Modelling Policies in MDPs in Reproducing Kernel Hilbert Space

Guy Lever
University College London
London, UK
g.lever@cs.ucl.ac.uk

Ronnie Stafford
University College London
London, UK
ronnie.stafford@gmail.com

Abstract

We consider modelling policies for MDPs in (vector-valued) reproducing kernel Hilbert function spaces (RKHS). This enables us to work "non-parametrically" in a rich function class, and provides the ability to learn complex policies. We present a framework for performing gradient-based policy optimization in the RKHS, deriving the functional gradient of the return for our policy, which has a simple form and can be estimated efficiently. The policy representation naturally focuses on the relevant region of state space defined by the policy trajectories, and does not rely on a-priori defined basis points; this can be an advantage in high dimensions where suitable basis points may be difficult to define a-priori. The method is adaptive in the sense that the policy representation will naturally adapt to the complexity of the policy being modelled, which is achieved with standard efficient sparsification tools in an RKHS. We argue that finding a good kernel on states can be easier then remetrizing a high dimensional feature space. We demonstrate the approach on benchmark domains and a simulated quadrocopter navigation task.

1 INTRODUCTION

Gradient methods are a popular means of performing policy optimization in reinforcement learning (RL) and control problems modelled as Markov decision processes (Sutton et al., 1999), and have achieved significant practical success (e.g. Kober and Peters, 2011; Deisenroth and Rasmussen, 2011; Kohl and Stone, 2004). Some problems with the approach persist however. Firstly, practitioners usually choose a parametric class to model policies, such as linear functions on some predefined feature space. It can be difficult to pick a suitable feature space a-priori and the ability of such parametric classes to represent complex functions can be poor. Recently there has been interest in policy gradient methods in very rich policy classes such as those parameterized by neural networks (e.g. Wierstra et al., 2010; Heess et al., 2012; Silver et al., 2014). Secondly, steepest gradient ascent is not invariant to a rescaling of parameters and a poor choice of parameter scaling can adversely affect the performance of steepest ascent. Identifying a suitable scaling has been identified as an important problem and practitioners often rely on remetrizing, using, for example, an inverse Fisher information matrix as a preconditioner, as in natural gradients (Kakade, 2001). Such preconditioning can be expensive in large parameter spaces.

In this work we present an alternative method of modelling policies "non-parametrically"[1] in vector-valued reproducing kernel Hilbert spaces (RKHS), which addresses the problems discussed above. The key advantages of this approach are:

1. **The policy space can be a very rich function class**. For example, kernels can be chosen such that any continuous policy can be accurately modelled in the RKHS. The policy gradient is an entire function in the RKHS, not restricted by any a-priori chosen parameterization of the class.
2. **The gradient of the log policy has a simple analytic form**. The policy gradient can be easily and efficiently estimated.
3. **Complex policies can be compactly represented**. The representation of the policy within the RKHS can be adaptive to the complexity of the problem, finding a compact representation if sufficient, or a complex representation if needed, and focus on the region of state space defined by the policy trajectory. This can be achieved using standard, efficient sparsification methods in an RKHS, see Section 2.2. We do not have to

Appearing in Proceedings of the 18^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2015, San Diego, CA, USA. JMLR: W&CP volume 38.

[1] By which we mean policies can be drawn from a potentially infinite-dimensional function space.

specify a large feature space a-priori, or deal with a large number of parameters.

4. **It may alleviate the need for remetrization via preconditioning**. The chosen RKHS function class is endowed with a meaningful norm which is often useful (capturing our preferred notion of "smoothness" of functions) and one would not normally want to remetrize the function space (as one would then have a different RKHS). While steepest ascent generally takes the policy gradient w.r.t. the Euclidean metric on parameter space and the natural gradient algorithm takes the policy gradient w.r.t. a sensible distance measure between the induced trajectory *distributions*, we take the policy gradient w.r.t. a sensible metric between the *functions* defining policies. Choosing a kernel on moderately sized state spaces which induces a useful norm on functions can be more straightforward than remetrizing a high dimensional, abstract feature space.

Non-parametric methods using reproducing kernel Hilbert spaces of functions are of fundamental importance throughout machine learning (Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002) and it is surprising the functional policy gradient in an RKHS has not been thoroughly investigated. The specific contributions of this paper are to develop the framework for learning policies in vector-valued reproducing kernel Hilbert spaces, deriving the functional policy gradient analytically, which has a particularly simple form and can be computed efficiently. We develop sparsification tools for representing complex policies compactly and adaptively. We discuss compatible function approximation. We provide an 'actor-critic' algorithm for policy optimization in an RKHS. We analyze connections to gradient preconditioning methods.

We compare the approach to existing alternative methods on benchmark problems and a challenging simulated quadrocopter navigation task.

1.1 Prior Work

Modelling policies in a RKHS was first considered in Bagnell and Schneider (2003); Bagnell (2004), though the policy takes a different form. We compare the two approaches in detail in Section 3 and experimentally.

