Non-Parametric Policy Learning for High-Dimensional State Representations

Herke van Hoof[+] HOOF@IAS.TU-DARMSTADT.DE
Gerhard Neumann[+] NEUMANN@IAS.TU-DARMSTADT.DE
Jan Peters[+*] PETERS@IAS.TU-DARMSTADT.DE
[+] *Intelligent Autonomous Systems Institute, TU Darmstadt, Darmstadt, Germany*
[*] *Max Planck Institute for Intelligent Systems, Tübingen, Germany*

Editor:

Abstract

Learning complex control policies from high-dimensional sensory input is a challenge for reinforcement learning algorithms. Non-parametric methods can help to address this problem, but many current approaches rely on unstable greedy maximization. In this paper, we develop a kernel-based reinforcement learning algorithm that performs robust policy updates. We show that our method outperforms related approaches, and is able to learn an underpowered swing-up task task directly from high-dimensional image data.

1. Introduction

Learning continuous valued control policies directly from high-dimensional sensory input presents a major obstacle to applying reinforcement learning (RL) methods effectively in realistic settings. Current approaches for continuous domains typically rely on human-designed features for value function approximations or specialized parametric policies.

Furthermore, many methods require a transition model, which, in the general case, is unknown. A recent approach (Grünewälder et al., 2012) embeds the transition distributions in a reproducing kernel Hilbert space (RKHS). This allows efficient estimation of expected values and avoids the need to define features. Such RKHS embeddings have been successfully used in value-function methods (Grünewälder et al., 2012). These methods update the policy greedily with respect to the approximated value function, which can cause instabilities in the learning progress. Such instabilities can be avoided by bounding the information loss between subsequent state-action distributions (Peters et al., 2010).

In this paper[1], we employ kernel-based methods to find bounded policy updates according to the information-theoretic objective proposed in Peters et al. (2010). This method contributes *a more stable learning progress* for non-parametric reinforcement learning. Simultaneously, it *avoids the use of hand-crafted features.*

The resulting robust RL method performs well in continuous state-action spaces with high-dimensional sensory representations. In our experiments, we show that our method outperforms relevant baselines on a reaching task and an underpowered swing-up task. We also show it is applicable to a task with high dimensional pixel images state representation.

1. This paper is a short version of van Hoof et al. (2015).

1.1 Notation and Background

In a Markov decision process (MDP), an agent in state $\mathbf{s}$ selects an action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$ according to a (possibly stochastic) policy π and receives a reward $\mathcal{R}_{\mathbf{s}}^{\mathbf{a}} \in \mathbb{R}$. We will assume continuous state-action spaces: $\mathbf{s} \in \mathcal{S} = \mathbb{R}^{D_\mathrm{s}}$, $\mathbf{a} \in \mathcal{A} = \mathbb{R}^{D_\mathrm{a}}$. If the Markov decision process is ergodic, for each policy π, there exists a stationary distribution $\mu_\pi(\mathbf{s})$ such that $\int_{\mathcal{S}}\int_{\mathcal{A}} \mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(s)\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s} = \mu_\pi(\mathbf{s}')$, where $\mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}} = Pr(\mathbf{s}'|\mathbf{a},\mathbf{s})$. The goal of a reinforcement learning agent is to choose a policy such that the joint state-action distribution $p(\mathbf{s},\mathbf{a}) = \mu_\pi(\mathbf{s})\pi(\mathbf{a}|\mathbf{s})$ maximizes the average reward $J(\pi) = \int_{\mathcal{S}}\int_{\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})\mathcal{R}_{\mathbf{s}}^{\mathbf{a}}\mathrm{d}\mathbf{a}\mathrm{d}\mathbf{s}$.

