. The effect is that the volume near the center of the search space is expanded, while the volume near the boundary is shrunk. Compared to (Snoek et al., 2014; Binois et al., 2015), where warping functions were introduced with many kernel parameters to be learned, we do not train transformations. Also, we avoid learning many additional kernel parameters for better efficiency and scalability. Because of the transformation, the proposed BOCK solves also the issue of flat optimization surfaces of the acquisition function in high dimensional spaces (Rana et al., 2017). And compared to REMBO (Wang et al., 2016; Binois et al., 2015), BOCK does not rely on assumptions of low dimensionality of the latent search space.

3. Method

3.1. Prior assumption and search space geometry

The flexibility of a function f on a high-dimensional domain X can be, and usually is, enormous. To control the flexibility and make the optimization feasible some reasonable assumptions are required. A standard assumption in Bayesian Optimization is the *prior assumption* (Swersky, 2017), according to which the optimum of $f(\mathbf{x})$ should lie somewhere near the center of the search space X. Since the search space is set with the *prior assumption* in mind, it is reasonable for Bayesian Optimization to spend more evaluation budget in areas near the center of X.

It is interesting to study the relation of the *prior assumption* and the geometry of the search space. The ratio of the volume of two concentric balls $B(\mathbf{c}; R-\delta R)$ and $B(\mathbf{c}; R)$, with a radius difference of δR, is

$$\frac{\text{volume}(B(\mathbf{c}; R-\delta R))}{\text{volume}(B(\mathbf{c}; R))} = o((1-\delta)^D), \qquad (4)$$

which rapidly goes to zero with increasing dimensionality D. This means that the volume of $B(\mathbf{c}; R)$ is mostly concentrated near the boundary, which in combination with Gaussian processes' behavior of high predictive variance at points far from data, creates the *boundary issue* (Swersky, 2017).

It follows, therefore, that with a transformation of the search space we could avoid excessively biasing our search towards large values of R.

3.2. Cylindrical transformation of search space

The search space geometry has a direct influence on the kernel $K(\mathbf{x}, \mathbf{x}')$ of the Gaussian Process surrogate model, and, therefore, its predictive variance $\sigma^2(\mathbf{x})$, see eq. (3). A typical design choice for Gaussian Processes (Snoek et al., 2012; 2014; González et al., 2016) are stationary kernels, $K(\mathbf{x}, \mathbf{x}') \propto f(\mathbf{x}-\mathbf{x}')$. Unfortunately, stationary kernels are not well equipped to tackle the *boundary issue*. Specifically, while stationary kernels compute similarities only in terms of relative locations $\mathbf{x}-\mathbf{x}'$, the boundary issue dictates the use of location-aware kernels $K(\mathbf{x}, \mathbf{x}')$ to recognize whether $\mathbf{x}, \mathbf{x}'$ lie near the boundary or the center areas of the search space.

A kernel that can address this should have the following two properties. First, the kernel must define the similarity between two points $\mathbf{x}, \mathbf{x}'$ in terms of their absolute locations, namely the kernel has to be non-stationary. Second, the kernel must transform the geometry of its input (*i.e.*, the search space for the Gaussian Process surrogate model) such that regions near the center and the boundaries are equally represented. To put it otherwise, we need a geometric transformation of the search space that expands the region near the center while contracting the regions near the boundary. A transformation with these desirable properties is the cylindrical one, separating the radius and angular components of a point $\mathbf{x}$, namely

$$T(\mathbf{x}) = \begin{cases} (\|\mathbf{x}\|_2, \mathbf{x}/\|\mathbf{x}\|_2) & \text{for } \|\mathbf{x}\|_2 \neq 0 \\ (0, \mathbf{a}_{arbitrary}) & \text{for } \|\mathbf{x}\|_2 = 0 \end{cases} \qquad (5)$$
$$T^{-1}(r, \mathbf{a}) = r\mathbf{a}$$

where $\mathbf{a}_{arbitrary}$ is an arbitrarily chosen vector with unit ℓ_2-norm. Although polar coordinate (Padonou & Roustant, 2016) appears to be able to used for the same purpose, our specific choice of transformation does not suffer from numerical instability. [1]

After applying the geometric transformation we arrive at a new kernel $K_{cyl}(\mathbf{x}_1, \mathbf{x}_2)$, which we will refer to as the cylindrical kernel. The geodesic similarity measure (kernel)

[1] In high dimensional spaces, the inverse transformation from spherical to rectangular coordinate entails multiplication of many trigonometric functions, causing numerical instabilities because of large products of small numbers.

of K_{cyl} on the *transformed cylinder*, $T(X)$, is defined as

$$\begin{aligned} K_{cyl}(\mathbf{x}_1, \mathbf{x}_2) &= \widetilde{K}(T(\mathbf{x}_1), T(\mathbf{x}_2)) \\ &= K_r(r_1, r_2) \cdot K_a(\mathbf{a}_1, \mathbf{a}_2) \end{aligned} \tag{6}$$

where the final kernel decomposes into a 1-D radius kernel K_r measuring the similarity of the radii of r_1, r_2 and a angle kernel K_a.

