KERNEL-BASED LIFELONG POLICY GRADIENT REINFORCEMENT LEARNING

Rami Mowakeaa[†] *Seung-Jun Kim*[†] *Darren K. Emge*[⋆]

[†] Dept. of Computer Science and Electrical Engineering
University of Maryland, Baltimore County
Baltimore, MD, USA
{ramo1, sjkim}@umbc.edu
[⋆] Combat Capabilities Development Command
Chemical Biological Center RDCB-DRC-P
Gunpowder, MD, USA
darren.k.emge.civ@mail.mil

ABSTRACT

Policy gradient methods have been widely used in reinforcement learning (RL), especially thanks to their facility to handle continuous state spaces, strong convergence guarantees, and low-complexity updates. Training of the methods for individual tasks, however, can still be taxing in terms of the learning speed and the sample trajectory collection. Lifelong learning aims to exploit the intrinsic structure shared among a suite of RL tasks, akin to multitask learning, but in an efficient online fashion. In this work, we propose a lifelong RL algorithm based on the kernel method to leverage nonlinear features of the data based on a popular union-of-subspace model. Experimental results on a set of simple related tasks verify the advantage of the proposed strategy, compared to the single-task and the parametric counterparts.

Index Terms— Reinforcement learning, lifelong learning, kernel methods, policy gradients, dictionary learning.

1. INTRODUCTION

Policy gradient (PG) methods have gained much attention due to their suitability to reinforcement learning (RL) applications with continuous domains such as in robotics [1]. More recently, nonparametric PG methods have been proposed to capitalize on the vast flexibility of the more general class of functions living in a reproducing kernel Hilbert space (RKHS) [2, 3]. Despite these advantages, and strong convergence guarantees [4], PG methods may require extensive interactions with the environment by the learning agent, resulting in slow convergence. Beginning with the original REINFORCE [5], many approaches have been put forth to achieve faster and more robust learning such as the actor-critic method [6], functional approximation of the value function [7], the use of natural gradients [4], and the expectation maximization approach [8].

While these approaches target the performance of RL applied to a single task, considering multiple related tasks jointly or in sequence provides additional opportunities for improving not only in the rate of learning, but in the quality of the resulting learners as well. Transfer learning has been applied to RL by leveraging experience gained in past tasks towards new yet similar tasks [9, 10, 11]. Multitask learning, in contrast, has been applied to RL, allowing joint analysis of a set of related tasks in pursuit of greater performance across them. Hierarchical Bayesian frameworks have been proposed to analyze a collection of tasks from a fixed, yet unspecified distribution, where the shared structures in the dynamics and the reward/value functions are captured [12, 13]. In [14], multitask RL is cast as a variational inference problem and a distributed solver is developed with quadratic convergence. Other methods target multitask RL in partially observable environments [15], and tackle inverse RL setups [16].

Lifelong learning (LL) extends the goal of multitask learning to efficient online solutions tailored to streaming data and tasks, and has been applied recently to supervised learning problems [17, 18, 19, 20]. LL seeks to leverage experience gained from past tasks towards new tasks, and incorporate knowledge obtained from new tasks toward old tasks, while processing the tasks sequentially, in an efficient online manner. LL has been applied to PG RL in the parametric case [21], where a union-of-subspace model was employed to extract the shared knowledge among sequential tasks.

In this work, a novel kernel-based lifelong PG RL algorithm is developed, which targets the RL tasks, where the policies are sought in a flexible RKHS. We utilize a union-of-subspace model in the RKHS to capture the shared skills across tasks [22], and apply an online kernel dictionary learning method to deal with the tasks arriving sequentially to the learner [23]. Functional stochastic gradient descent is employed to pursue a low-cost solution to minimize an expected risk functional. To alleviate the rapid increase in computational and memory complexity associated with any kernel method, a sparsification approach is also adopted to attain parsimonious representations of the shared skill library in function space. Preliminary numerical tests based on a collection of spring-mass-damper tasks verify the advantage of the proposed kernel LL strategy, over the parametric LL counterpart, as well as the parametric/nonparametric single-task learning approaches.

The rest of this paper is organized as follows. In Sec. 2, the problem formulation is put forth. In Sec. 3, the proposed algorithm is derived. The results of the numerical experiments are presented in Sec. 4. Conclusions are provided in Sec. 5.

This work was supported in part by the MSI STEM Research & Development Consortium (MSRDC)/U.S. Army under Grant W911SR-14-2-0001, and in part by the National Science Foundation under Grant 1547347.