To avoid overly greedy optimization and instabilities in the learning process, the Kullback-Leibler (KL) divergence of a sampling distribution $q(\mathbf{s},\mathbf{a})$ from the state-action distribution $p(\mathbf{s},\mathbf{a})$ can be bounded (Peters et al., 2010), leading to the sample-based solution

$$\pi(\mathbf{a}_i|\mathbf{s}_i)\mu_\pi(\mathbf{s}_i) \propto q(\mathbf{s}_i,\mathbf{a}_i)\exp\left(\frac{\delta(\mathbf{s}_i,\mathbf{a}_i,V)}{\eta}\right), \quad \delta(\mathbf{s},\mathbf{a},V) = \mathcal{R}_{\mathbf{s}}^{\mathbf{a}} + \mathbb{E}_{\mathbf{s}'}[V(\mathbf{s}')|\mathbf{s},\mathbf{a}] - V(\mathbf{s}) \tag{1}$$

where $V(\mathbf{s})$ and η denote Lagrangian multipliers, and δ denotes the Bellman error (Peters et al., 2010). These multipliers are obtained through minimization of the dual function

$$g(\eta,V) = \eta\epsilon + \eta\log\left(\sum_{i=1}^{n}\frac{1}{n}\exp\left(\frac{\delta(\mathbf{s}_i,\mathbf{a}_i,V)}{\eta}\right)\right), \quad (\mathbf{s}_i,\mathbf{a}_i) \sim q(\mathbf{s},\mathbf{a}) \tag{2}$$

The Lagrangian multiplier $V(\mathbf{s})$ is a function of $\mathbf{s}$ and resembles a value function. To calculate the Bellman error δ, the transition distribution is required. As this distribution is generally not known, δ needs to be approximated. The dual function (2) depends implicitly on reference distribution q through the samples.

1.2 Problem Statement

In this paper, we aim at developing a method applicable in continuous state-action MDPs with high-dimensional state representations. We assume hand-coded feature functions and parametric policies are not available and the transition and reward models of the MDP are unknown. Furthermore, we will concentrate on infinite-horizon problems.

2. Using Conditional Embeddings to solve Continuous MDPs

In this section, first, we show how to solve this optimization problem with respect to a non-linear function V. Then, we show how to use conditional embeddings to approximate the expectated values in Eq. (1).

Functional form of V. We need to minimize the dual problem in (2): $(\mu^*, V^*) = \arg\max g(\eta, V)$. We will assume that $V^* \in \mathcal{F}$ for some reproducing kernel Hilbert space (RKHS) $\mathcal{F}$ with kernel k_s, i.e., we assume V^* is of the form

$$V^* = \sum_{\tilde{\mathbf{s}}\in\tilde{\mathcal{S}}} \alpha_{\tilde{\mathbf{s}}} k_\mathrm{s}(\tilde{\mathbf{s}},\cdot), \tag{3}$$

for some set $\tilde{\mathcal{S}}$ and scalars α. The kernel k_s implicitly defines a feature map $\boldsymbol{\phi}(\mathbf{s}) = k_\mathrm{s}(\mathbf{s},\cdot)$. The implicit definition has the advantage that we do not need to explicitly specify a feature vector. Instead, we can use a kernel that works in a large number of tasks.

Embedding the transition model. Since the transition model $\mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}}$ is unknown, we need to approximate $\mathbb{E}_{\mathbf{s}'}[V(\mathbf{s}')|\mathbf{s},\mathbf{a}]$. To do so efficiently we represent $\mathcal{P}_{\mathbf{ss}'}^{\mathbf{a}}$ by the expected implicit features $\boldsymbol{\mu}_{\mathbf{s}'|\mathbf{s},\mathbf{a}} = \mathbb{E}_{\mathbf{s}'}[\boldsymbol{\phi}(\mathbf{s}')|\mathbf{s},\mathbf{a}]$ (Song et al., 2013). In order to learn the conditional operator, we will use a kernel over the state-action space of the form $\boldsymbol{\psi}(\mathbf{s},\mathbf{a}) = k_{\mathrm{s}}(\mathbf{s},\cdot)k_{\mathrm{a}}(\mathbf{a},\cdot)$. Given a sample $\{(\mathbf{s}_1,\mathbf{a}_1,\mathbf{s}'_1),\ldots,(\mathbf{s}_n,\mathbf{a}_n,\mathbf{s}'_n)\}$, the empirical conditional embedding

