Kernel Learning for Sample Constrained Black-Box Optimization

Rajalaxmi Rajagopalan, Yu-Lin Wei, Romit Roy Choudhury

Department of Electrical & Computer Engineering
University of Illinois Urbana-Champaign
{rr30,yulinlw2,croy}@illinois.edu

Abstract

Black box optimization (BBO) focuses on optimizing unknown functions in high-dimensional spaces. In many applications, sampling the unknown function is expensive, imposing a tight sample budget. Ongoing work is making progress on reducing the sample budget by learning the shape/structure of the function, known as kernel learning. We propose a new method to learn the kernel of a Gaussian Process. Our idea is to create a continuous kernel space in the latent space of a variational autoencoder, and run an auxiliary optimization to identify the best kernel. Results show that the proposed method, *Kernel Optimized Blackbox Optimization* (KOBO), outperforms state of the art by estimating the optimal at considerably lower sample budgets. Results hold not only across synthetic benchmark functions but also in real applications. We show that a hearing aid may be personalized with fewer audio queries to the user, or a generative model could converge to desirable images from limited user ratings.

1 Introduction

Many problems involve the optimization of an *unknown* objective function. Examples include personalizing content x to maximize a user's satisfaction $f(x)$, or training deep learning models with hyperparameters x to maximize their performance $f(x)$. Function $f(x)$ is unknown in these cases because it is embedded inside the human brain (for the case of personalization) or too complex to derive (for hyperparameter tuning). However, for any chosen sample x_i, the value of $f(x_i)$ can be evaluated. For hearing-aid personalization, say, evaluating the function would entail playing audio with some hearing-compensation filter x_i and obtaining the audio clarity score $f(x_i)$ from the user.

Bayesian methods like Gaussian Process Regression (GPR) are de facto approaches to black-box optimization (BBO). Using a set of function evaluations, conventional GPR (Frazier 2018) learns a probabilistic surrogate model $\hat{f}(x)$ for $f(x)$. The optimum is estimated on this surrogate as $\hat{x}^* = \text{argmin} -\hat{f}(x)$. In most BBO problems, $f(x)$ is expensive to evaluate, hence a strict sample or query budget B is of interest. Techniques that lower this budget have garnered recent attention. One idea is to exploit domain knowledge about the rough shape of $f(x)$, i.e., select a GPR kernel that models this shape. With humans, for example, $f(x)$ may have a staircase structure as they may not perceive differences in certain neighborhoods of x, but their ratings may change just outside that neighborhood. If GPR's surrogate model $\hat{f}(x)$ captures this staircase structure in its kernel, sample efficiency can improve. However, in the absence of domain knowledge, *can the optimal GPR kernel* $\mathbf{K}^*$ *be learnt, using the same sample queries used to estimate* x^**?*

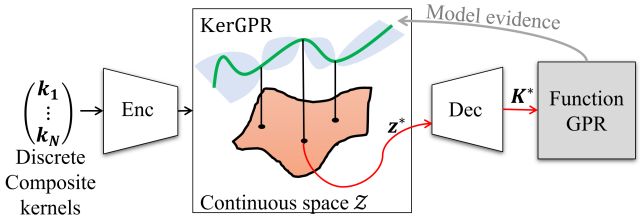


Figure 1: KerGPR in VAE latent space gives K^* to fGPR

A growing body of research (Grosse et al. 2012)(Kim and Teh 2016)(Teng et al. 2020) is concentrating on kernel learning. One effective approach is Automatic Statistician (AS) (Duvenaud et al. 2013) where authors compose complex kernels by combining simple ones through a context-free grammar and design a search method over the countably infinite complex kernels (more in Section 5). Subsequent improvements over AS have used *Hellinger distance* as a measure of kernel similarity (Malkomes, Schaff, and Garnett 2016). This similarity metric guides an optimization-based search over the space of composite kernels. To reduce search complexity, (Gardner et al. 2017) exploits additive structures in the search space and employs MCMC methods to discover kernels. All these techniques operate on discrete spaces, delivering a categorical composition of simple kernels.

A *continuous* space of kernels naturally lends itself to optimization methods. Our contribution is in designing such a continuous kernel space and running an auxiliary optimization on it to discover $\mathbf{K}^*$. To this end, we first synthesize many discrete kernels by adding or multiplying a set of simple "basis" kernels, and then use a variational autoencoder (VAE) to learn a low-dimensional continuous manifold for the discrete kernels. This manifold lives in the latent space of the VAE, as shown by the orange region in Figure 1. Con-

ventional optimization on this latent kernel space is difficult since we lack an objective function, but given a kernel, we can evaluate its effectiveness using *model evidence* (i.e., how well a given kernel agrees with available samples from $f(x)$). Thus, optimizing over the kernel space can also be designed as a blackbox optimization problem, hence we run a kernel space GPR *(KerGPR)* to output an optimal $\mathbf{K}^*$. The main GPR module, *Function GPR*, uses $\mathbf{K}^*$ to model the surrogate function $\hat{f}(x)$, and queries the surrogate at more points. The new model evidence is then passed back to KerGPR (see Fig. 1) to update the optimal kernel. The iteration terminates when the query budget is expended. *Function GPR* then outputs the minimum of the surrogate $\hat{f}(x)$. Results show that KOBO consistently outperforms SOTA methods (namely MCMC (Gardner et al. 2017), BOMS (Malkomes, Schaff, and Garnett 2016), and CKS (Duvenaud et al. 2013)) in terms of the number of queries needed to reach the optimal. The gain from kernel learning is also significant compared to the best static kernels. Experiments are reported across synthetic benchmark functions and from real-world audio experiments with U=6 users. Volunteers were asked to rate the clarity of audio clips, and using $B \leq 25$ ratings, KOBO prescribed a personalized filter that maximized that user's personal satisfaction. The performance gain is robust across extensive experiments, suggesting that KOBO could be deployed in real-world black-box applications where sample-budget is of prime concern.