2. PROBLEM FORMULATION

2.1. Policy Gradient Reinforcement Learning

The RL problem can be formulated as a sequential decision making problem modeled using a Markov decision process (MDP) defined by the tuple $(\mathcal{X}, \mathcal{A}, p, r, H)$, where $\mathcal{X} \subset \mathbb{R}^p$ is the bounded set of possible states, $\mathcal{A} \subset \mathbb{R}$ is the set of possible actions, $p : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}^+$ is the state transition probability density function describing the environment dynamics, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the instantaneous reward function, and $H \in \mathbb{N}$ is the length of the episode (trajectory). At a given time step h, an agent in state $\mathbf{x}_h$ selects an action $a_h \in \mathcal{A}$ according to a policy distribution $\pi_{\boldsymbol{\theta}} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^+$ parameterized by $\boldsymbol{\theta}$. At the next time step $h+1$, the agent is transitioned to state $\mathbf{x}_{h+1}$ according to p and receives a reward $r_{h+1} = r(\mathbf{x}_{h+1}, a_h)$. The sequence of $\{\mathbf{x}_h\}_{h=0}^{H}$ and $\{a_h\}_{h=0}^{H-1}$ form a trajectory $\boldsymbol{\tau}$ over a finite horizon H. The goal of the RL agent is to learn a policy distribution $\pi_{\boldsymbol{\theta}}$ that maximizes the expected reward given by

$$\tilde{\mathcal{J}}(\boldsymbol{\theta}) = \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \mathcal{R}(\boldsymbol{\tau}) d\boldsymbol{\tau} \tag{1}$$

where $\mathbb{T}$ is the set of trajectories, the trajectory reward $\mathcal{R}(\boldsymbol{\tau}) := \frac{1}{H}\sum_{h=1}^{H} r_h$, and the trajectory density $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$ is given by

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = p_0(\mathbf{x}_0) \prod_{h=0}^{H-1} p(\mathbf{x}_{h+1}|\mathbf{x}_h, a_h) \pi_{\boldsymbol{\theta}}(a_h|\mathbf{x}_h) \tag{2}$$

with p_0 being the initial state distribution.

PG algorithms are often derived through a lower bound on (1) using Jensen's inequality as [24, 8]

$$\log \tilde{\mathcal{J}}(\tilde{\boldsymbol{\theta}}) = \log \int_{\mathbb{T}} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau}) \mathcal{R}(\boldsymbol{\tau}) d\boldsymbol{\tau} \tag{3}$$

$$\geq \int_{\mathbb{T}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \mathcal{R}(\boldsymbol{\tau}) \log \frac{p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} d\boldsymbol{\tau} + \text{const.} \tag{4}$$

$$\propto -\mathcal{D}(p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\mathcal{R}(\boldsymbol{\tau}) || p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})) := -\mathcal{J}_{\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}\right) \tag{5}$$

where $\mathcal{D}(\cdot||\cdot)$ is the Kullback-Leibler (KL) divergence, and the terms not dependent on variable $\tilde{\boldsymbol{\theta}}$ are neglected. The lower bound in (5) shows that the expected reward in (1) can be maximized, given a baseline policy $\pi_{\boldsymbol{\theta}}$, by minimizing the KL divergence between the reward-weighted trajectory distribution under policy $\pi_{\boldsymbol{\theta}}$ and the trajectory distribution under the desired policy $\pi_{\tilde{\boldsymbol{\theta}}}$. Then, $\pi_{\boldsymbol{\theta}}$ is substituted with $\pi_{\tilde{\boldsymbol{\theta}}}$, new trajectories are sampled, and the process is repeated until convergence.

In order to seek nonparametric policies via kernel methods, we postulate a RKHS $\mathcal{H}$, determined by a positive semidefinite kernel function $\kappa(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathcal{A}$, which is given through a feature transform $\phi : \mathcal{X} \to \mathcal{H}$ by $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle$, and a norm $\|\phi(\mathbf{x})\|_{\mathcal{H}} = \sqrt{\langle \phi(\mathbf{x}), \phi(\mathbf{x}) \rangle}$.

2.2. Multitask Policy Gradient Reinforcement Learning

Consider a stream of RL tasks defined by MDPs $(\mathcal{X}, \mathcal{A}, p^{(t)}, r^{(t)}, H^{(t)})$, $t \in \{1, \ldots, T\}$, which are defined on shared state and action spaces but can differ in their transition densities, reward functions, and horizon lengths. The purpose of multitask policy gradient reinforcement learning is to estimate a collection of policies $\{\pi_{\boldsymbol{\theta}^{(t)}}\}_{t=1}^{T}$ that maximize the per-task expected reward of (1), leveraging the structure shared among the tasks. In this work, we capture the shared structure of the tasks using the union-of-subspace model of policy parameterization as in

$$\boldsymbol{\theta}_{\text{st}}^{(t)} \approx \mathbf{L}\mathbf{s}^{(t)} \tag{6}$$

where $\boldsymbol{\theta}_{\text{st}}^{(t)} \in \mathcal{H}$ is the optimal single-task learning (STL) solution when task t is considered alone, $\mathbf{L} := [\boldsymbol{\ell}_1, \ldots, \boldsymbol{\ell}_K]$ is a shared skill library of K atoms with $\boldsymbol{\ell}_k \in \mathcal{H}$, and $\mathbf{s}^{(t)} \in \mathbb{R}^K$ is a sparse code that describes the task parameterization through the shared library.