For the angle kernel $K_a(\mathbf{a}_1, \mathbf{a}_2)$, we opt for a continuous radial kernel on the (hyper-)sphere (Jayasumana et al., 2014),

$$K_d(\mathbf{a}_1, \mathbf{a}_2) = \sum_{p=0}^{P} c_p (\mathbf{a}_1^T \mathbf{a}_2)^p, \quad c_p \geq 0, \ \forall p \tag{7}$$

with trainable kernel parameters of $c_0, \cdots, c_P$ and P user-defined. The advantages of a continuous radial kernel is two-fold. First, with increasing P a continuous radial kernel can approximate any continuous positive definite kernel on the sphere with arbitrary precision (Jayasumana et al., 2014). Second, the cylindrical kernel has $P+1$ parameters, which is independent of the dimensionality of X. This means that while the continuous radial kernel retains enough flexibility, only few additional kernel parameters are introduced, which are independent of the dimensionality of the optimization problem and can, thus, easily scale to more than 50 dimensions. This compares favorably to Bayesian optimization with ARD kernels that introduce at least d kernel parameters for a d-dimensional search space.

Although the boundary issue is mitigated by the cylindrical transformation of the search space, the prior assumption (good solutions are expected near the center) can be promoted. To this end, and to reinforce the near-center expansion of the cylindrical transformation, we consider input warping (Snoek et al., 2014) on the radius kernel $K_r(r_1, r_2)$. Specifically, we use the cumulative distribution function of the Kumaraswamy distribution, $Kuma(r|\alpha, \beta) = 1 - (1 - r^\alpha)^\beta$ (with $\alpha > 0, \beta > 0$),

$$\begin{aligned} K_r(r_1, r_2) &= K_{base}(Kuma(r_1|\alpha, \beta), Kuma(r_1|\alpha, \beta)) \\ &= K_{base}(1 - (1 - r_1^\alpha)^\beta, 1 - (1 - r_2^\alpha)^\beta | \alpha, \beta) \end{aligned} \tag{8}$$

where the non-negative a, b are learned together with the kernel parameters. K_{base} is the base kernel for measuring the radius-based similarity. Although any kernel is possible for K_{base}, in our implementations we opt for the Matérn52 kernel used in Spearmint (Snoek et al., 2012). By making radius warping concave and non-decreasing, K_r and, in turn, K_{cyl} focus more on areas with small radii.

Overall, the transformation of the search space has two effects. The first effect is that the volume is redistributed, such that areas near the center are expanded, while areas near the boundaries are contracted. Bayesian optimization's attention in the search space, therefore, is also redistributed from the boundaries to the center of the search space. The second effect is that the kernel similarity changes, such that the predictive variance depends mostly on the angular difference between the existing data points and the ones to be evaluated. An example is illustrated in Fig. 1, where our dataset comprises of $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2\}$ and the acquisition function must select between two points, $\mathbf{x}_{*,1}$ and $\mathbf{x}_{*,2}$. Whereas in the original Euclidean geometry (Figure 1 to the left) $\mathbf{x}_{*,1}$ is further away from $\mathcal{D}$, thus having higher predictive variance, in the cylindrical geometry both $\mathbf{x}_{*,1}$ and $\mathbf{x}_{*,2}$ are equally far, thus reducing the artificial preference to near-boundary points.

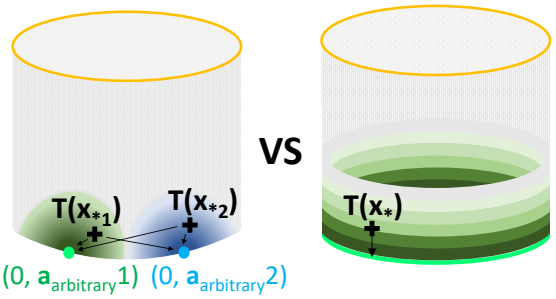

Figure 2. Similarity to the center point in *transformed geometry*.

3.3. Balancing center over-expansion

The transformation T maps an annulus $A(\mathbf{0}; R - \delta R, R)$ of width δR to the cylinder $C(R - \delta R, R; \mathbf{0}, 1)$, where $(\mathbf{0}, 1)$ is the center and the radius of the cylinder. For almost any point in the original ball there is a one-to-one mapping to a point on the cylinder. The only exception is the extreme case of the ball origin, which is mapped to the 0-width sphere $C(0, 0; 0, 1) = \{(0, \mathbf{a}) | \|\mathbf{a}\| = 1\}$ on the base of the cylinder (bright green circle in the Figure 2 to the right). Namely, the center point $\mathbf{x}_{center}$ is overly expanded, corresponding to a set of points. Because of the one-to-many correspondence between $\mathbf{x}_{center}$ and $C(0, 0; 0, 1)$, an arbitrary point is selected in eq. (5).