$$\hat{\boldsymbol{\mu}}_{S'|s,a} = \sum_{i=1}^{n} \beta_i(\mathbf{s},\mathbf{a})\boldsymbol{\phi}(\mathbf{s}'_i), \quad \boldsymbol{\beta}(\mathbf{s}_i,\mathbf{a}_i) = (\mathbf{K}_{\mathrm{sa}} + \lambda I)^{-1}\mathbf{k}_{\mathrm{sa}}(\mathbf{s}_i,\mathbf{a}_i) \tag{4}$$

where $\boldsymbol{\beta}(\mathbf{s}_i,\mathbf{a}_i)$ are the learned conditional embedding strengths (Song et al., 2013), $\boldsymbol{\Psi} = [\boldsymbol{\psi}(\mathbf{s}_1,\mathbf{a}_1),\ldots,\boldsymbol{\psi}(\mathbf{s}_n,\mathbf{a}_n)]$, $\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{s}'_1),\ldots,\boldsymbol{\phi}(\mathbf{s}'_n)]$, $\mathbf{K}_{\mathrm{sa}} = \boldsymbol{\Psi}^T\boldsymbol{\Psi}$ and $\mathbf{k}_{\mathrm{sa}}(\mathbf{s},\mathbf{a}) = \boldsymbol{\Psi}^T\boldsymbol{\psi}(\mathbf{s},\mathbf{a})$.

Optimizing the dual Since k_{s} is a reproducing kernel and $V \in \mathcal{F}$, the expected value of V can be approximated using the embedded distribution (Song et al., 2013), i.e.,

$$\mathbb{E}_{\mathbf{s}'}[V(\mathbf{s}')|\mathbf{s},\mathbf{a}] = \left\langle V, \hat{\boldsymbol{\mu}}_{S'|s,a}\right\rangle_{\mathcal{F}} = \sum_{i=1}^{n}\beta_i(\mathbf{s},\mathbf{a})V(\mathbf{s}'_i).$$

In the dual g (Eq. 2), V is now only evaluated at sampled s_i and s'_i. The generalized representer theorem tells us that there is at least one optimum of the form (3) with $\tilde{\mathcal{S}}$ the set of sampled states. Consequently, $\mathbb{E}_{\mathbf{s}'|\mathbf{s},\mathbf{a}}[V(\mathbf{s}')] - V(\mathbf{s}) = \boldsymbol{\alpha}^T\tilde{\mathbf{K}}_{\mathrm{s}}\boldsymbol{\beta}(\mathbf{s},\mathbf{a}) - \boldsymbol{\alpha}^T\mathbf{k}_{\mathrm{s}(\mathbf{s})}$, where $\tilde{\mathbf{K}}_{\mathrm{s}}$ is the Gram matrix with entries $[\tilde{\mathbf{K}}_{\mathrm{s}}]_{ji} = k_{\mathrm{s}}(\tilde{\mathbf{s}}_j,\mathbf{s}'_i)$, and $[\mathbf{k}_s(\mathbf{s})]_j = k_{\mathrm{s}}(\tilde{\mathbf{s}}_j,\mathbf{s})$. The dual problem, which is convex in $\boldsymbol{\alpha}$, can now be stated as

$$g(\eta,\boldsymbol{\alpha}) = \eta\epsilon + \eta\log\left(\sum_{i=1}^{n}\frac{\exp\left(\frac{\delta(\mathbf{s}_i,\mathbf{a}_i,\boldsymbol{\alpha})}{\eta}\right)}{n}\right), \delta(\mathbf{s},\mathbf{a},\boldsymbol{\alpha}) = \mathcal{R}_{\mathbf{s}}^{\mathbf{a}} + \boldsymbol{\alpha}^T\left(\tilde{\mathbf{K}}_{\mathrm{s}}\boldsymbol{\beta}(\mathbf{s},\mathbf{a}) - \mathbf{k}_{\mathrm{s}}(\mathbf{s})\right), \tag{5}$$

