Optimizing Dynamic Structures with Bayesian Generative Search

Minh Hoang [1] **Carl Kingsford** [2]

Abstract

Kernel selection for kernel-based methods is prohibitively expensive due to the NP-hard nature of discrete optimization. Since gradient-based optimizers are not applicable due to the lack of a differentiable objective function, many state-of-the-art solutions resort to heuristic search or gradient-free optimization. These approaches, however, require imposing restrictive assumptions on the explorable space of structures such as limiting the active candidate pool, thus depending heavily on the intuition of domain experts. This paper instead proposes **DTERGENS**, a novel generative search framework that constructs and optimizes a high-performance composite kernel expressions generator. **DTERGENS** does not restrict the space of candidate kernels and is capable of obtaining flexible length expressions by jointly optimizing a generative termination criterion. We demonstrate that our framework explores more diverse kernels and obtains better performance than state-of-the-art approaches on many real-world predictive tasks.

1. Introduction

At the core of most machine learning (ML) algorithms, practitioners will often find two universally essential components: model specification and (hyper)-parameter optimization. While there has been reasonable progress on automating the latter (Bergstra et al., 2013; Kandasamy et al., 2015; Wang et al., 2017; Hoang et al., 2018), model selection remains an art due to the complexity of a highly structured space of model architectures and the lack of a differentiable objective. In deep learning, for example, the space of architectures encompasses neural network design choices such as the number of hidden layers, dimension of each hidden layer and its activation function (Elsken et al., 2018). For kernel-based methods such as support vector machine (SVM) (Hearst, 1998), Gaussian processes (GP) (Rasmussen & Williams, 2006) or sparse Gaussian processes (Quiñonero-Candela & Rasmussen, 2005; Quiñonero-Candela et al., 2007; Hensman et al., 2013; Gal & Turner, 2015; Low et al., 2015; Hoang et al., 2015; 2017; Yu et al., 2019), we are interested in the space of semi-positive definite composite kernels closed under addition and multiplication. In this paper, we focus on the latter, which is important in many learning disciplines such as active learning (Castro et al., 2005; Krause & Guestrin, 2007; Castro, 2007; Low et al., 2012; Cao et al., 2013; Hoang et al., 2014a;b), distributed learning (Chen et al., 2013; Gal et al., 2014; Hoang et al., 2016; Allamraju & Chowdhary, 2017; Hoang et al., 2019a;b) and reinforcement learning (Poupart et al., 2006; Poupart & Vlassis, 2008; Akchurina, 2009; Ziebart et al., 2010; Hoang & Low, 2013; Engel et al., 2013; da Motta Salles Barreto et al., 2014; Zoph & Le, 2016).

1.1. Related works

In each example above, the space of model architecture typically does not have a well-defined distance metric and is therefore not amenable to popular gradient-based optimization techniques such as gradient descent and coordinate descent (Lu et al., 2018). To sidestep this issue, existing solutions for model selection resort to either heuristic search (Duvenaud et al., 2013; Idrissi et al., 2016) or gradient-free optimization methods (Malkomes et al., 2016; Kandasamy et al., 2018; Lu et al., 2018). These methods, however, either (a) compromise search efficiency (Idrissi et al., 2016; Malkomes et al., 2016); or (b) restrict the space of valid structure candidates, thus potentially leaving out high-performing candidates (Duvenaud et al., 2013; Kandasamy et al., 2018; Lu et al., 2018). More specifically, there are 4 lines of relevant work which are detailed below:

1. Naïve random search, grid search and genetic algorithms (Idrissi et al., 2016), despite being frequently employed due to their ease of implementation, are generally slow to converge, hence less practical, under complex and high-

[1] Computer Science Department, School of Computer Science, Carnegie Mellon University, USA [2] Computational Biology Department, School of Computer Science, Carnegie Mellon University, USA. Correspondence to: Minh Hoang <qhoang@andrew.cmu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

dimensional structure space.

2. Duvenaud et al. (2013) instead takes advantage of the composition rule for semi-positive definite kernels to formulate the kernel selection problem as an informed tree search guided by model likelihood score. This approach, however, requires fixing a default set of model parameters during the search and does not take into account the performance gain from optimizing these parameters.

3. Kandasamy et al. (2018) and Malkomes et al. (2016) factor in parameter optimization for neural architecture search and kernel search respectively through black-box Bayesian Optimization (BO). These methods, however, must rely on random mutation operators to navigate between structures in a user-determined active set of candidates, hence compromising both the efficiency of the search (i.e., sensitive to randomization) and limiting the valid pool of structures to the active candidates.

4. Lu et al. (2018) uses Variational Autoencoder (VAE) (Kingma & Welling, 2013) to learn an embedding of the composite kernel structures space onto a continuous latent representation space. Employing BO on this latent space helps to circumvent the need to navigate between discrete structures. Its decoder, however, is limited to generating composite kernels with upper-bounded dimensions. While setting this upper bound to be arbitrarily large can alleviate this problem, VAE would then require a prohibitively large training set to learn an accurate embedding.

1.2. Our contributions

To overcome these shortcomings, our framework reformulates the structure discrete optimization problem as an optimization task over the parameter space of a latent generative process (Section 3.1), which takes the form of an *open-ended structure generator* guided by a *data-driven termination policy* (Section 3.2). This allows us to employ BO on a well-behaved embedding space of generative parameters, thus bypassing both the need to select an initial set of active candidates (Kandasamy et al., 2018; Malkomes et al., 2016) and to rely on heuristic operators to navigate between discrete structures (Kandasamy et al., 2018; Malkomes et al., 2016; Idrissi et al., 2016).

Furthermore, unlike other embedding-based approaches such as Lu et al. (2018) which imposes restrictive assumption on the space of interest, our approach adds flexibility and expressiveness through embedding kernels with a recursive generative process. Specifically, our latent embedding maps to a subspace of kernel expressions (i.e., all possible expressions on an infinite generative trajectory characterized by its generative parameters) instead of a single expression (Lu et al., 2018). Distillation of high-performing kernels from this subspace is guided by a stochastic termination policy which determines the best stopping point on the trajectory. This enables exploration of arbitrarily complex expressions and, to the best of our knowledge, makes our framework the first kernel selection method that places no structural restriction on the search space.

We further show that both the generative parameters and the termination policy can be optimized in tandem by exploiting their dynamic in the generative component. In particular, with respect to a fixed policy, we devise a dynamic BO algorithm for optimizing generative parameters that is capable of adapting to the constant policy updates (Section 3.3). Alternately, given each sample trajectory collected by the BO step, we devise a non-myopic update for the policy distribution via modelling the dynamic between these two components (Section 3.4). Together, this workflow is described as the **D**ynamic **TER**mination **GEN**erative **S**earch (**DTERGENS**) algorithm for composite kernel selection (Algorithm 1), which delivers our main contribution.

Last but not least, we demonstrate that our method is able to produce complex kernels which significantly improve predictive performance of multiple predictive tasks over state-of-the-art structure search methods. Our results show a wider range of structures being explored by **DTERGENS** and more rapid rates of improvement as compared to other methods. Finally, we show that **DTERGENS** is also able to recover known well-performing kernels on artificially designed predictive tasks (Section 4).

