Policy Search in Reproducing Kernel Hilbert Space

Ngo Anh Vien and **Peter Englert** and **Marc Toussaint**
Machine Learning and Robotics Lab
University of Stuttgart, Germany
{vien.ngo, peter.englert, marc.toussaint}@ipvs.uni-stuttgart.de

Abstract

Modeling policies in reproducing kernel Hilbert space (RKHS) renders policy gradient reinforcement learning algorithms non-parametric. As a result, the policies become very flexible and have a rich representational potential without a predefined set of features. However, their performances might be either non-covariant under reparameterization of the chosen kernel, or very sensitive to step-size selection. In this paper, we propose to use a general framework to derive a new RKHS policy search technique. The new derivation leads to both a natural RKHS actor-critic algorithm and a RKHS expectation maximization (EM) policy search algorithm. Further, we show that kernelization enables us to learn in partially observable (POMDP) tasks which is considered daunting for parametric approaches. Via sparsification, a small set of "support vectors" representing the history is shown to be effectively discovered. For evaluations, we use three simulated (PO)MDP reinforcement learning tasks, and a simulated PR2's robotic manipulation task. The results demonstrate the effectiveness of the new RKHS policy search framework in comparison to plain RKHS actor-critic, episodic natural actor-critic, plain actor-critic, and PoWER approaches.

1 Introduction

Policy search methods in reinforcement learning (RL) have recently been shown to be very effective in very high dimensional robotics problems [Williams, 1992; Kober *et al.*, 2013]. In such applications, the policy is compactly represented in a parametric space and effectively optimized using gradient-descent [Sutton *et al.*, 1999]. Theoretically, policy gradient methods have strong convergence guarantees. In larger problems, their performance and computation can be further enhanced when combined with function approximation. The function approximation techniques might be linear [Kober *et al.*, 2013] or non-linear [Silver *et al.*, 2014; Lillicrap *et al.*, 2015].

Nonetheless, parametric approaches require a careful design of the features, which is inflexible in practical use and inefficient in large problems. One can overcome this inflexibility and limited adaptability by using nonparametric techniques, esp. by modeling policy in RKHS (reproducing kernel Hilbert space) function space [Bagnell and Schneider, 2003a; Lever and Stafford, 2015; van Hoof *et al.*, 2015]. Moreover, an adaptively compact policy representation can be achieved via sparsification in RKHS. For the case of a multi-dimensional output policy (multi-dimensional actions), a vector-valued kernel can be used [Micchelli and Pontil, 2005].

However, standard policy gradient methods might face difficult convergence problems. The issue stems from either using a non-covariant gradient or from selecting an adhoc step-size. The latter is a standard problem in any gradient-based method and is classically addressed using line-search, which may be sample-efficient. The step-size selection problem can be avoided by using EM-inspired techniques instead [Vlassis *et al.*, 2009; Kober and Peters, 2011]. Using a non-covariant gradient means that the gradient direction might change significantly even with a simple re-parameterization of the policy representation. Fortunately, the inherent non-covariant behaviour due to coordinate transformation [Amari, 1998] can be fixed using the natural gradient with respect to one particular Riemannian metric, namely the natural metric or the Fisher information metric [Kakade, 2001; Bagnell and Schneider, 2003b; Peters and Schaal, 2008].

The EM-inspired and policy gradient algorithms have further been shown to be special cases of a common policy framework, called free-energy minimization [Kober and Peters, 2011; Neumann, 2011]. Being inspired by this general derivation, in this paper we propose a policy search framework in RKHS which results in two RKHS policy search algorithms: RKHS natural actor-critic and RKHS EM-inspired policy search. We derive the following

- The natural gradient in RKHS is computed with respect to the Fisher information metric, which is a by-product of considering the path-distribution manifold in policy update. We show that plain functional policy gradients are non-covariant under the change of the kernel's hyperparameters, e.g. variance in RBF kernels or free parameters in polynomial kernels. We prove that our functional natural policy gradient is a covariant update rule. As a result, we design a new Natural Actor-Critic in the RKHS framework, called RKHS-NAC. Using a simple example we show that using RKHS-AC requires very careful choice of the kernel's hyperparameters, as it drastically affects the performance and converging policy quality. By contrast, RKHS-NAC free us from that painful problem.

- The RKHS EM-inspired algorithm is considered as a kernelized PoWER (Policy Learning by Weighting Exploration with the Returns). As a result, both mean and variance functionals of the policy are updated analytically based on a notion of bounds on policy improvements.
- The general RKHS policy search framework turns out to be an efficient technique to learning in POMDPs where the policy is modeled based on a sparse set of observed histories. This nonparametric representation renders our standard RKHS policy search algorithms a promising approach in POMDP learning which is daunting for parametric ones.

We compare our new framework to the recently introduced RKHS actor-critic framework (RKHS-AC) [Lever and Stafford, 2015], other parametric approaches such as the episodic natural actor-critic [Peters and Schaal, 2008], the actor-critic frameworks [Konda and Tsitsiklis, 1999], and PoWER [Kober and Peters, 2011]. The comparisons are done on benchmark RL problems in both MDP and POMDP environments, and a robotic manipulation task. The robotic task is implemented on a simulated PR2 platform.

2 Background

In this section, we briefly present related background that will be used in the rest of the current paper.

