Journal of Artificial Intelligence Research 75 (2022) 597-636 Submitted 04/2022; published 10/2022

Low-Rank Representation of Reinforcement Learning Policies

Bogdan Mazoure BOGDAN.MAZOURE@MAIL.MCGILL.CA
Thang Doan THANG.DOAN@MAIL.MCGILL.CA
Tianyu Li TIANYU.LI@MAIL.MCGILL.CA
School of Computer Science
McGill University
Montreal, QC, Canada

Vladimir Makarenkov MAKARENKOV.VLADIMIR@UQAM.CA
Département d'Informatique
Université du Québec à Montréal
Montreal, QC, Canada

Joelle Pineau JPINEAU@CS.MCGILL.CA
Doina Precup DPRECUP@CS.MCGILL.CA
School of Computer Science
McGill University
Montreal, QC, Canada

Guillaume Rabuseau GUILLAUME.RABUSSEAU@UMONTREAL.CA
Department of Computer Science
and Operations Research - Mila
CIFAR AI Chair
Université de Montréal
Montreal, QC, Canada

Abstract

We propose a general framework for policy representation for reinforcement learning tasks. This framework involves finding a low-dimensional embedding of the policy on a reproducing kernel Hilbert space (RKHS). The usage of RKHS based methods allows us to derive strong theoretical guarantees on the expected return of the reconstructed policy. Such guarantees are typically lacking in black-box models, but are very desirable in tasks requiring stability and convergence guarantees. We conduct several experiments on classic RL domains. The results confirm that the policies can be robustly represented in a low-dimensional space while the embedded policy incurs almost no decrease in returns.

1. Introduction

In the reinforcement learning (RL) framework, the goal of a rational agent consists in maximizing the expected rewards in a dynamical system by finding a suitable conditional distribution known as *policy*. The later can be found using policy iteration and any suitable function approximator, ranging from linear models to neural networks [59, 37]. Albeit neural networks being the state-of-the-art learner for most performance-based tasks [51, 8], this class of blackbox models provide few guarantees on the final performance, which should be imposed on a policy in risk-sensitive tasks [24]. While the same reasoning can be applied to the value function, the problem of value function representation is complementary to that of

policy representation. For instance, in the specific case of Boltzmann policies (to which we restrict most of our theoretical and empirical findings), both are arguably near-identical, as the policy is essentially the projection of the value function onto the probability simplex. This observation allows us to focus on the policy representation problem without discarding value functions [67]. Note that, however, value and policy learning are not always identical, and examples where learning the optimal policy is much more complex than the relatively simple value function can be constructed.

In many application domains such as self-driving cars or robotic controllers [50, 2, 10, 34], it is often crucial to have a robust policy. One criteria of such policy is that the variance of the expected returns should be as small as possible. Unfortunately, due to the non-convexity of most deep learning objective functions, it is difficult to enforce and theoretically validate such constraints. Moreover, the high complexity of these models often leads to high variance and has been shown empirically [44].

One approach to mitigate this issue is to represent complex policies using decision rules [6, 34, 4]. However, their major downsides are non-convex loss functions (preventing in-depth error analysis), and access to external oracle information (e.g. unbiased gradients).

In order to address the need for performance guarantees, we propose a principled way to represent a broad class of policy density functions as points in a Reproducing Kernel Hilbert Space (RKHS). As we show in the paper, one advantage of this representation is that truncation of the embedding leads to a reduction in variance of the expected returns. In addition, the theoretical framework of RKHS offers powerful tools for analyzing the error introduced by the truncation. In general, our framework achieves two desirable properties: (i) our method produces policies with lower return variance than the original policy; (ii) policies can be embedded in lower-dimensional subspaces without significant drops in performance. We show both of these properties theoretically as well as empirically.

The use of RKHS and kernel methods in reinforcement learning problems is not uncommon. Non-parametric kernel density estimators have previously been used to represent fundamental RL quantities such as the value function [62, 66], transition dynamics [27, 43, 35, 5] and predictive representations of states [49, 12] known as PSRs. Moreover, some works leverage spectral learning and RKHS to extend the classical PSR setting [11, 14, 36]. One can even recover the original distribution from the kernel mean estimator through the tools provided by [33], which turns out to be useful in message passing [56]. The main difference of our work with algorithms operating in the value function space such as, for example, [62], is that we are concerned with finding a low-rank representation of the policy, as it is the quantity which will be deployed in real-world scenarios (as opposed to the value function which is ancillary to training the policy). For instance, specifically for the class of Boltzmann policies, both approaches are near-identical, with the additional policy decoding step from the kernelized value function for the value-based approaches, a computational overhead which our method avoids by directly operating in policy space.