2 Problem Formulation and Background

Problem Formulation

Consider an *unknown* real-valued function $f : \mathcal{H} \rightarrow \mathbf{R}$ where $\mathcal{H} \subseteq \mathbf{R}^N$, $N \geq 500$. Let x^* be the minimizer of $f(x)$. We aim to estimate x^* using a budget of B queries. Thus, the optimization problem is,

$$\underset{\hat{x} \in \mathcal{H}}{\operatorname{argmin}} \quad ||f(\hat{x}) - f(x^*)||_2 \qquad \text{s.t.} \qquad Q \leq B \tag{1}$$

where Q is the number of times the objective function is evaluated/queried, and the sample budget is $B \ll N$. Function f may be non-convex, may not have a closed-form expression, and its gradient is unavailable (hence, a black-box optimization problem). Bayesian approaches like GPR suit such problems but require choosing a kernel to model the function structure; a poor choice incurs more queries for optimization. Since queries can be expensive (e.g., when users need to answer many queries, or a NeuralNet needs re-training for each hyper-parameter configuration), lowering Q is of growing interest. Kernel learning aims to address this problem.

Background: Gaussian Process Regression (GPR)

Bayesian optimization (BO) (Frazier 2018) broadly consists of two modules: (1) *Gaussian Process Regression* that learns a Gaussian posterior distribution of the likely values of $f(x)$ at any point of interest x. (2) *Acquisition function*, a sampling strategy that prescribes the point at which f should be evaluated (or sampled) next. We briefly discuss GPR to motivate the kernel learning problem.

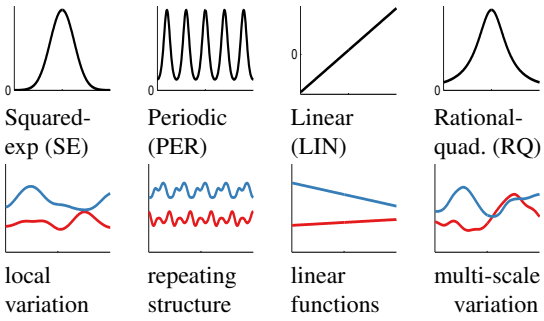


Figure 2: Row 1 shows simple (or base) kernels (Duvenaud et al. 2013). Row 2 shows corresponding surrogates drawn from a GP with the above kernel.

GPR Prior & Posterior: GPR generates a probabilistic surrogate model by defining a Gaussian distribution ($\mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$) over infinite candidate functions. At initialization, i.e., when no function-sample are available, the prior distribution over the candidate functions is defined by $\boldsymbol{\mu} = \mathbf{0}$ and a covariance matrix $\mathbf{K}$. This matrix is computed using a kernel function k as, $\mathbf{K}_{ij}$=$k(x_i, x_j)$. The kernel essentially favors candidate functions that are similar to the kernel's own shape/structure; these candidates are assigned a higher likelihood. An expert with domain knowledge about the structure of $f(x)$ can choose the kernel judiciously, resulting in better surrogate functions $\hat{f}(x)$. Better $\hat{f}(x)$ will ultimately reduce the number of queries needed to optimize the objective $f(x)$.
Once the function f has been observed for a set of samples $\mathcal{X} = \{x_1, x_2, \ldots, x_K\}$, i.e., we know $\mathcal{F} = \{f(x_1), f(x_2), \ldots, f(x_K)\}$, the prior is updated to form the posterior distribution (Eqn. 2) over the candidate functions. The posterior mean $\boldsymbol{\mu}$ is the most likely surrogate of the function f.

$$P(\mathcal{F}|\mathcal{X}) \sim \mathcal{N}(\mathcal{F}|\boldsymbol{\mu}, \mathbf{K}) \tag{2}$$

where, $\boldsymbol{\mu}$ = $\{\mu(x_1), \mu(x_2), \ldots, \mu(x_K)\}$, $\mathbf{K}_{ij}$=$k(x_i, x_j)$, and k represents a kernel function.
Predictions: To make predictions $\hat{\mathcal{F}} = f(\hat{\mathcal{X}})$ at new points $\hat{\mathcal{X}}$, GPR uses the current posterior (Eqn. 2) to define the conditional distribution of $\hat{\mathcal{F}}$ as: $P(\hat{\mathcal{F}}|\mathcal{F}, \mathcal{X}, \hat{\mathcal{X}}) \sim \mathcal{N}(\hat{\mathbf{K}}^T\mathbf{K}^{-1}\mathcal{F}, \hat{\hat{\mathbf{K}}} - \hat{\mathbf{K}}^T\mathbf{K}^{-1}\hat{\mathbf{K}})$ The details and proof of all the above are clearly explained in (Wang 2020)).

Kernel Selection

Kernels model the possible shape of the unknown function based on a set of observations $(\mathcal{X}, \mathcal{F})$ of the unknown function. Figure 2 illustrates example kernels on the top row; the bottom row shows candidate functions that GPR can derive using the corresponding kernel. In general, a class of kernels $\mathcal{K}$ produces a family of (GPR) surrogates that fit the function observations $(\mathcal{X}, \mathcal{F})$. Of course, each surrogate is associated to a likelihood which can improve with additional observations.
The goal of kernel selection is to select one kernel $\mathbf{K}^* \in \mathcal{K}$ that best explains the function observations $(\mathcal{X}, \mathcal{F})$. Let's denote $\mathcal{L} : \mathcal{K} \rightarrow \mathbf{R}$ to be a model evidence that measures

how well a kernel $\mathbf{K}$ fits the observations. We assume that evaluating $\mathcal{L}(\mathbf{K})$ for all kernels in $\mathcal{K}$ is too expensive. The kernel selection problem is then,

$$\mathbf{K}^* = \underset{\mathbf{K} \in \mathcal{K}}{\operatorname{argmax}}\, \mathcal{L}(\mathbf{K}) \tag{3}$$

This problem is difficult to optimize with Bayesian approaches when the kernel space $\mathcal{K}$ is discrete (Parker and Rardin 2014). This is because the function $\mathcal{L}(\mathbf{K})$ is only defined at the feasible points and cannot be queried arbitrarily. In contrast, it is possible to deduce a continuous function's behavior in a neighborhood of a point; in the discrete case, the behavior of the objective may change significantly as we move from one feasible point to another. This motivates transforming the problem in Eqn. 3 into a problem in continuous space on which optimization can be applied.
In this paper, the "model evidence" $\mathcal{L}$ is chosen to be GPR posterior in Eqn. 2 as it generates the surrogate that best describes the observations informed by the chosen kernel.