2.1 Markov Decision Process

A Markov decision process (MDP) is defined as 5-tuple $\{\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma\}$, where $\mathcal{S} \in \Re^m$ is a state space, $\mathcal{A} \in \Re^n$ is an action space, $\mathcal{T}(s, a, s') = p(s'|s, a)$ is a transition function, $\mathcal{R}(s, a, s')$ is a reward function, and a discount factor γ. An RL agent goes to optimize an optimal policy π, without knowing T, R, that maximizes a cumulative discounted reward.

$$U(\pi) = \int p(\xi; \pi) R(\xi) d\xi = \int p(\xi; \pi) \sum_{t=0}^{\infty} \gamma^t r_t d\xi \tag{1}$$

where $\xi = \{s_0, a_0, s_1, a_1, \ldots\}$ is a trajectory, $s_t \in \mathcal{S}, a_t \in \mathcal{A}$. We denote $R(\xi) = \sum_{t=0}^{\infty} \gamma^t r_t$ the cumulative discounted reward of a trajectory ξ, and $p(\xi; \pi)$ the trajectory distribution given π. In this paper, we use Gaussian policies

$$\pi(a|s) \sim e^{-\frac{1}{2}\big(h(s)-a\big)^\top \Sigma^{-1} \big(h(s)-a\big)} \tag{2}$$

where $h(s) : \mathcal{S} \mapsto \Re^n$ is a vector-valued function of the current state s, and Σ is an $n \times n$ covariance matrix. For instance, the linear parametrization approach of $h(s)$ assumes that the policy π is parametrized by a parameter space $\theta \in \mathcal{R}^d$, and depends linearly on predefined features $\phi_i(s)$, as

$$h(s) = \sum_{i=1}^{d} \theta_i \phi_i(s) \tag{3}$$

Similarly, nonparametric approaches with a reproducing kernel K represent $h(s) = \langle K(s), h \rangle$ (the reproducing property).

2.2 A Policy Search Framework

We briefly describe a general policy framework whose main derivation is to maximize a lower bound on the expected return at the new parameter value θ'.

$$\log U(\theta) = \log \int p(\xi; \theta) R(\xi) d\xi$$
$$= \int p(\xi; \theta') \log \frac{p(\xi; \theta) R(\xi)}{p(\xi; \theta')} d\xi + \mathrm{KL}\left(p(\xi; \theta') || p(\xi|R; \theta)\right)$$

where $\mathrm{KL}\left(p(\xi; \theta') || p(\xi|R; \theta)\right)$ is the Kullback-Leibler divergence, and assuming that the rewards are positive. In order to maximize $U(\theta)$, one can choose $p(\xi; \theta') \propto p(\xi|R; \theta)$ to make the KL zero (E-step). Substituting this result into the above inequality to obtain new performance $U(\theta')$ (M-step) as

$$\begin{aligned} \log U(\theta') &\geq \int p(\xi; \theta) R(\xi) \log \frac{p(\xi; \theta')}{p(\xi; \theta)} d\xi \\ &\propto -\mathrm{KL}\big(p(\xi; \theta) R(\xi) || p(\xi; \theta')\big) = \mathcal{L}(\theta'; \theta) \end{aligned} \tag{4}$$

This derivation can result in two different types of algorithms: 1) policy gradient, e.g. REINFORCE [Williams, 1992] and (Natural) Actor-Critic Algorithms [Konda and Tsitsiklis, 1999; Peters and Schaal, 2008]; 2) EM-inspired policy search, e.g. PoWER [Kober and Peters, 2011], MCEM [Vlassis *et al.*, 2009], VIP [Neumann, 2011]. All these approaches optimize $U(\theta')$ by maximizing the lower bound $\mathcal{L}(\theta', \theta)$, hence take the derivative $\nabla_{\theta'} \mathcal{L}(\theta'; \theta)$.

(Natural) Actor-Critic Algorithms

These methods use iterative update rules to increment θ along the direction of the gradient ascent $\nabla_\theta U(\theta) = \lim_{\theta' \to \theta} \nabla_{\theta'} L(\theta'; \theta)$. Using the policy gradient theorem [Sutton *et al.*, 1999; Baxter and Bartlett, 2001], we can derive that

$$\nabla_\theta U(\theta) = \mathbb{E}\Big[\sum_{t=0}^{H} \nabla_\theta \log \pi(a_t|s_t) A^\pi(s_t, a_t)\Big] \tag{5}$$

where $A^\pi(s_t, a_t) = \big(Q^\pi(s_t, a_t) - V(s_t)\big)$ is called an advantage function, and $Q^\pi(s, a)$ is the state-action Q-value function. Using function approximation, the advantage function $A^\pi(s, a)$ is approximated as $\tilde{A}(s, a) = \sum_{i=1}^{d} w_i \psi_i(s)$, in which the basis functions are $\psi_i(s) = \nabla_{\theta_i} \log \pi(a|s)$ in order to make the function approximation compatible with the policy parameterization. The parameter θ is updated along the direction of the gradient ascent $\nabla_\theta U(\pi)$. Interleaving updates of the weights θ (the actor's parameter) and w (the critic's parameter) lead to the actor-critic framework.

With compatible function approximation, the actor update using Amari's natural gradient [Amari, 1998] is proven to become $\theta \leftarrow \theta + \alpha w$. This leads to the natural actor-critic framework, e.g. the episodic natural actor-critic framework [Peters and Schaal, 2008]

EM-Inspired Policy Search

The methods eRWR [Peters and Schaal, 2007], Cost-regularized Kernel Regression (CrKR) [Kober *et al.*, 2010], and PoWER [Kober and Peters, 2011] introduce a notion of weighting exploration by changing the policy defined in Eq. 2 to $a = \big(\theta + \epsilon\big)^\top \phi(s)$, where $\epsilon \sim \mathcal{N}(\epsilon; 0, \Sigma)$. For brevity, we assume that the variance Σ is a diagonal matrix of σ_i. The policy is improved by computing the next parameter value θ' that makes $\nabla_{\theta'} \mathcal{L}(\theta'; \theta)$ equal to zero. As a result, the update in the form of the reward weighted exploration with the returns is (for dimension i-th)

$$\theta'_i = \theta_i + \frac{\mathbb{E}\Big[\sum_{t=0}^{H} \epsilon_{i,t} \sigma_i^{-1} A^\pi(s_t, a_t)\Big]}{\mathbb{E}\Big[\sum_{t=0}^{H} \sigma_i^{-1} A^\pi(s_t, a_t)\Big]} \tag{6}$$

If taking the derivative of the lower bound w.r.t. Σ' (or σ_i), we would also receive a similar update expression for the variance Σ'.