2. Background and Notations

2.1. Kernels and Composite Kernels

Many kernel-based models such as Gaussian processes (Rasmussen & Williams, 2006) and SVM (Cortes & Vapnik, 1995) employ the notion of a covariance function $k(\mathbf{x}, \mathbf{x}') = \mathrm{cov}(g(\mathbf{x}), g(\mathbf{x}'))$ to define the similarity between latent function values given corresponding inputs. The covariance matrix evaluated over two finite sets of observations $\boldsymbol{\tau} = [\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_n]$ and $\boldsymbol{\tau}' = [\mathbf{x}'_1, \mathbf{x}'_2 \ldots \mathbf{x}'_m]$ is further denoted by $\mathbf{K}_{\boldsymbol{\tau}\boldsymbol{\tau}'}$ whose (i, j)-entry stores the value of $k(\mathbf{x}_i, \mathbf{x}'_j)$ for all $i \in [n]$ and $j \in [m]$. For convenience, we use the shorthand $\mathbf{K}_{\boldsymbol{\tau}}$ for $\mathbf{K}_{\boldsymbol{\tau}\boldsymbol{\tau}'}$ when $\boldsymbol{\tau} \equiv \boldsymbol{\tau}'$.

As formalized in Duvenaud et al. (2013), a composite kernel is the result of applying summation and multiplication (recursively) on a finite set of base kernels, which includes the common squared exponential (k_{SE}), periodic (k_{PER}), linear (k_{LIN}), and rational quadratic (k_{RQ}) kernels. The entire space of such composite kernels can therefore be succinctly characterized by the following kernel composing rules:

1. Every base kernel $k \in \mathcal{K}_{\mathcal{B}} \triangleq \{k_{\mathrm{PER}}, k_{\mathrm{LIN}}, k_{\mathrm{RQ}}, k_{\mathrm{SE}}\}$ is a valid kernel.

2. A valid kernel k can be created by adding existing valid kernels, $k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}')$.

3. A valid kernel k can be created by multiplying existing valid kernels, $k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}') \times k_2(\mathbf{x},\mathbf{x}')$.

For ease of notation, we generally drop the input argument and instead use the shorthand $k = k_1 + k_2$ and $k = k_1 \times k_2$ when referring to these composing rules. Let $\mathcal{K}_{\mathcal{C}}$ denote the space of composite kernels defined above, any valid composite kernel $k \in \mathcal{K}_{\mathcal{C}}$ can then be expressed as a sum-of-product of base kernels in $\mathcal{K}_{\mathcal{B}}$ (Duvenaud et al., 2013). That is, for any composite kernel k, there exists a finite collection of base kernels $k_{uv} \in \mathcal{K}_{\mathcal{B}}$ for which,

$$k(\mathbf{x},\mathbf{x}') \;=\; \sum_{u=1}^{m}\left[\prod_{v=1}^{n_u} k_{uv}(\mathbf{x},\mathbf{x}')\right] \tag{1}$$

where m denotes the number of product terms in the expression and n_u denotes the number of base kernels in the u^{th} product term. To generate kernel expressions of this form, we design a system of nested recurrent neural networks (RNN) that recursively grows the sum-of-product expression by appending new base kernel units to it until a termination condition is reached (Section 3.2), upon which the system returns a single terminal expression. For completeness, we will refer to any kernel expression produced by this generator prior to the terminal state as an intermediate kernel expression. That is, the composite kernel k' is an intermediate expression with respect to k or $k' \subseteq_{\mathcal{K}} k$ iff

$$k'(\mathbf{x},\mathbf{x}') \;=\; \sum_{u=1}^{m'}\left[\prod_{v=1}^{n'_u} k_{uv}(\mathbf{x},\mathbf{x}')\right] \tag{2}$$

for some $m' \leq m$ and $n'_u \leq n_u$ and the terminal expression k defined in Eq. (1) above[1].

2.2. Bayesian Optimization

Many optimization problems in ML are black-box optimization tasks of the form $\arg\max_{\mathbf{x}} F(\mathbf{x})$ where we neither have (or do not want to compute for complexity reason) an analytical expression for the derivative $F'(x)$ nor its derivatives. As such, problems of this class are not amenable to gradient-based optimization techniques. Evaluation of the objective function is also restricted to sampling some (possibly) noisy response at a point x and is typically expensive, hence making grid search and evolutionary algorithms highly impractical (Mockus et al., 1978).

Bayesian optimization (BO) is a sequential approach designed to optimize such black-box functions without an analytical/tractable derivative. The BO algorithm first prescribes a prior Bayesian belief (which is usually modeled using GP (Rasmussen & Williams, 2006) across existing BO literature) for the black-box objective function, i.e. $F(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), k(\mathbf{x},\mathbf{x}'))$ where μ and σ are respectively the prior GP mean and covariance functions. Using this probabilistic model, the BO algorithm then construct an acquisition function $\alpha(\mathbf{x};\mu,k)$ which leverages the expected outcome distribution and its uncertainty (with respect to the current belief) to guide exploration of the input space. Specifically, every BO iteration selects a new input candidate to be queried for response via maximizing this acquisition function. That is,

$$\mathbf{x}_{t+1} \;=\; \arg\max_{\mathbf{x}} \alpha\Big(\mathbf{x};\mu_t,k_t\Big) \tag{3}$$

where μ_t and k_t parameterize the current GP posterior (of iteration t) and the collected response $F(\mathbf{x}_{t+1})$ will be used to update this belief. A popular acquisition function is the upper confidence bound (UCB) (Srinivas et al., 2010):

$$\overline{\alpha}_t(\mathbf{x}) \;=\; \mu_t(\mathbf{x}) \;+\; \beta_t\sqrt{k(\mathbf{x},\mathbf{x})} \tag{4}$$

where t denotes the current iteration of the algorithm and the parameter β_t balances the trade-off between exploitation (of candidates with high expected outcome) and exploration (of candidates with high uncertainty). Interested readers are referred to the BO literature for other choices of acquisition functions (Snoek et al., 2012; Hoang et al., 2018; Wang & Jegelka, 2017; Hernández-Lobato et al., 2014)

3. Automated Kernel Selection

In this section, we describe the technical components and workflow of our proposed method (see Fig. 1). First, section 3.1 reformulates the kernel selection objective as a hyper-parameter optimization task for a kernel synthesis routine, which is amenable to Bayesian Optimization (BO).

Section 3.2 outlines the design of an *open-ended kernel generator* $G(\theta)$, which employs two nested RNNs $G_p(\theta_p)$ and $G_s(\theta_s)$ parameterized by $\theta = (\theta_p, \theta_s)$ to synthesize composite kernels via the generative grammar defined in Eq. (1). The role of $G(\theta_p)$ and $G(\theta_s)$ is to expand the current kernel expression at each step via addition/multiplication. As such, given an instantiation of generative parameter θ, $G(\theta)$ synthesizes an infinitely growing expression via sequentially adding base kernel units to the existing structure.

To avoid generating an infinitely long kernel, $G(\theta)$ is further augmented with a stochastic decision making process, which we refer to as *termination policy* π. The terminal expression (i.e., stopping point of this procedure) is obtained when π decides to stop the generating process (Section 3.2).

The remaining of this section discusses our optimization framework in two sequential steps: (1) a BO variant that optimizes for θ while accounting for the constantly updating

[1] The order of base kernels given by subscript uv is not arbitrary as it reflects the sequential nature of the generator.