We optimize for η and $\boldsymbol{\alpha}$ using standard second order techniques. Note that, if we choose kernels $\mathbf{k}_{\mathrm{s}}(\mathbf{s}_i,\mathbf{s}_j) = \tilde{\phi}(\mathbf{s}_i)^T\tilde{\phi}(\mathbf{s}_j)$ and $\mathbf{k}_{\mathrm{sa}}((\mathbf{s}_i,\mathbf{a}_i),(\mathbf{s}_j,\mathbf{a}_j)) = \mathbb{I}((\mathbf{s}_i,\mathbf{a}_i) = (\mathbf{s}_j,\mathbf{a}_j))$, we obtain the original REPS formulation by Peters et al. (2010) as a special case. In these equations, $\mathbb{I}$ is the indicator function and $\tilde{\phi}$ is a set of hand-crafted features.

Generalizing the Sample-Based Policy The parameters resulting from the optimization, η and $\boldsymbol{\alpha}$, can be inserted back in (5) to yield the desired probabilities $p(\mathbf{s}_i,\mathbf{a}_i)$. Since states and actions are continuous, we need to generalize from these weighted samples to nearby data points. To this end, we fit a generalizing stochastic policy $\tilde{\pi}(\mathbf{a}|\mathbf{s})$ by minimizing the expected Kullback-Leibler divergence $\mathbb{E}_{\mathbf{s}}\left[\mathrm{KL}(\pi(\mathbf{a}|\mathbf{s})||\tilde{\pi}(\mathbf{a}|\mathbf{s}))\right]$ of the generalizing policy from the sample-based policy $\pi(\mathbf{a}|\mathbf{s})$. Due to its flexibility and its good performance on many practical problems, the cost-sensitive Gaussian processes introduced in the cost-regularized kernel regression (CrKR) algorithm (Kober et al., 2011) as generalizing policies. Algorithm 1 shows how the different steps of our approach fit together.

3. Related Work

RKHS embeddings have been used to model transition dynamics for reinforcement learning (Grünewälder et al., 2012). This method considers only discrete action sets and uses greedy

Algorithm 1 REPS with RKHS embeddings

```
for $i = 1, \dots, \text{max\_iteration}$ do
    generate rollouts according to $\tilde{\pi}_{i-1}$
    minimize kernel-based dual:                  $\eta^*, \boldsymbol{\alpha}^* \leftarrow \arg\min g(\eta, \boldsymbol{\alpha})$    Eq. 5
    calculate kernel embedding strengths:        $\boldsymbol{\beta}_j \leftarrow (\mathbf{K}_{\text{sa}} + \lambda \mathbf{I})^{-1} \mathbf{k}_{\text{sa}}(\mathbf{s}_j, \mathbf{a}_j)$    Eq. 4
    calculate kernel-based Bellman errors:       $\delta_j \leftarrow \mathcal{R}_j + \boldsymbol{\alpha}^{*T} \left( \tilde{\mathbf{K}}_{\text{s}} \boldsymbol{\beta}_j - \mathbf{k}_{\text{s}}(\mathbf{s}_j) \right)$    Eq. 5
    fit a generalizing non-parametric policy $\tilde{\pi}_i$
end for
```

maximization with respect to approximated value functions. Rawlik et al. (2013) presents a method that considers continuous actions, but assumes the environments injects observable control noise and that the system is control-affine. Another non-parametric method, proposed by Pazis and Parr (2011), assumes the value function is Lipschitz. This method is model free and only works for deterministic dynamics. The actions chosen by these methods are generally deterministic, such that if exploration of the state space is required heuristics such as ϵ-greedy or a softmax policy should be used.

Policy-search methods can be applied to address this shortcoming, by iteratively improving a policy. Bagnell and Schneider (2003) developed a policy gradient method using RKHS embedding of a desirability function that defines a policy. However, their approach is restricted to discrete actions, and cannot exploit learnable system dynamics. Lever and Stafford (2015) introduced another policy gradient approach, which searches for the mean function of a Gaussian process policy within an RKHS. A disadvantage is that a schedule to decay the co-variance of the Gaussian towards zero has to be manually defined.