2.3 Actor Critic in RKHS

As recently introduced by [Bagnell and Schneider, 2003a; Lever and Stafford, 2015], the policies are parameterized by functionals in reproducing kernel Hilbert spaces. More specifically, the functional $h(s)$ in Eq. 2 is an element of a vector-valued RKHS $\mathcal{H}_K$

$$h(\cdot) = \sum K(s_i, \cdot)\alpha_i, \quad \text{where } \alpha_i \in \mathcal{A}(\in \mathcal{R}^n) \tag{7}$$

where the kernel K is defined as a mapping $\mathcal{S} \times \mathcal{S} \mapsto \mathcal{L}(\mathcal{A})$ [Micchelli and Pontil, 2005], $\mathcal{L}(\mathcal{A})$ is the space of linear operators on $\mathcal{A}$. The simplest choice of K might be $K(s, s') = \kappa(s, s')\mathrm{I}_n$, where I_n is an $n \times n$ identity matrix, and κ is a scalar-valued kernel [Schölkopf and Smola, 2002].

The functional gradient, i.e. the Fréchet derivative, of the term $\log \pi(a_t|s_t)$ is computed as

$$\nabla \log_h \pi_h(a_t, s_t) = K(s_t, \cdot)\Sigma^{-1}(a_t - h(s_t)) \in \mathcal{H}_K \tag{8}$$

Therefore, the gradient of the performance measure w.r.t. h using likelihood ratio methods is approximated by a trajectory set ξ_i of H steps as

$$\begin{aligned} \nabla_h U(\pi_h) &\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_h \log P(\xi_i; h) R(\xi_i) \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{H} A(s_t, a_t) K(s_t, \cdot)\Sigma^{-1}(a_t - h(s_t)) \end{aligned} \tag{9}$$

As a result, $\nabla_h U(\pi_h) \in \mathcal{H}_K$ and the functional gradient update at iteration k is

$$h_{k+1} \leftarrow h_k + \alpha \nabla_h U(\pi_{h_k}) \tag{10}$$

where α is a step-size. After each gradient update, the number of centres (s_t) of the next h_{k+1} might increase rapidly. One can attain sparse representation via sparsification. In addition, Lever and Stafford proved that the compatible kernel function for $A^\pi(s, a)$, where K_h is a compatible kernel that is constructed based on K as

$$K_h\big((s, a), (s', a')\big) = \Big(K(s, s')\Sigma^{-1}(a - h(s))\Big)^\top \Sigma^{-1}(a' - h(s')) \tag{11}$$

As a result of compatible function approximation, there exists a feature space $\mathcal{F}_\phi \equiv \mathcal{H}_K$ that contains $A_h(s, a)$ and the weight w and satisfies

$$\begin{aligned} A_h(s, a) &= \langle w, \nabla_h \log \pi(a|s)\rangle_{\mathcal{F}_\phi} \\ &= \langle w, K(s, \cdot)\Sigma^{-1}(a - h(s))\rangle_{\mathcal{H}_K} \end{aligned} \tag{12}$$

There are a number of ways to regress $A_h(s, a)$ from data generated from π_h. Lever et. al. [Lever and Stafford, 2015] used kernel pursuit matching [Vincent and Bengio, 2002] for both function regression and sparsification. Accordingly, they propose the compatible RKHS Actor-Critic framework, RKHS-AC.

However, the RKHS-AC's gradient update is not covariant w.r.t. the change of the kernel's hyperparameter. We show an example use of RKHS-AC in the toy problem in Fig. 1 (see the experiment section for the problem description). We use two types of kernel to represent our functional policy in Eq. 2 with different values of hyperparameters: polynomial kernel $\kappa(s, s') = (s^\top s' + c)^d$, where $d = 5.$, and $c = 0.01; 0.2; 1.0$; and RBF kernel $\kappa(s, s') = \exp(-\|s - s'\|/(2 * var))$, where $var = 0.01; 0.1; 2.0$. Note that the changes of the hyperparameters c, var lead to transformation of the coordinate system in the same corresponding feature space. The results show that RKHS-AC is non-covariant. RKHS-AC does not guarantee good improvement in all setting. For instance, in the cases of $c = 0.01$ and $var = 2.0$ some features are very peaky, this would make the policy overwhelmingly exploit non-optimal actions at early stage of learning.