1.2 Background

1.2.1 Markov Decision Processes

We recall some basic concepts associated with reinforcement learning and control problems. In RL an agent acts in an environment by sequentially choosing actions over a sequence of time steps, in order to maximize a cumulative reward. We model this as a *Markov decision process* (MDP) which comprises: a *state space* $\mathcal{S}$, an *action space* $\mathcal{A}$ and we denote the state-action space $\mathcal{Z} := \mathcal{S} \times \mathcal{A}$, with $z = (s, a)$, $z_t = (s_t, a_t)$ etc., an *initial state distribution*, with density $p_1(s_1)$, a stationary *transition dynamics distribution* with conditional density[2] $p(s_{t+1}|Z_t = z_t)$ satisfying the Markov property $p(s_{t+1}|Z_i = z_i, \forall i \leq t) = p(s_{t+1}|Z_t = z_t)$, and a *reward function* $r : \mathcal{Z} \to \mathbb{R}$. Given an MDP an *agent* controls a *trajectory* $\xi := (z_1, z_2, ...)$ over $\mathcal{Z}$ by sequentially selecting actions $a_t \in \mathcal{A}$ at each time step according to a chosen stationary stochastic *policy* $\pi_w : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, where $\mathcal{P}(\mathcal{A})$ is the set of probability measures on $\mathcal{A}$ and $w \in \mathcal{W}$ is a parameterizing variable for policies. (Overloading notation) we denote by $\pi_w(a_t|s_t)$ the conditional probability density at a_t associated to the policy π_w. The agent's goal is to obtain a policy which maximizes the cumulative discounted reward,

$$U(w) := U(\pi_w) := \mathbb{E}[r_\gamma(\xi)] \tag{1}$$
$$= \int p(\xi; \pi_w) \sum_{t=1}^{\infty} \gamma^{t-1} r(z_t) \mathrm{d}\xi,$$

where $p(\xi; \pi_w)$ is the distribution over trajectories $z_1, z_2, ...$ when following π_w and $r_\gamma(\xi) := \sum_{t=1}^{\infty} \gamma^{t-1} r(z_t)$.

The optimization methods we focus on in this work are the policy gradient methods. These are iterative schemes which (in the most basic form), given a current policy π_w, suggest to increment the parameter w in the direction of steepest ascent of $U(\pi_w)$:

$$w \leftarrow w + \eta \nabla_w U(\pi_w), \tag{2}$$

for some $\eta \in \mathbb{R}$. We can write the update direction as

$$\nabla_w U(\pi_w) = \int p(\xi; w) r_\gamma(\xi) \nabla_w \log p(\xi; w) \mathrm{d}\xi$$
$$= \int p(\xi; w) r_\gamma(\xi) \sum_{t=1}^{\infty} \nabla_w \log \pi_w(a_t|s_t) \mathrm{d}\xi. \tag{3}$$

The policy gradient theorem (Sutton et al., 1999) implies that

$$\nabla_w U(\pi_w) = \int \rho_w(z) Q^{\pi_w}(z) \nabla_w \log \pi_w(a|s) \mathrm{d}z \tag{4}$$

where $Q^{\pi_w}(z) := \sum_{t=1}^{\infty} \mathbb{E}_{Z_t \sim p_t(\cdot|Z_1 = z)}[\gamma^{t-1} r(Z_t)]$ and $\rho_w(z) := \sum_{t=1}^{\infty} \gamma^{t-1} p_t(z; w)$ and where $p_t(z; w)$ is the marginal density over a single state-action pair at time point t following π_w.

1.2.2 (Vector-valued) Reproducing Kernel Hilbert Spaces

We will model functions $h : \mathcal{S} \to \mathcal{A}$ as elements of a reproducing kernel Hilbert space (RKHS). Since the action

[2] When it is clear from context we will drop the random variable in the conditional density and write $p(s_{t+1}|z_t) = p(s_{t+1}|Z_t = z_t)$ for convenience.

space $\mathcal{A}$ might in general be a vector space of dimension greater than 1 we will need the framework of vector-valued RKHS, which are sets of functions mapping an input space $\mathcal{X}$ to a vector space $\mathcal{Y}$. These spaces are the natural generalization of real-valued RKHSs, which is the special case $\mathcal{Y} = \mathbb{R}$.

Following Micchelli and Pontil (2005): let $\mathcal{X}$ be a set and $\mathcal{Y}$ a Hilbert space with inner product $\langle \cdot, \cdot \rangle_{\mathcal{Y}}$. then a Hilbert space $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$ is an RKHS when, for every $y \in \mathcal{Y}$ and $x \in \mathcal{X}$, the linear functional $h \mapsto \langle y, h(x) \rangle_{\mathcal{Y}}$ is continuous. There is therefore (by the Reisz lemma) an element $K(x|y) \in \mathcal{H}$ such that for all $h \in \mathcal{H}$ it holds that $\langle y, h(x) \rangle_{\mathcal{Y}} = \langle K(x|y), h \rangle_{\mathcal{H}}$. This is called the "reproducing property" of K in $\mathcal{H}$. We denote $\mathcal{H} = \mathcal{H}_K$. Further, the inner product in $\mathcal{H}_K$ satisfies $\langle K(x|y), K(x'|y') \rangle_K = \langle y, K(x, x')y' \rangle_{\mathcal{Y}}$, where $K(x, x')y := K(x|y)(x') \in \mathcal{Y}$, and where the *kernel function* $K(x, x') : \mathcal{Y} \to \mathcal{Y}$ is in the space $\mathcal{L}(\mathcal{Y})$ of linear operators on $\mathcal{Y}$. We will generally prefer the notation $K(x, \cdot)y$ for the function $K(x|y)$. $\mathcal{H}_K$ is then the completion (w.r.t. its norm) of the linear span of the set $\{K(x|y) \ : \ x \in \mathcal{X}, y \in \mathcal{Y}\}$, and an $h \in \mathcal{H}_K$ is typically specified by an expansion (possibly infinite) of the form:

$$h(\cdot) = \sum_i K(x_i|y_i) = \sum_i K(x_i, \cdot)y_i. \tag{5}$$

Vector-valued RKHSs are typically specified by choosing the operator-valued kernel $K : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$, which defines the entire function class and its norm.

The special case that $\mathcal{Y} = \mathbb{R}$ with the Euclidean inner product corresponds to the well known setting of a real-valued RKHS (Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002), where for clarity $K(x, x')$ is a real number and $K(x|y) = yK(x, \cdot)$. Another simple special case is when $\mathcal{Y} = \mathbb{R}^n$, and $K(x, x') = \hat{K}(x, x')\boldsymbol{I}$, where $\hat{K}$ is a scalar-valued kernel and $\boldsymbol{I}$ the identity matrix. In general however, K can be operator-valued.