Deisenroth and Rasmussen (2011) describe a non-parametric model-based iterative method. However, their method requires the reward function to be known and to be of squared exponential form, and does not address the exploration problem.

REPS is an alternative that performs well on a variety of problems (Lioutikov et al., 2014; Peters et al., 2010; Daniel et al., 2013). These approaches assume the Lagrangian multiplier V is linear in manually-defined features and do not consider non-parametric policies. So far, work on learned transition models for REPS has been limited. The transition dynamics have been approximated by deterministic single-sample outcomes (Daniel et al., 2013; Peters et al., 2010), or by time-dependent linear models (Lioutikov et al., 2014).

4. Experiments

We evaluate our contribution on a reaching task, as well as two variations of the underpowered pendulum swing-up task. We will first discuss elements of our set-up that are the same across tasks, and then discuss implementation specifics and results for each tasks separately.

4.1 Experiment Set-up

We do not consider it possible for our agent to explore by choosing arbitrary state-action pairs. Instead, as shown in Alg. 1, from an initial state distribution our agent explores using its stochastic policy. After every 10 rollouts, the model learner and the policy of the agents are updated. To bootstrap the model and the policy, the agent is given 30 rollouts using a random exploratory policy initially. To avoid excessive computations, we include a simple forgetting mechanism that only keeps the latest 30 rollouts at any time. Each roll-out contains 49 samples on expectation. In our experiments, for every method, we performed 10 trials. The model and policy are refined incrementally in every iteration.

4.2 Comparative Methods

We compared multiple methods to learn the tasks. On the one hand, we consider the non-parametric value-function based methods introduced by Grünewälder et al. (2012) and Pazis and Parr (2011). We also compare to versions of REPS that use the sample-based model approximation introduced by Daniel et al. (2013), and one that uses fixed features.

Sample based model. For near-deterministic systems, the observed $f(\mathbf{s}'_i)$ can be used as coarse approximation for $\mathbb{E}_{\mathbf{s}'}\left[f(\mathbf{s}')|\mathbf{s}_i, a_i\right]$.

Feature based REPS Instead of the non-parametric form of V assumed in this paper, we can follow earlier work and define a fixed feature basis consisting of radial basis functions distributed according to a grid over the state-action space.

Approximate value iteration. We compare an on-policy version of the method by Grünewälder et al. (2012) by replacing the maximum operator by a softmax to ensure exploration. We also evaluate the proposed off-policy variant that uses a grid of training points over the state-action space, which is a richer set of inputs than that obtained by the other methods under comparison.

Non-parametric approximate linear programming. We compare to an on-policy variant of the method by Pazis and Parr (2011), where exploration is assured by adding Gaussian noise in a fraction ϵ of selected actions.

4.3 Reaching Task Experiment

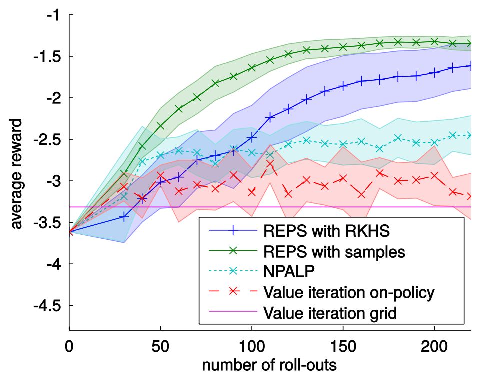
In the reaching task, we simulate a simple deterministic two-link planar robot. The robot gets negative reinforcement according to the square of the applied action and of the distance of its end-effector to the desired Cartesian position.