Unfortunately, the dependency on a point that is both arbitrary *and* fixed incurs an arbitrary behavior of K_{cyl} as well. For any point $\mathbf{x}_* \in X \setminus \{\mathbf{0}\}$ the kernel $K_{cyl}(\mathbf{x}_{center}, \mathbf{x}_*)$ changes arbitrarily, depending on the choice of $\mathbf{a}_{arbitrary}$, see Figure 2. Having a fixed arbitrary point, therefore, is undesirable as it favors points lying closer to it. To this end, we define $\mathbf{a}_{arbitrary}$ as the angular component of a test point $\mathbf{x}_*$, $\mathbf{a}_{arbitrary} = \mathbf{x}_* / \| \mathbf{x}_* \|$, thus being not fixed anymore. Geometrically, this is equivalent to using the point in $C(0, 0; 0, 1)$ closest to $T(\mathbf{x}_*)$, see Figure 2 to the right. This implies that, if the origin is in the dataset, the Gram matrix needed for computing the predictive density now depends on the angular location of the test point under consideration.

Similar method constructing gram matrix dependent with prediction point is proposed in (Pronzato & Rendas, 2017), where the functional form of the kernel changes according to prediction point, while training data changes in BOCK. This may look somewhat unconventional but still well behaved (the kernel is still positive definite and the predictive mean and variance change smoothly). More details can be found in the supplementary material.

4. Experiments

In Bayesian optimization experiments, we need to define *(a)* how to train the surrogate model, *(b)* how to optimize the acquisition function and *(c)* how to set the search space. For BOCK we use Gaussian Process surrogate models, where following (Snoek et al., 2012; 2014) we train parameters of BOCK with MCMC (slice sampling (Murray & Adams, 2010; Neal, 2003)) . For the acquisition function, we use the Adam (Kingma & Ba, 2014) optimizer, instead of L-BFGS-B (Zhu et al., 1997). To begin the optimization we feed 20 initial points to Adam. To select the 20 initial points, a sobol sequence (Bratley & Fox, 1988) of 20,000 points is generated on the cube (we used the cube for fair comparison with others). The acquisition function is evaluated on these points and the largest 20 points are chosen as the initial ones. Instead of using a static sobol sequence in the entire course of Bayesian optimization (Snoek et al., 2012; 2014), we generate different sobol sequences for different evaluations, as fixed grid point impose too strong constraints in high dimensional problems. In the d-dimensional space, our search space is a ball $B(\mathbf{0}, \sqrt{d})$ circumscribing a cube $[-1, 1]^d$, which is the scaled and translated version of the typical search region, unit cube $[0, 1]^d$. Our search space is much larger than a cube. By generating sobol sequence on the cube, the reduction of the *boundary issue* mostly happens at corners of the cube $[-1, 1]^d$. The implementation is available online (https://github.com/ChangYong-Oh/HyperSphere)

4.1. Benchmarking

First, we compare different Bayesian Optimization methods and BOCK on four benchmark functions. Specifically, following (Eggensperger et al., 2013; Laguna & Martí, 2005) we use the repeated Branin, repeated Hartmann6 and Levy to assess Bayesian Optimization in high dimensions. To test the ability of Bayesian Optimization methods to optimize functions with more complex structure and stronger intra-class dependencies, we additionally include the Rosenbrock benchmark, typically used as benchmark for gradient-based optimization (Laguna & Martí, 2005). The precise formulas for the four benchmark functions are added to the supplementary material. We solve the benchmark functions in 20 and 100 dimensions [2] , using 200 and 600 function evaluations respectively for all Bayesian Optimization methods. We compare the proposed BOCK with the following Bayesian Optimization methods using publicly available software: SMAC (Hutter et al., 2011), TPE (Bergstra et al., 2011), Spearmint (Snoek et al., 2012), Spearmint+ (Snoek et al., 2014), additive BO (Kandasamy et al., 2015), elastic BO (Rana et al., 2017). We also report an in-house improved Spearmint implementation, which we refer to as Matérn. [3]

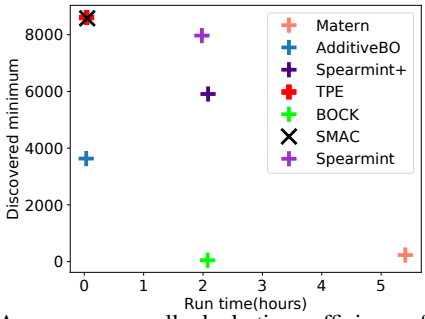

Figure 3. Accuracy vs wall clock time efficiency for the 20-dimensional Rosenbrock benchmark. BOCK is the closest to the optimum operating point $(0, 0)$. Matérn is also accurate enough, although considerably slower, while SMAC and additive BO are faster but considerably less accurate.