2 MODELLING POLICIES IN AN RKHS

The policies we consider in this work are stochastic Gaussian policies,

$$\pi_{h,\boldsymbol{\Sigma}}(a|s) := \frac{1}{Z} e^{-\frac{1}{2}(h(s)-a)^\top \boldsymbol{\Sigma}^{-1}(h(s)-a)}, \tag{6}$$

parameterized by deterministic functions, $h \in \mathcal{H}$, $h : \mathcal{S} \to \mathcal{A} \subseteq \mathbb{R}^m$, and a covariance $\boldsymbol{\Sigma}$ on $\mathbb{R}^m$. The function $h(\cdot)$ will be an element of an RKHS $\mathcal{H}_K$,

$$h(\cdot) = \sum_i K(s_i, \cdot)\alpha_i \in \mathcal{H}_K, \tag{7}$$

where $s_i \in \mathcal{S}$ and $\alpha_i \in \mathcal{A}$. Our primary focus is on optimizing $\pi_{h,\boldsymbol{\Sigma}}$ w.r.t. h, and for compactness often write $\pi_h = \pi_{h,\boldsymbol{\Sigma}}$ when clear from context. We suppose $\mathcal{S}$ is an arbitrary set but assume for simplicity that $\mathcal{A} = \mathbb{R}^m$. Since $\mathcal{A}$ will often have dimension greater than one, we require $\mathcal{H}_K$ to be a class of vector-valued RKHS functions (see Section 1.2.2). To specify the RKHS we pick a kernel $K : \mathcal{S} \times \mathcal{S} \to \mathcal{L}(\mathcal{A})$. The most basic choice of K is to pick $K(s, s') = \kappa(s, s')\boldsymbol{I}$, where κ is a scalar-valued kernel, but general linear operators $K(s, s')$ are covered by the theory: these would typically capture correlations between the action space. For example, the movement of joints is often highly correlated (consider grasping tasks), and the operator $K(s, s')$ could be used to encode prior knowledge of such correlations. Note that, because of our choice of $\mathcal{A}$, Z does not depend upon h.

2.1 Policy Gradients in RKHS

We now consider how to compute the steepest ascent direction of the return (4) w.r.t. h when the policy is modelled as in (6) and h is modelled non-parametrically in a (vector-valued) RKHS $\mathcal{H}_K \subseteq \mathcal{A}^{\mathcal{S}}$. Importantly, the gradient will be an entire function in $\mathcal{H}_K$. Recalling (4), to compute the steepest ascent direction we first need to compute the gradient of $\log \pi_{h,\boldsymbol{\Sigma}}(a|s)$ which is then integrated together with the Q-function. This will be a functional gradient and the notion of the Fréchet derivative[3] is sufficient for us.

Given the policy form (6) we have,

$$\log \pi_{h,\boldsymbol{\Sigma}}(a|s) = -\log Z - \frac{1}{2}(h(s) - a)^\top \boldsymbol{\Sigma}^{-1}(h(s) - a),$$

where $h \in \mathcal{H}_K$, $h : \mathcal{S} \to \mathcal{A} = \mathbb{R}^m$. The following result provides the gradient of the log policy with respect to the function h.

Claim 2.1. *The derivative of the map $f : \mathcal{H}_K \to \mathbb{R}$, $f : h \mapsto \log \pi_{h,\boldsymbol{\Sigma}}(a|s)$, at h, is the bounded linear map $Df|_h : \mathcal{H}_K \to \mathbb{R}$ defined by,*

$$\begin{aligned} Df|_h : g &\mapsto (a - h(s))^\top \boldsymbol{\Sigma}^{-1} g(s) \\ &= \langle K(s, \cdot)\boldsymbol{\Sigma}^{-1}(a - h(s)), g \rangle_K. \end{aligned} \tag{8}$$

Thus the direction of steepest ascent is the function,

$$\nabla_h(\log \pi_{h,\boldsymbol{\Sigma}}(a|s)) = K(s, \cdot)\boldsymbol{\Sigma}^{-1}(a - h(s)) \in \mathcal{H}_K. \tag{9}$$

Proof. We just check that (8) satisfies the definition of the

[3] The Fréchet derivative is the derivative for functions on a Banach space. Let $\mathcal{V}$ and $\mathcal{W}$ be Banach spaces, and $\mathcal{U} \subset \mathcal{V}$ be an open subset of $\mathcal{V}$. A function $f : \mathcal{U} \to \mathcal{W}$ is called Fréchet differentiable at $x \in \mathcal{U}$ if there exists a bounded linear operator $Df|_x : \mathcal{V} \to \mathcal{W}$ such that,

$$\lim_{r \to 0} \frac{\|f(x + r) - f(x) - Df|_x(r)\|_{\mathcal{W}}}{\|r\|_{\mathcal{V}}} = 0.$$

Fréchet derivative: we have that

$$f(h+g) = \log \pi_{h+g,\mathbf{\Sigma}}$$
$$= -\log Z - \frac{1}{2}(h(s)+g(s)-a)^\top \mathbf{\Sigma}^{-1}(h(s)+g(s)-a),$$

so with $Df|_h(g)$ defined as in (8) we have that

$$\begin{aligned}\frac{|f(h+g)-f(h)-Df|_h(g)|}{||g||_K} &= \frac{g(s)^\top \mathbf{\Sigma}^{-1} g(s)}{2||g||_K} \\ &= \frac{\langle g, K(s,\cdot)\mathbf{\Sigma}^{-1}g(s)\rangle_K}{2||g||_K} \\ &\leq \frac{||g||_K(\mathbf{\Sigma}^{-1}g(s))^\top K(s,s)\mathbf{\Sigma}^{-1}g(s)}{2||g||_K} \\ &= (\mathbf{\Sigma}^{-1}g(s))^\top K(s,s)\mathbf{\Sigma}^{-1}g(s)/2 \\ &\to 0, \end{aligned} \tag{10}$$

where the last line we took the limit as $g \to 0$ and the inequality is due to Cauchy-Schwarz. □