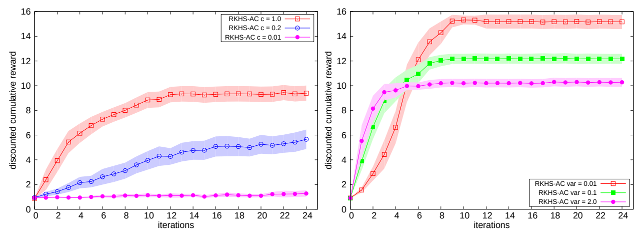

Figure 1: Discounted cumulative reward in Toy domain with polynomial and RBF kernels. The horizon H is set to 20. We report the mean performances averaged over 50 runs, and their 95% confidence interval in shading areas

3 Policy Search in RKHS

Inspired by the derivation of the parametric policy search framework, we derive a new policy search framework which enables policy modeling in RKHS. Assume that the controller is $a \sim \mathcal{N}(a; h(s), \Sigma(s))$, where $a \in \Re^n$, $h(\cdot) \in \mathcal{H}_K$ as defined in Eq. 7, and $\Sigma(\cdot)$ is a matrix-valued function. By reshaping $\Sigma(\cdot)$ to a vector-valued function, one could embed it into a another vector-valued RKHS. For brevity, we assume that $\Sigma(\cdot)$ is a diagonal matrix-valued function which allows independent exploration in each dimension of actions. The following derivation still applies for the cases of general matrices. The variance on each dimension is a scalar-valued functional $g_i(\cdot) \in \mathcal{H}_{K_i}$ with a reproducing kernel K_i, $\forall i = 1, \ldots, n$.

The lower bound of the expected return is similarly derived as

$$\log U(h') \geq \int p(\xi; h)R(\xi) \log \frac{p(\xi; h')}{p(\xi; h)} d\xi \propto \mathcal{L}(h'; h) \tag{13}$$

Its derivative is

$$\nabla_{h'}\mathcal{L}(h'; h) = \mathbb{E}\left[\sum_{t=0}^{H} \nabla_{h'} \log \pi(a_t|s_t) R(\xi)\right] \tag{14}$$

We next derive two RKHS policy search algorithms based on the above notion of bounds on policy improvement.

3.1 (Natural) Policy Gradients in RKHS

The RKHS policy gradient of Lever and Stafford [Lever and Stafford, 2015] can exactly be re-derived by computing the derivative of $U(h)$ in Eq. 10 if we set $\nabla_h U(h) = \lim_{h' \to h} \nabla L(h'; h)$ (see Eq. 14).

In this paper, we are interested in applying the natural gradient method [Amari, 1998] to update the functional policy. The key idea is to constraint the change of $p(\xi; h)$ after each gradient update. For vanilla gradient, the functional gradient direction is $g = \nabla_h U(\pi_h)$. With the constraint of a small change in the trajectory distribution, we compute the steepest gradient direction by solving an optimization problem,

$$\max_{h'} . \quad L(h'; h), \quad s.t.\ \mathrm{KL}\big(p(\xi; h)||p(\xi; h')\big) \leq \delta \tag{15}$$

Using the second-order Taylor expansion for the KL-divergence constraint, we can prove that the *Fisher information operator* [Nordebo *et al.*, 2010] is

$$\begin{aligned} G &= \mathbb{E}_{\xi \sim P(\xi;h)} \Big[\nabla^2_h \log P(\xi;h)\Big] \\ &= \mathbb{E}_{\xi \sim P(\xi;h)} \Big[\nabla_h \log P(\xi;h) \otimes \log \nabla_h P(\xi;h)\Big] \end{aligned} \tag{16}$$

in which we use the fact: $\int p(\xi) d\xi = 1$ twice.

We now prove an important property of G in order to derive further results: *the bounded property*. Consequently, there exists the inverse operator G^{-1} of G according to the bounded inverse theorem [Reed and Simon, 2003].

Proposition 1 *The linear operator $G : \mathcal{H} \mapsto \mathcal{H}$ is bounded.*

Proof: The linear operator G is a mapping from a Hilbert space $\mathcal{H}$ to $\mathcal{H}$, hence according to the Riesz representation theorem [Reed and Simon, 2003] there exists the adjoint of G. The adjoint is a continuous linear operator $G^* : \mathcal{H} \mapsto \mathcal{H}$ such that $\langle Gh, g\rangle = \langle h, G^* g\rangle$, where $h, g \in \mathcal{H}$. Moreover, we can easily see that G^*G defines a Gram operator, hence it is positive definite and self-adjoint. Thus, G^*G is a bounded linear operator [Reed and Simon, 2003]. In other words, there exists a number $M > 0$ such that $\|G^*Gh\| \leq M\|h\|$. From those results of G and G^*G, we can derive

$$\begin{aligned} \langle Gh, Gh\rangle &= \langle h, G^*Gh\rangle \\ &\leq \|h\|\|G^*Gh\| \leq M\|h\|^2 \end{aligned}$$

that means the linear operator G is also bounded. □

From a set of N trajectories $\{\xi_i\}$, G can be estimated as

$$\begin{aligned} \hat{G} &= \frac{1}{N}\sum_{i=1}^{N} \nabla_h \log P(\xi_i, h) \otimes \nabla_h \log P(\xi_i, h) \\ &= \frac{1}{N}\Phi(\xi)\Phi(\xi)^\top \end{aligned} \tag{17}$$

where $\Phi(\xi) = [\nabla_h \log P(\xi_1; h), \ldots, \nabla_h \log P(\xi_N; h)]$. Using Lagrange multipliers for the optimization problem in (15), the natural functional gradient is

$$\begin{aligned} \nabla^{NG}_h U(\pi_h) &= G^{-1}\nabla_h U(\pi_h) \\ &\approx \frac{1}{N}\Big(\frac{1}{N}\Phi(\xi)\Phi(\xi)^\top + \lambda I\Big)^{-1}\Phi(\xi)R \end{aligned} \tag{18}$$

where $R = [R(\xi_1), \ldots, R(\xi_N)]$, and λ is a regularizer. Using the Woodbury identity for Eq. 18, we obtain