We focus on four aspects: *(a)* accuracy, *(b)* efficiency (wall clock time) vs accuracy, *(c)* scalability (number of dimensions) vs efficiency, and *(d)* robustness of BOCK to hyper-pararameters and other design choices. We study (a) in all four benchmark functions. For brevity, we report (b)-(d) on the Rosenbrock benchmark only, the hardest of the four benchmark functions for all Bayesian Optimization methods in terms of accuracy, and report results the rest of the benchmark functions in the supplementary material.

Accuracy. We first present the results regarding the accuracy of BOCK and the Bayesian Optimization baselines in Table 1. BOCK and Matérn outperform others with large margin in discovering near optimal solutions. For benchmark functions with complicated dependencies between variables, such as the repeated Hartmann6 and Rosenbrock, BOCK consistently discovers smaller values compared to other baselines, while not being affected by an increasing number of dimensions. What is more, BOCK is on par even with methods that are designed to exploit the specific geometric structures, if the same geometric structures can be found in the the evaluated functions. For instance, the repeated Branin and Levy have an additive structure, where the same low dimensional structure is repeated. The non-ARD kernel of Matérn can exploit such special, additive structures. BOCK is able to reach a similar near-optimum solution without being explicitly designed to exploit such structures.

[2] We also solve the 50-dimensional cases. As conclusions are similar, we add these results to the supplementary material.

[3] Differences with standard Spearmint: (a) a non-ARD, Matérn52 kernel for the surrogate model, (b) dynamic search grid generation per evaluation, (c) Adam (Kingma & Ba, 2014) instead of L-BFGS-B (Zhu et al., 1997), (d) more updates for optimizer.

Table 1. Bayesian Optimization on four benchmark functions for 20 and 200 dimensions, with the exception of Spearmint+ (Snoek et al., 2014) and Elastic BO (Rana et al., 2017) evaluated only on the 20-dimensional cases because of prohibitive execution times). For benchmark functions with complicated dependencies between variables (repeated Hartmann6, Rosenbrock), BOCK consistently discovers good solutions compared to other baselines, while not being affected by an increasing number of dimensions. Also, BOCK matches the accuracies of methods, like Matérn, designed to exploit specific geometric structures, *e.g.* the additive structures of repeated Branin and Levy. We conclude that BOCK is accurate, especially when we have no knowledge of the geometric landscape of the evaluated functions.

BENCHMARK	REPEATED BRANIN		REPEATED HARTMANN6		ROSENBROCK		LEVY	
DIMENSIONS	20	100	20	100	20	100	20	100
MINIMUM	0.3979	0.3979	-3.3223	-3.3223	0.0000	0.0000	0.0000	0.0000
SMAC	15.95±3.71	20.03±0.85	-1.61±0.12	-1.16±0.19	8579.13± 58.45	8593.09± 18.80	2.35±0.00	9.60±0.04
TPE	7.59±1.20	23.55±0.73	-1.74±0.10	-1.01±0.10	8608.36± 0.00	8608.36± 0.00	2.35±0.00	9.62±0.00
SPEARMINT	5.07±3.01	2.78±1.06	-2.60±0.42	-2.55±0.19	7970.05± 1276.62	8608.36± 0.00	1.88±0.59	4.87±0.35
SPEARMINT+	6.83±0.32	-	-2.91±0.25	-	5909.63± 2725.76	-	2.35±0.00	-
ADDITIVE BO*	5.75±0.93	14.07±0.84	-3.03±0.13	-1.69±0.22	3632.25± 1642.71	7378.27± 305.24	2.32±0.02	9.59±0.04
ELASTIC BO	6.77±4.85	-	-2.85±0.57	-	5346.96± 2494.89	-	1.35±0.34	-
MATÉRN	0.41±0.00	0.54±0.06	-3.29±0.04	-2.91±0.26	230.25± 187.41	231.42± 28.94	0.38±0.13	2.17±0.18
BOCK	0.50±0.12	1.03±0.17	-3.30±0.02	-3.16±0.10	47.87± 33.94	128.69± 52.84	0.54±0.13	6.78±2.16

* ADDITIVE BO (KANDASAMY ET AL., 2015) REQUIRES A USER-SPECIFIED "MAXIMUM GROUP SIZE" TO DEFINE THE ADDITIVE STRUCTURE. IN EACH EXPERIMENT WE TRIED 5 DIFFERENT VALUES AND REPORTED THE BEST RESULT.