By combining the cost functions of all T tasks together with the model in (6), the multitask RL problem can be formulated as

$$\min_{\mathbf{L}} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \mathcal{J}_{\boldsymbol{\theta}^{(t)}}\left(\mathbf{L}\mathbf{s}^{(t)}\right) + \mu \left\|\mathbf{s}^{(t)}\right\|_1 \right\} + \lambda \|\mathbf{L}\|_{\mathcal{H}}^2 \tag{7}$$

where $||\cdot||_1$ is the ℓ_1-norm promoting sparsity in $\{\mathbf{s}^{(t)}\}$, $\|\mathbf{L}\|_{\mathcal{H}}^2 := \sum_{k=1}^{K} \|\boldsymbol{\ell}_k\|_{\mathcal{H}}^2$ controls the complexity of $\mathbf{L}$, and $\boldsymbol{\theta}^{(t)}$ is the baseline policy from which $\boldsymbol{\theta}_{\text{st}}^{(t)}$ is computed. Positive parameters μ and λ adjust the strengths of the regularizations.

In practice, the per-task cost $\mathcal{J}_{\boldsymbol{\theta}^{(t)}}$ is evaluated using the sample trajectories from the corresponding task. Thus, it can be seen that the objective in (7) depends on the samples of the entire set of tasks. To alleviate this issue, a strategy similar to those used in [21, 20] is employed here. Specifically, by performing a second-order Taylor series expansion of $\mathcal{J}_{\boldsymbol{\theta}^{(t)}}(\tilde{\boldsymbol{\theta}})$ around the single task optimum $\boldsymbol{\theta}_{\text{st}}^{(t)}$, and dropping some constants, the multitask learning formulation can be written as

$$\min_{\mathbf{L}} \frac{1}{T} \sum_{t=1}^{T} \min_{\mathbf{s}^{(t)}} \left\{ \left\|\mathbf{L}\mathbf{s}^{(t)} - \boldsymbol{\theta}_{\text{st}}^{(t)}\right\|_{\mathbf{H}_{\text{st}}^{(t)}}^2 + \mu \left\|\mathbf{s}^{(t)}\right\|_1 \right\} + \lambda \|\mathbf{L}\|_{\mathcal{H}}^2 \tag{8}$$

where $\|\mathbf{v}\|_{\mathbf{U}}^2 := \mathbf{v}^\top \mathbf{U} \mathbf{v}$, and

$$\mathbf{H}_{\text{st}}^{(t)} := \nabla_{\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\theta}}}^2 \mathcal{J}_{\boldsymbol{\theta}^{(t)}}\left(\tilde{\boldsymbol{\theta}}\right)\Big|_{\tilde{\boldsymbol{\theta}}=\boldsymbol{\theta}_{\text{st}}^{(t)}} \tag{9}$$

$$= -\mathbb{E}\left[\mathcal{R}(\boldsymbol{\tau}) \sum_{h=0}^{H-1} \nabla_{\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\theta}}}^2 \log \pi_{\tilde{\boldsymbol{\theta}}}(a_h|\mathbf{x}_h)\right]\Bigg|_{\tilde{\boldsymbol{\theta}}=\boldsymbol{\theta}_{\text{st}}^{(t)}} \tag{10}$$

with the expectation taken with respect to $\boldsymbol{\tau} \sim p_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{\tau})$ is the Hessian evaluated at the single-task optimum. We assume that the policies $\pi_{\tilde{\boldsymbol{\theta}}}$ are twice differentiable. For example, in much of the policy gradient literature, the policies are assumed to be Gaussian as in

$$\pi_{\tilde{\boldsymbol{\theta}}}(a_h|\mathbf{x}_h) = C \exp\left\{-\frac{1}{2\sigma^2}\left(a_h - \left\langle \tilde{\boldsymbol{\theta}}, \phi(\mathbf{x}_h)\right\rangle\right)^2\right\} \tag{11}$$

where C is an appropriate normalization constant and σ^2 is the variance of the distribution, which controls the amount of exploration conducted by the policy. Then, the gradient can be expressed as

$$\nabla_{\tilde{\boldsymbol{\theta}}} \mathcal{J}_{\boldsymbol{\theta}^{(t)}}\left(\tilde{\boldsymbol{\theta}}\right) = -\mathbb{E}\left[\mathcal{R}(\boldsymbol{\tau}) \sum_{h=0}^{H-1} \frac{1}{\sigma^2}\left(a_h - \left\langle \tilde{\boldsymbol{\theta}}, \phi(\mathbf{x}_h)\right\rangle\right)\phi(\mathbf{x}_h)\right] \tag{12}$$

and the Hessian as

$$\nabla_{\tilde{\boldsymbol{\theta}},\tilde{\boldsymbol{\theta}}}^2 \mathcal{J}_{\boldsymbol{\theta}^{(t)}}\left(\tilde{\boldsymbol{\theta}}\right) = \mathbb{E}\left[\mathcal{R}(\boldsymbol{\tau}) \sum_{h=0}^{H-1} \frac{1}{\sigma^2} \phi(\mathbf{x}_h)\phi^\top(\mathbf{x}_h)\right]. \tag{13}$$