Substituting (9) into the integral (4), the ascent direction is therefore given by the integral,

$$\nabla_h U(\pi_h) = \int \rho_h(z) Q^{\pi_h}(z) K(s,\cdot)\mathbf{\Sigma}^{-1}(a-h(s))\mathrm{d}z. \tag{11}$$

Note that the steepest ascent direction is a function. To evaluate this integral we can estimate $Q^{\pi_h}(z)$ and approximate the integral by sampling T state-action pairs $\{z_k = (s_k, a_k)\}_{k=1}^T$ from $(1-\gamma)\rho_h(\cdot)$ and approximating with the sum[4]:

$$(1-\gamma)\nabla_h U(\pi_h)$$
$$\approx \sum_{k=1}^{T} Q^{\pi_h}(z_k) K(s_k,\cdot)\mathbf{\Sigma}^{-1}(a_k - h(s_k)) \in \mathcal{H}_K. \tag{12}$$

Note that the gradient is a function in $\mathcal{H}_K$ with an expansion of the form (7).

[4]We remark that in computing the gradient of the log policy we had to obtain the derivative of the point evaluation functional $\delta_s : h \mapsto h(s)$. The derivative of the point evaluation functional $h \mapsto \langle h, K(s,\cdot)\rangle$ in an RKHS can be identified with the element $K(s,\cdot)$ of the function space, ensuring the finite sum (12) is indeed an element of the RKHS and a valid direction to increment the policy. In many other function classes this would not be the case; in fact, the *defining property* of an RKHS is the fact that point evaluation is a continuous linear functional. i.e. for Hilbert function spaces that are not RKHS, replacing the policy gradient integral with a sum as in (12) would not be guaranteed to be an element of our function space, and we would not be able to approximate the integral as a sum, and remain in the function space. RKHS are precisely those Hilbert function spaces in which we can use the approximation (12).

2.2 Compact Representations Via Sparsification in the RKHS

Unlike a parametric model with a fixed size of the expansion, successively updating the policy (7) via (2) by adding increments (12) will lead to an increasingly complex function representation. This means we can model complex policies, but also that our policies will become slow to evaluate. We must therefore keep the size of expansion of our policy (7), and the gradient (12) under control. We achieve this by sparsifying these expansions i.e. given any $h(\cdot) = \sum_{i=1}^N K(s_i,\cdot)\alpha_i \in \mathcal{H}_K$, we seek a $\hat{h}(\cdot) = \sum_{i=1}^N K(s_i,\cdot)\beta_i \in \mathcal{H}_K$ where only $n \ll N$ of the β_i are non zero. This can be achieved efficiently using (a vector-valued version of) the kernel matching pursuit algorithm (Vincent and Bengio, 2002). This method adds basis features $K(s_i,\cdot)$, and coefficients β_i to the sparse expansion sequentially by greedily adding the feature which maximally reduces the error $\sum_{i\in\mathcal{I}} ||h(s_i) - \hat{h}(s_i)||_{\mathcal{A}}^2$ (where $\mathcal{I}$ is some subsample of $\{1,...,N\}$) at each stage. See Appendix B for a description of the vector-valued version of the method.

In fact this allows us to find compact policy representations *adaptively*: we can define a tolerance for matching pursuit so that basis features and weights will be added to the compact policy expansion only if the error is reduced by more than the tolerance level, so the expansion becomes complex if the policy we are fitting requires it, but otherwise remains compact. This is a distinct advantage over parametric versions of the policy (7) where the number of basis points would usually be a-priori fixed, which could be too small (meaning we cannot represent the optimal policy), or too large (slow to evaluate, and more difficult to evaluate gradients).

2.3 Compatible Function Approximation

To use the form of the policy gradient theorem (11) to compute an ascent direction, we must approximate the Q-function Q^{π_h}. Thus we now consider *compatible function approximation* (Sutton et al., 1999) in this setting; that is, we consider estimates $\hat{Q}^{\pi_h}$ of Q^{π_h} such that replacing Q^{π_h} with $\hat{Q}^{\pi_h}$ in (11) the integral still evaluates to the policy gradient.

Theorem 2.2, below, provides one way to achieve compatible function approximation in our setting. We first define our approximation class for Q^{π_h}: Given the vector-valued RKHS $\mathcal{H}_K \subseteq \mathcal{A}^{\mathcal{S}}$ used to model policies of the form (6), (7) and a function $h \in \mathcal{H}_K$, define the scalar-valued kernel $K_h : (\mathcal{S}\times\mathcal{A})\times(\mathcal{S}\times\mathcal{A}) \to \mathbb{R}$,

$$K_h((s,a),(s',a'))$$
$$:= \left(K(s,s')\mathbf{\Sigma}^{-1}(a-h(s))\right)^\top \mathbf{\Sigma}^{-1}(a'-h(s')), \tag{13}$$

and denote the RKHS associated to K_h by $\mathcal{H}_K^h \subset \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$.