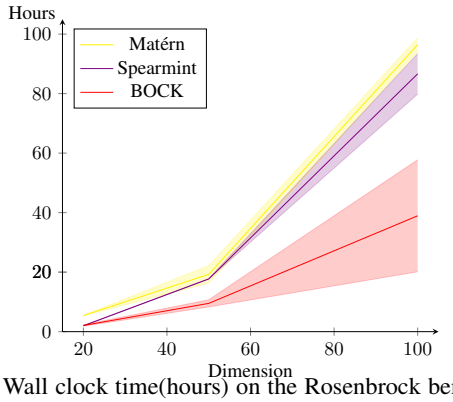

Figure 4. Wall clock time(hours) on the Rosenbrock benchmark for an increasing the number of dimensions (20, 50 and 100 dimensions, using 200, 400 and 600 function evaluations respectively for all methods). The solid lines and colored regions represent the mean wall clock time and one standard deviation over these 5 runs. As obtaining the evaluation score $y = f(\mathbf{x}_*)$ on these benchmark functions is instantaneous, the wall clock time is directly related to the computational efficiency of algorithms. In this figure, we compare BOCK and BOs with relative high accuracy in all benchmark functions, such as Spearmint and Matérn. BOCK is clearly more efficient, all the while being less affected by the increasing number of dimensions.

We conclude that BOCK is accurate, especially when we have no knowledge of the geometric landscape of the evaluated functions. In the remaining of the experiments we focus on the Bayesian Optimization methods with competitive performance, namely BOCK, Spearmint and Matérn.

Efficiency vs accuracy. Next, we compare in Figure 3 the accuracy of the different Bayesian Optimization methods as a function of their wall clock times for the 20-dimensional case for Rosenbrock. As the function minimum is $f(\mathbf{x}_{opt}) = 0$, the optimal operating point is at $(0, 0)$. BOCK is the closest to the optimal point. Matérn is the second most accurate, while being considerably slower to run. SMAC (Hutter et al., 2011) and AdditiveBO (Kandasamy et al., 2015) are faster than BOCK, however, they are also considerably less accurate.

Scalability. In Figure 4 we evaluate the most accurate Bayesian Optimization methods from Table 1 (Spearmint, Matérn and BOCK.) with respect to how scalable they are, namely measuring the wall clock time for an increasing number of dimensions. Compared to Spearmint BOCK is less affected by the increasing number of dimensions. Not only the BOCK surrogate kernel requires fewer parameters, but also the number of surrogate kernel parameters is independent of the number of input dimensions, thus making the surrogate model fitting faster. BOCK is also faster than Matérn, although the latter uses a non-ARD kernel that is also independent of the number of input dimensions. Presumably, this is due to a better, or smoother, optimization landscape after the cylindrical transformation of geometry of the input space, affecting positively the search dynamics. We conclude that BOCK is less affected by the increasing number of dimensions, thus scaling better.

Robustness. To study the robustness of BOCK to design choices, we compare three BOCK variants. The first is the standard BOCK as described in Section 3. The second variant, BOCK-W, removes the input warping on the radius component. The third variant, BOCK+B, includes an additional boundary treatment to study whether further reduction of the predictive variance is beneficial. Specifically, we reduce the predictive variance by adding "fake" data. [4] We present results in Table 2.

[4]Predictive variance depends only on the inputs $\mathbf{x}$, not the evaluations $y = f(\mathbf{x})$. Thus we can manipulate the predictive variance only with input data. BOCK+B uses one additional "fake data", which does not have output value(evaluation), in its predictive variance. BOCK's predictive variance $\sigma^2(\mathbf{x}_* \mid \mathcal{D})$ becomes $\sigma^2(\mathbf{x}_* \mid \mathcal{D} \cup \{(R\,\mathbf{x}_* / \| \mathbf{x}_* \|, \sim)\})$ in BOCK+B on the search space of the ball $B(\mathbf{0}; R)$, where $(R\,\mathbf{x}_* / \| \mathbf{x}_* \|, \sim)$ is the fake data.