$$\mathcal{L}(\mathbf{K}) = P(\mathcal{F}|\mathcal{X}, \mathbf{K}) \tag{4}$$

3 Kernel Learning in KOBO

Intuition and Overview: Our prime objective is to create a continuous space $\mathcal{Z}$ corresponding to the discrete space $\mathcal{K}$, thus simplifying the optimization in Eqn. 3. We propose to achieve this using a Variational Autoencoder (VAE) which can take discrete inputs ($\mathbf{K} \in \mathcal{K}$) and learn (encoder) a continuous latent space $\mathcal{Z}$ from which the inputs can be faithfully reconstructed (decoder). If a large number of discrete kernels are created and represented sufficiently using a scheme that offers some notion of mutual similarity between kernels (i.e., representations defining a kernel space $\mathcal{K}$), then we expect the VAE to give us a continuous representation of such kernels $\mathcal{Z}$. This approach satisfies our objective since VAEs are expected to ensure *continuity* and *completeness* of their latent space, i.e., (i) two close points in the latent space cannot decode to completely different results, and (ii) a point sampled from the latent space must decode to a valid result. When the VAE is trained, we have successfully transformed the discrete optimization in $\mathcal{K}$ (Eqn. 4) to an easier continuous one in $\mathcal{Z}$.
Building on this intuition, KOBO's kernel learner is composed of 3 modules as shown in Figure 3:

(1) ***Kernel Combiner*** creates composite kernels $\mathbf{K} \in \mathcal{K}$ that form the discrete kernel space $\mathcal{K}$.

(2) ***Kernel Space Variational Autoencoder (KerVAE)*** trained on kernels generated by Kernel Combiner transforms discrete kernel space $\mathcal{K}$ to continuous space $\mathcal{Z}$.

(3) ***Kernel Space GPR (KerGPR)***: Optimizes model evidence (Eqn. 4) on $\mathcal{Z}$; this gives z^* which decodes to the optimal kernel $\mathbf{K}^*$.

Figure 3 connects all the modules to give a complete overview of KOBO. The main objective function (Eqn. 1) is optimized with a **Function GPR (fGPR)**. fGPR uses a simple Square-Exponential (SE) kernel to obtain a batch of observations $\mathcal{D}_n$. The model evidence is then passed to the kernel learning pipeline. The Kernel Combiner takes simple

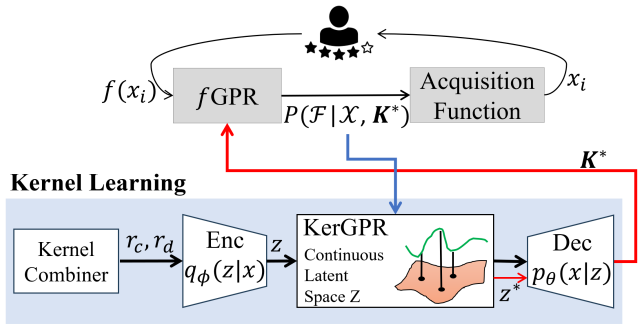


Figure 3: System flow: KOBO iterates across f*GPR* on top and *KerGPR* below that runs in the KerVAE latent space. The blue arrow denotes the model evidence input to KerGPR, and the red arrow denotes the optimal kernel $\mathbf{K}^*$ supplied by KerGPR to fGPR.

kernels and observations $\mathcal{D}_n$ as inputs, and outputs a discrete space of composite kernels $\mathbf{K}_\mathcal{C} \in \mathcal{K}$. This is guided by a context-free-grammar described later. The KerVAE is trained on this discrete space and generates the corresponding continuous latent space $\mathcal{Z}$. KerGPR running on $\mathcal{Z}$ optimizes the model evidence (from fGPR) to find the optimal kernel $\mathbf{K}^*$ which is prescribed to fGPR. fGPR uses this $\mathbf{K}^*$ to obtain a new batch of observations $\mathcal{D}_{n+1}$ and update model evidence, which is again passed to the kernel learning pipeline. The cycle iterates until fGPR has expended the sample budget B. At this point, fGPR outputs the minimum of its surrogate model. The following discussions expand on Kernel Combiner, KerVAE, and KerGPR.

Kernel Combiner

Complex kernels $\mathbf{k}_\mathcal{C}$ can be expressed as operations on the context-free grammar of base kernels $\mathcal{B}$ (Hopcroft, Motwani, and Ullman 2001; Duvenaud et al. 2013). Given $\mathcal{B} = \{A, B, C, D, E\}$, and a set of operators $\mathcal{O} = \{\text{add}, \text{multiply}, \text{end}, \dots\}$, the Kernel Combiner generates a composite kernel $\mathbf{k}_\mathcal{C}$ by drawing kernels from $\mathcal{B}$ and operators from $\mathcal{O}$ with probabilities $p_\mathcal{B}, p_\mathcal{O}$. An example $\mathbf{k}_\mathcal{C} = A * C + B * D$. In general, to form a kernel space $\mathcal{K}$, the Kernel Combiner develops a unique representation for each $\mathbf{k}_\mathcal{C}$.
Grammar-based Representation: Given a composite kernel $\mathbf{k}_\mathcal{C}$, its grammar-based representation is a vector r_c, designed as follows. Let $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}$ be five base kernels in $\mathcal{B}$. These are simple kernels like Square-exponential, Periodic, Rational Quadratic, etc. Any composite kernel $\mathbf{k}_\mathcal{C}$ created from the base kernels is expressed in the form of Eqn. 5. The code r_c is then the vector of indices, i.e., $r_c = [a_1, b_1, c_1, d_1, e_1, a_2, b_2, c_2, d_2, e_2, a_3, b_3, c_3, d_3, e_3]$.

$$\begin{aligned} \mathbf{k}_\mathcal{C} = {} & \mathbf{A}^{a_1} * \mathbf{B}^{b_1} * \mathbf{C}^{c_1} * \mathbf{D}^{d_1} * \mathbf{E}^{e_1} \\ & + \mathbf{A}^{a_2} * \mathbf{B}^{b_2} * \mathbf{C}^{c_2} * \mathbf{D}^{d_2} * \mathbf{E}^{e_2} \\ & + \mathbf{A}^{a_3} * \mathbf{B}^{b_3} * \mathbf{C}^{c_3} * \mathbf{D}^{d_3} * \mathbf{E}^{e_3} \quad \dots \end{aligned} \tag{5}$$

If a composite kernel is, say, $\mathbf{k}'_\mathcal{C} = \mathbf{A}^2 * \mathbf{E} + \mathbf{C} * \mathbf{D}$, then its Grammar-based representation would be $r'_c =$

$[2,0,0,0,1,0,0,1,1,0,0,0,0,0,0,0]$. Note: elements of the code vectors can also be fractions.