$$\begin{aligned} \nabla^{NG}_h U(\pi_h) &\approx \Phi(\xi)\Big(\Phi(\xi)^\top\Phi(\xi) + \lambda N I\Big)^{-1} R \\ &= \Phi(\xi)\Big(\Lambda + \lambda N I\Big)^{-1} R \\ &= \sum_{i=1}^{N}\sum_{t=0}^{H} K(s_t, \cdot)\Sigma^{-1}(a_t - h(s_t))\hat{R}_i \end{aligned}$$

where Λ is a Gram matrix of trajectories which is built based on the trajectory kernel function $K_\xi(\xi_i, \xi_j)$; and $\hat{R} = \Big(\Lambda + \lambda N I\Big)^{-1} R$. We now define the scalar-valued trajectory kernel $K_\xi(\xi_i, \xi_j)$. Note that the gradient of log trajectory distribution is

$$\nabla_h \log P(\xi_i; h) = \sum_{t=0}^{H} K(s_t, \cdot)\Sigma^{-1}(a_t - h(s_t)) \tag{19}$$

Therefore,

$$\begin{aligned} K_\xi(\xi_i, \xi_j) &= \sum_{i,j}\Big(K(s_i, s_j)\Sigma^{-1}(a_i - h(s_i))\Big)^\top \Sigma^{-1}(a_j - h(s_j)) \\ &= \sum_{i=1}^{T_i}\sum_{j=1}^{T_j} K_h\Big((s_i, a_i), (s_j, a_j)\Big) \end{aligned}$$

The resulting kernel of trajectories is a sum of kernel functions of all visited state-action pairs $K_h\big((s,a),(s',a')\big)$ defined in Eq. 11. As a result of summing of many proper kernels, the trajectory kernel is proper and specifies a RKHS $\mathcal{H}_{K_\xi}$ of trajectory feature maps.

Finally, the plain natural policy gradient in RKHS, reported in Algorithm 1, has the following update

$$h_{t+1} = h_t + \alpha \nabla^{NG}_h U(\pi_h) \tag{20}$$

3.2 Episodic Natural Actor-Critic in RKHS

We now use the above results to propose a new episodic natural actor-critic framework that enables policy modeling in RKHS.

The Actor Update

First, we rewrite the Fisher information operator in Eq. 16 by substituting the definition of $\nabla \log P(\xi)$ in Eq. 5 to obtain

$$\begin{aligned} G &= \mathbb{E}_{P(\xi,h)}\Big[\sum_{t=0}^{H}\nabla^2_h \log \pi(a_t|s_t)\Big] \\ &= \mathbb{E}_{P(\xi,h)}\Big[\sum_{t=0}^{H}\nabla_h \log \pi(a_t|s_t)\nabla_h \log \pi(a_t|s_t)^\top\Big] \end{aligned} \tag{21}$$

where we use the fact: $\int \pi(a|s) da = 1$ twice. On the other hand, we replace $A(s,a)$ in Eq. 12 into Eq. 9 to obtain

$$\begin{aligned} \nabla_h U(\pi) &= \mathbb{E}_{P(\xi,h)}\Big[\sum_{t=0}^{H}\langle w, \nabla_h \log \pi(a_t|s_t)\rangle \nabla \log \pi(a_t|s_t)\Big] \\ &= \mathbb{E}_{P(\xi,h)}\Big[\sum_{t=0}^{H}\nabla_h \log \pi(a_t|s_t)\rangle \nabla \log \pi(a_t|s_t)^\top w\Big] \\ &= Gw \end{aligned}$$

As a result, the natural gradient in RKHS used in the actor update is

$$\nabla^{NG}_h U(\pi_h) = G^{-1}Gw = w \in \mathcal{H}_K \tag{22}$$

The Critic Evaluation

Similar to eNAC [Peters and Schaal, 2008], we write the entire rollout of Bellman equations for an episode, and put the definition of $A^\pi(s_t, a_t)$ in Eq. 12 to get

$$\begin{aligned} \sum_{t=0}^{H}\gamma^t r(s_t, a_t) &= \sum_{t=0}^{H}\gamma^t A^\pi(s_t, a_t) + V^\pi(s_0) \\ &= \langle w, \sum_{t=0}^{H}\gamma^t K(s_t, \cdot)\Sigma^{-1}(a_t - h(s_t))\rangle + V^\pi(s_0) \end{aligned}$$

where $V^\pi(s_0)$ is the value function of the starting state. One could use another kernel to estimate $V(s_0)$. We propose to use kernel matching pursuit algorithm [Vincent and Bengio, 2002] to regress $A^\pi(s,a)$ by sampling a set of trajectories from the policy π_h.

Putting all ingredients together, we derive the RKHS natural actor-critic framework (RKHS-NAC), as described in Algorithm 1. The algorithm uses the kernel matching pursuit method to sparsify the new functional h_{k+1}.