3. LIFELONG PG RL IN RKHS

3.1. Library Update in Function Space

In the LL setup, new tasks arrive continuously in a streaming fashion. Collecting a batch of tasks and solving for all the tasks jointly through (8) may incur significant delay and computational burden. A viable alternative is to derive an online learning algorithm, which enjoys low-cost updates as well. Specifically, assuming that the tasks arrive independently and with identical distributions, and invoking the law of large numbers with $T \to \infty$, (8) can be re-written as

$$\min_{\mathbf{L}} \mathbb{E}\left[\min_{\mathbf{s}} \|\mathbf{L}\mathbf{s} - \boldsymbol{\theta}_{\text{st}}\|^2_{\mathbf{H}_{\text{st}}} + \mu \|\mathbf{s}\|_1\right] + \lambda \|\mathbf{L}\|^2_{\mathcal{H}}. \quad (14)$$

We seek solutions to (14) using the stochastic gradient descent (SGD) method in function space [25, 20]. To simplify the presentation, let us assume that task t arrives at iteration t. In order to compute the instantaneous gradient, consider first the solution to the inner minimization problem, i.e., the sparse coding problem, for a particular task t, given by

$$\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}} \left\|\boldsymbol{\theta}_{\text{st}}^{(t)} - \mathbf{L}^{(t-1)}\mathbf{s}\right\|^2_{\mathbf{H}_{\text{st}}^{(t)}} + \mu\|\mathbf{s}\|_1 \quad (15)$$

where $\mathbf{L}^{(t-1)}$ is the library from iteration $t-1$. Then, the instantaneous gradient of the objective of (14) at iteration t is given by

$$2\mathbf{H}_{\text{st}}^{(t)}\left(\mathbf{L}^{(t-1)}\mathbf{s}^{(t)} - \boldsymbol{\theta}_{\text{st}}^{(t)}\right)\mathbf{s}^{(t)\top} + 2\lambda\mathbf{L}^{(t-1)} \quad (16)$$

and the library update using SGD at time t is given by

$$\mathbf{L}^{(t)} = (1-\lambda\eta)\mathbf{L}^{(t-1)} - \eta\mathbf{H}_{\text{st}}^{(t)}\left(\mathbf{L}^{(t-1)}\mathbf{s}^{(t)} - \boldsymbol{\theta}_{\text{st}}^{(t)}\right)\mathbf{s}^{(t)\top} \quad (17)$$

where a factor of 2 in the gradient has been absorbed into the step size parameter η. Upon convergence, we refer to the product $\hat{\boldsymbol{\theta}}^{(t)} = \mathbf{L}\mathbf{s}^{(t)}$ as the LL policy for task t.

3.2. Library Update via Dual Representation

Thanks to the Representer Theorem, the STL solution to each task t can be described in its dual form, using a linear combination of the lifted features of the state vectors in the sample trajectories [26]. Denote by N_t the number of trajectories sampled for task t. Let $\mathbf{X}_n^{(t)} := [\mathbf{x}_{n,1}^{(t)}, \ldots, \mathbf{x}_{n,H^{(t)}}^{(t)}] \in \mathbb{R}^{p\times H^{(t)}}$ be the collection of state vectors from the n-th trajectory. Define also $\mathbf{X}^{(t)} := [\mathbf{X}_1^{(t)}, \ldots, \mathbf{X}_{N_t}^{(t)}] \in \mathbb{R}^{p\times H^{(t)}N_t}$. Likewise, the collection of the lifted features is defined as $\boldsymbol{\Phi}(\mathbf{X}_n^{(t)}) := [\boldsymbol{\phi}(\mathbf{x}_{n,1}^{(t)}), \ldots, \boldsymbol{\phi}(\mathbf{x}_{n,H^{(t)}}^{(t)})]$ and $\boldsymbol{\Phi}(\mathbf{X}^{(t)}) := [\boldsymbol{\Phi}(\mathbf{X}_1^{(t)}), \ldots, \boldsymbol{\Phi}(\mathbf{X}_{N_t}^{(t)})]$. Now, upon introducing a coefficient vector $\mathbf{w}_{\text{st}}^{(t)} \in \mathbb{R}^{H^{(t)}N_t}$, the STL optimum can be expressed as

$$\boldsymbol{\theta}_{\text{st}}^{(t)} = \boldsymbol{\Phi}(\mathbf{X}^{(t)})\mathbf{w}_{\text{st}}^{(t)}. \quad (18)$$