Our contributions are the following:

- We propose a decomposition of policy densities within a separable Hilbert space and derive a set of performance bounds which, when used together, provide a heuristic to pick the dimensionality of the embedding. For practical reasons, we restrict our analysis to the space of square-integrable functions.

- The truncation of the RKHS embedding is shown to reduce variance of returns, when second order moment conditions are satistfied.
- We interpret the performance error bounds through three separate error terms: truncation (Thm. 4), discretization (Thm. 6), and pruning (Thm. 7).
- Empirical evidence on continuous control tasks supports our theoretical findings.

To the best of our knowledge, this is the first work proposing a general framework to learn an RKHS policy embedding with performance and long-term behaviour guarantees.

2. Preliminary

In this section, we provide a brief introduction to reinforcement learning, density approximation, as well as singular value decomposition.

2.1 Reinforcement Learning

We consider a problem modeled by a discounted Markov Decision Process (MDP) $\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, r, P, \beta, \gamma \rangle$, where $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; given states s, $s' \in \mathcal{S}$, action $a \in \mathcal{A}$, $\mathbb{P}(s'|s,a)$ is the transition probability of transferring s to s' under the action a and $r(s,a)$ is the reward collected at state s after executing the action a; β is the initial state distribution and γ is the discount factor. Through the paper, we assume that r is a bounded function.

The agent in RL environment often execute actions according to some policies. We define a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$ such that $\pi(a|s)$ is the conditional probability that the agent takes action $a \in \mathcal{A}$ after observing the state $s \in \mathcal{S}$. The state value function V_π and the state-action value function Q_π are defined as:

$$V_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k) \middle| s_t = s \right],$$

$$Q_\pi(s,a) = \mathbb{E}_\pi \left[\sum_{k=t}^{\infty} \gamma^{k-t} r(s_k, a_k) \middle| s_t = s, a_t = a \right]$$

The gap between Q_π and V_π is known as the advantage function:

$$A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s),$$

where $a \sim \pi(\cdot|s)$. The coverage of policy π is defined as the stationary state visitation probability $\rho_\pi(s) = \lim_{t\to\infty} \mathbb{P}(s_t = s)$ under π. The objective is to find a policy π which maximizes the expected discounted reward:

$$\eta(\pi) = \mathbb{E}_\beta[V_\pi(s)]. \tag{1}$$

Analogously, we let a Markov chain be defined by the tuple (P,β) where P is the transition probability and β is the initial state distribution. Note that the Markov chain (P^π,β) where $P^\pi = \mathbb{E}_{a\sim\pi}[\mathbb{P}(s'|s,a)]$, induced by rolling out π in $\mathcal{M}$ has a stationary distribution if the corresponding Markov chain is ergodic.

2.2 Function Approximation in Inner Product Spaces

The theory of inner product spaces has been widely used to characterize approximate learning of Markov decision processes [45, 63]. In this subsection, we provide a short overview of inner product spaces of square-integrable functions on a closed interval I denoted $L^2(I)$.

The $L^2(I)$ space is known to be separable for $I = [0, 1]$, i.e., it admits a countable orthonormal basis of functions $\{\omega_k\}_{k=1}^{\infty}$, such that $\langle \omega_i, \omega_j \rangle = \delta_{ij}$ for all i, j and δ being Kronecker's delta. This property makes possible computations with respect to the inner product

$$\langle f, g \rangle = \int_{t \in I} f(t)\bar{g}(t)dt, \tag{2}$$

and corresponding norm $||f|| = \sqrt{\langle f, f \rangle}$, for $f, g \in L^2(I)$ and conjugate map $\bar{g}$.

That is, for any $f \in L^2(I)$, there exist scalars $\{\xi_k^f\}_{k=1}^{\infty}$ such that its representation in the RKHS is given by

$$\hat{f}(t) = \sum_{k=1}^{\infty} \xi_k^f \omega_k(t), \quad \xi_k^f = \langle f, \omega_k \rangle \ . \tag{3}$$

If $\hat{f}_K(t) = \sum_{k=1}^{K} \xi_k^f \omega_k(t)$, then adding $\xi_{K+1}\omega_{K+1}$ to $\hat{f}_K$ can be thought of as adding a new orthogonal component.

Eq. 3 suggests a simple embedding rule, where one would project a density function onto a fixed basis and store only the first $\{\xi_k^f\}_{k=1}^{K}$ coefficients. Which components to pick will depend on the nature of the basis: harmonic and wavelet bases are ranked according to amplitude, while singular vectors are sorted by corresponding singular values.