This encoding scheme has two advantages: (1) Each code r_c preserves its composition, i.e., given the code vector r_c, the base kernels and the operators used to construct $\mathbf{k}_\mathcal{C}$ can be interpreted. (2) The code space is continuous, hence, a code $r_c'' = [2,1,0,0,1,0,0,1,1,0,0,0,0,0,0,0]$ – which is only a flip of the second element in r_c' – results in $\mathbf{k}_\mathcal{C}'' = \mathbf{A}^2 * \mathbf{B} * \mathbf{E} + \mathbf{C} * \mathbf{D}$. In general, a small change in the code produces a small modification to the kernel composition (which makes the VAE's task: $\mathcal{K} \rightarrow \mathcal{Z}$ easier). Past work (Garrido-Merchán and Hernández-Lobato 2020)(Lu et al. 2018) have used one-hot encoding to represent kernel matrices, however, such one-hot schemes suffer from the lack of continuity (as shown in Figure 5(a) and (b) in the Appendix).

However, the context-free grammar r_c does not encode any information from the objective function to be modeled by the complex kernels. We add this to the representation next.

Data-based Representation: Given the available function observations $(\mathcal{X}, \mathcal{F})$, for each $\mathbf{k}_\mathcal{C}$, we compute the "distances" between its GPR covariance matrix $M_\mathcal{C}$ and the covariance matrix of each base kernel, $M_{b \in \mathcal{B}}$ (the covariance matrix is computed as $\mathbf{K}_{ij} = k(x_i, x_j)$, $k(\cdot)$ is the kernel function and x_i, x_j are any two observations). We use the Forbenius norm to compute the matrix distances. This representation is denoted as $r_d \in \mathcal{R}^{|\mathcal{B}|}$.

$$r_d = ||M_\mathcal{C} - M_{b \in \mathcal{B}}||_F \tag{6}$$

The function information is now encoded in r_d as the covariance matrix is computed using the kernel and the function observations/samples. Kernels that model the function's data similarly need to be in the same neighborhood so that the model evidence is sufficiently smooth. Figure 5(c) and (d) (in the Appendix) shows the advantage of encoding the objective function's data in the kernel code.

Thus, the kernel space $\mathcal{K}$ consisting of complex kernels is a subset of the space of all positive semi-definite (PSD) matrices $\mathbf{S}$, $\mathcal{K} \subseteq \mathbf{S}$. By restricting our search to $\mathcal{K}$– not $\mathbf{S}$ – the kernels generated through context-free grammar compositions, we can scale the GPR covariance matrix as new function observations arrive (simple kernel functions have closed-form expressions that are expanded to complex kernels for any number of observations). Therefore, our kernel learning problem in Eqn. 3 becomes a kernel selection problem. The final representation of a composite kernel $\mathbf{k}_\mathcal{C}$ is $r = [r_c, r_d]$. This is used to train the KerVAE to generate the continuous kernel space $\mathcal{Z}$.

Kernel Space Variational Autoencoder (KerVAE) and GPR (KerGPR)

The KerVAE learns a continuous latent space $\mathcal{Z}$ of the discrete kernel space $\mathcal{K}$ and has two main components: (1) a probabilistic encoder that models $q_\phi(z|x) \sim p_\theta(x|z)p(z)$ parameterized by ϕ where $p(z)$ is the prior over the latent space, and (2) a decoder that models the likelihood $p_\theta(x|z)$ parameterized by θ. The parameters of $q_\phi(z|x), p_\theta(x|z)$ are optimized by joint maximization of the ELBO loss (Kingma and Welling 2013),

$$\mathcal{L}(\phi, \theta, x) = \mathbf{E}_{q_\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z|x)] \tag{7}$$

Q	$f_1(x) \sim A * A * B + C$	$f_2(x) \sim C + D$	$f_3(x) \sim D * B + D$
5	$A * B$	A	A
10	$A * B + C$	$A + D$	$A * B * D + D$
15	$A * A * B + D$	$A * C + D$	$A * B + D$
20	$A * B + C * D$	$A * C + D$	$B * D + D$
25	$A * A * B + C * D$	$A * C + D$	$B * D + D$

Table 1: KOBO learning the ground truth kernel. $\{A, B, C, D, E\} = \{\text{SE}, \text{PER}, \text{RQ}, \text{MAT}, \text{LIN}\}$

The KerVAE is re-trained after accumulating v function observations as the data representation r_d is re-computed for every new set of observations $\mathcal{D}_v = (\mathcal{X}_v, \mathcal{F}_v)$. Once KerVAE is trained, KerGPR is used to determine the optimal kernel $z^* \in \mathcal{Z}$. The KerVAE decoder decodes z^* to r_c^* and r_c^* is easily mapped to $\mathbf{K}^* \in \mathcal{K}$ due to the grammar's interpretability.

KerGPR: Optimizing on KerVAE's latent kernel space $\mathcal{Z}$ is also a black-box problem (similar to Eqn. 1) because the objective can *only* be evaluated for a given kernel k (i.e., by first decoding a given z to k, and computing k's model evidence $\mathcal{L}(\mathbf{K})$ in Eqn. 4). We use GPR to find z^* in the latent space and decode to the optimal kernel $\mathbf{K}^*$. Thus, our optimization objective is:

$$\mathbf{K}^* = \text{Dec}(\underset{z \in \mathcal{Z}}{\text{argmax}}\, P(\mathcal{F}|\mathcal{X}, \text{Dec}(z))) \tag{8}$$

where, Dec(.) is the KerVAE decoder that maps a point from $\mathcal{Z}$ to the $\mathcal{K}$ space. We use the simple SE kernel in KerGPR as it does not benefit from recursive kernel learning (explained in Technical Appendix). The optimal kernel $\mathbf{K}^*$ is then used by Function GPR (fGPR) — the GPR posterior in Eqn. 2 — to generate surrogates that closely model the unknown function structure.