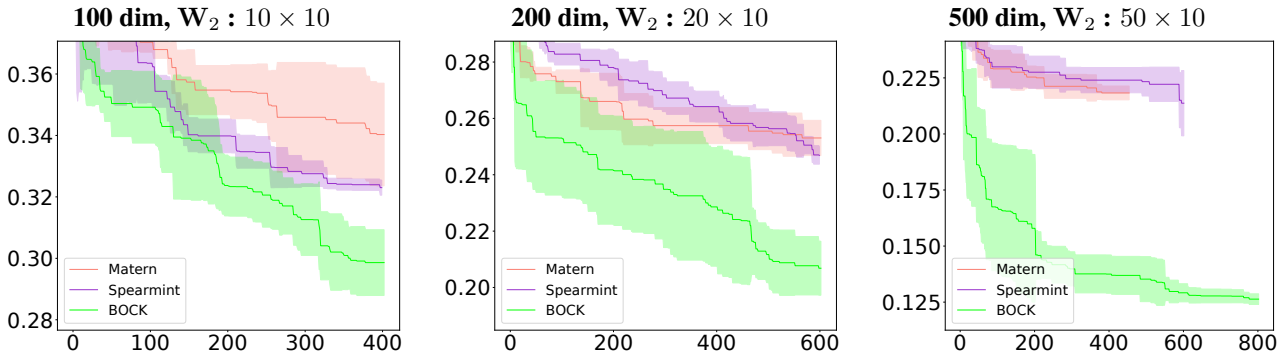

Figure 5. Training on MNIST a two-layer neural network: $784 \xrightarrow{\mathbf{W}_1,\mathbf{b}_1} N_{hidden} \xrightarrow{\mathbf{W}_2,\mathbf{b}_2} 10$. For all experiments, $\mathbf{W}_1$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are optimized with Adam (Kingma & Ba, 2014) and $\mathbf{W}_2$ with Bayesian Optimization. In this experiment, Bayesian Optimization repeats the following steps. *(a)* A new $\mathbf{W}_2$ is suggested by BOCK. *(b)* Given this $\mathbf{W}_2$, the $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{b}_2$ are fine-tuned by SGD on the training set. *(c)* The loss on the validation is returned as the evaluation on $\mathbf{W}_2$. Therefore, in this experiment, Bayesian Optimization algorithms optimize the validation loss. We observe that BOCK can optimize successfully a modestly sized neural network layer. BOCK consistently finds a better solution than existing Bayesian optimization algorithms. In high dimensional cases, BOCK outperforms other algorithms with a significant margin. We conclude that BOCK is capable of optimizing in high-dimensional and complex spaces.

Table 2. Comparison between different BOCK variants on Rosenbrock. Excluding input warping results in slight instabilities, while including additional boundary treatments brings only marginal benefits.

DIMENSIONS	20	50	100
BOCK	47.87± 33.94	29.65±11.56	128.69± 52.84
BOCK-W	1314.03± 1619.73	51.14±58.18	157.89± 161.92
BOCK+B	48.87± 18.33	33.90±21.69	87.00± 36.88

Removing the input warping on the radius is hurting the robustness, as BOCK-W tends to reach slightly worse minima than BOCK. However, introducing further boundary treatments has a marginal effect.

Further, we assess the sensitivity of BOCK with respect to the hyperparameter P in eq.(5). For $P = 3, 5, 7, 9$, we observe that higher P tends to give slightly better minima, while increasing the computational cost.

For clarity of presentation, as well as to maintain the experimental efficiency, in the rest of the experiments we focus on BOCK with $P = 3$.

4.2. Optimizing a neural network layer

As BOCK allows for accurate and efficient Bayesian Optimization for high-dimensional problems, we next perform a stress test, attempting to optimize neural network layers of 100, 200 and 500 dimensions. Specifically, we define a two-layered neural network with architecture: $784 \xrightarrow{\mathbf{W}_1,\mathbf{b}_1} N_{hidden} \xrightarrow{\mathbf{W}_2,\mathbf{b}_2} 10$, using ReLU as the intermediate non-linearity.

In this experiment, we split the data set into train (first 45000 images of MNIST train data set), validation (next 5000 images of MNIST train data set) and test (10000 images of MNIST test data set). For all Bayesian optimization experiments $\mathbf{W}_1$, $\mathbf{b}_1$ and $\mathbf{b}_2$ are optimized with Adam (Kingma & Ba, 2014) and $\mathbf{W}_2$ with Bayesian Optimization. The training proceeds as follows. First, Bayesian Optimization suggests a $\mathbf{W}_2$ based on evaluations on the **validation** set. Given this $\mathbf{W}_2$ we train on the **train** sets the $\mathbf{W}_1, b_1, b_2$ with Adam, then repeat.

We show the validation loss in Figure 5, where we report mean and standard deviation over 5 runs for all methods. We observe that BOCK clearly outperforms Spearmint and Matérn in terms of validation loss in Figure 5, with the gap increasing for higher $\mathbf{W}_2$ dimensions. This show that BOCK effectively optimizes a target quantity (validation loss) and thus is a competitive optimizer even for high dimensional problems.

Evaluation of generalization performance is given in Table 3. For Bayesian Optimization algorithms, given each $\mathbf{W}_2$ optimizing loss on the **validation** set, we train $\mathbf{W}_1$, $\mathbf{b}_1$ and $\mathbf{b}_2$ on the **train+validation** set 5 times with Adam. For SGD, we train a network with Adam on the **train+validation** set and report test loss 5 times. We compare BOCK with the competitive Spearmint and Matérn on both validation loss and test loss. To the best of our knowledge we are the first to apply Gaussian Process-based Bayesian Optimization in so high-dimensional and complex, representation learning spaces. [5].

Somewhat surprisingly, BOCK is able to match and even outperform the Adam-based SGD in terms of generalization. There are three reasons for this. First, in this experiment, all Bayesian optimization algorithms directly optimize the loss

[5]To our knowledge, running Bayesian Optimization on 200 or 500 dimensional problems has only been tried with methods assuming low effective dimensionality (Wang et al., 2016; Chen et al., 2012)

Table 3. Test loss with optimized $\mathbf{W}_2$. Bayesian Optimization methods train $\mathbf{W}_1$, $\mathbf{b}_1$ and $\mathbf{b}_2$ on **train+validation** set for each optimized $\mathbf{W}_2$ 5 times (25 runs in total). SGD is trained 5 times on the **train+validation** sets and its test loss is reported.