The choice of the basis $\{\omega_k\}_{k=1}^{\infty}$ plays a crucial role in the quality of approximation. The properties of the function f (e.g. periodicity, continuity) should also be taken taken into account when picking ω. Some examples of well-known orthonormal bases of $L^2(I)$ are *Fourier* $\omega_k(t) = e^{2\pi ikt}$ [57] and *Haar* $\omega_{k,k'}(t) = 2^{k/2}\omega(2^k t - k')$ [28]. Other examples include but are not limited to Hermite and Daubechies bases [16]. Moreover, it is known that a mixture model with countable number of components is a universal density approximator [40]; in such case, the basis consists of Gaussian density functions and is not necessarily orthonormal.

Moreover, matrix decomposition algorithms such as the truncated singular value decomposition (SVD) are popular projection methods due to their robustness and extension to finite-rank bounded operators [68], since they learn both the basis vectors and the corresponding linear weights. Although SVD has previously been used in embedding schemes [39, 26], most works apply it on the weights of the function rather than the outputs. Similarly to the density approximators above, a key appeal of SVD is that the error rate of truncation can be controlled efficiently through the magnitude of the minimum truncated singular values[1].

While convergence guarantees are mostly known for the closed interval $I = [0, 1]$, multiple works have studied properties of the previously discussed basis functions on the whole real line and in multiple dimensions [21]. Weaker convergence results in Hilbert spaces can be stated with respect to the space's inner product. If, for every $g \in L^2(I)$ and sequence of functions $f_1, .., f_n \in L^2(I)$,

$$\lim_{n \to \infty} \langle f_n, g \rangle \to \langle f, g \rangle, \tag{4}$$

[1] see Eckart-Young-Mirsky theorem

then the sequence f_n is said to converge weakly to f as $n \to \infty$. Stronger theorems are known for specific bases but require stronger assumptions on f.

In order to allow our learning framework to have strong convergence results in the inner product sense as well as a countable set of basis functions, we restrict ourselves to separable Hilbert spaces.

3. Framework

In this section, we first introduce a framework to represent policies in an RKHS with desirable error guarantees. The size of this embedding can be varied through the coefficient truncation step. We show that this truncation has the property to reduce the return variance without drastically affecting the expected return. Next, we propose a discretization and pruning steps as a tractable alternative to working directly in the continuous space. Finally, we derive a practical algorithm which leverages all aforementioned steps.

3.1 Formulation as RKHS Problem

By Mercer's theorem [41], any symmetric positive-definite kernel function κ with associated integral operator T_κ can be decomposed as:

$$\begin{aligned}\kappa(x,y) &= \sum_{k=1}^{\infty} (\sqrt{\lambda_k} e_k(x))(\sqrt{\lambda_k} e_k(y)) \\ &= \sum_{k=1}^{\infty} \omega_k(x)\omega_k(y),\end{aligned} \tag{5}$$

where λ_k and e_k are the eigenvalues and eigenfunctions of T_κ, respectively.

It follows by Moore–Aronszajn theorem [3] that there exists a unique Hilbert space $\mathcal{H}$ of functions for which κ is the kernel.

Moreover, the inner product associated with $\mathcal{H}$ is:

$$\langle f, g\rangle_{\mathcal{H}} := \sum_{k=1}^{\infty} \frac{\langle f, e_k\rangle_{L_2}\langle g, e_k\rangle_{L_2}}{\lambda_k} = \sum_{k=1}^{\infty} \frac{\xi_k^f \xi_k^g}{\lambda_k}. \tag{6}$$

This allows us to state performance bounds between policies embedded in an RKHS in terms of their projection weights $\xi_1^\pi, \xi_2^\pi, ..$, as shown in the next section.

Consider the conditional distribution $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. For a fixed s, the function $\pi(\cdot|s)$ belongs to a Hilbert space $\mathcal{H}_s$ of functions $f : \mathcal{A} \to \mathbb{R}$. The following lemmas show how the distance between two policies can be computed as a function of their coefficients in the RKHS representations.

Lemma 1. *Let s be a state in $\mathcal{S}$. Let $\mathcal{H}_s$ be an RKHS, the integral operator of which has eigenvalues $\{\lambda_k\}_{k=1}^{\infty}$ in decreasing order and eigenfunctions $\{e_k\}_{k=1}^{\infty}$. Let $\xi_k^{\pi_i} = \langle \pi_i, e_k\rangle$ and $\pi_1, \pi_2 \in \mathcal{H}_s$ such that $\pi_1 = \pi_1(\cdot|s), \pi_2 = \pi_2(\cdot|s)$. Then, the following holds*

$$||\pi_1 - \pi_2||_{\mathcal{H}_s}^2 = \sum_{k=1}^{\infty} \frac{(\xi_k^{\pi_1} - \xi_k^{\pi_2})^2}{\lambda_k}. \tag{7}$$