Algorithm 1 RKHS Policy Search Framework

```
Given the kernel $K$, initialize $h_0 = 0$
for $k = 0, 1, 2, \ldots$ do
    Sample $N$ trajectories $\{\xi_i\}$ from $\pi_{h_k}$
    Estimate $A^{\pi}(s,a)$ using $\{\xi_i\}_{i=1}^{N}$ to obtain $w \in \mathcal{H}_K$
    RKHS-NPG    : $h_{k+1} = h_k + \alpha\nabla_h^{NG} U(\pi_{h_k})$
    RKHS-NAC    : Update the policy $h_{k+1} = h_k + \alpha_k w$
    RKHS-PoWER: Update $h_{k+1}, g_{k+1}$ as in Eq. 24
    Sparsify $h_{k+1} \in \mathcal{H}_K$
end for
```

3.3 EM-Inspired Policy Search in RKHS

The major problem of the policy gradient approaches, like RKHS-AC and RKHS-NAC, is the step-size α_k. As inspired by PoWER, we propose an alternative solution for maximizing the lower bound $L(h'; h)$ by setting its derivative $\nabla_{h'} L(h'; h)$ to zero and solving for h' to obtain

$$\mathbb{E}_{\pi_h}\left\{\sum_{t=0}^{H} A(s_t,a_t)K(s,\cdot)\Sigma^{-1}(s)\big(a_t - h'(s_t)\big)\right\} = 0$$

Hence,

$$h' = h + \mathbb{E}_{\pi_h}\left\{\sum_{t=0}^{H} A(s_t,a_t)K(s,\cdot)\Sigma^{-1}(s)K^{\top}(s,\cdot)\right\}^{-1} \times \mathbb{E}_{\pi_h}\left\{\sum_{t=0}^{H} A(s_t,a_t)K(s,\cdot)\Sigma^{-1}(s)\big(a_t - h(s_t)\big)\right\}$$

where we exploited the reproducing property $h'(s) = \langle h', K(s,\cdot)\rangle$. The update rule for the variance functionals $\Sigma(\cdot)$ or $g_i(\cdot)$ are similarly derived as

$$g_i' = \frac{\mathbb{E}_{\pi_h}\left[\sum_{t=0}^{H} \epsilon_t^{i2} A(s_t,a_t)K_i(s_t,\cdot)K_i(s_t,s_t)^{-1}\right]}{\mathbb{E}_{\pi_h}\left[\sum_{t=0}^{H} \epsilon_t^{i2} A(s_t,a_t)\right]} \in \mathcal{H}_{K_i}$$

where $\epsilon_t^i = \big(a_t - h(s_t)\big)_i$ (the i-th dimension). For an example of how to estimate h' and Σ' given a set of N trajectories $(s_{jt}, a_{jt})_{j=1,t=0}^{N,H}$ sampled from π_h, we assume $K(s,s') = \kappa(s,s')\mathrm{I}_n$ for simplicity purposes, where $\kappa(s_i,s_j)$ is a scalar-valued kernel. The estimates for the i-th dimension are [1]

$$h_i' = h_i + \sum_{j=1}^{N}\sum_{t=0}^{H} \kappa(s_{jt},\cdot)\alpha_{jt}^{i} \tag{23}$$

$$g_i' = \sum_{j=1}^{N}\sum_{t=0}^{H} K_i(s_{jt},\cdot)\beta_{jt}^{i} \tag{24}$$

where we denote $\Phi = \big(\kappa(s_{1,0},\cdot),\ldots,\kappa(s_{N,H},\cdot)\big)$, $A_1 = \mathrm{diag}\,\big(A(s_{1,0},a_{1,0})/g_i(s_{1,0}),\ldots,A(s_{N,H},a_{N,H})/g_i(s_{N,H})\big)$, $A_2 = \big(\epsilon_{1,0}^{i},\ldots,\epsilon_{N,H}^{i}\big)^{\top}$, $\Gamma = \Phi^{\top}\Phi$ is a Gram matrix where $\Gamma_{ij} = \kappa(s_i,s_j)$, to compute $\alpha^i = (\lambda A_1^{-1} + \Gamma)^{-1}A_2$ (the constant λ is a regularizer) [2], and

$$\beta_{jt}^{i} = \frac{\epsilon_{jt}^{i2}A(s_{jt},a_{jt})}{K_i(s_{jt},s_{jt})\sum_{k=1}^{N}\sum_{l=0}^{H}\epsilon_{kl}^{i2}A(s_{kl},a_{kl})}$$

The overall algorithm, named RKHS-PoWER, is reported in Algorithm 1.

[1]The results are based on the use of the Woodbury matrix identity

[2]α^i is a $N.H \times 1$ vector, is reshaped to $N \times H$ matrix

3.4 Policy Search in POMDP Environment

One of the very interesting characteristics of the RKHS policy search framework is learning in POMDP environments. POMDP learning requires action selection at time t: $\mu(a_t|h_t)$ to depend on the history $h_t = \{o_0, a_0, \ldots, o_t\}$, where o_t is an observation. Feature design in POMDP learning is elusive for standard parametric policy search approaches, as the history space is very large and high-dimensional. It's common for the parametric methods to put regular features on each state dimension. Hence parametric methods suffer from not only the *curse of dimensionality* (the number of features is exponential on the number of state dimensions) but also the *curse of history* (the number of features is exponential on the length of a history state). In contrast, our RKHS policy search approaches discover history features adaptively and sparsely. One may find this learning similar to the non-controlled POMDP learning using conditional embeddings of predictive state representation [Boots *et al.*, 2013]. In our experiment, we fix the length of h_t (the number of history steps) and use an RBF kernel to measure the distance between two histories.

4 Experiments

We evaluate and compare on four domains (three learning in MDP and one learning in POMDP): a Toy domain, the benchmark Inverted Pendulum domain, a simulated PR2 robotic manipulation task, and the POMDP inverted Pendulum. The presented methods are: RKHS-NAC, RKHS-AC [Lever and Stafford, 2015], PoWER [Kober and Peters, 2011], eNAC [Peters and Schaal, 2008], and actor-critic. All methods use a discount factor $\gamma = 0.99$ for learning and report the cumulative discounted reward. The number of centres used in RKHS methods is comparable to the number of bases used in parametric methods. All tasks use the RBF kernel. The bandwidth of RBF kernels is chosen using the *median trick*.