4 Evaluation and Results

KOBO Versus Simple Kernels

Metric: We use the metric of **Regret**, defined as the difference between the predicted and true minimum, $(f(\hat{x}^*) - f(x^*))$. We compare KOBO's regret against 5 popular base kernels $\mathcal{B} = \{\text{SE}, \text{PER}, \text{RQ}, \text{MAT}, \text{LIN}\}$ which respectively denote Square-Exponential (SE), Periodic (PER), Rational Quadratic (RQ), Matérn (MAT), and Linear (LIN) kernels. All KOBO experiments are initialized with the SE kernel.

Synthetic Baseline Functions: We report results from 3 types of synthetic functions $f(x)$ that are popular benchmarks (Kim 2020) for black-box optimization: ■ Staircase functions in $N = 2000$ dimensions; they exhibit non-smooth structures (Al-Roomi 2015). ■ Smooth benchmark functions such as BRANIN commonly used in Bayesian optimization research (Sonja Surjanovic 2013). ■ Periodic functions such as MICHALEWICZ that exhibit repetitions in their shape (Sonja Surjanovic 2013). The top row of Figure 4 visualizes these functions (more details on evaluation parameters in the Technical Appendix). All reported results are an average of 10 runs.

Results: Figures 4(bottom row) plots Regret for the Staircase, Smooth(BRANIN), and the MICHALEWICZ functions, respectively. KOBO minimizes

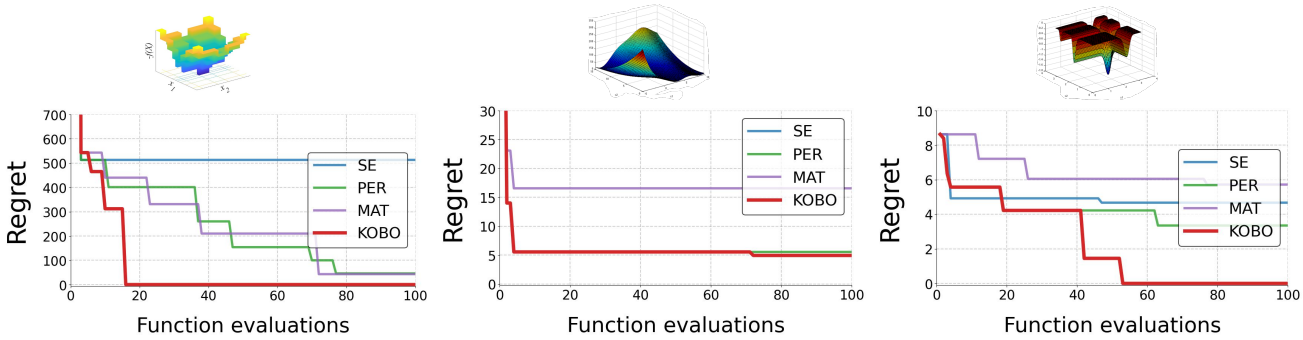


Figure 4: Comparison of KOBO and conventional BO using SE, PER, RQ, and Matérn kernels (Bottom Row) for (a) Staircase, (b) Smooth Branin, and, (c) Periodic Michalewicz functions (Top Row).

Regret at significantly fewer function evaluations (or samples), especially for Staircase and MICHALEWICZ. For a smooth function like BRANIN, KOBO's gain is understandably less since the SE and PER kernels naturally fit the smooth shape. When real world functions exhibit complex (non-smooth) structures and when function evaluations are expensive, KOBO's advantage is desirable.

KOBO Versus SOTA Baselines

Another Metric: Since KOBO learns the kernel in the latent space, we will use Model Evidence in addition to Regret. Model Evidence is the normalized probability of generating the observed data $\mathcal{D}$ given a kernel model $\mathbf{K}$, i.e., $\log(P(\mathbf{f}|\mathcal{X},\mathbf{K}))/|\mathcal{D}|$ (Malkomes, Schaff, and Garnett 2016). Computing the exact model evidence is generally intractable in GPs (Rasmussen, Williams et al. 2006)(MacKay et al. 1998). We use the Bayesian Information Criterion (BIC) to approximate the model evidence as $\log(P(\mathbf{f}|\mathcal{X},\mathbf{K})) = -\frac{1}{2}\mathbf{f}^T\mathbf{K}^{-1}\mathbf{f} - \frac{1}{2}\log((2\pi)^N|\mathbf{K}|)$, where N is the dimensions of the input space $\mathcal{H} \subseteq \mathbf{R}^N$.
We will plot Regret against the number of "Function Evaluations" (on the X axis), but for Model Evidence, we will plot it against the number of "Latent Evaluations". Recall that Model Evidence is the metric used in the latent space of KerVAE to find the "best" kernel $\mathbf{K}^*$. Hence "Latent Evaluations" denotes the number of latent space samples z and the corresponding kernels $\text{Dec}(z) = \mathbf{K}$ sampled by KerGPR to find $\mathbf{K}^*$. This reflects the computation overhead of KOBO.
SOTA Baselines (details in Technical Appendix):
(1) MCMC: The MCMC kernel search (Gardner et al. 2017; Abdessalem et al. 2017) applies the Metropolis-Hastings algorithm (Gardner et al. 2017) on the space of composite kernels $\mathbf{k}_\mathcal{C}$, using model evidence as the function. The proposal distribution is defined as: given a kernel $\mathbf{k}$, it is either added or multiplied to a kernel from $\mathcal{B}$ (chosen with p).
(2) CKS: The Automatic Statistician/Compositional Kernel Search (CKS) (Duvenaud et al. 2013) method takes advantage of the fact that complex kernels are generated as context-free grammar compositions of positive semi-definite matrices (closed under addition and multiplication); the kernel selection is then a tree search guided by model evidence. CKS searches over the discrete kernel space $\mathcal{K}$ using a greedy strategy that, at each iteration, chooses the kernel with the highest model evidence. This kernel is then expanded by composition to a set of new kernels. The search process repeats on this expanded list.
(3) BOMS: The Bayesian Optimization for Model Search (BOMS) (Malkomes, Schaff, and Garnett 2016), unlike CKS' greedy strategy, is a meta-learning technique, which, conditioned on observations $\mathcal{D}$, establishes similarities among the kernels in $\mathcal{K}$, i.e., BOMS constructs a kernel between the kernels ("kernel kernel"). Like KOBO, BOMS performs BO in $\mathcal{K}$ by defining a Gaussian distribution: $P(g) = \mathcal{N}(g;\mu_g,\mathbf{K}_g)$, where g is the model evidence, μ_g is the mean, and $\mathbf{K}_g$ is the covariance (defined by "kernel kernel" function). $\mathbf{K}_g$ is constructed by defining a heuristic similarity measure between two kernels: *Hellinger distance*.
Results: Figure 5(Row 1) shows that KOBO lowers Regret faster than all SOTA baselines for the three benchmark functions. For Staircase, KOBO attains the global minimum in about 17 function evaluations in contrast to MCMC, which incurs 28, BOMS 32, and CKS 43. For Michalewiez, KOBO attains the minimum in about 10 fewer samples than MCMC. While BOMS and CKS do not attain the minimum but get close to it. However, KOBO's performance gain is not as pronounced for Branin due to its smooth structure as evidenced in Figure 4.
Figure 5(Row 2) compares Model Evidence for the same benchmarks. For Staircase, KOBO's KerGPR achieves significantly higher Model Evidence in 20 iterations compared to MCMC, i.e., KOBO's optimal kernel $\mathbf{K}^*$ better explains the observed data. For Branin, KOBO can match the Model Evidence of baselines, and performs modestly better for Michalewiez. The results illustrate that KOBO's performance is superior because a continuous search space learned by KerVAE simplifies the KerGPR optimization to determine $\mathbf{K}^*$, implying that KOBO presents an efficient search of the discrete kernel space $\mathcal{K}$ in contrast to sub-optimal search techniques like greedy search (CKS), or heuristic metrics for kernel optimization (BOMS).