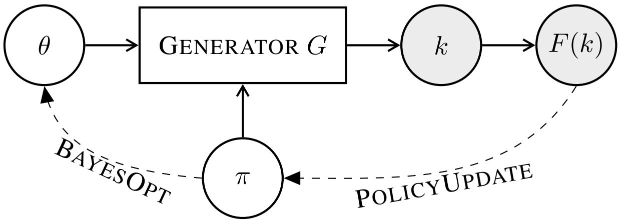


Figure 1. The generic workflow of **DTERGENS**. Given policy π, we employ BO to obtain generative weight candidate θ (Section 3.3). Using the observed generative trajectory, we alternately update the policy distribution (Section 3.4).

belief of π (Section 3.3); and (2) an optimization routine for the posterior $p(\pi \mid \theta)$ given the kernel samples collected from the BO step (Section 3.4).

3.1. Reformulating Kernel Selection

We consider a learning scenario for some kernel-based predictive model ξ with respect to some arbitrary dataset τ and training protocol ϕ (i.e., train-test split, parameter optimization technique, training budget and other model-specific parameters). Let $\Omega = \{\xi, \tau, \phi\}$ denote these configurations and $F_\Omega(k)$ denote the performance measure of the predictive scenario (i.e., predictive accuracy) characterized by Ω for some kernel $k \in \mathcal{K}_\mathcal{C}$. The kernel selection problem can then be formally described as

$$k_* \quad = \quad \arg\max_{k \in \mathcal{K}_\mathcal{C}} F_\Omega(k) \qquad (5)$$

Optimizing over the discrete domain of kernel expressions is, however, intractable as previously motivated (Section 1). To work around this, we model k as the output of a generative process $G(\theta, \pi)$ (whose design will be detailed in Section 3.2) conditioned on parameter θ and termination policy π. Proposition 1 subsequently suggests, under the expressiveness assumption that any valid composite kernel can be generated (given a certain choice of θ and π), the original discrete kernel selection objective can be cast as an equivalent optimization problem on the domain of θ and π.

Proposition 1 (Objective Reformulation)**.** *For some arbitrary composite kernel generative process $G : \Theta \times \Pi \rightarrow \mathcal{K}_\mathcal{C}$, if $\forall\, k \in \mathcal{K}_\mathcal{C},\ \exists\, \theta, \pi \in \Theta \times \Pi$ such that $G(\theta, \pi) = k$, then:*

$$\max_{k \in \mathcal{K}_\mathcal{C}} F_\Omega(k) \quad \equiv \quad \max_{\theta \in \Theta} \max_{\pi \in \Pi} F_\Omega \circ G(\theta, \pi) \qquad (6)$$

Proof. Let $(\theta^*, \pi^*) = \arg\max\ F_\Omega \circ G(\theta, \pi)$. Suppose there exists k such that $F_\Omega(k) > F_\Omega \circ G(\theta^*, \pi^*)$. By the above expressiveness assumption, there also exists (θ', π') such that $k = G(\theta', \pi')$, which implies the contradiction $F_\Omega \circ G(\theta', \pi') > \max_\theta \max_\pi\ F_\Omega \circ G(\theta, \pi)$. □

Remark 1. Essentially, the reformulation above reveals a latent generative process whose parameters are related to the model predictive accuracy in a possibly more well-behaved manner. We hypothesize that, as manifestations of this latent process, kernel expressions only contain truncated information which makes direct optimization on the kernel space difficult. On the other hand, modelling the relationship between predictive accuracy and these generative parameters allow recovery of such latent information which in turn enable efficient search.

To further exploit the intrinsic relationship between π and θ, our framework optimizes each with respect to the other instead of jointly searching for these variables with BO (which does not factor in the designed interaction of π and θ). Explicitly, given fixed π, θ is optimized using a modified BO routine which takes into account this dynamic via the posterior $p(\pi \mid \theta)$ (Section 3.3). Alternately, given new observations sampled from each BO iteration, the policy posterior $p(\pi \mid \theta)$ can be obtained via MLE (Section 3.4). The outline of this workflow is illustrated in Fig. 1 and detailed in Algorithm 1. To lay the groundwork for our algorithmic development, we discuss the design of our kernel generator $G(\theta, \pi)$ next (see Fig. 2).

3.2. Open-ended Composite Kernel Generator

Composite kernel expressions (Section 2.1) naturally manifest as tree structures with multiple secondary branches (i.e., each represents a product of base kernel units) connected via one primary skeleton (i.e., represents the sum of these product terms). To generate such structures, we construct G (Fig. 2) by concatenating two nested recurrent neural networks (RNN) $G_p(\theta_p)$ and $G_s(\theta_s)$ parameterized by $\theta = [\theta_p, \theta_s]$ [2]. Termination of these RNN units is determined by respectively querying policies $\pi = [\pi_p, \pi_s]$ at every generative step. We describe the components of G as follow:

Primary Unit $G_p(\theta_p)$: Given a constant primer input $\mathbf{x}_0 \in \mathbb{R}^{d_p}$ (e.g., $\mathbf{x}_0 = \mathbf{1}$), $G_p(\theta_p)$ generates a sequence of subsequent feedback inputs $\mathbf{x}_1, \mathbf{x}_2 \ldots \subseteq \mathbb{R}^{d_p}$ and corresponding outputs $\mathbf{x}^0_1, \mathbf{x}^0_2 \ldots \subseteq \mathbb{R}^{d_s}$. The latter is passed on as inputs to $G_s(\theta_s)$. At every generative state $\mathbf{x}_t$ and corresponding intermediate kernel expression $k_{\mathbf{x}_t}$ (composing all base units so far), G_p either: (a) with probability $\pi_p(k_{\mathbf{x}_t})$, terminates the generative process and returns a terminal expression; or (b) with probability $1 - \pi_p(k_{\mathbf{x}_t})$, invokes $G_s(\theta_s)$ to generate a new secondary branch.

Secondary Unit $G_s(\theta_s)$: Given some input $\mathbf{x}^0_t$ produced

[2] While LSTM and GRU can be used in place of the RNN cell we employed, both alternatives are more suited to model time series data (due to their forget mechanism) whereas the selection of the base kernel units tend not to exhibit temporal properties.

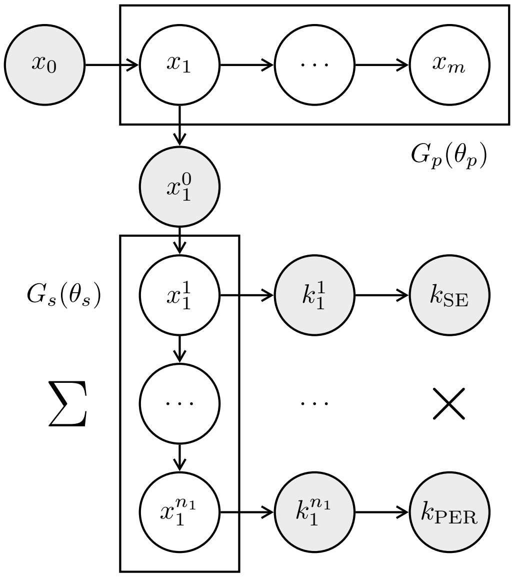


Figure 2. Schematic of the open-ended kernel structure generator with nested recurrent neural network components $G_s(\theta_s)$ and $G_p(\theta_p)$. In each component, states move forward with a probability drawn from the corresponding termination policy. The termination states at each branch ($x_t^{n_t}$) and the primary skeleton (x_m) are stochastically determined by termination policy π.

by $G_p(\theta_p)$, $G_s(\theta_s)$ generates a sequence of feedback inputs $\mathbf{x}_t^1, \mathbf{x}_t^2 \ldots \subset \mathbb{R}^{d_s}$ and the corresponding one-hot encoding of base kernels $k_t^i \in \mathcal{K}_{\mathcal{B}}$ (obtained via softmax activation). At every generative state $\mathbf{x}_t^i$ and corresponding intermediate kernel expression $k_{\mathbf{x}_t^i}$ (composing all base units so far), $G_s(\theta_s)$ either: (a) with probability $\pi_s(k_{\mathbf{x}_t^i})$, terminates the immediate branch and invokes $G_p(\theta_p)$; or (b) with probability $1 - \pi_s(k_{\mathbf{x}_t^i})$, generates a new base kernel unit k_t^{i+1} and appends it to the current product term (i.e., secondary branch).