Lemma 2. *Let $(\xi_1^{\pi_1}, \xi_1^{\pi_2}), .., (\xi_K^{\pi_1}, \xi_K^{\pi_2})$ be the projection weights of π_1, π_2 onto $\mathcal{H}_s$ such that, for $p, q \in \mathbb{N}^+$, if $p > q$ then $\xi_p^{\pi_i} < \xi_q^{\pi_i}$ for $i = 1, 2$. If there exists a $K \in \mathbb{N}^+$ such that for all $k > K$, $|\xi_k^{\pi_1} - \xi_k^{\pi_2}| < \varepsilon^k \sqrt{\lambda_k}$ for some real $\varepsilon > 0$, then the following holds*

$$\sum_{k=K+1}^{\infty} \frac{(\xi_k^{\pi_1} - \xi_k^{\pi_2})^2}{\lambda_k} \leq \frac{\varepsilon^{2(K+1)}}{1-\varepsilon^2}. \tag{8}$$

The truncated embedding of size K can be formed by setting the sequence of coefficients $\{\xi_k^{\pi}\}_{k=K}^{\infty}$ to zero, obtaining

$$\hat{\pi}_K(t) = \sum_{k=1}^{K} \xi_k \omega_k(t) \ . \tag{9}$$

As shown in Experiments A.5, truncating the embedding at a given rank can have desirable properties on the variance without incurring drops in performance.

3.2 Truncating RKHS Embeddings Can Reduce Variance of Returns

In this subsection, we show under which conditions one can expect a reduction in variance of returns of the RKHS policy, thus helping in sensitive tasks such as medical interventions or autonomous vehicles [24].

Any policy π in the RKHS can be decomposed as a finite sum:

$$\pi(a|s) = \underbrace{\sum_{k=1}^{K} \xi_k \omega_k(s,a)}_{\hat{\pi}_K} + \underbrace{\sum_{k=K+1}^{\infty} \xi_k \omega_k(s,a)}_{\varepsilon_K}, \ \forall s, a \in \mathcal{S} \times \mathcal{A}. \tag{10}$$

Our framework allows to derive variance guarantees by first considering the *random return* variable Z of the policy π given some state and action (see **(author?)** [8]):

$$Z(\pi) = r(S_0, A_0) + \sum_{t=1}^{\infty} \gamma^t r(S_t, A_t), \tag{11}$$

with $S_t \sim P(\cdot|s_{t-1}, a_{t-1}), A_t \sim \pi(\cdot|s_t)$ and $S_0, A_0 \sim \beta$ for some initial distribution β.

The variance of $Z(\pi)$ over the entire trajectory is hard to compute [61] since it requires to solve a Bellman equation for $\mathbb{E}[Z^2(\pi)]$. Instead, we look at the variance over initial state-action pairs sampled from β. Since we do not know the distribution of $\pi(a|s)$ but only that of (a, s) under β, we compute all expectations using the Law of the Unconscious Statistician [48, LOTUS].

Lemma 3. *Let $\mathbb{V}_\beta[X] = \mathbb{E}_\beta[X^2] - (\mathbb{E}_\beta[X])^2$ be the variance operator with respect to β. Let π be a policy and $\hat{\pi}_K$ its K component approximation. Let σ be an entry-wise positive and differentiable normalization function $\sigma : \mathbb{R} \to [0, 1]$ with derivative σ' s.t. $\pi(\cdot|s) = \sigma(Q_\pi(\cdot|s))$. If the following conditions are satisfied*

1. *$\sigma'(\eta(\hat{\pi}_K)) \leq \sigma'(\eta(\pi))$ (monotonically decreasing σ' on $(0, \infty)$);*

2. $\sqrt{\frac{\mathbb{E}_\beta[\varepsilon_K^2]}{3}} \geq \mathbb{E}_\beta[\varepsilon_K]$ *(second moment condition);*

3. $\mathbb{V}_\beta[\pi] = (\sigma'(\eta(\pi)))^2\mathbb{V}_\beta[Q_\pi] + O(\sigma''(\eta(\pi)))^2) \approx (\sigma'(\eta(\pi)))^2\mathbb{V}_\beta[Q_\pi]$ *(Second order Taylor residual is small),*

then the following holds:

$$\mathbb{V}_\beta[Q_\pi] \geq \mathbb{V}_\beta[Q_{\hat{\pi}_K}] \tag{12}$$

There are no further restrictions on the form of σ; in our experiments, we use the softmax function $\sigma(\boldsymbol{x}_i) = \frac{e^{\boldsymbol{x}_i}}{\sum_{j=1}^d e^{\boldsymbol{x}_j}}$. Lemma 3 tells us the relation between the variance of returns collected by either the true or truncated policies (detailed analysis in Appendix).