Is K^* indeed learning the structure of $f(x)$?

Synthetic Functions: If we knew the objective function $f(x)$, we could verify if $\mathbf{K}^*$ has learnt its structure. To test

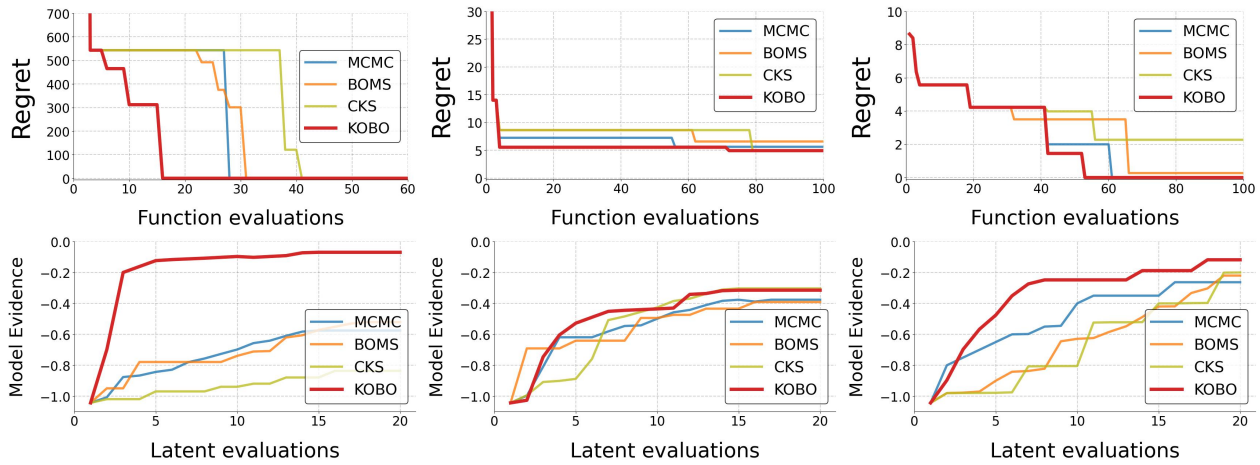


Figure 5: *Comparison of* KOBO*, MCMC, CKS, and BOMS for Staircase (Col 1), Branin (Col 2), and Michalewiez (Col 3) functions:* (Row 1) `Regret` (Row 2) `Model Evidence`.

this, we sample a function from a GP with a known kernel K^+ and pretend that to be $f(x)$; we check if KOBO's $\mathbf{K}^*$ converges to K^+. Table 1 reports results from 3 N-dimensional synthetic functions, shown in the top row ($N = 2000$). These synthetic objective functions were sampled from a GP that uses different but *known* kernels. The subsequent rows show KerVAE's learnt kernel after Q observations/queries. With more Q, KerGPR closely matches the ground truth kernel.

Learning Real-world CO_2 Emission Data: Figure 6's blue curve plots real CO_2 emissions data over time (Thoning, Tans, and Komhyr 1989). We test if KOBO's $\mathbf{K}^*$ can learn the structure of this blue curve from partial data. Figure 6(a,b,c) show results when KOBO has observed the first 20%, 40%, and 60% of the data, respectively. With the first 20%, $\mathbf{K}^* = \text{SE} * \text{PER} + \text{RQ}$, hence KOBO's red curve captures the periodic structure of the early data. When the first 40% of the data is observed, KOBO captures the downward linear trend of the CO_2 data resulting in $\mathbf{K}^* = \text{SE} * \text{PER} + \text{PER} + \text{LIN}$. With 60% of the data, $\mathbf{K}^* = \text{SE} * \text{PER} * \text{RQ} + \text{PER} * \text{LIN} + \text{LIN}$ models the interplay between the function's periodic structure and linear trends. A conventional Periodic kernel (PER), shown in black in Figure 6(c), is only able to capture the periodic structure, not the upward linear trend, even with 60% of the data.

User Experiments: (1) Audio Personalization, and (2) Image Recommendation

(1) Audio: We apply KOBO to audio personalization for real volunteers. We deliberately corrupt audio played to the user with the aim of helping the user pick a filter h^* that cancels the effect of the corruption – equalization – and recovers the original audio; hence maximizing the user's audio satisfaction. Therefore, a GPR employed in the space of all audio filters $\mathcal{H}$, maximizes the user's satisfaction $f(h)$ at h^*. At each iteration, the corrupted audio is filtered with a new h' (recommended by GPR) and played to the user. The user's rating (0 to 10) of the perceived audio quality serves as the function observations $f(h')$. User feedback is finite and the frequency selective nature of human hearing (Antoine Lorenzi 2003) makes optimizing $f(h), h \in \mathcal{R}^{4000}$, well suited for kernel learning methods like KOBO.