Termination Policy π: Let $\tau = \{\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_n\}$ be the set of training inputs specified by the black-box model and $\mathbf{K}_\tau \triangleq [k(\mathbf{x}_i, \mathbf{x}_j)]_{ij}$ be the corresponding covariance matrix given some kernel function $k(\mathbf{x}, \mathbf{x}')$. We respectively obtain the primary and secondary termination probability given corresponding policy parameters $\mathbf{w}_p$ and $\mathbf{w}_s$ as below:

$$\begin{aligned} \pi_s(k; \mathbf{w}_s, \tau) &= \sigma\left(\mathbf{w}_s^\top \mathbf{K}_\tau \mathbf{w}_s\right) \\ \pi_p(k; \mathbf{w}_p, \tau) &= \sigma\left(\mathbf{w}_p^\top \mathbf{K}_\tau \mathbf{w}_p\right) \end{aligned} \tag{7}$$

where $\sigma(t) \triangleq 1/(1 + \exp(-t))$ denotes the sigmoid function. These data-driven parameterizations imply that termination decisions are task-specific and not arbitrarily obtained as an universal rule. We note that under these parameterizations, the notations $p(\pi_p|\theta)$ and $p(\pi_s|\theta)$ are implicitly equivalent to $p(\mathbf{w}_p|\theta)$ and $p(\mathbf{w}_s|\theta)$.

On top of being task-specific, termination decisions also need to be sensitive to generative weights θ. This is because one intermediate expression can turn into different terminal expressions under different generative trajectories (dictated by θ). To avoid imposing myopic policies, we adopt the Bayesian approach and model the dynamic between π and θ via the conditional distributions $p(\mathbf{w}_s|\theta_s; \eta_s)$ and $p(\mathbf{w}_p|\theta; \eta_p)$ parameterized by η_s and η_p, respectively – see Section 3.4 for a detailed description of (η_s, η_p).

Algorithm 1 DTERGENS - Kernel Selection

```
Input: $\mathcal{F}_\Omega, n_{\text{BO}}, \gamma = \{n_w, n_\epsilon, n_\ell, \lambda_p, \lambda_s\}$
$\mathcal{G} \leftarrow$ CREATEGENERATOR
$\eta_p, \eta_s \leftarrow$ INITIALIZEPOLICY
for $t = 1$ to $n_{\text{BO}}$ do
    $\theta^t \leftarrow \text{argmax}_\theta\ f_{\text{acq}}(\theta)$
    SAMPLE $\pi_p \sim p(\pi_p \mid \theta_p^t)$
    SAMPLE $\pi_s \sim p(\pi_s \mid \theta_s^t)$
    $\mathbf{k}_t \leftarrow \mathcal{G}(\theta^t, \pi_p, \pi_s)$
    $\mathbf{k}^* \leftarrow \text{argmax}_{\mathbf{k} \subseteq_{\mathcal{K}} \mathbf{k}_t}\ \mathcal{F}_\Omega(\mathbf{k})$
    POLICYUPDATE$(\mathbf{k}^*, \theta^t, \gamma)$
    BAYESIANUPDATE$\big(\theta^t, \mathcal{F}_\Omega(\theta^t), p(\pi_p|\theta_p^t), p(\pi_s|\theta_s^t)\big)$
end for
```

We then leverage these dynamics to construct a dynamic BO algorithm (for optimizing generative weight θ) that accounts for policy updates (Section 3.3). Alternately, we use the generative trajectories obtained from this BO algorithm to update these distributions, thus allowing us to learn the dynamics between θ and π as we optimize for θ (Section 3.4).

Remark 2. To avoid $\mathcal{O}(n^2)$ computation of $\mathbf{k}_\mathcal{D}$ for each candidate $\mathbf{K}$, we approximate the above parameterization with a kernel evaluated on a subset $\tau(S)$ of S randomly sampled data points. Particularly:

$$\begin{aligned} \sigma(\mathbf{w}_s^\top \mathbf{K}_\tau \mathbf{w}_s) &\simeq \sigma(\mathbf{w}'^\top_s \mathbf{K}_{\tau(S)} \mathbf{w}'_s) \\ \sigma(\mathbf{w}_p^\top \mathbf{K}_\tau \mathbf{w}_p) &\simeq \sigma(\mathbf{w}'^\top_p \mathbf{K}_{\tau(S)} \mathbf{w}'_p) \end{aligned} \tag{8}$$

where $\mathbf{w}'_s$ and $\mathbf{w}'_p$ have reduced dimension to match that of $\mathbf{K}_{\tau(S)}$. This allows us to evaluate the termination decisions in $\mathcal{O}(S^2)$ without suffering heavy performance trade-off, as empirically shown in Section 4.

3.3. Generative Parameter Optimization

In this section, we detail our BO algorithm to optimize generative weight θ. Formally, given termination policies $\pi = [\pi_p, \pi_s]$ respectively distributed by posteriors $p(\pi_p \mid \theta)$ and $p(\pi_s \mid \theta)$, our objective can be formally described as:

$$\theta^* = \arg\max_\theta g_\pi(\theta) \equiv \arg\max_\theta F_\Omega \circ G(\theta, \pi) \tag{9}$$

where the predictive accuracy function F_Ω and the generator G have been previously defined (Section 3.1).

We adopt the standard practice of BO (Snoek et al., 2012) and impose a Gaussian Process (GP) (Rasmussen & Williams, 2006) prior on the black-box g_π, i.e., $g_\pi \sim \mathcal{GP}(\mu, k_{\mathrm{BO}})$ where μ and k_{BO} respectively denote its mean and covariance functions. The BO algorithm iteratively obtains the next best candidate θ (to be explored) by maximizing a surrogate acquisition function, whose evaluation depends on the posterior of this GP. The black-box evaluation $g_\pi(\theta)$ is then used to update the GP posterior.