In practice, Condition 1 is satisfied when σ' is decreasing on $(0, \infty)$, rewards are strictly positive and $\eta(\hat{\pi}_K) \leq \eta(\pi)$. Condition 2 is not restrictive, and is, for example, satisfied by the Student's t distribution with $\nu > 1$ degrees of freedom. Condition 3 is a classical assumption in variance analysis [9] and holds if the expansion is done near the expectation of the Q function. The requirement that π directly depends on Q is fairly common in reinforcement learning, specifically when dealing with maximum-entropy policies [29] and does not restrict much the class of learnable policies. For example, parametrizing the softmax function with a temperature parameter $T \in \mathbb{R}^+$ yields $\sigma(\boldsymbol{x}_i; T) = \frac{e^{\boldsymbol{x}_i/T}}{\sum_{j=1}^d e^{\boldsymbol{x}_j/T}}$ broadens the class of achievable policies while maintaining Lemma 3 results.

The variance reduction property of truncated RKHS embeddings stated in Lemma 3 can be contrasted with the class of policy gradient methods [60], in which the variance of the gradients directly impacts the quality of the learned policy. The core idea consists in taking the gradient of returns $\eta(\pi_\theta)$ with respect to the policy parameters θ:

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \psi_t \nabla_\theta \log \pi_\theta(a_t|s_t)], \tag{13}$$

where ψ_t is any "sensible" estimate of the rewards-to-go in the current episode. The variance of this gradient is impacted by stochasticity of two terms, ψ_t and $\nabla_\theta \log \pi_\theta(a_t|s_t)$. The case where ψ_t is taken to be the Monte-Carlo returns estimate of the trajectory [REINFORCE, 64] has the problem of large variance, which can potentially be mitigated by replacing ψ_t with the returns of the truncated policy and appealing to Lemma 3.

As we show in the next section, even after the truncation step, our framework still provides important performance guarantees.

3.3 Truncation Bound

Storing an approximation of the ground truth policy implies a trade-off between performance and memory allocation. Our framework allows to control the difference in collected reward using the following theorem that projects π onto the first K basis functions.

Theorem 4. *Let $s \in \mathcal{S}$ and $\mathcal{H}_s$ be the associated RKHS. Let $\pi_1, \pi_2 \in \mathcal{H}_s$ be represented by the coefficients $\{\xi_k^{\pi_1}\}_{k=1}^\infty$ and $\{\xi_k^{\pi_2}\}_{k=1}^\infty$ and let ε such that Lemma 2 holds. Let $M_s > 0$ be*

such that $|\pi_1(a|s) - \pi_2(a|s)| \leq M_s||\pi_1 - \pi_2||_{\mathcal{H}_s}$ *for all* $a \in \mathcal{A}$. *Let* $\Delta^K_{\mathcal{H}_s} = \sum_{k=1}^K \frac{(\xi_k^{\pi_1} - \xi_k^{\pi_2})^2}{\lambda_k}$ *and* $\bar{\epsilon} = \max_{s,a} |Q_{\pi_2}(s,a) - V_{\pi_2}(s)|$. *Then*

$$|\eta(\pi_2) - \eta(\pi_1)| \leq \frac{4|\mathcal{A}|^2\bar{\epsilon}\gamma}{(1-\gamma)^2}\left\{\max_{s\in\mathcal{S}}(M_s\Delta^K_{\mathcal{H}_s})^2 + O(\varepsilon^{2K}\max_{s\in\mathcal{S}} M_s\Delta^K_{\mathcal{H}_s})\right\} + \mathbb{E}_{\rho_{\pi_2}}[M_s\Delta^\infty_{\mathcal{H}_s}\sum_{a\in\mathcal{A}} A_{\pi_2}(s,a)] \,.$$

As a special case, we can consider $\pi_2 = \hat{\pi}_K$ to be the policy π_1 projected onto the first top K bases of $\mathcal{H}_s$. In this case, the first term vanishes and we are only left with the last term, which corresponds to the compounding error made on the remaining set of bases. Specifically, Theorem 4 reduces to the following result.

Corollary 5. *Let* $s \in \mathcal{S}$ *and* $\mathcal{H}_s$ *be the associated RKHS. Let* $\pi_1 \in \mathcal{H}_s$ *be represented by the coefficients* $\{\xi_k^{\pi_1}\}_{k=1}^\infty$ *and* π_2 *be its projection onto the first top* K *bases of* $\mathcal{H}_s$. *Let* ε *such that Lemma 2 holds and* $M_s > 0$ *be such that* $|\pi_1(a|s) - \pi_2(a|s)| \leq M_s||\pi_1 - \pi_2||_{\mathcal{H}_s}$ *for all* $a \in \mathcal{A}$. *Let* $\Delta^K_{\mathcal{H}_s} = \sum_{k=1}^K \frac{(\xi_k^{\pi_1} - \xi_k^{\pi_2})^2}{\lambda_k}$ *and* $\bar{\epsilon} = \max_{s,a} |Q_{\pi_2}(s,a) - V_{\pi_2}(s)|$. *Then*

$$|\eta(\pi_2) - \eta(\pi_1)| \leq \mathbb{E}_{\rho_{\pi_2}}[M_s\Delta^\infty_{\mathcal{H}_s}\sum_{a\in\mathcal{A}} A_{\pi_2}(s,a)] \,.$$