Hearing Loss									
Q	U1			U2			U3		
	SE	KOBO	PER	SE	KOBO	PER	SE	KOBO	PER
5	6	**6**	6	8	**8**	8	6	**6**	6
10	6	**8**	6	8	**8**	8	6	**7**	7
15	6	**10**	7	8	**10**	8	7	**9**	7
20	10	**10**	10	9	**10**	9	7	**10**	10
25	10	**10**	10	10	**10**	10	9	**10**	10
Random Audio Corruption									
Q	U1			U2			U3		
	SE	KOBO	PER	SE	KOBO	PER	SE	KOBO	PER
5	1	**1**	1	3	**3**	3	1	**1**	1
10	2	**3**	1	4	**4**	4	2	**2**	3
15	2	4	5	4	**10**	4	2	**2**	3
20	4	**10**	5	4	**10**	9	5	**8**	4
25	8	**10**	9	8	**10**	9	10	**8**	7

Table 2: Audio personalization results (the $U = 3$ volunteers (rest in Appendix) did not know which kernel was in use).

We invited 6 volunteers to two experiment sessions. In the first, the audio was corrupted with a "hearing loss" audiogram (CDC 2011); in the second, a "random" corruption filter was used (more details in Technical Appendix). By querying the user Q times, each time with a new filter h', KOBO expects to learn K^*, and in turn, maximize the user's satisfaction. We report `Regret` against increasing Q, and compare to conventional GPR optimizers with simple base kernels $\{\text{SE}, \text{PER}\}$. The audio demos at various stages of the optimization is made public: https://keroptbo.github.io/.

(2) Images: We also apply KOBO to image recommendation. The motivation is that users are often unable to articulate precisely what images they want, however, given an image they can express their satisfaction. We formulate this as a black-box optimization problem in the image space by posing the question: *if a user rates few pictures generated by*

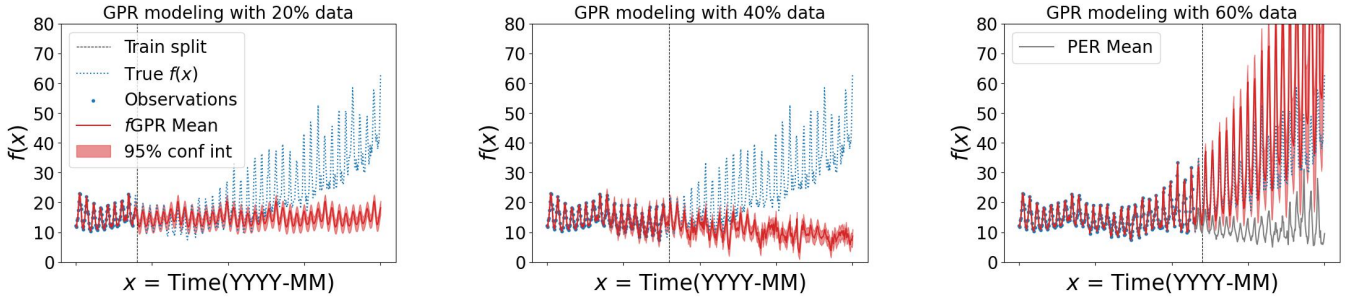


Figure 6: The blue curve is real-world CO_2 emissions data from (Thoning, Tans, and Komhyr 1989). The red curve is KOBO's prediction of the blue curve after observing (a) 20% (b) 40% (c) 60% of the blue data.

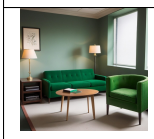
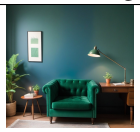
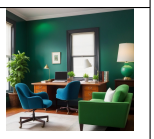
x_{start}	$x_{Q=5}$	$x_{Q=15}$	$x_{Q=25}$
Prompt: Office room with a desk, a blue chair, a lamp on the desk, and a green couch on the side			
			
Prompt: Garden with a fountain in the center and surrounded by trees and flowers of different colors			
			

Table 3: Prompt-based image generation user results

an AI model, can the model find the "best" picture to maximize user satisfaction? To realize this, we use a pre-trained VAE-based image generator (Esser, Rombach, and Ommer 2020) (ImGen). The user issues a crude prompt of what they want, allowing ImGen to display an initial image, x_{start}. The user scores this image as $f(x_{start})$ and KOBO is triggered. Table 3 displays x_{start} and the images recommended after $Q = 5, 15, 25$ queries. For a subjective measure of optimality, we asked users to describe their ideal image upfront. KOBO's recommendation at $Q \geq 15$ seems to match the prompts quite well (more results in Appendix).

5 Related Work

A body of works in BO has explored kernel learning. Closest to KOBO are BOMS (Malkomes, Schaff, and Garnett 2016), Automatic Statistician (AS) and their variants (Duvenaud et al. 2013; Grosse et al. 2012; Kim and Teh 2016), and MCMC (Gardner et al. 2017; Abdessalem et al. 2017) discussed (and used as baselines) earlier. In other work (Teng et al. 2020), authors treat the kernel as a random variable and learn its belief from the data. (Zhen et al. 2020) introduces kernels with random Fourier features for meta-learning tasks. The kernel features are learned as latent variables of a model to generate adaptive kernels. In contrast, KOBO uses variational inference as an auxiliary module to only learn a *continuous* latent kernel space; the KerGPR optimization primarily drives the kernel learning.

Authors in (Kandasamy, Schneider, and Póczos 2015; Gardner et al. 2017; Mutny and Krause 2018; Wang et al. 2018) have improved BO performance in high-dimensional spaces by modeling the function via additive kernels. The objective is decomposed into a sum of functions in low-dimensional space. KOBO's comprehensive space of kernels from additive and multiplicative compositions is capable of modeling more complex function structures. Finally, (Kusner, Paige, and Hernández-Lobato 2017), (Gómez-Bombarelli et al. 2018), and (Gonzalez et al. 2015) perform optimization in a continuous latent space learned by VAEs to circumvent categorical data. Authors of (Garrido-Merchán and Hernández-Lobato 2020) use one-hot encoding approximations for BO of categorical variables. KOBO borrows from these ideas but applies them to kernel learning.