4.1 Toy Domain

The Toy domain was introduced by Lever and Stafford [Lever and Stafford, 2015]. It has a state space $\mathcal{S} \in [-4.0; 4.0]$, an action space $\mathcal{A} \in [-1.0; 1.0]$, the starting state $s_0 = 0$, and a reward function $r(s,a) = \exp(-|s-3|)$. The dynamics is $s_{t+1} = s_t + a_t + \epsilon$, where ϵ is a small Gaussian noise. We set $N = 10$, $H = 20$. All controllers use 20 centres. The optimal value is around 14.5 [Lever and Stafford, 2015].

We first study the covariant behaviour of the natural RKHS policy gradient method, as not seen in the standard RKHS-AC described in Section 2.3. Both algorithms use line-search over a grid of 50 choices. The results are computed over 50 runs as depicted in Fig. 2. The performance of RKHS-NAC in all setting for both polynomial and RBF kernels ensures that RKHS-NAC is covariant.

Figure 3 reports the comparisons of other methods without line-search. We chose the best step-size for each algorithm to ensure their best performance (AC and RKHS-AC use 0.005; eNAC and RKHS-NAC use 0.1). Arbitrary choice of the step-size may yield instability and local optimality which is not a problem with the EM-inspired methods. All RKHS methods perform comparably well and slightly better than AC and eNAC. PoWER improves slowly but very stably.

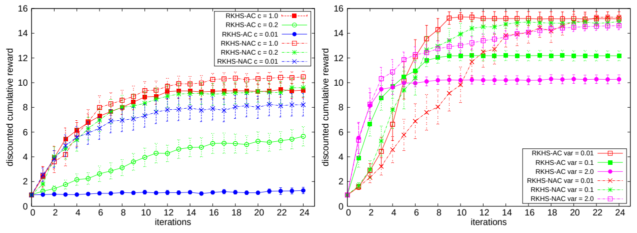

Figure 2: Toy domain with the use of (left) polynomial kernel, (right) RBF kernel.

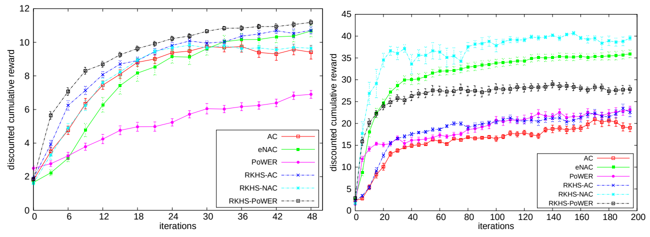

Figure 3: (left) Toy Domain. (right) Inverted Pendulum

4.2 Inverted Pendulum

This task [Lever and Stafford, 2015] has an action domain $[-3, 3]$, a state space $s = (\theta, \omega)$ which are angular position $\theta \in [-\pi, \pi]$ and velocity $\omega \in [-4\pi, 4\pi]$ values. The initial state is always at $s_0 = (-\pi, 0)$. The reward function is $r(s, a) = \exp(-0.5\theta^2)$ and The dynamics is: $\theta' = \theta + 0.05\omega + \epsilon$; $\omega' = \omega + 0.05a + 2\epsilon$, where ϵ is a small Gaussian noise with a standard deviation of 0.02. Each transition applies the above dynamics two times (for the same action value). We use $N = 10$, $H = 100$ whose optimal return is roughly 46. We use 40 RBF centres in all controllers. The averaged performance is obtained over 15 runs. If the pendulum is balanced, the return should be at least 25. All methods use carefully selected step-sizes to achieve their best performance.

The mean performance and the mean's first deviation of all algorithms are reported in Fig. 3. While RKHS-AC, AC, and PoWER are still struggling in finding the optimal policy after 200 iterations, RKHS-NAC, RKHS-PoWER and eNAC can achieve a return over 25 after 20 iterations. The performance of RKHS-NAC and eNAC are very promising in this domain which are very close to the optimal performance.

4.3 POMDP Inverted Pendulum

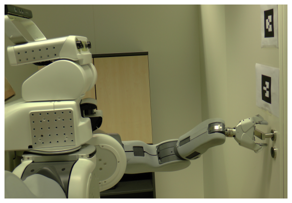

Figure 4: The PR2 robot is opening a door.

The third task is the POMDP Inverted Pendulum. We keep using the original dynamics and reward functions and add an observation function to create a learning task in POMDP. We assume that the agent can only observe a noisy angular position $o_t = \theta_t + 2\epsilon$. We use 150 centres for all algorithms and line-search over a grid of 50 step-sizes, and the length 20 for a history state. We use only eNAC as the best parametric contender whose state space is 3-step history states and put 4 centres on each state dimension (4^6 features). Figure 5 reports the average performances over 10 runs and their first deviations. RKHS-NAC performs best as this task has a very large history state space where bandwidth setting is very important. Because RKHS-NAC is covariant under such transformation which is yielded from changes of the bandwidth.

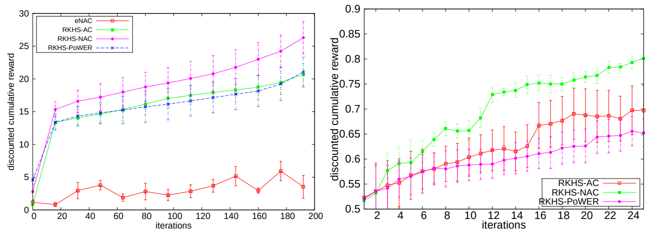

Figure 5: Results for (left) the POMDP inverted pendulum;(right) the simulated robotic manipulation task.