This theorem implies that representing any policy within the top K bases of the Hilbert space yields an approximation error polynomial in ε due to picking the linear coefficients by decreasing magnitude. Since $M_s = ||\kappa(\cdot, s)||_{\mathcal{H}_s} \leq \sup_{s\in\mathcal{S}} ||\kappa(\cdot, s)||_{\mathcal{H}_s} < \infty$ when all function in $\mathcal{H}_s$ are bounded (see [58]), the right hand side of Theorem 4 is finite.

While Theorem 4 holds in the continuous case, most projection algorithms such as Fast Fourier Transform, wavelet transform and SVD operate on a discretized version of the function. The next section describes the error of switching from the continuous to the discrete case.

3.4 Discretization Error Bound

So far, the theoretical formulation of our algorithm was in the continuous space of states and actions. However, most efficient algorithms to project a function onto an orthonormal basis operate in the discrete space [57, 16], since it is often hard or even impossible to explicitly construct eigenfunctions for an arbitrary RKHS. For this reason, we introduce the discretization step, which allows to leverage these highly optimized methods without sacrificing the approximation guarantees from the continuous domain.

An important step in our algorithm is the component-wise grouping of similar states. This step is crucial, since it allows us to greatly simplify the calculations of the error bounds and re-use existing discrete projection algorithms such as Fast Fourier Transform. However, when the discretization is done naively, it can result in slower computation times due to the curse of dimensionality of the state-action space (to mitigate this, we propose the pruning step in the next section). We use an approach known as *quantile bining* [42]. We assume that state components assigned to the same bin have a similar behaviour, a condition which holds for simple environments and greatly simplifies our proposed algorithm.

Recall that the cumulative distribution function $\Pi_X(x) = \mathbb{P}[X \leq x]$ and the corresponding quantile function $\mathcal{Q}_X(p) = \inf\{x \in \mathbb{R} : p \leq \Pi_X(x)\}$. In practice, we approximate $\mathcal{Q}, \Pi$ with the empirical quantile and density functions $\hat{\mathcal{Q}}, \hat{\Pi}$, respectively.

Theorem 6. *Let Π_S, Π_A be cumulative policy functions over $\mathcal{S}, \mathcal{A}$ and let $\lceil\Pi\rceil_S, \lceil\Pi\rceil_A$ be their empirical estimates discretized using quantile bining over states and actions into b_S and b_A clusters, respectively. Let $\mathcal{Q}_S, \mathcal{Q}_A$ be the corresponding empirical quantile functions. Then, the volume of the discretization error is:*

$$E_{\Pi,\lceil\Pi\rceil} = \sum_{i=1}^{b_S}\sum_{j=1}^{b_A}\left\{\left|\int_{q_{i-1}^S}^{q_i^S} \lceil\hat{\Pi}\rceil_S(s)ds - \frac{i}{b_S}(q_i^S - q_{i-1}^S)\right| \cdot \left|\int_{q_{i,j-1}^A}^{q_{i,j}^A} \lceil\hat{\Pi}\rceil_A(a)da - \frac{j}{b_A}(q_{i,j}^A - q_{i,j-1}^A)\right|\right\}, \tag{14}$$

where $q_i^S = \lceil\hat{\mathcal{Q}}_S\rceil\left(\frac{i}{b_S}\right)$ and $q_{i,j}^A = \lceil\hat{\mathcal{Q}}_{A|i}\rceil\left(\frac{j}{b_A}\right)$.

Note that the process of computing $E_{\Pi,\lceil\Pi\rceil}$ is fundamentally similar to a Riemann sum. Therefore, an argument identical to Theorem 4 can be used to map Theorem 6 into the error space of returns.