We verify that $\mathcal{H}_K^h$ is indeed an RKHS; to do this it is sufficient to demonstrate the existence of a feature map $\phi : \mathcal{S} \times \mathcal{A} \to \mathcal{F}_\phi$, and an inner product $\langle \cdot, \cdot \rangle_{\mathcal{F}_\phi}$ in $\mathcal{F}_\phi$ such that $K_h((s,a),(s',a')) = \langle \phi(s,a), \phi(s',a') \rangle_{\mathcal{F}_\phi}$ (e.g. Schölkopf and Smola, 2002). Thus we define

$$\phi : (s,a) \mapsto K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s)) \in \mathcal{H}_K =: \mathcal{F}_\phi, \quad (14)$$

so that by the reproducing property of K in $\mathcal{H}_K$,

$$\begin{aligned}\langle \phi(s,a), \phi(s',a') \rangle_{\mathcal{F}_\phi} \\ &= \langle K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s)), K(s',\cdot)\mathbf{\Sigma}^{-1}(a' - h(s')) \rangle_{\mathcal{H}_K} \\ &= K_h((s,a),(s',a')),\end{aligned}$$

as required. Further, for any $q \in \mathcal{F}_\phi$ there exists a $w \in \mathcal{F}_\phi$ such that,

$$\begin{aligned} q(s,a) &= \langle w, \phi(s,a) \rangle_{\mathcal{F}_\phi} \\ &= \langle w, K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s)) \rangle_{\mathcal{H}_K}. \end{aligned} \quad (15)$$

We now demonstrate that $\mathcal{H}_K^h$ is a compatible approximation architecture:

Theorem 2.2. *Suppose that $\hat{Q}^{\pi_h} \in \mathcal{H}_K^h$ is such that,*

$$\hat{Q}^{\pi_h} = \underset{Q \in \mathcal{H}_K^h}{\operatorname{argmin}} \int \rho_h(z)(Q(z) - Q^{\pi_h}(z))^2 \mathrm{d}z. \quad (16)$$

Then $\hat{Q}^{\pi_h}$ is a compatible function approximator for Q^{π_h}, i.e.

$$\nabla_h U(\pi_h) = \int \rho_h(z)\hat{Q}^{\pi_h}(z)K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s))\mathrm{d}z. \quad (17)$$

Proof. From (15) there exists a $w_h \in \mathcal{H}_K$ such that,

$$\hat{Q}^{\pi_h}(s,a) = \langle w_h, K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s)) \rangle_{\mathcal{H}_K}. \quad (18)$$

Therefore $\nabla_{w_h}\hat{Q}^{\pi_h}(s,a) = K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s)) \in \mathcal{H}_K$. And now from (16),

$$\begin{aligned} 0 &= \int \rho_h(z)(\hat{Q}^{\pi_h}(z) - Q^{\pi_h}(z))\nabla_{w_h}\hat{Q}^{\pi_h}(z)\mathrm{d}z \\ &= \int \rho_h(z)(\hat{Q}^{\pi_h}(z) - Q^{\pi_h}(z))K(s,\cdot)\mathbf{\Sigma}^{-1}(a - h(s))\mathrm{d}z \end{aligned}$$

which implies (17). □

Thus, compatible function approximation can be achieved by using kernel regression to estimate Q^{π_h} using the *compatible kernel* (13). Many practical methods exist to do this and in our experiments we choose efficient kernel matching pursuit algorithm (Vincent and Bengio, 2002) to regress $\hat{Q}^{\pi_h}$ quickly in the desired RKHS $\mathcal{H}_K^h$.

2.4 An Actor-Critic Algorithm with RKHS Policies

We now combine these elements to present a full practical policy gradient scheme based on modelling policies in an RKHS: the algorithm pseudocode is given in Algorithm 1.

Algorithm 1 Compatible RKHS Actor-Critic

Input: MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, r, P\}$; kernel $K : \mathcal{S} \times \mathcal{S} \to \mathcal{L}(\mathcal{A})$; initial covariance $\mathbf{\Sigma}_1$
Initialize: $h_1 = 0$
Parameters: tolerance δ
for $j = 1, 2, ...$ **do**
  Collect data from the system using policy π_{h_j}
  Compatible Q approximation: Fit $Q^{\pi_{h_j}}$ with $\hat{Q}^{\pi_{h_j}}$ in $\mathcal{H}_K^h$ as in Section 2.3
  Compute policy gradient: Approximate the policy gradient $\nabla_h U(\pi_{h_j,\mathbf{\Sigma}_j})$ using (12)
  Update policy: $h_{j+1} = h_j + \eta_j \nabla_{h_j} U(\pi_{h_j,\mathbf{\Sigma}_j})$ for step size η_j
  Sparsify policy: Sparsify $h_{j+1} \in \mathcal{H}_K$ with tolerance δ as in Section 2.2
  Update $\mathbf{\Sigma}$: reduce $\mathbf{\Sigma}_j$ (or take a gradient step w.r.t. $\mathbf{\Sigma}_j$)
end for

3 COMPARISON TO OTHER APPROACHES

We briefly discuss connections to a simple parametric version of our policy, an alternative method of modelling policies in an RKHS and to the natural gradient algorithm.

3.1 Comparison to the Parametric Case

We first compare to a similar *parametric* approach. Restricting to the real-valued RKHS function case, and putting $\mathcal{S} = \mathbb{R}^d$, for simplicity, a parametric approach might consider policies of the form (6) but where the function h is given as a linear function over features $\phi_i(\cdot)$, where

$$\begin{aligned} h(s) &= \sum_{i=1}^{n} w_i \phi_i(s) \\ \phi_i(s) &= K(s, c_i). \end{aligned} \quad (19)$$