Assuming that a Gaussian policy is used for $\pi_{\boldsymbol{\theta}^{(t)}}$, let $R_n^{(t)} := \mathcal{R}(\boldsymbol{\tau}_n^{(t)})$ be the reward of the n-th trajectory $\boldsymbol{\tau}_n^{(t)}$, $\mathbf{1}_{H^{(t)}}$ be a vector of 1's of length $H^{(t)}$ and

$$\boldsymbol{\mathcal{J}}_t'' := \frac{1}{\sigma^2}\text{diag}\left\{R_1^{(t)}\mathbf{1}_{H^{(t)}}, \ldots, R_{N_t}^{(t)}\mathbf{1}_{H^{(t)}}\right\} \in \mathbb{R}^{H^{(t)}N_t \times H^{(t)}N_t}. \quad (19)$$

Input: λ, μ, ϵ, η, and $\{\boldsymbol{\tau}_n^{(t)}\}_{n=1}^{N_t}$, $t = 1, 2, \ldots, T$
Output: $\mathbf{D}^{(T)}$, $\mathbf{A}^{(T)}$, $\{\mathbf{s}^{(t)}\}_{t=1}^{T}$
1: Initialize $\mathbf{D}^{(0)}$ and $\mathbf{A}^{(0)}$ randomly
2: For $t = 1, 2, \ldots, T$
3: Obtain STL solution $\boldsymbol{\theta}_{\text{st}}^{(t)}$
4: Compute $\mathbf{H}_{\text{st}}^{(t)}$ from (20)
5: Compute sparse code $\mathbf{s}^{(t)}$ via (22)
6: Let $\check{\mathbf{D}}^{(t)} = [\mathbf{D}^{(t-1)}, \mathbf{X}^{(t)}]$
7: Compute $\check{\mathbf{A}}^{(t)}$ as the r.h.s. of (23)
8: Obtain sparsified pool $\mathbf{D}^{(t)}$ (via, e.g., destructive KOMP)
9: Compute $\mathbf{A}^{(t)}$ from (27)
10: End For

Table 1. Kernel lifelong PG RL algorithm.

Then, the Hessian in (13) can be approximated from samples $\boldsymbol{\Phi}(\mathbf{X}^{(t)})$ as

$$\mathbf{H}_{\text{st}}^{(t)} = \frac{1}{N_t}\boldsymbol{\Phi}(\mathbf{X}^{(t)})\boldsymbol{\mathcal{J}}_t''\boldsymbol{\Phi}(\mathbf{X}^{(t)})^\top. \quad (20)$$

To derive the library updates, consider the pool of samples $\mathbf{D}^{(t-1)} := [\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(t-1)}]$ taken up to task $t-1$ and invoke the Representer Theorem to represent $\mathbf{L}^{(t-1)}$ as [cf. (8)]

$$\mathbf{L}^{(t-1)} = \boldsymbol{\Phi}(\mathbf{D}^{(t-1)})\mathbf{A}^{(t-1)}. \quad (21)$$

At time t, upon obtaining $(\boldsymbol{\theta}_{\text{st}}^{(t)}, \mathbf{H}_{\text{st}}^{(t)})$, further define $\mathbf{K}_{t,t} := \boldsymbol{\Phi}(\mathbf{X}^{(t)})^\top\boldsymbol{\Phi}(\mathbf{X}^{(t)})$ as the per-task kernel matrix and $\mathbf{K}_{t,1:t-1} := \boldsymbol{\Phi}(\mathbf{X}^{(t)})^\top\boldsymbol{\Phi}(\mathbf{D}^{(t-1)})$ as the task-pool kernel matrix. Then, (15) can be written in dual form as

$$\mathbf{s}^{(t)} = \arg\min_{\mathbf{s}} \left\|\boldsymbol{\mathcal{J}}_t''^{\frac{1}{2}}\mathbf{K}_{t,t}\mathbf{w}_{\text{st}}^{(t)} - \boldsymbol{\mathcal{J}}_t''^{\frac{1}{2}}\mathbf{K}_{t,1:t-1}\mathbf{A}^{(t-1)}\mathbf{s}\right\|_2^2 + \mu\|\mathbf{s}\|_1 \quad (22)$$

and library $\mathbf{L}^{(t)} = \boldsymbol{\Phi}(\mathbf{D}^{(t)})\mathbf{A}^{(t)}$ can be updated as [cf. (17)]

$$\mathbf{A}^{(t)} = \begin{bmatrix} (1-\lambda\eta)\mathbf{A}^{(t-1)} \\ -\frac{\eta}{N_t}\boldsymbol{\mathcal{J}}_t''\left(\mathbf{K}_{t,1:t-1}\mathbf{A}^{(t-1)}\mathbf{s}^{(t)} - \mathbf{K}_{t,t}\mathbf{w}_{\text{st}}^{(t)}\right)\mathbf{s}^{(t)\top} \end{bmatrix}. \quad (23)$$

Since $\mathbf{A}^{(t)}$ (and $\mathbf{D}^{(t)}$) grow by $H^{(t)}N_t$ rows (columns, respectively) at each iteration, a method to control the growth in complexity of the representation of $\mathbf{L}$ is needed.