4.4 Robotic Manipulation Task

This task is a door-opening task on a simulated PR2 robot (using the physics simulation ODE internally) and is only a comparison of the three RKHS methods. We give a demonstration via kinesthetic teaching on a physical Willow-Garage PR2 platform whose setting is as seen in Fig. 4. As action space we define a two dimensional space that defines the gripper position on the door handle. The first action is the horizontal position of the finger on the door and the second action is the finger opening width. Both actions transform the demonstration trajectory. The resulting trajectory is then executed on the simulator to return a reward value. The reward function consists of two parts: $e^{-(R_1+R_2)}$. The first part measures how close the current trajectory is to the demonstration, R_1. The second part is a binary penalty cost that tells whether the door is opened successful, R_2. We set $N = 10, H = 1$.

Figure 5 reports the average performance over 5 runs with a fixed step-size $\alpha = 0.005$ for RKHS-AC and RKHS-NAC. Each run starts from different initial poses of the robot. All three algorithms are making improvement, RKHS-PoWER's performance is improving slowly but looks more stable.

5 Conclusion

We have proposed a new policy search framework that enables policy modeling in reproducing kernel Hilbert space. The main insight is in its derivation that makes the previous RKHS policy gradient framework a special case, and additionally results in two new algorithms: natural actor-critic in RKHS and the EM-inspired policy search in RKHS. While RKHS-NAC is a fix for the non-covariant RKHS-AC algorithm, the RKHS-PoWER is able to mitigate the step-size problem. Their experimental results are very promising due to the covariant behaviour and the new application track in POMDPs. The covariant behaviour seems to play a very important role in RKHS methods, because the bandwidth setting issue is not as easy as in parametric methods, especially in very large problems.

Acknowledgment

This work was supported by the DFG (German Science Foundation) under grant number 409/9-1 within the priority program Autonomous Learning SPP 1527 and the EU-ICT Project 3rdHand 610878. We would like to thank Guy Lever (UCL) for fruitful discussions.

References

[Amari, 1998] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, February 1998.

[Bagnell and Schneider, 2003a] J. Andrew Bagnell and Jeff Schneider. Policy search in reproducing kernel hilbert space. Technical Report CMU-RI-TR-03-45, Robotics Institute, Pittsburgh, PA, November 2003.

[Bagnell and Schneider, 2003b] J. Andrew Bagnell and Jeff G. Schneider. Covariant policy search. In *IJCAI*, pages 1019–1024, 2003.

[Baxter and Bartlett, 2001] Jonathan Baxter and Peter L. Bartlett. Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res. (JAIR)*, 15:319–350, 2001.

[Boots *et al.*, 2013] Byron Boots, Geoffrey J. Gordon, and Arthur Gretton. Hilbert space embeddings of predictive state representations. In *UAI*, 2013.

[Kakade, 2001] Sham Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems 14 (NIPS)*, pages 1531–1538, 2001.

[Kober and Peters, 2011] Jens Kober and Jan Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2):171–203, 2011.

[Kober *et al.*, 2010] Jens Kober, Erhan Oztop, and Jan Peters. Reinforcement learning to adjust robot movements to new situations. In *Robotics: Science and Systems VI*, 2010.

[Kober *et al.*, 2013] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *I. J. Robotic Res.*, 32(11):1238–1274, 2013.

[Konda and Tsitsiklis, 1999] Vijay R. Konda and John N. Tsitsiklis. Actor-critic algorithms. In *Advances in Neural Information Processing Systems 12 (NIPS)*, pages 1008–1014, 1999.

[Lever and Stafford, 2015] Guy Lever and Ronnie Stafford. Modelling policies in mdps in reproducing kernel hilbert space. In *AISTATS*, 2015.

[Lillicrap *et al.*, 2015] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.

[Micchelli and Pontil, 2005] Charles A. Micchelli and Massimiliano Pontil. On learning vector-valued functions. *Neural Computation*, 17(1):177–204, 2005.

[Neumann, 2011] Gerhard Neumann. Variational inference for policy search in changing situations. In *ICML*, pages 817–824, 2011.

[Nordebo *et al.*, 2010] Sven Nordebo, Andreas Fhager, Mats Gustafsson, and Börje Nilsson. Fisher information integral operator and spectral decomposition for inverse scattering problems. *Inverse Problems in Science and Engineering, Taylor & Francis*, 18(8), 2010.

[Peters and Schaal, 2007] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *ICML*, pages 745–750, 2007.

[Peters and Schaal, 2008] Jan Peters and Stefan Schaal. Natural actor-critic. *Neurocomputing*, 71(7-9):1180–1190, 2008.

[Reed and Simon, 2003] Michael Reed and Barry Simon. *Functional Analysis*. Elsevier, 2003.

[Schölkopf and Smola, 2002] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels Support Vector Machines, Regularization, Optimization, and Beyond*. Adaptive Computation and Machine Learning series, MIT Press, 2002.

[Silver *et al.*, 2014] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *ICML*, pages 387–395, 2014.

[Sutton *et al.*, 1999] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12 (NIPS)*, pages 1057–1063, 1999.

[van Hoof *et al.*, 2015] Herke van Hoof, Jan Peters, and Gerhard Neumann. Learning of non-parametric control policies with high-dimensional state features. In *AISTATS*, 2015.

[Vincent and Bengio, 2002] Pascal Vincent and Yoshua Bengio. Kernel matching pursuit. *Machine Learning*, 48(1-3):165–187, 2002.

[Vlassis *et al.*, 2009] Nikos Vlassis, Marc Toussaint, Georgios Kontes, and Savas Piperidis. Learning model-free robot control by a monte carlo EM algorithm. *Auton. Robots*, 27(2):123–130, 2009.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.