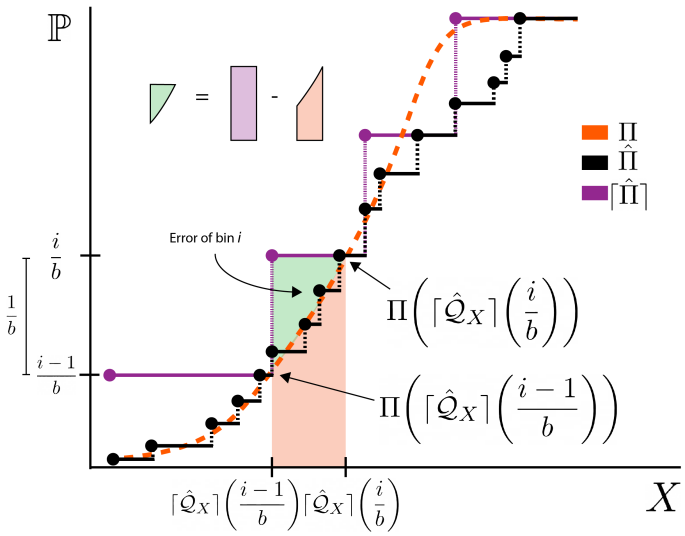

Figure 1: Schematic view of the discretization error

Theorem 6 and Figure 1 show how the discretization error induced by Algorithm 1 can be computed across all states and actions in the training set. Note that samples used to compute both distribution functions are i.i.d. inter-, but not intra-trajectories. This means that the classical result from [20] might not hold. For the rest of the paper, we make the dependence of $E_{\Pi,\lceil\Pi\rceil}(b_S, b_A)$ on b_S, b_A explicit in the notation.

The discretization step enables us to use powerful discrete projection algorithms, but as a side effect can drastically expand the state space. However, we observed empirically that most policies tend to cover a small subset of the state space. Using this information, we introduce a state-space pruning method based on the steady state distribution, which is addressed in the followed section.

3.5 Pruning Error Bound

The discretization step from the previous section greatly simplifies the calculations of the projection weights. However, it can potentially make the new state-action space quite large. In this section, we introduce the pruning step based on the long-run distribution of the policy, and then quantify the impact of removing unvisited states both on the performance and coverage of the policy.

Pruning unvisited states Since policies tend to concentrate around the optimum as training progresses [1], pruning those states would not significantly hinder the overall performance of the embedded policy. Let $\rho_\pi(s)$ denote the number of times π visits state s in the long run and $N(s,a)$ the number of times action a is taken from s over N trajectories.

This leads us to define π_{pruned} as a policy which acts uniformly at random below some state visitation threshold. Precisely, if $S_c = \{s : \rho_\pi(s) \leq c\}$ and $\mathbb{1}(\cdot)$ is the indicator function, then

$$\pi_{\text{pruned}}(a|s) = \{1 - \mathbb{1}_{S_0}(s)\}\frac{N(s,a)}{N(s)} + \mathbb{1}_{S_0}(s)\frac{1}{b_A}\,. \tag{15}$$

Note that, in most practical settings with large state and action spaces, choosing $c = 0$ prunes a considerable amount of state-action pairs. Pruning rarely visited states can drastically reduce the number of parameters, while maintaining high probability performance guarantees for π_{pruned}. The following theorem quantifies the effect of pruning states on the performances of the policy.

Theorem 7 (Policy pruning, [54])**.** *Let N be the number of rollout trajectories of the policy π, r_M be the largest reward. With probability $1 - 2\delta$, the following holds:*

$$|\eta(\pi) - \eta(\pi_{pruned})| \leq \frac{2r_M}{1-\gamma}\sqrt{\frac{3|\mathcal{S}||\mathcal{A}| + 4\log\frac{1}{\delta}}{2N}}\,. \tag{16}$$

This theorem allows one to discretize only the subset of frequently visited states while still ensuring a strong performance guarantee with high probability.

Instead of pruning based on samples from $\rho_{\lceil\pi\rceil}$, which induces a computational overhead since we need to perform an additional set of rollouts, we instead prune states based on ρ_π. The approximation error induced by this switch can be understood through the scope of Thm. 8.

Impact of pruning on state visitation Any changes done to the ground truth policy such as embedding it into an RKHS will affect its stationary distribution and hence the long-run coverage of the policy. In this subsection, we provide a result on the error made on the stationary distributions as a function of error made in the original policies. In particular, since $\eta(\pi) = \langle \rho_\pi, \mathbf{r}^\pi \rangle$ for expected reward $\mathbf{r}^\pi$, characterizing the error in ρ_π is of great importance: it directly links the policy approximation error with change in expected performance.

Under a fixed policy π, an MDP induces a Markov chain defined by the *expected transition model* $(\mathbf{P}_\pi)_{ss'} = \sum_{a\in\mathcal{A}} \pi(a|s)P(s'|s,a)$. In tensor form, it can be represented as $\mathbf{A}(\mathbf{T}_{(2)}\mathbf{\Pi})$, where $\mathbf{\Pi}_{as} = \pi(a|s)$ is the policy matrix, $\mathbf{A} = \mathbf{I} \odot \mathbf{I}$ is the Khatri-Rao product and $\mathbf{T}_{(2)}$ is the second mode of the transition tensor $\mathbf{T}_{sas'} = P(s'| = s, a)$.

If the Markov chain defined above is irreducible and homogeneous, then its *stationary distribution* corresponds to the state occupancy probability[2] given by the principal left eigenvector of $\mathbf{P}_\pi$.