Rather than take the gradient in the RKHS $\mathcal{H}_K$ the standard approach would be to optimize the weights $\boldsymbol{w}$ and centres $\{c_i\}$. Putting $\theta = (\boldsymbol{w}, \{c_i\}) \in \mathbb{R}^{n+dn}$ (with a Euclidean norm) it is straightforward to compute the necessary derivatives with respect to θ (assuming the kernel is differentiable), but recalling the discussion of Section 1.2.1, there is no obvious way to choose a scaling of these parameters, whereas modelling everything directly in an RKHS essentially fixes the scaling in a principled way by adopting the RKHS norm on functions as that determining the scaling. Secondly, in high dimensional state spaces (consider a state representation of images with thousands of pixels, for instance) the size of the parameter vector would become huge, making scaling and preconditiong difficult and computationally expensive, but in such spaces it is vital to optimize the position of centres as a small number of

well positioned centres would be hard to choose a-priori. Thirdly, it is not necessarily easy to choose n a-priori, picking n too large would be computationally expensive (especially if we precondition, see Section 3.3), too small could mean we are unable to fit important policies. Recalling the discussion in Section 2.2, we can fit to the complexity of our problem adaptively. A final key issue with working parametrically is that the ideal position of centres must be reachable from the initialization by continuous repositioning since they are updated by following a gradient direction *through state space*. This could be a particular problem in high dimensional complex state spaces where the ideal position of centres might be unreachable without passing through poor positions, and gradually repositioning centres could be slow. When parameterizing in an RKHS we do not have this problem: we do not rely on repositioning and new centres can immediately emerge anywhere on the support of our current trajectory through state space.

3.2 Comparison to Prior Work

An alternative policy form, explored in Bagnell and Schneider (2003); Bagnell (2004)[5] is

$$\pi_f(a|s) := \frac{1}{Z} e^{f(s,a)}, \tag{20}$$

where $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is an element of an RKHS on $\mathcal{S} \times \mathcal{A}$,

$$f(s,a) = \sum_{i=1}^{n} \alpha_i K((s_i, a_i), (s, a)). \tag{21}$$

The policy gradient is (see Appendix A)

$$\frac{\nabla_f \pi_f(a|s)}{\pi_f(a|s)} = K((a,s),\cdot) - \mathbb{E}_{A\sim\pi_f(\cdot|s)}[K((A,s),\cdot)].$$

Contrasting (6) and (20), the latter is in one sense more general than (6) – distributions that are not Gaussians can be modelled. However, in MDPs, optimal deterministic policies always exist, so more general distributions are not needed to model the optimal policy. Indeed an optimal policy which is a deterministic function may be difficult to learn under (20) since it would require, for each s, the map $a \mapsto f(s,a)$ to be a delta peak in the action space, which would usually not correspond to an RKHS function f, or be easily approximated by an RKHS function with a compact expansion (it would be a very complex function requiring a large expansion of the form (21) and slow to evaluate). Under (6), to learn a deterministic function we simply decay $\mathbf{\Sigma}$ towards zero. Indeed, experimentally we observe that we need a much larger expansion of (21) to learn well than in the method we propose. Note also that the gradient of the log policy contains an expectation, which we must sample in order to estimate, which is both expensive and may lead to inaccuracies.

[5] Strictly speaking they consider POMDPs and so the kernel is defined on observation space.

3.3 Comparison with Natural Gradients

For a normed parameter space $\mathcal{W}$, the steepest ascent direction $\nabla_w U(\pi_w)$ is not invariant to rescaling of individual components of the parameter vector w. This arises since

$$\frac{\nabla_w U(w)}{||\nabla_w U(w)||} = \underset{v:||v||=1}{\operatorname{argsup}} \lim_{\epsilon\to 0} \frac{U(w+\epsilon v) - U(w)}{\epsilon} \tag{22}$$

thus the direction of steepest ascent of (1) depends upon the choice of norm $||\cdot||$ on the parameter space, and in particular *prefers lower norm directions*: if a rescaling decreases the norm in a particular direction, that direction will feature more in the gradient. This causes problems such as the plateau effect, when the objective function (1) contains ridges, and gradient ascent will zig-zag.

One popular approach to remedy this is to take the natural gradient (Kakade, 2001) in which the parameter space is remetrized to measure a discrepancy between the policy *distributions* under different parameters – and this is invariant to rescaling parameters. In the approach we present gradients are functional gradients taken in the Hilbert function space – w in (22) is an RKHS function – and thus the gradient direction is affected by the Hilbert space norm over functions parameterizing our policies. In fact the RKHS norm encodes a notion of "smoothness" of functions, which is generally useful throughout machine learning for expressing a preference for smooth functions via regularization. Here, the functional gradient will prefer smoother policies in the policy space, in the sense specified by the RKHS norm. If this norm is sensible we can avoid the post-hoc remetrization using a preconditioner, which can be expensive in large parameter spaces. Choosing a suitable kernel on a state space, inducing a sensible norm on the RKHS, can be quite straightforward compared to remetrizing a high dimensional feature space. For example, state spaces might be lower dimensional, and practitioner may have good intuition for sensible kernels; in fact many kernels specifically tailored to the application domain may exist.

4 EXPERIMENTS

We present results for four MDPs; a synthetic **Toy** task, the familiar benchmarks **Mountain Car**, **Inverted Pendulum** and a simulated **Quadrocopter UAV** navigation task. We define $\mathcal{R} \in [0,1]$ in all the reward functions and discount factor $\gamma = 0.99$ across all tasks. Average cumulative rewards for each task are presented for each iteration of policy improvement.

Our proposed RKHS actor-critic controller, *RKHS2* (Section 2.4), is presented along with existing comparisons: a parametric controller *Para* (Section 3.1 with weight gradient updates); a parametric controller with a natural gradient pre-conditioner *Nat* (Section 3.3 with weight gradient up-