6 Limitations and Conclusion

■ **Trading Computation for Sample Efficiency:** KOBO incurs rounds of computation in estimating the model evidence $\mathcal{L}$. However, this does not affect sample efficiency, since KerVAE training is sample-independent. Thus, KOBO's advantage is in reducing the sample evaluations of $f(x)$ (e.g., user burden) and not in total CPU cycles.

■ **Overfitting to Simple Functions:** As iterations progress, KerGPR might learn a kernel more complex than the actual target function f (see $f_1(x)$ in Table 1). Choosing a complex kernel expands the surrogate function space, and may need more samples to converge. To avoid kernel overfitting, we can regularize the kernel complexity, i.e., the length and norm of the kernel grammar codes.

■ **Latent Space Interpretability:** Current latent space learned by KerVAE is abstract and lacks interpretability. An interpretable latent space should offer improvements to KerGPR, facilitating the use of simpler optimizers compared to the expensive Bayesian Optimization in the latent space.

To conclude, we propose KOBO, a kernel learning method for GPR. We design a continuous latent space of kernels (using a VAE), and optimize the space via an auxiliary GPR to output an optimal kernel $\mathbf{K}^*$. This optimal kernel better models the structure of the objective function, which ensures sample efficiency. We show sample applications and believe the ideas could be extended to others.

Acknowledgements

We thank Foxconn and NSF (grant 2008338, 1909568, 2148583, and MRI-2018966) for funding this research. We are also grateful to the reviewers for their insightful feedback.

References

Abdessalem, A. B.; Dervilis, N.; Wagg, D. J.; and Worden, K. 2017. Automatic kernel selection for gaussian processes regression with approximate bayesian computation and sequential monte carlo. *Frontiers in Built Environment*, 3: 52.

Al-Roomi, A. R. 2015. Unconstrained Single-Objective Benchmark Functions Repository.

Antoine Lorenzi, B. C. 2003. Human Frequency Discrimination.

CDC. 2011. Centers for Disease Control and Prevention (CDC). National Center for Health Statistics (NCHS). National Health and Nutrition Examination Survey Data, Hyattsville, MD.

Duvenaud, D.; Lloyd, J.; Grosse, R.; Tenenbaum, J.; and Zoubin, G. 2013. Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*, 1166–1174. PMLR.

Esser, P.; Rombach, R.; and Ommer, B. 2020. Taming Transformers for High-Resolution Image Synthesis. arXiv:2012.09841.

Frazier, P. I. 2018. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*.

Gardner, J.; Guo, C.; Weinberger, K.; Garnett, R.; and Grosse, R. 2017. Discovering and exploiting additive structure for Bayesian optimization. In *Artificial Intelligence and Statistics*, 1311–1319. PMLR.

Garrido-Merchán, E. C.; and Hernández-Lobato, D. 2020. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380: 20–35.

Gómez-Bombarelli, R.; Wei, J. N.; Duvenaud, D.; Hernández-Lobato, J. M.; Sánchez-Lengeling, B.; Sheberla, D.; Aguilera-Iparraguirre, J.; Hirzel, T. D.; Adams, R. P.; and Aspuru-Guzik, A. 2018. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2): 268–276.

Gonzalez, J.; Longworth, J.; James, D. C.; and Lawrence, N. D. 2015. Bayesian optimization for synthetic gene design. *arXiv preprint arXiv:1505.01627*.

Grosse, R.; Salakhutdinov, R. R.; Freeman, W. T.; and Tenenbaum, J. B. 2012. Exploiting compositionality to explore a large space of model structures. *arXiv preprint arXiv:1210.4856*.

Hopcroft, J. E.; Motwani, R.; and Ullman, J. D. 2001. Introduction to automata theory, languages, and computation. *Acm Sigact News*, 32(1): 60–65.

Kandasamy, K.; Schneider, J.; and Póczos, B. 2015. High dimensional Bayesian optimisation and bandits via additive models. In *International conference on machine learning*, 295–304. PMLR.

Kim, H.; and Teh, Y. W. 2016. Scalable structure discovery in regression using gaussian processes. In *Workshop on Automatic Machine Learning*, 31–40. PMLR.

Kim, J. 2020. Benchmark Functions for Bayesian Optimization.

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kusner, M. J.; Paige, B.; and Hernández-Lobato, J. M. 2017. Grammar variational autoencoder. In *International conference on machine learning*, 1945–1954. PMLR.

Lu, X.; Gonzalez, J.; Dai, Z.; and Lawrence, N. D. 2018. Structured variationally auto-encoded optimization. In *International conference on machine learning*, 3267–3275. PMLR.

MacKay, D. J.; et al. 1998. Introduction to Gaussian processes. *NATO ASI series F computer and systems sciences*, 168: 133–166.

Malkomes, G.; Schaff, C.; and Garnett, R. 2016. Bayesian optimization for automated model selection. *Advances in neural information processing systems*, 29.

Mutny, M.; and Krause, A. 2018. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, 31.

Parker, R. G.; and Rardin, R. L. 2014. *Discrete optimization*. Elsevier.

Rasmussen, C. E.; Williams, C. K.; et al. 2006. *Gaussian processes for machine learning*, volume 1. Springer.

Sonja Surjanovic, D. B. 2013. Virtual Library of Optimization Functions.

Teng, T.; Chen, J.; Zhang, Y.; and Low, B. K. H. 2020. Scalable variational Bayesian kernel selection for sparse Gaussian process regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 5997–6004.

Thoning, K. W.; Tans, P. P.; and Komhyr, W. D. 1989. Atmospheric carbon dioxide at Mauna Loa Observatory: 2. Analysis of the NOAA GMCC data, 1974–1985. *Journal of Geophysical Research: Atmospheres*, 94(D6): 8549–8565.

Wang, J. 2020. An intuitive tutorial to Gaussian processes regression. *arXiv preprint arXiv:2009.10862*.

Wang, Z.; Gehring, C.; Kohli, P.; and Jegelka, S. 2018. Batched large-scale Bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, 745–754. PMLR.

Zhen, X.; Sun, H.; Du, Y.; Xu, J.; Yin, Y.; Shao, L.; and Snoek, C. 2020. Learning to learn kernels with variational random features. In *International Conference on Machine Learning*, 11409–11419. PMLR.