3.3. Sparsification

Despite its strength and flexibility, kernel SGD suffers from a rapid growth in complexity as additional samples are observed, which is also known as the curse of kernelization. In order to restrain the growth in complexity, the pool of samples $\mathbf{D}^{(t)}$ must be sparsified by throwing away the samples that do not contribute much in the representation of $\mathbf{L}$ [27]. Recently, a sparsification technique that can ensure convergence at the expense of small error in the functional approximation was developed, which is adopted here too [28].

With some abuse of notation, let $\mathbf{D}^{(t-1)} \in \mathbb{R}^{p\times M_{t-1}}$ denote the *sparsified* pool at iteration $t-1$ from now on. Then, the tentative pool at iteration t is given by

$$\check{\mathbf{D}}^{(t)} = [\mathbf{D}^{(t-1)}, \mathbf{X}^{(t)}] \quad (24)$$

and the corresponding library is defined as $\check{\mathbf{L}}^{(t)} := \boldsymbol{\Phi}(\check{\mathbf{D}}^{(t)})\check{\mathbf{A}}^{(t)}$, where $\check{\mathbf{A}}^{(t)}$ is obtained as the right-hand side of (23). The goal is

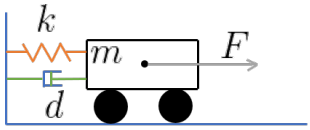
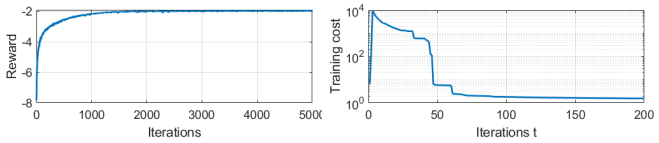


Fig. 1. A horizontal spring-mass-damper system.



(a) (b)

Fig. 2. Average convergence curves. (a) kernel STL (b) kernel LL

then to obtain a smallest subset $\mathbf{D}^{(t)}$ of $\check{\mathbf{D}}^{(t)}$ for which the projection error

$$\min_{\mathbf{L}\in\text{span}(\boldsymbol{\Phi}(\mathbf{D}^{(t)}))} \|\mathbf{L}-\check{\mathbf{L}}^{(t)}\|_{\mathcal{H}}^2 \tag{25}$$

does not exceed a pre-specified tolerance ϵ. In [28], a greedy algorithm called destructive kernel orthogonal matching pursuit (KOMP) was proposed to perform this subset search approximately.

Defining $\mathbf{K}_{\mathbf{D}^{(t)},\mathbf{D}^{(t)}} := \boldsymbol{\Phi}(\mathbf{D}^{(t)})^\top\boldsymbol{\Phi}(\mathbf{D}^{(t)})$ and $\mathbf{K}_{\mathbf{D}^{(t)},\check{\mathbf{D}}^{(t)}} := \boldsymbol{\Phi}(\mathbf{D}^{(t)})^\top\boldsymbol{\Phi}(\check{\mathbf{D}}^{(t)})$, projection (25) can be computed via

$$\mathbf{A}^{(t)} = \arg\min_{\mathbf{A}} \|\boldsymbol{\Phi}(\mathbf{D}^{(t)})\mathbf{A}-\boldsymbol{\Phi}(\check{\mathbf{D}}^{(t)})\check{\mathbf{A}}^{(t)}\|_{\mathcal{H}}^2 \tag{26}$$

$$= \mathbf{K}_{\mathbf{D}^{(t)},\mathbf{D}^{(t)}}^{-1}\mathbf{K}_{\mathbf{D}^{(t)},\check{\mathbf{D}}^{(t)}}\check{\mathbf{A}}^{(t)} \tag{27}$$

(again with some abuse of notation.) Finally, the library at iteration t is given as $\mathbf{L}^{(t)} = \boldsymbol{\Phi}(\mathbf{D}^{(t)})\mathbf{A}^{(t)}$. The overall algorithm is listed in Table 1.