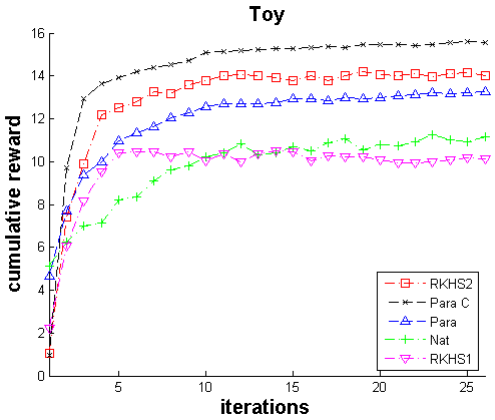

Figure 1: Toy cumulative reward, $H = 20$

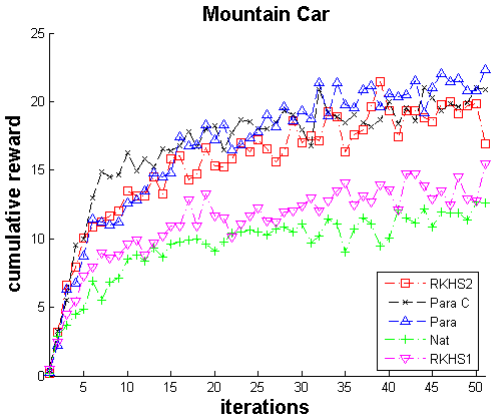

Figure 2: Mountain car cumulative reward, $H = 100$

dates); a parametric controller with adjustable centres *Para C* (Section 3.1 with weight and centre updates), and the existing RKHS controller (section 3.2) *RKHS1*. Each controller is a compatible actor-critic of the same generic form as Algorithm 1. Step sizes η_j were chosen by performing a line search over a grid of 50 candidates. For fairness of comparison we use the same kernels (Gaussians) and the same number of centres in each controller (we use a number of centres sufficient for these controllers to generally achieve the tasks), and the same number of samples were used to estimate the Q-function. We restrict the maximum number of centres in the RKHS methods to be equal to the parametric versions, however in practice one would set the number of centres adaptively as in Section 2.2. For the two RKHS controllers we sparsified the gradient before performing the line search for the step size (which is more efficient than sparsifying every policy) – with a budget of centres we found it was best to sparsify the gradient more than the policy. Where appropriate each policy is randomly initialised.

4.1 Toy, Mountain Car and Pendulum Benchmarks

The Toy benchmark is a Markov chain on interval $\mathcal{S} \in [-4, 4]$ where $\mathcal{A} \in [-1, 1]$ and $r(s, a) = e^{-|s-3|}$. The dynamics are $s' = s + a + \epsilon$ where ϵ is small amount of Gaussian noise, and the agent is reset to $s = 0$ if it leaves the interval. Over a horizon of $H = 20$ optimal return is around 14, doing nothing receives almost no reward. The controllers used 20 centres. Fig. 1 shows results averaged over 50 experiments.

The second benchmark is the Mountain Car problem (see e.g. Singh and Sutton, 1996). The agent controls a car located at the bottom of a valley and the objective is to drive to the top of a hill but does not have enough power to achieve this directly and must therefore climb the opposite hill a short distance before accelerating toward the goal. States $s = (x, v)$ are position and velocity, $\mathcal{S} = (-1.2, 0.7) \times (-0.07, 0.07)$, $\mathcal{A} = [-1, 1]$ and $r(s, a) = e^{-8(x-0.6)^2}$ and $s_0 = (-0.5, 0)$. Dynamics are $x' = x + v + \epsilon_1$, $v' = v + 0.001a - 0.0025\cos(3x) + \epsilon_2/10$, where ϵ_1, ϵ_2 are Gaussian random variables with standard deviation 0.02. For a horizon $H = 100$ optimal return is around 25. Doing nothing receives almost no reward. The controllers used 50 centres. Results shown in Fig. 2 are averaged over 30 experiments.

The third benchmark is the under-actuated Inverted Pendulum swing-up problem. An agent applies torque $\mathcal{A} = [-3, 3]$ to a pendulum where states $s = (\theta, \omega)$ are angular position and velocity, $\Theta = [-\pi, \pi)$ (angle from the vertical) and $\Omega = [-4\pi, 4\pi]$. Reward is $r(s, a) = e^{-0.5\theta^2}$ and $s_0 = (-\pi, 0)$. Dynamics are $\theta' = \theta + \omega\Delta t + \epsilon_1$ and $\omega' = \omega + \dot{\omega}\Delta t + 2\epsilon_2$, $\Delta t = 0.05$ and ϵ_1, ϵ_2 are Gaussian random variables with standard deviation 0.02. We transitioned twice using the dynamics with the current action before allowing the policy to select its next action, therefore one MDP transition is equal to twice the application of the dynamics equations. Over a horizon $H = 400$ optimal reward is around 80, anything over 60 means the pendulum is kept balanced and doing nothing receives almost no reward. The controllers used 100 centres. Results shown in Fig. 3 are averaged over 15 experiments for only four controllers that were practical in this slightly more difficult task: we abandoned the existing *RKHS1* controller due to a high computational expense of selecting actions.

The proposed RKHS actor-critic algorithm is competitive with the best algorithms on each benchmark.

4.2 Quadrocotper UAV

The fourth experiment is a Quadrocotper navigation task which uses a simulator (De Nardi, 2013) specifically calibrated to model the dynamics of a single or multiple Pelican^{TM} platforms. We restrict our experiment to a single platform which is a higher dimensional problem,