DIMENSIONS	100	200	500
SPEARMINT	0.3219 ± 0.0420	0.2246 ± 0.0172	0.1812 ± 0.0201
MATÉRN	0.3189 ± 0.0334	0.2350 ± 0.0130	0.2012 ± 0.0183
BOCK	0.2847 ± 0.0314	0.1778 ± 0.0156	0.0993 ± 0.0034
ADAM-SGD	0.2389 ± 0.0167	0.1551 ± 0.0067	0.1199 ± 0.0071

Table 4. Frobenius Norm of optimized $\mathbf{W}_2$. From 5 runs of Bayesian optimization, we have 5 optimized $\mathbf{W}_2$ and mean and std of 5 of them are reported.

DIMENSIONS	100	200	500
SPEARMINT	7.4491 ± 0.3768	7.8577 ± 1.3314	11.6091 ± 2.4745
MATÉRN	6.0120 ± 0.2843	8.6870 ± 0.2014	13.1189 ± 0.3077
BOCK	2.8805 ± 1.1378	3.3917 ± 1.6424	1.9467 ± 0.2183
ADAM-SGD	3.9734 ± 0.2350	5.1930 ± 0.2975	7.1933 ± 0.1111

on the **validation** set using only **train** data. Second, to evaluate test loss with $\mathbf{W}_2$ which optimizes the loss on the **validation** set by Bayesian optimization algorithms, $\mathbf{W}_1, b_1, b_2$ are trained on **train+validation**, which prevent overfitting to **validation** set. Thirdly, since BOCK prefers an optimum near the center, BOCK has an implicit L2-regularization effect. We compared Frobenius norm of optimized $\mathbf{W}_2$ in Table 4, in which BOCK results in $\mathbf{W}_2$ with consistently small Frobenius norm.

It is noteworthy that BOCK can optimize such high-dimensional and complex (representation learning) functions with Bayesian Optimization (Figure 5). We conclude that BOCK is able to optimize complex, multiple-optima functions, such as neural network layers and that BOCK materializes regularization methods useful in neural network training.

4.3. Hyper-optimizing stochastic depth ResNets

As BOCK allows for accurate and efficient Bayesian Optimization, in our last experiment we turn our attention to a practical hyperparameter optimization application. Stochastic Depth ResNet (SDResNet) (Huang et al., 2016) was shown to obtain better accuracy and faster training by introducing a stochastic mechanism that randomly suppresses ResNet blocks (ResBlock) (He et al., 2016). The stochastic mechanism for dropping ResBlocks is controlled by a vector $\mathbf{p} \in [0, 1]^t$ of probabilities for t ResBlocks, called "death rate". In (Huang et al., 2016) a linearly increasing (from input to output) death rate was shown to improve accuracies.

Instead of pre-defined death rates, we employ BOCK to find the optimal death date vector for SDRes-110 on CIFAR100 (Krizhevsky & Hinton, 2009). We first train an SDResNet for 250 epochs and linear death rates with exactly the same configuration in (Huang et al., 2016) up to 250 epochs. In this experiment BOCK has access to the **training and validation set only**. Then, per iteration BOCK first proposes the next candidate p based on evaluation on the validation set. Given the candidate $\mathbf{p}$ we run 100 epochs of SGD on the training set and repeat with an annealed learning rate (0.01 for 50 epochs, then 0.001 for 50 more). We initialize the death rate vector to $\mathbf{p} = [0.5, 0.5, ..., 0.5]$. We report the final accuracies computed in the **unseen test set** in Table 5, using only 50 evaluations.

Table 5. Using BOCK to optimize the "death rates" of a Stochastic Depth ResNet-110, we improve slightly the accuracy on CIFAR100 while reducing the expected depth of the network.

METHOD	TEST ACC.	VAL. ACC.	EXP. DEPTH
RESNET-110	72.98±0.43	73.03±0.36	110.00
SDRESNET-110+LINEAR	74.90±0.15	75.06±0.04	82.50
SDRESNET-110+BOCK	75.06±0.19	75.21±0.05	74.51±1.22

We observe that BOCK learns a vector $\mathbf{p}$ that results in an improved validation accuracy compared to SDResNet, all the while allowing for a lower expected depth. The improved validation accuracy materializes to an only slightly better test accuracy, however. One reason is that optimization is not directly equivalent to learning, as also explained in Section 4.2. What is more, it is likely that the accuracy of SDResNet-110 on CIFAR-100 is maxed out, especially considering that only 50 evaluations were made. We conclude that BOCK allows for successful and efficient Bayesian Optimization even for practical, large-scale learning problems.