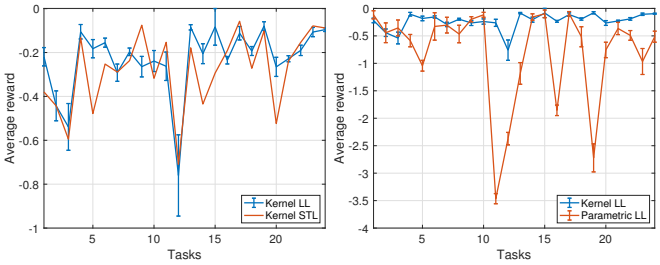
4. NUMERICAL EXPERIMENTS

To demonstrate the effectiveness of the proposed algorithm, it is tested with a spring-mass-damper system illustrated in Fig. 1, which is commonly considered in the RL literature. The system consists of a mass of m kg, connected horizontally on a fixed surface to a spring with a spring constant of k N/m and a damper with a damping constant of d N·s/m. A collection of $T = 24$ tasks were generated with the parameters drawn uniformly from $m \in [0.5, 5]$, $k \in [1, 10]$, and $d \in [0.01, 0.2]$, which allows for a variety of system responses. Denote the displacement of the mass by x, its velocity by $\dot{x}$ and the state vector by $\mathbf{x} = [x, \dot{x}]^\top$. The objective of each task is to move the mass from its initial state $[x_{\text{init}}, 0]^\top$ to a target state $[x_{\text{targ}}, 0]^\top$ by applying forces $\{F_h\}$ over horizon $h = 1, \ldots, 50$. We chose x_{init} randomly in $\{1, \ldots, 10\}$ and x_{targ} in $\{1, \ldots, 5\}$. The reward obtained at time step h is given by $r_h := -\sqrt{(x_h - x_{\text{targ}})^2 + (\dot{x}_h)^2}$.

The STL optima $\{\boldsymbol{\theta}_{\text{st}}^{(t)}\}$ were computed using a natural actor-critic (NAC) algorithm [4, 29] with the state vector appended by a constant 1 to allow for an affine offset in the case of parametric STL. The exploration variance is set to $\sigma^2 = 50^2$ for parametric policies and $\sigma^2 = 5^2$ for nonparametric policies. A Gaussian radial basis function (RBF) kernel was adopted, whose bandwidth parameter was chosen by observing the STL performances on a subset of 8 tasks. The regularization parameters λ and μ were selected via 10-fold cross-validation using a subset of 8 tasks chosen at random.

	pSTL	pLL	kSTL	kLL
Avg. reward	-0.8809	-0.8132	-0.2781	-0.2307
% improvement	0%	7.69%	68.43%	73.81%

Table 2. Average rewards and percentage improvements.



(a) (b)

Fig. 3. Per-task performances.

Then, 10 independent runs were performed on all tasks using the best regularization parameters. To evaluate either a STL or a LL, sample trajectories were taken using the learned policy but without the exploration, that is, with $\sigma^2 = 0$. We used $K = 10$.

In Fig. 2, the learning curves are presented to show convergence. In Fig. 2(a), the average reward (1) is shown for the NAC-based kernel STL algorithm. Fig. 2(b) depicts the objective of (8), where random permutations of T tasks were presented repeatedly to the proposed kernel LL algorithm.

In Table 2, the average rewards over 24 tasks and 10 independent runs of the parametric STL (pSTL), the parametric LL (pLL), the kernel STL (kSTL), and the kernel LL (kLL) algorithms are listed. Also shown is the percentage improvement with reference to the pSTL algorithm. It is noted that in both parametric and kernel cases, LL achieves performance gain by exploiting the shared structure in the collection of STLs, even though the STLs themselves have been fully trained. Furthermore, the advantage of using nonparametric learning algorithms is clearly observed over the parametric counterparts, which highlights the usefulness of considering rich families of functions for PG RL, even for the relatively simple tasks involving spring-mass-damper systems.

In Fig. 3(a), the reward of each of the 24 tasks using kLL is compared to that of kSTL. The error bars represent the standard deviations from 10 independent runs of the LL algorithm. It is observed that the LL strategy significantly improves the performance of the STL for some tasks—a clear manifestation of the multitask learning advantage, where the skills learned from other tasks are transferred to improve the performance for challenging tasks. Fig. 3(b) shows the rewards, averaged over 10 runs, for the kLL and the pLL algorithms. This highlights again the advantage of employing kernel learning in a lifelong setting.

5. CONCLUSION

A kernel-based lifelong PG RL algorithm has been proposed, which can exploit a shared union-of-subspace structure in RKHS across related tasks that arrive sequentially over time. An online learning algorithm was derived based on the SGD method in function space to yield low-cost updates. The prohibitive growth in computational and memory complexity inherent in kernel learning has been mitigated using a sparsification strategy. The numerical tests verified the advantage of the proposed algorithm, compared to both the nonparametric STL and the parametric lifelong counterparts.

6. REFERENCES

[1] M. P. Deisenroth, G. Neumann, and J. Peters, "A survey on policy search for robotics," *Foundations and Trends in Robotics*, vol. 2, no. 1–2, pp. 1–142, 2013.

[2] G. Lever and R. Stafford, "Modelling policies in MDPs in reproducing kernel Hilbert space," in *Proc. Int. Conf. Artificial Intell. Stat.*, San Diego, CA, May 2015, pp. 590–598.