We use the commonly used exponential-squared (or Gaussian) kernel for angular velocities $\dot{\boldsymbol{\theta}}$ and actions. However, for the angles $\boldsymbol{\theta}$ we need a kernel that represents its periodicity. We chose the kernel $k_{\mathrm{p}}(x_i, x_j) = \exp(-\sum_d \sin((x_i^{(d)} - x_j^{(d)})/(2\pi))^2/[\mathbf{l}]_d^2)$, with $\mathbf{l}$ free parameters. Consequently, the kernel $k((\boldsymbol{\theta}_i, \dot{\boldsymbol{\theta}}_i, a_i), (\boldsymbol{\theta}_j, \dot{\boldsymbol{\theta}}_j, a_j)) = k_{\mathrm{p}}(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) k_{\mathrm{se}}(\dot{\boldsymbol{\theta}}_i, \dot{\boldsymbol{\theta}}_j) k_{\mathrm{se}}(a_i, a_j)$.

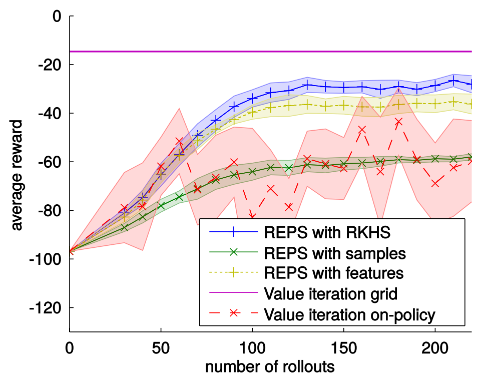
Feature-based REPS needs a grid over the complete state-action space. Because of practical difficulties with the six-dimensional state-action space, we omitted this method.

Results of the reaching task The results of the reaching task are shown in Figure 1a. As this task is deterministic, the sample-based model is optimal and provides a upper bound. Since REPS with RKHS embeddings needs to iteratively learn the model, its convergence is slower.

The baseline methods NPALP and value iteration using RKHS embeddings obtain good performance after the first couple of iterations, but are not suitable for iterative learning as



(a) Learning progress of the different methods on the reaching task. As this environment is deterministic, sample-based approximations of the transition functions are optimal. Error bars show twice the standard error of the mean.



(b) Learning progress of the different methods on the swing-up task. Our non-parametric relative entropy method outperforms the other on-policy learners. Error bars show twice the standard error of the mean.

Figure 1: Comparative results on two tasks with low-dimensional sensing.

they stop improving after that. The grid based value iteration scheme fails because of the high dimensionality of the state space.

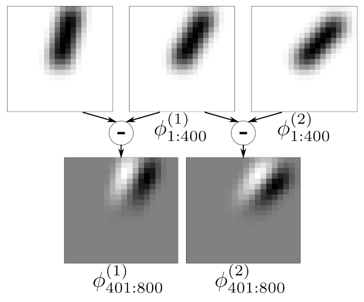
4.4 Low-Dimensional Swing-up Experiment

In this experiment, we simulate a pendulum with a torque a applied at the pivot. Additive noise is applied to the controls. The algorithms are given the angle θ and the angular velocity $\dot{\theta}$ directly, so that the state $\mathbf{s} = [\theta, \dot{\theta}]^T$. The kernels used are the same as in the reaching task experiment (Sec. 4.3). The NPALP method was not designed for stochastic systems and is consequently omitted.

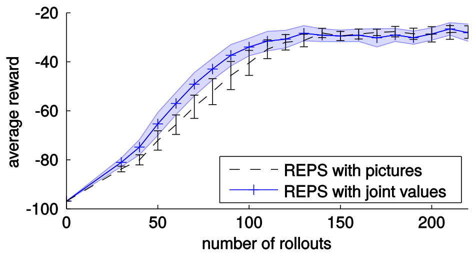
Results of the low-dimension swing-up. A comparison between the methods is shown in Fig. 1b. The value iteration methods starts out competitively, but fails to keep improving the policy. Large variance indicates the learning process is unstable. The bounded policy update in REPS makes learning progress smooth by limiting information loss, and trades off exploration and exploitation.

The sample-based model learner performs considerably worse in this experiment, as it cannot account for stochastic transitions. The variant with fixed features performs well initially, but in later iterations the non-parametric can perform better as it focuses its representative power on frequently visited parts of the state-space.