In our context, each candidate θ_t obtained at iteration t is also transcribed into some composite kernel expression $k_t \triangleq G(\theta_t, \pi)$, which will be used to update the policy posterior (Section 3.4). To account for this dynamic landscape of θ (subject to constantly shifting π), we adapt this vanilla BO algorithm by decomposing the GP covariance into three components, one of which characterizes the kernel distance between two instances of generative weights whereas the other captures the shifts in termination policies. Explicitly, given candidates θ_i and θ_j and their corresponding policy posteriors (at time of discovery) $p_i(\mathbf{w}_p|\theta_i), p_i(\mathbf{w}_s|\theta_i), p_j(\mathbf{w}_p|\theta_j)$ and $p_j(\mathbf{w}_s|\theta_j)$, the kernel distance between these candidates is given as:

$$k_{\mathrm{BO}}(\theta_i, \theta_j) \;=\; k_{\pi_p}(p_i, p_j) \cdot k_{\pi_s}(p_i, p_j) \cdot k_\theta(\theta_i, \theta_j) \quad (10)$$

where $\mathbf{k}_\theta$ is the standard Gaussian kernel and, letting D_{JS} denote the Jensen-Shannon divergence between two distributions, we define:

$$\begin{aligned} k_{\pi_p}(p_i, p_j) &\triangleq \mathrm{D}_{\mathrm{JS}}\Big(p_i(\mathbf{w}_p \mid \theta_i) \,\|\, p_j(\mathbf{w}_p \mid \theta_j)\Big) \\ k_{\pi_s}(p_i, p_j) &\triangleq \mathrm{D}_{\mathrm{JS}}\Big(p_i(\mathbf{w}_s \mid \theta_i) \,\|\, p_j(\mathbf{w}_s \mid \theta_j)\Big) \end{aligned} \quad (11)$$

3.4. Learning Policy Posteriors

Following the setup in Section 3.2, we let $\eta_p \triangleq \mathrm{vec}(\zeta_p, \Sigma_p)$, $\eta_s \triangleq \mathrm{vec}(\zeta_s, \Sigma_s)$ and respectively assume Gaussian parameterization for the policy posteriors:

$$\begin{aligned} p(\mathbf{w}_p \mid \theta; \eta_p) &= p(\mathbf{w}_p \mid \theta_p; \eta_p) \triangleq \mathcal{N}(\mathbf{w}_p; \theta_p^\top \zeta_p, \Sigma_p) \\ p(\mathbf{w}_s \mid \theta; \eta_s) &= p(\mathbf{w}_s \mid \theta_s; \eta_s) \triangleq \mathcal{N}(\mathbf{w}_s; \theta_s^\top \zeta_s, \Sigma_s) \end{aligned} \quad (12)$$

This section details an update iteration of η_p and η_s given a new candidate θ and its corresponding generated kernel $k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{m} \prod_{j=1}^{n_i} k_{ij}(\mathbf{x}, \mathbf{x}')$ where $k_{ij} \in \mathcal{K}_{\mathcal{B}}$. An apparent challenge is the lack of optimization objective because it is hard to obtain the optimal stopping point for an infinite generative trajectory.

To sidestep this challenge, we instead adopt the best estimation in hindsight k^*, which is the best performing intermediate kernel expression with respect to the sampled trajectory, i.e., $k^* \triangleq \arg\max_{k' \subseteq_{\mathcal{K}} k} \; F_\Omega(k')$. This serves as an estimation to the optimal expression in the infinite subspace defined by θ. We argue that high-performing kernels are likely to produce covariance matrices (on training set τ) reasonably similar to $\mathbf{K}^*_\tau$ which motivates the following design of our loss functions:

$$\begin{aligned} \ell_p(\pi_p; \eta_p) &\triangleq \mathbb{E}_\pi\Big[\langle G_{\pi_s}(\theta, \pi_p), \mathbf{k}^* \rangle_\tau\Big] \\ \ell_s(\pi_s; \eta_s) &\triangleq \mathbb{E}_\pi\Big[\langle G_{\pi_p}(\theta, \pi_s), \mathbf{k}^* \rangle_\tau\Big] \end{aligned} \quad (13)$$

where G_{π_s} and G_{π_p} respectively denotes fixing the π_s and π_p components of generator G and $\langle \mathbf{K}, \mathbf{K}' \rangle_\tau \triangleq \|\mathbf{K}_\tau - \mathbf{K}'_\tau\|_{\mathrm{Fro}}$ is the Frobenius norm of the difference between the data covariance matrices corresponding to composite kernels k and k'. We write $\ell_p(\pi_p; \eta_p)$ to show the dependence on η_p (and likewise for ℓ_s) since $\pi_p = \sigma(\mathbf{w}_p^\top \mathbf{K}_\tau \mathbf{w}_p)$ and the posterior distribution of $\mathbf{w}_p$ is parameterized by η_p. As generator termination is stochastic (Section 3.2), we approximate these expectations with empirical losses $\bar{\ell}_p$ and $\bar{\ell}_s$ by averaging over n_ℓ independent simulations.

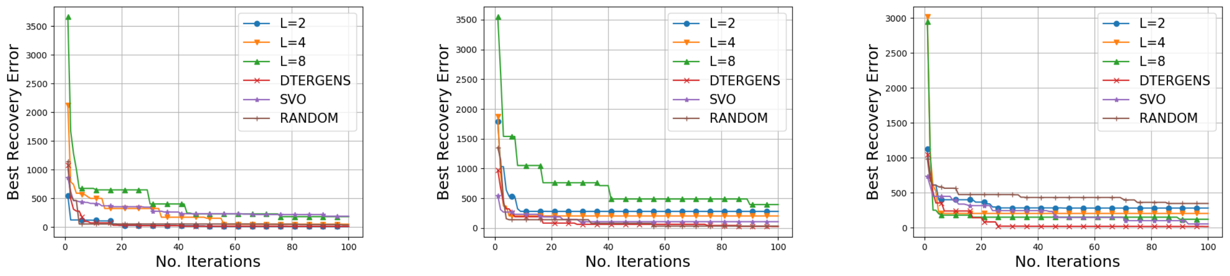
Finally, this allows us to formulate the optimization tasks of η_p and η_s as minimizing the empirical losses $\bar{\ell}_p, \bar{\ell}_s$, expected over the corresponding posterior policy distributions:

$$\begin{aligned} \eta_p^* &= \underset{\eta_p}{\mathrm{argmin}}\; \mathbb{E}_{p(\mathbf{w}_p|\theta;\eta_p)}\left[\bar{\ell}_p(\pi_p; \eta_p)\right] \\ \eta_s^* &= \underset{\eta_s}{\mathrm{argmin}}\; \mathbb{E}_{p(\mathbf{w}_s|\theta;\eta_s)}\left[\bar{\ell}_s(\pi_s; \eta_s)\right] \end{aligned} \quad (14)$$

As standard practice, the optimal parameters η_p and η_s can be obtained by gradient descent update. The expected loss gradients $\nabla_{\eta_p}\bar{\ell}_p$ and $\nabla_{\eta_s}\bar{\ell}_s$, however, do not have analytical forms due to having a generator component and require simulation to compute. We circumvent this problem by employing the randomized gradient technique (Nesterov & Spokoiny, 2017), which estimates derivative at a point by averaging over the instantaneous rates of change with respect to some ϵ-perturbations around it. In particular, we derive our approximate randomized gradient update for the k^{th} entry of η_p as follow:

$$\begin{aligned} \eta_p^k &= \eta_p^k - \gamma_p \nabla_{\eta_p^k} \mathbb{E}\left[\bar{\ell}_p(\pi_p; \eta_p)\right] \\ &\simeq \eta_p^k - \frac{\gamma_p}{n_\epsilon} \sum_{j=1}^{n_\epsilon} \frac{\mathbb{E}\left[\bar{\ell}_p(\pi_p; \eta_p + \epsilon_j \mathbf{e}_k) - \bar{\ell}_p(\pi_p; \eta_p)\right]}{\epsilon_j} \\ &\simeq \eta_p^k - \frac{\gamma_p}{n_w n_\epsilon} \sum_{j=1}^{n_\epsilon} \sum_{i=1}^{n_w} \frac{\bar{\ell}_p(\pi_p^i; \eta_p + \epsilon_j \mathbf{e}_k) - \bar{\ell}_p(\pi_p^i; \eta_p)}{\epsilon_j} \end{aligned}$$

where $\epsilon_j \sim \mathcal{N}(0, 1)$ is a random Gaussian perturbation, $\mathbf{e}_k$ is the one-hot vector with a single 1-entry at the k^{th} position and $\bar{\ell}_p(\pi_p^i; \eta_p + \epsilon_j \mathbf{e}_k)$ denotes the empirical loss obtained with respect to the i^{th} sample of $\mathbf{w}_p$ drawn from the perturbed posterior parameterized by $\eta_p + \epsilon_j \mathbf{e}_k$. Finally, γ_p, n_w and n_ϵ respectively denote the learning rate, number of $\mathbf{w}$ samples and number of ϵ samples per update.