[3] S. Paternain, J. Bazerque, A. Small, and A. Ribeiro, "Stochastic policy gradient ascent in reproducing kernel Hilbert spaces," *arXiv:1807.11274*, 2018.

[4] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomputing*, vol. 71, pp. 1180–1190, 2008.

[5] R. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[6] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Adv. Neural Info. Process. Syst.*, Denver, CO, 2000, pp. 1008–1014.

[7] R. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Adv. Neural Info. Process. Syst.*, Denver, CO, 2000, pp. 1057–1063.

[8] J. Kober and J. Peters, "Policy search for motor primitives in robotics," in *Adv. Neural Info. Process. Syst.*, Vancouver, BC, Dec. 2009, pp. 849–856.

[9] D. Abel, Y. Jinnai, S. Guo, G. Konidaris, and M. Littman, "Policy and value transfer in lifelong reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Stockholm, Sweden, Jul. 2018, pp. 20–29.

[10] F. Fernández and M. Veloso, "Learning domain structure through probabilistic policy reuse in reinforcement learning," *Progress Artific. Intell.*, vol. 2, pp. 13–27, 2013.

[11] M. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, 2009.

[12] A. Wilson, A. Fern, S. Ray, and P. Tadepalli, "Multi-task reinforcement learning: a hierarchical Bayesian approach," in *Proc. Int. Conf. Mach. Learn.*, Corvallis, OR, Jun. 2007, pp. 1015–1022.

[13] A. Lazaric and M. Ghavamzadeh, "Bayesian multi-task reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Haifa, Israel, Jun. 2010, pp. 599–606.

[14] R. Tutunov, D. Kim, and H. Ammar, "Distributed multi-task reinforcement learning with quadratic convergence," in *Adv. Neural Info. Process. Syst.*, Montreal, QC, Dec. 2018, pp. 8907–8916.

[15] H. Li, X. Liao, and L. Carin, "Multi-task reinforcement learning in partially observable stochastic environments," *J. Mach. Learn. Res.*, vol. 10, pp. 1131–1186, 2009.

[16] C. Dimitrakakis and C. Rothkopf, "Bayesian multitask inverse reinforcement learning," in *Proc. Eur. Workshop Reinforcement Learn.*, Athens, Greece, Sep. 2011, pp. 273–284.

[17] G. Pillonetto, F. Dinuzzo, and G. De Nicolao, "Bayesian online multitask learning of Gaussian processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 2, pp. 193–205, 2008.

[18] P. Ruvolo and E. Eaton, "ELLA: An efficient lifelong learning algorithm," in *Proc. Int. Conf. Mach. Learn.*, Atlanta, GA, Jun. 2013, pp. 507–515.

[19] C. Clingerman and E. Eaton, "Lifelong learning with Gaussian processes," in *Proc. Eur. Conf. Mach. Learn. Principles Practice Knowl. Discovery Databases*, Skopje, Macedonia, Sep. 2017, pp. 690–704.

[20] S.-J. Kim and R. Mowakeaa, "Kernel-based efficient lifelong learning algorithm," in *Proc. IEEE Data Sci. Workshop*, Minneapolis, MN, Jun. 2019.

[21] H. Ammar, E. Eaton, P. Ruvolo, and M. Taylor, "Online multi-task learning for policy gradient methods," in *Proc. Int. Conf. Mach. Learn.*, Beijing, China, Jun. 2014, pp. 1206–1214.

[22] H. Nguyen, V. Patel, N.Nasrabadi, and R. Chellappa, "Design of non-linear kernel dictionaries for object recognition," *IEEE Trans. Image Process.*, vol. 22, no. 12, pp. 5123–5135, Dec. 2013.

[23] S.-J. Kim, "Online kernel dictionary learning," in *Proc. IEEE Global Conf. Signal and Info. Process.*, Orlando, FL, Dec. 2015, pp. 103–107.

[24] P. Dayan and G. Hinton, "Using expectation-maximization for reinforcement learning," *Neural Computation*, vol. 9, no. 2, pp. 271–278, 1997.

[25] J. Kivinen, A. Smola, and R. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[26] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, New York, NY, 2004.

[27] P. Honeine, C. Richard, and J. C. M. Bermudez, "On-line nonlinear sparse approximation of functions," in *Proc. IEEE Int. Symp. Info. Theory*, Nice, France, Jun. 2007, pp. 956–960.

[28] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "Parsimonious online learning with kernels via sparse projections in function space," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, New Orleans, LA, 2017, pp. 4671–4675.

[29] N. Vien, P. Englert, and M. Toussaint, "Policy search in reproducing kernel Hilbert space.," in *Proc. Int. Joint Conf. Artific. Intell.*, New York, NY, Jul. 2016, pp. 2089–2096.