5. Conclusion

We propose BOCK, Bayesian Optimization with Cylindrical Kernels. Many of the problems in Bayesian Optimization relate to the *boundary issue* (too much value near the boundary), and the *prior assumption* (optimal solution probably near the center). Because of the boundary issue, not only much of the evaluation budget is unevenly spent to the boundaries, but also the *prior assumption* is violated. The basic idea behind BOCK is to transform the ball geometry of the search space with a cylindrical transformation, expanding the volume near the center while contracting it near the boundaries. As such, the Bayesian optimization focuses less on the boundaries and more on the center.

We test BOCK extensively in various settings. On standard benchmark functions BOCK is not only more accurate, but also more efficient and scalable compared to state-of-the-art Bayesian Optimization alternatives. Surprisingly, optimizing a neural network up to 500 dimensions with BOCK allows for even better parameters than SGD with Adam (Kingma & Ba, 2014) with respect to both validation loss and test loss. And hyper-optimizing the "death rate" of stochastic depth ResNet (Huang et al., 2016) results in smaller ResNets while maintaining accuracy.

We conclude that BOCK allows for accurate, efficient and scalable Gaussian Process-based Bayesian Optimization.

References

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

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

Binois, M., Ginsbourger, D., and Roustant, O. A warped kernel improving robustness in bayesian optimization via random embeddings. In *International Conference on Learning and Intelligent Optimization*, pp. 281–286. Springer, 2015.

Bratley, P. and Fox, B. L. Algorithm 659: Implementing sobol's quasirandom sequence generator. *ACM Transactions on Mathematical Software (TOMS)*, 14(1):88–100, 1988.

Chen, B., Castro, R., and Krause, A. Joint optimization and variable selection of high-dimensional gaussian processes. *arXiv preprint arXiv:1206.6396*, 2012.

Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., and Leyton-Brown, K. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, volume 10, 2013.

González, J., Dai, Z., Hennig, P., and Lawrence, N. Batch bayesian optimization via local penalization. In *Artificial Intelligence and Statistics*, pp. 648–657, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hennig, P. and Schuler, C. J. Entropy search for information-efficient global optimization. *Journal of Machine Learning Research*, 13(Jun):1809–1837, 2012.

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. Predictive entropy search for efficient global optimization of black-box functions. In *Advances in neural information processing systems*, pp. 918–926, 2014.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. Deep networks with stochastic depth. In *European Conference on Computer Vision*, pp. 646–661. Springer, 2016.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. Sequential model-based optimization for general algorithm configuration. *LION*, 5:507–523, 2011.

Jayasumana, S., Hartley, R., Salzmann, M., Li, H., and Harandi, M. Optimizing over radial kernels on compact manifolds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3802–3809, 2014.

Kandasamy, K., Schneider, J., and Póczos, B. High dimensional bayesian optimisation and bandits via additive models. In *International Conference on Machine Learning*, pp. 295–304, 2015.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. 2009.

Kushner, H. J. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

Laguna, M. and Martí, R. Experimental testing of advanced scatter search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33(2): 235–255, 2005.

Močkus, J. On bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pp. 400–404. Springer, 1975.

Murray, I. and Adams, R. P. Slice sampling covariance hyperparameters of latent gaussian models. In *Advances in Neural Information Processing Systems*, pp. 1732–1740, 2010.

Neal, R. M. Slice sampling. *Annals of statistics*, pp. 705–741, 2003.

Padonou, E. and Roustant, O. Polar gaussian processes and experimental designs in circular domains. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):1014–1033, 2016.

Pronzato, L. and Rendas, M.-J. Bayesian local kriging. *Technometrics*, 59(3):293–304, 2017.

Rana, S., Li, C., Gupta, S., Nguyen, V., and Venkatesh, S. High dimensional bayesian optimization with elastic gaussian process. In *International Conference on Machine Learning*, pp. 2883–2891, 2017.

Rasmussen, C. E. and Williams, C. K. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, 2006.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and de Freitas, N. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104 (1):148–175, 2016.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pp. 2951–2959, 2012.

Snoek, J., Swersky, K., Zemel, R., and Adams, R. Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, pp. 1674–1682, 2014.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, pp. 2171–2180, 2015.

Springenberg, J. T., Klein, A., Falkner, S., and Hutter, F. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems*, pp. 4134–4142, 2016.

Srinivas, N., Krause, A., Kakade, S. M., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv preprint arXiv:0912.3995*, 2009.

Swersky, K., Snoek, J., and Adams, R. P. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pp. 2004–2012, 2013.

Swersky, K. J. *Improving Bayesian Optimization for Machine Learning using Expert Priors*. PhD thesis, 2017.

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.

Wang, Z., Hutter, F., Zoghi, M., Matheson, D., and de Feitas, N. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

Wang, Z., Li, C., Jegelka, S., and Kohli, P. Batched high-dimensional bayesian optimization via structural kernel learning. *arXiv preprint arXiv:1703.01973*, 2017.

Zhu, C., Byrd, R. H., Lu, P., and Nocedal, J. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560, 1997.