The grid-based value iteration method works well. However, providing the grid is only possible in simulation, as without an existing controller it is generally not possible to start the dynamical system with arbitrary position and velocity.



(a) Illustration of high-dimensional features. First row: Low-resolution images of the pendulum are rendered to yield the first 400 dimensions of the feature vector. Second row: the difference images form the next 400 dimensions of the feature vector.



(b) Learning progress of our method on the underpowered pendulum swing-up directly from joint values and from rendered images. Shaded area is twice the standard error of the mean.

Figure 2: Features and results for the high-dimensional swing-up task

4.5 High-Dimensional Swing-up Experiment

In a more challenging version of the swing-up, the agent only has access to images of the pendulum. We render a blurred image of the pendulum in its current state, as well as a difference image between visual representations of the pendulum in its current- and previous state (to provide a notion of angular velocity). The state representation, illustrated in Fig. 2a, is 800 dimensional.

Employing polynomial features or a grid of radial basis functions in such a big space would be infeasible. All comparative methods that performed reasonably rely on a grid, so we will compare our method only to our solution from the previous experiment. We chose squared-exponential kernels for all dimensions. To reduce the number of hyperparameters, we constrain the bandwidth on all pixels per image to be the same.

Results for the high-dimensional swing-up. The results of the evaluation of our approach, together with the sample-based baseline method, are shown in Fig. 2b. We see that the learning progress is a bit slower than progress on the low-dimensional task. However, the final solution is equally good.

5. Discussion and Future Work

In this paper, we have developed a policy search method with smooth, robust updates to solve continuous MDPs. This method uses learned non-parametric models and allows the use of non-parametric policies, avoiding hand-crafted features.

Many tasks concerning sensory data hava a high extrinsic dimensionality, but are intrinsically low-dimensional. Kernel-based algorithms perform all operations on kernel values, so they are invariant to the extrinsic dimensionality.

We show the resulting algorithm to be able to outperform other algorithms on a reaching task and a pendulum swing-up task with control noise. Our algorithm does this using only on-policy samples, i.e., without the need to sample arbitrary state-action pairs. We also show the algorithm is able to solve the task using only rendered images, with an 800-dimensional representation of the state space.

We are currently working to scale up the algorithm using efficient approximations to scale the algorithm to larger data sets. We are also investigating how to address specific problems in real-world robotic tasks like delays and hysteresis.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007–2013) under grant agreements #610967 (TACMAN) and #270327 (CompLACS).

References

J.A. Bagnell and J. Schneider. Policy search in reproducing kernel Hilbert space. Technical Report RI-TR-03-45, CMU, 2003.

C. Daniel, G. Neumann, O. Kroemer, and J. Peters. Learning sequential motor tasks. In *ICRA*, 2013.

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

S. Grünewälder, G. Lever, L. Baldassarre, M. Pontil, and A. Gretton. Modelling transition dynamics in MDPs with RKHS embeddings. In *ICML*, pages 535–542, 2012.

J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *International Joint Conference on Artificial Intelligence*, pages 2650–2655, 2011.

G. Lever and R. Stafford. Modelling policies in MDPs in reproducing kernel Hilbert space. In *AIstats*, pages 590–598, 2015.

R. Lioutikov, A. Paraschos, G. Neumann, and J. Peters. Sample-based information-theoretic stochastic optimal control. In *ICRA*, 2014.

J. Pazis and R. Parr. Non-parametric approximate linear programming for MDPs. In *AAAI*, pages 459–464, 2011.

J. Peters, K. Muelling, and Y. Altun. Relative entropy policy search. In *AAAI*, pages 1607–1612, 2010.

K. Rawlik, M. Toussaint, and S. Vijayakumar. Path integral control by reproducing kernel Hilbert space embedding. In *IJCAI*, 2013.

L. Song, K. Fukumizu, and A. Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *Signal Processing Magazine, IEEE*, 30(4):98–111, 2013.

H. van Hoof, J. Peters, and G. Neumann. Learning of non-parametric control policies with high-dimensional state features. In *AIstats*, 2015.