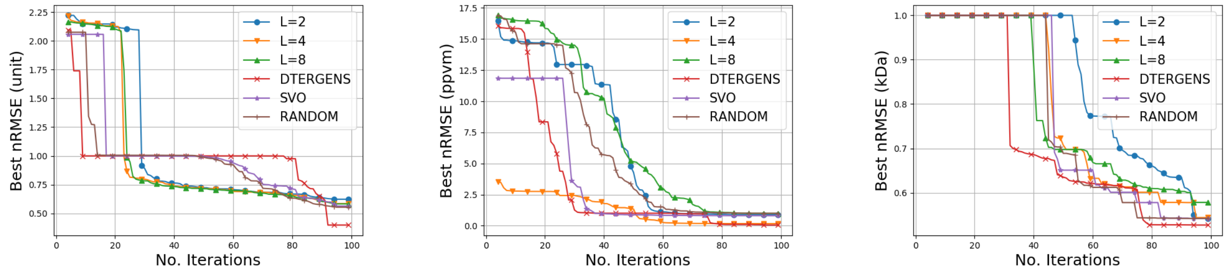


(a)

(b)

(c)

Figure 3. Graphs of best kernel recovery error found over 100 iterations with various kernel selection methods on three synthetic datasets constructed from hand-crafted kernels: (a) $\mathrm{RQ} \times \mathrm{RQ}$; (b) $\mathrm{PER} \times \mathrm{RQ} \times \mathrm{LIN} \times \mathrm{LIN}$; and (c) $\mathrm{LIN} \times \mathrm{RQ} \times \mathrm{LIN} + \mathrm{PER} \times \mathrm{LIN} + \mathrm{RQ} \times \mathrm{SE}$.



(a)

(b)

(c)

Figure 4. Graphs of best nRMSE found over 100 iterations with various kernel selection methods on (a) DIABETES dataset (Efron et al., 2004); (b) MAUNA dataset (Keeling & Whorf, 2004); and (c) PROTEIN dataset (Rana, 2013).

4. Experiments

This section evaluates and reports the empirical performance of our kernel selection framework **DTERGENS** on a synthetic kernel recovery task and kernel selection for regression on three real-world datasets:

1. The DIABETES dataset (Efron et al., 2004) containing 442 diabetes patient records (i.e., inputs) with 10 variables: age, sex, body mass index, average blood pressure and six blood serum measurements. The target output variable is a quantitative measure of disease progression one year after baseline.

2. The MAUNA LOA (Mauna Loa Atmospheric Carbon Dioxide) dataset (Keeling & Whorf, 2004) measuring monthly average carbon-dioxide concentration (in ppvm) from continuous air samples collected over a 42-year period at the Mauna Loa Observatory, Hawaii.

3. The PROTEIN dataset (Rana, 2013) featuring 45730 observations of protein tertiary structures, each consists of 9 physicochemical properties. The target output variable is the size of residue (in kDa).

To demonstrate the efficacy of **DTERGENS**, we compare our performance with the following benchmarks: (a) random search over the space of kernels with max length $L \leq 10$ (baseline); (b) **SVO**: Structure Variationally-Encoded Optimization (Lu et al., 2018) where we train the VAE component using 25000 randomly generated kernel expressions with max length $L \leq 10$ (to show the advantage of generative search); and (c) our own algorithm with no stopping policy and fixing expression length[3] $L = 2, 4, 8$ (to show the advantage of having adaptive termination policies for the generative components).

4.1. Black-box Model Configuration

For all experiments, we demonstrate the performance of our framework on the black-box model Variational DTC Sparse Gaussian Process (vDTC) (Hensman et al., 2013) with the following configurations: (1) 80-10-10 train-test-validation split (i.e., we use the validation fold to compute BO feedback and the test fold to evaluate final performance); (2) 100 randomly selected inducing inputs; (3) kernel hyperparameters are optimized using L-BFGS over 100 iterations.

[3] Termination of secondary component is chosen at random; termination of primary component is guaranteed upon reaching length L; vanilla BO is used to optimize generative weights θ.

4.2. Generator Architecture

For primary unit G_p, we use a RNN cell consisting of: (1) 5-dimensional input layer; (2) two hidden layers with 10 and 20 dimensions; (3) two 5-dimensional output layer, one of which encodes input to secondary unit G_s while the other encodes the primary component's feedback input. We use another RNN cell to parameterize G_s for which the first three layers are similar to the primary unit. This cell has two output layers where one encodes the secondary component's feedback input (5-dimensional) and the other encodes the one-hot representation of a base kernel (4-dimensional).

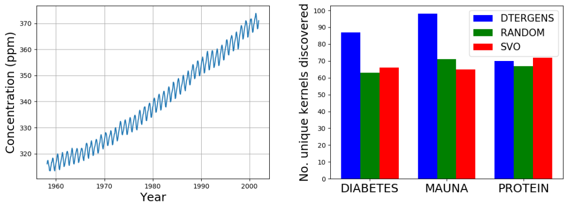
We use reLU activation for all non-output layers, softmax activation for the kernel output layer of G_s and tanh activation for the remaining output layers. Finally, we optimize a total number of 917 RNN parameters using REMBO (Wang et al., 2016) (which has been adapted to account for the dynamic optimization landscape - Section 3.3). Other BO variants such as ADD-GP-UCB (Kandasamy et al., 2015) or DEC-HBO (Hoang et al., 2018) can also be used to handle this high-dimensional setting.

4.3. Synthetic Kernel Recovery

In this experiment, we investigate how well various kernel selection methods recover a known covariance matrix given synthetic data randomly drawn from its corresponding distribution. Unlike most real-world settings where a ground truth kernel is not known and performance evaluation relies on possibly noisy predictive accuracy, this scenario provides a gold standard for kernel selection and allows us to directly measure the success of various contending methods.

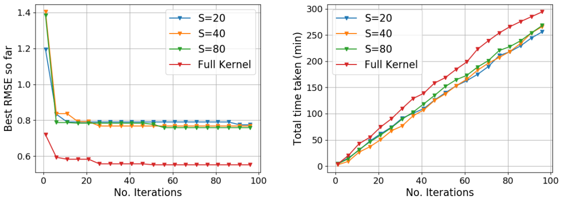
Explicitly, given a randomly chosen kernel k^* (with arbitrarily initialized hyper-parameters) and n input observations $\tau = \{\mathbf{x}_1, \mathbf{x}_2 \ldots \mathbf{x}_n\} \subseteq \mathbb{R}^d$ i.i.d drawn from $\mathcal{N}(\mathbf{0}, \mathbf{I})$, we subsequently draw corresponding output observations $\mathcal{Y} = \{y_1, y_2 \ldots y_n\}$ from $\mathcal{Y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}^*_\tau)$ where $\mathbf{K}^*_\tau$ denotes the data covariance matrix constructed with k^*. We then apply kernel selection for vDTC prediction on this synthetic dataset with the standard black-box configurations (Section 4.1) and measure our recovery error for any selected kernel k by $\ell_{\text{syn}}(k) = \|\mathbf{K}_\tau - \mathbf{K}^*_\tau\|_{\text{Fro}}$. Fig. 3 shows the best recovery error achieved over a span of 100 search iterations with 3 different kernels: (1) $k_{\text{RQ}} \times k_{\text{RQ}}$; (2) $k_{\text{PER}} \times k_{\text{RQ}} \times k_{\text{LIN}} \times k_{\text{LIN}}$; and (3) $k_{\text{LIN}} \times k_{\text{RQ}} \times k_{\text{LIN}} + k_{\text{PER}} \times k_{\text{LIN}} + k_{\text{RQ}} \times k_{\text{SE}}$.