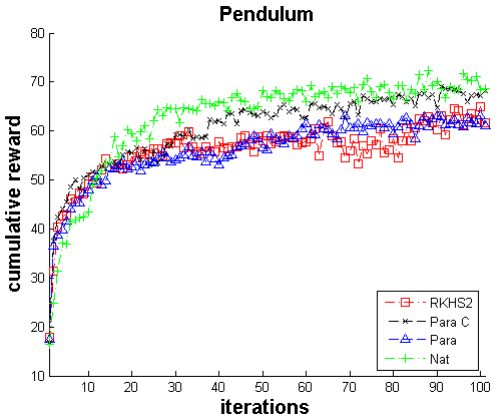

Figure 3: Inverted Pendulum cumulative reward, $H = 400$

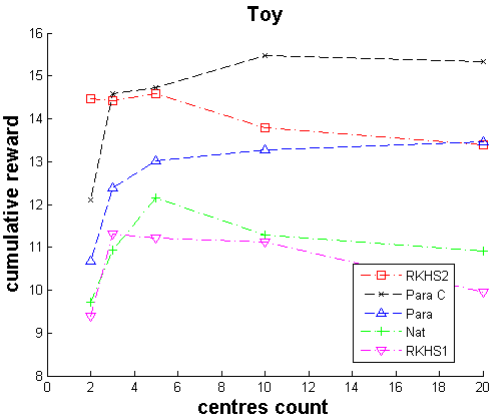

Figure 5: Toy experiments with fewer centres

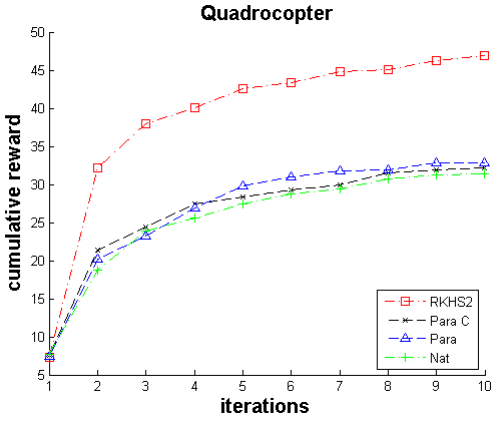

Figure 4: UAV navigation cumulative reward, $H = 50$

$\mathcal{S} \subset \mathbb{R}^{13}$, compared to the previous tasks. Explicitly $s = (x, y, z, \theta\, \phi, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\theta}, \dot{\phi}, \dot{\psi}, F)$ which consists of platform position $s_{xyz} \in \mathbb{R}^3$, platform roll, pitch and yaw $s_{\theta\phi\psi} \in \mathbb{R}^3$, associated time derivatives and the thrust applied to all rotors $F \in \mathbb{R}$. The action space is three dimensional $\mathcal{A} \subset \mathbb{R}^3$ which represent desired velocity vectors for the platform. The simulator mimics the architecture of a real platform such that a PID controller receives these desired velocities at the agent's rate and translates them into low level commands issued directly to the rotor blades at a rate of about $50Hz$, in attempt to attain those velocities, which creates complex dynamics for the system. A target location is defined at coordinates x_{targ} such that we define $r(s, a) = e^{-\frac{1}{2\sigma^2}||x_{targ} - s_{xyz}||^2}$. The controllers used 100 centres.

All existing controllers are unable to compete with the proposed *RKHS2* on this high dimensional task – see Fig. 4, averaged over 25 experiments. One key possible advantage the proposed controller has here is its ability to immediately add new centres anywhere along its trajectory support, rather than by moving existing centres, which may be an advantage on this higher dimensional task.

4.3 Learning with Fewer Centres

Finally we consider the effect of reducing the budget of centres in the Toy MDP. Results over a range of centres are shown in Fig. 5. As expected the RKHS actor-critic does not degrade in performance as the number of centres is reduced, whereas the other controllers can degrade significantly. This suggests that the RKHS parameterization is able to identify optimal basis points well.

5 CONCLUSIONS AND EXTENSIONS

We have presented a method of modelling policies in RKHS function spaces, allowing us to work non-parametrically in policy gradient methods in reinforcement learning, with rich representation potential and without the need to a-priori specify parameters. We have derived the policy gradient for our method which is en entire function. We have demonstrated how the representation of the policy can be adaptive to the problem at hand using efficient sparse approximation techniques, and have demonstrated how to perform compatible Q estimation in our framework by deriving the *compatible approximation kernel*. We have verified the method against competitor methods in benchmark domains and on a simulated quadrocopter navigation task. The approach seems to have an advantage in high dimensional state spaces where picking good basis points to represent policies a-priori can be difficult: rather than gradually moving around a-priori chosen basis points our method provides a principled way to add basis points along experienced trajectories.

Acknowledgements

This work was supported by the European Community Seventh Framework Programme (FP7/2007-2013) under grant agreement 270327 (CompLACS).

References

Bagnell, J. (2004). *Learning Decisions: Robustness, Uncertainty and Approximations*. PhD thesis, Carnegie Mellon University.

Bagnell, J. A. D. and Schneider, J. (2003). Policy search in reproducing kernel hilbert space. Technical Report CMU-RI-TR-03-45, Robotics Institute, Pittsburgh, PA.

De Nardi, R. (2013). The qrsim quadrotor simulator. Technical Report RN/13/08, Department of Computer Science, University College London, Gower Street, London UK.

Deisenroth, M. P. and Rasmussen, C. E. (2011). Pilco: A model-based and data-efficient approach to policy search. In *ICML*, pages 465–472.

Heess, N., Silver, D., and Teh, Y. W. (2012). Actor-critic reinforcement learning with energy-based policies. In *Proceedings of the Tenth European Workshop on Reinforcement Learning, EWRL 2012, Edinburgh, Scotland, June, 2012*, pages 43–58.

Kakade, S. (2001). A natural policy gradient. In *NIPS*, pages 1531–1538.

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

Kohl, N. and Stone, P. (2004). Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA*, pages 2619–2624.

Mallat, S. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415.

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

Schölkopf, B. and Smola, A. (2002). *Learning With Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. Adaptive Computation and Machine Learning Series. Mit Press.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 387–395.

Singh, S. P. and Sutton, R. S. (1996). Reinforcement learning with replacing eligibility traces. *Machine Learning*, 22(1-3):123–158.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063.

Vincent, P. and Bengio, Y. (2002). Kernel matching pursuit. *Machine Learning*, 48(1-3):165–187.

Wierstra, D., Förster, A., Peters, J., and Schmidhuber, J. (2010). Recurrent policy gradients. *Logic Journal of the IGPL*, 18(5):620–634.