In all experiments, **DTERGENS** consistently achieve the lowest recovery error after 100 iterations compared to other methods. Random search performs competitively when the ground truth kernels are short (i.e., $L = 2, 4$) and relatively easy to hit via randomization. It expectedly performs the worst when the ground truth kernel is longer (i.e., $L = 7$). We also observe that fixing L maintains a competitive performance for **DTERGENS** only when L matches the length of the ground truth kernel and is outperformed by other methods otherwise. This implies, without domain expert knowledge, discovery of constant length kernels is generally a bad strategy. Lastly, we observe that **SVO** is most significantly outperformed by **DTERGENS** in the first experiment. We reason that this is because the trained VAE could be biased to produce longer kernels, hence it is more difficult **SVO** to find a latent embedding that decodes to a length 2 kernel. **DTERGENS** does not incur this problem because its termination policy is also learned as it collects information about the embedding space.



(a) (b)

Figure 5. (a) The linear-periodic trend of the MAUNA dataset; and (b) No. unique kernels discovered by different methods (i.e., **DTERGENS**, **SVO** and random search) on three datasets.



(a) (b)

Figure 6. (a) nRMSE vs. number of sampled data points for policy update; and (b) total time taken vs. number of sampled data points for policy update for **DTERGENS** on DIABETES dataset.

4.4. Kernel Selection for Regression

We investigate the performance of kernel selection for regression tasks using vDTC (Hensman et al., 2013) on three real-world datasets DIABETES (Efron et al., 2004), MAUNA (Keeling & Whorf, 2004) and PROTEIN (Rana, 2013). In all experiments, we measure performance by computing the root-mean-square-error (RMSE) of predictions with a discovered kernel $\mathbf{k}$ on the hold out test set, normalized by the RMSE obtained using the standard square exponential kernel (i.e., SE). Explicitly, our normalized RMSE (nRMSE) metric is given by $\text{nRMSE}(k) = \|y(k) - y\|_2 / \|y(k_{\text{SE}}) - y\|_2$ where $y(k)$ denotes the prediction made by vDTC as we set its kernel to be k and y denotes the ground truth test output.

Fig. 4 shows the comparative performance between **DTERGENS** and the competing methods. Across all datasets, **DTERGENS** consistently obtains the best performing kernel expression. On PROTEIN dataset, **DTERGENS** also shows the fastest convergence among all competing methods. On MAUNA dataset, **DTERGENS** performs competitively with fixing $L = 4$ and both variants of **DTERGENS** outperform **SVO**. More interestingly, the best kernel found for the MAUNA dataset is $k_{\mathrm{LIN}} \times k_{\mathrm{PER}} \times k_{\mathrm{PER}} + k_{\mathrm{RQ}} \times k_{\mathrm{PER}}$ which seems to reflect the linearly increasing periodic nature of the data (Fig. 5a).

Fig. 5b demonstrates the expressiveness of three kernel selection methods (**DTERGENS**, **SVO** and random search) measured by the number of unique kernels found over 100 iterations. As expected, random search consistently produces the same amount of unique expressions across all experiments. While **SVO** discovers approximately the same amount of unique kernels as does random search on all three datasets, it tends to outperform random search as its discovery is guided. Finally, we observe that **DTERGENS** consistently discovers more unique kernels and also outperforms the other methods. This finding asserts our earlier intuition on how adding expressiveness to the embedding method also helps to improve search efficiency (Section 1).

Lastly, Fig 6 shows the performance vs. run time trade-off for different numbers S of sampled data points for policy evaluation (see Remark 2, Section 4.1) as observed on the DIABETES dataset. As compared to policy evaluation with kernels evaluated on full dataset, using approximate kernels evaluated on a random subset of data saves 20% total time at a reasonable cost of less than 0.2 nRMSE loss. For larger datasets (such as PROTEIN), the saving is more pronounced as the cost of evaluating kernels becomes the bottleneck, however we do not show the results here because it takes too long to achieve a full run for PROTEIN dataset with no approximation.

5. Conclusion

We tackle the composite kernel selection problem for an arbitrary black-box model. The proposed **DTERGENS** algorithm reformulates the kernel search problem as a parameter optimization task for a recursive kernel generator equipped with optimal stopping mechanism. Operating on this well-behaved space of generative weights allows efficient traversing of the candidate space with Bayesian Optimization. **DTERGENS** is able to explore arbitrarily complex kernels and avoid putting restrictive assumptions on the candidate space and performs competitively against other state-of-the-art methods on many real-world datasets. Last but not least, although we do not derive a convergence analysis similar to well-known BO regret bounds due to the difficulty of quantifying the embedding reconstruction loss with a recursive kernel generator, we would like to address this issue in a future work.

6. Acknowledgement

This work was partially supported in part by the Gordon and Betty Moore Foundation's Data-Driven Discovery Initiative through Grant GBMF4554, by the US National Science Foundation (DBI-1937540), by the US National Institutes of Health (R01GM122935), and by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program.

References

Akchurina, N. Multiagent reinforcement learning: Algorithm converging to Nash equilibrium in general-sum discounted stochastic games. In *Proc. AAMAS*, pp. 725–732, 2009.

Allamraju, R. and Chowdhary, G. Communication efficient decentralized Gaussian process fusion for multi-UAS path planning. In *Proc. ACC*, pp. 4442–4447, 2017.

Bergstra, J., Yamins, D., and Cox, D. D. Making a science of model search: Hyperparameter optimization and hundreds of dimensions for vision architectures. In *Proc. ICML*, pp. 115–123, 2013.

Cao, N., Low, K. H., and Dolan, J. M. Multi-robot informative path planning for active sensing of environmental phenomena: A tale of two algorithms. In *Proc. AAMAS*, pp. 7–14, 2013.

Castro, R. *Active Learning and Adaptive Sampling for Non-Parametric Inference*. PhD thesis, Rice University, Houston, Texas, August 2007.

Castro, R., Willett, R., and Nowak, R. Faster rate in regression via active learning. In *Proc. NIPS*, pp. 179–186, 2005.

Chen, J., Cao, N., Low, K. H., Ouyang, R., Tan, C. K.-Y., and Jaillet, P. Parallel Gaussian process regression with low-rank covariance matrix approximations. In *Proc. UAI*, pp. 152–161, 2013.

Cortes, C. and Vapnik, V. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

da Motta Salles Barreto, A., Precup, D., and Pineau, J. Practical kernel-based reinforcement learning. *CoRR*, abs/1407.5358, 2014. URL http://arxiv.org/abs/1407.5358.

Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., and Zoubin, G. Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings* *of the 30th International Conference on Machine Learning*, volume 28, pp. 1166–1174, 2013.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression, Annals of Statistics, 407-499. https://www4.stat.ncsu.edu/ boos/var.select/diabetes.html, 2004.

Elsken, T., Metzen, J. H., and Hutter, F. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20:55:1–55:21, 2018.

Engel, Y., Mannor, S., and Meir, R. Reinforcement learning with Gaussian processes. In *International Conference on Machine Learning*, pp. 201–208, 2013.

Gal, Y. and Turner, R. Improving the Gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *Proc. ICML*, pp. 655–664, 2015.

Gal, Y., van der Wilk, M., and Rasmussen, C. Distributed variational inference in sparse Gaussian process regression and latent variable models. In *Proc. NIPS*, pp. 3257–3265, 2014.

Hearst, M. A. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, July 1998. ISSN 1541-1672. doi: 10.1109/5254.708428. URL http://dx.doi.org/10.1109/5254.708428.

Hensman, J., Fusi, N., and Lawrence, N. D. Gaussian processes for big data. In *Proc. UAI*, pp. 282–290, 2013.

Hernández-Lobato, J. M., Hoffman, M. W., and Ghahramani, Z. Predictive entropy search for efficient global optimization of black-box functions. In *Proc. NIPS*, pp. 918–926, 2014.

Hoang, M., Hoang, N., Low, B. K. H., and Kingsford, C. Collective model fusion for multiple black-box experts. In *International Conference on Machine Learning*, pp. 2742–2750, 2019a.

Hoang, Q. M., Hoang, T. N., and Low, K. H. A generalized stochastic variational Bayesian hyperparameter learning framework for sparse spectrum Gaussian process regression. In *Proc. AAAI*, pp. 2007–2014, 2017.

Hoang, T. N. and Low, K. H. A general framework for interacting Bayes-optimally with self-interested agents using arbitrary parametric model and model prior. In *Proc. IJCAI*, pp. 1394–1400, 2013.

Hoang, T. N., Low, K. H., Jaillet, P., and Kankanhalli, M. Nonmyopic ϵ-Bayes-optimal active learning of Gaussian processes. In *Proc. ICML*, pp. 739–747, 2014a.

Hoang, T. N., Low, K. H., Jaillet, P., and Kankanhalli, M. S. Active learning is planning: Non-myopic ϵ-Bayes-optimal active learning of Gaussian processes. In *Proc. ECML-PKDD Nectar Track*, pp. 494–498, 2014b.

Hoang, T. N., Hoang, Q. M., and Low, K. H. A unifying framework of anytime sparse Gaussian process regression models with stochastic variational inference for big data. In *Proc. ICML*, pp. 569–578, 2015.

Hoang, T. N., Hoang, Q. M., and Low, K. H. A distributed variational inference framework for unifying parallel sparse Gaussian process regression models. In *Proc. ICML*, pp. 382–391, 2016.

Hoang, T. N., Hoang, Q. M., Ouyang, R., and Low, K. H. Decentralized high-dimensional bayesian optimization with factor graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3231–3238, 2018.

Hoang, T. N., Hoang, Q. M., Low, K. H., and How, J. Collective online learning of gaussian processes in massive multi-agent systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 7850–7857, 2019b.

Idrissi, J., Amine, M., Hassan, R., Youssef, G., and Mohamed, E. Genetic algorithm for neural network architecture optimization. In *International Conference on Logistics Operations Management*, pp. 1–4, 05 2016. doi: 10.1109/GOL.2016.7731699.

Kandasamy, K., Schneider, J., and Póczos, B. High dimensional Bayesian optimization and bandits via additive models. In *Proc. ICML*, pp. 295–304, 2015.

Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., and Xing, E. Neural architecture search with bayesian optimisation and optimal transport. In *Conference on Neural Information Processing Systems (NeuRIPS)*, 02 2018.

Keeling, C. D. and Whorf, T. P. Atmospheric carbon dioxide concentrations derived from flask air samples at sites in the SiO network. https://www.openml.org/d/41187, 2004.

Kingma, D. and Welling, M. Auto-Encoding Variational Bayes. In *Proc. ICLR*, 2013.

Krause, A. and Guestrin, C. Nonmyopic active learning of Gaussian processes: An exploration-exploitation approach. In *Proc. ICML*, pp. 449–456, 2007.

Low, K. H., Chen, J., Dolan, J. M., Chien, S., and Thompson, D. R. Decentralized active robotic exploration and mapping for probabilistic field classification in environmental sensing. In *Proc. AAMAS*, pp. 105–112, 2012.

Low, K. H., Yu, J., Chen, J., and Jaillet, P. Parallel Gaussian process regression for big data: Low-rank representation meets Markov approximation. In *Proc. AAAI*, 2015.

Lu, X., Gonzalez, J., Dai, Z., and Lawrence, N. Structured variationally auto-encoded optimization. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3267–3275, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/lu18c.html.

Malkomes, G., Schaff, C., and Garnett, R. Bayesian optimization for automated model selection. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 2900–2908, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9. URL http://dl.acm.org/citation.cfm?id=3157382.3157422.

Mockus, J., Tiešis, V., and Žilinskas, A. The application of Bayesian methods for seeking the extremum. In Dixon, L. C. W. and Szegŏ, G. P. (eds.), *Towards Global Optimization 2*, pp. 117–128. North-Holland Publishing Company, 1978.

Nesterov, Y. and Spokoiny, V. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017.

Poupart, P. and Vlassis, N. Model-based bayesian reinforcement learning in partially observable domains. In *Proc. ISAIM*, 2008.

Poupart, P., Vlassis, N., Hoey, J., and Regan, K. An analytic solution to discrete Bayesian reinforcement learning. In *Proc. ICML*, pp. 697–704, 2006.

Quiñonero-Candela, J. and Rasmussen, C. E. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

Quiñonero-Candela, J., Rasmussen, C. E., and Williams, C. K. I. Approximation methods for Gaussian process regression. *Large-Scale Kernel Machines*, pp. 203–223, 2007.

Rana, P. S. Physicochemical Properties of Protein Tertiary Structure Data Set, 2013. URL http://archive.ics.uci.edu/ml/datasets/.

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

Snoek, J., Hugo, L., and Adams, R. P. Practical Bayesian optimization of machine learning algorithms. In *Proc. NIPS*, pp. 2960–2968, 2012.

Srinivas, N., Krause, A., Kakade, S., and Seeger, M. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, pp. 1015–1022, 2010.

Wang, Z. and Jegelka, S. Max-value entropy search for efficient Bayesian optimization. In *Proc. ICML*, pp. 3627–3635, 2017.

Wang, Z., Zoghi, M., Hutter, F., Matheson, D., and de Freitas, N. Bayesian optimization in a billion dimensions via random embeddings. *JAIR*, 55:361–387, 2016.

Wang, Z., Li, C., Jegelka, S., and Kohli, P. Batched high-dimensional Bayesian optimization via structural kernel learning. In *Proc. ICML*, pp. 3656–3664, 2017.

Yu, H., Nghia, T., Low, B. K. H., and Jaillet, P. Stochastic variational inference for bayesian sparse gaussian process regression. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2019.

Ziebart, B. D., Maas, A., Bagnell, J. A., and Dey, A. K. Maximum entropy reinforcement learning. In *Proc. AAAI*, pp. 1433–1438, 2010.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. URL http://arxiv.org/abs/1611.01578.