The following theorem bridges the error made on the reconstruction of long-run distribution as a function of the system's transition dynamics and distance between policies.

Theorem 8 (Approximate policy coverage)**.** *Let $||\cdot||_{S_p}$ be the Schatten p-norm and $||\cdot||_p$ be the vector p-norm. Let ρ_π (respectively $\rho_{\pi'}$) be the stationary distribution of policy π (respectively π'). If ($\mathbf{P}_\pi$,$|\mathcal{S}|^{-1}\mathbf{1}_{|\mathcal{S}|}$) and ($\mathbf{P}_{\pi'}$,$|\mathcal{S}|^{-1}\mathbf{1}_{|\mathcal{S}|}$) are irreducible, homogeneous Markov chains, then the following holds:*

$$||\rho_\pi^\top - \rho_{\pi'}^\top||_2 \leq ||\mathbf{Z}||_{S_\infty}||\mathbf{\Pi} - \mathbf{\Pi'}||_{S_1}||\mathbf{T}_{(3)}||_{S_2}, \tag{17}$$

where $\mathbf{Z} = (\mathbf{I} - \mathbf{P}_\pi + \mathbf{1}_{|\mathcal{S}|}\rho_\pi^\top)^{-1}$ and $\mathbf{1}_{|\mathcal{S}|}$ is a vector of all ones.

A detailed proof can be found in the appendix. Note that the upper bound depends on both the environment's structure as well as on the policy reconstruction quality. It is thus expected that, for MDPs with particularly large singular values of $\mathbf{T}_{(2)}$, the bound converges less quickly than for those with smaller singular values. See the appendix for visualizations of the bound on empirical domains. When looking at Thm.8, a natural question which arises: can we store a low-rank approximation to $\rho(s,a) = \pi(a|s)\rho_\pi(s)$ for $s \notin S_0$ instead of simply storing π? The main drawback of such an approach is that, in order to deploy the agent in real-world scenarios, the approximate policy still has to be recovered from the joint distribution. This will require to store two low-rank approximations: $\rho(s,a)$ and $\rho(s)$ for all $s \notin S_0, a \in \mathcal{A}$, thus drastically increasing the space complexity of the algorithm. Moreover, after careful inspection of Thm. 8, it can be seen that the error of storing the marginal state distribution for even slightly different policies is amplified by the norm of the transition tensor. This suggests that each time we are required to decode a policy from the joint state-action distribution, the policy error will compound with the error coming from encoding the environment dynamics..

A useful implication of Thm 8 is that the policy discretization error can be stated in terms of performance difference.

Corollary 9. *Let $\boldsymbol{r}_2^\pi = ||\mathbb{E}_\pi[r]||_2$ be the Euclidean norm of the expected reward under π, and let $E_{\Pi,\lceil\Pi\rceil}$ be the discretization error as defined in Thm. 6. If b_S, b_A are sufficiently large, then*

$$|\eta(\lceil\pi\rceil) - \eta(\pi)| \leq ||\mathbf{Z}||_{S_\infty} E_{\Pi,\lceil\Pi\rceil}(b_S, b_A)||\mathbf{T}_{(3)}||_{S_2}\boldsymbol{r}_2^\pi$$

Note that, for any given π, the expression converges to 0 if b_S, b_A are picked very large.

Equipped with these tools, we are ready to state the general policy embedding bound in the RKHS setting.

[2]The existence and uniqueness of ρ follow from the Perron-Frobenius theorem.

3.6 General Policy Embedding Bound

We are finally ready to use the previous performance bounds to state the general performance result. The difference in performance of ground-truth policy and truncated embedded policy can be decomposed as a discretization error, a pruning error and a projection error:

Corollary 10. *Let π be the ground truth policy, $\lceil\pi\rceil$ the discretized policy, π_{pruned} the pruned policy and $\hat{\pi}$ the embedded policy. With probability $1-2\delta$ and sufficiently large b_S, b_A, the following holds*

$$
\begin{aligned}
&|\eta(\pi)-\eta(\hat{\pi})| \\
&\leq \underbrace{\frac{4|\mathcal{A}|^2\bar{\epsilon}\gamma}{(1-\gamma)^2}\left\{\max_{s\in\mathcal{S}}(M_s\Delta^K_{\mathcal{H}_s})^2+O(\varepsilon^{2^K}\max_{s\in\mathcal{S}}M_s\Delta^K_{\mathcal{H}_s})\right\}}_{Thm.\ 4}+\underbrace{\mathbb{E}_{\rho_{\hat{\pi}}}[M_s\Delta^{\infty}_{\mathcal{H}_s}\sum_{a\in\mathcal{A}}A^{\hat{\pi}}(s,a)]}_{Thm.\ 4} \\
&+\underbrace{||\mathbf{Z}||_{S_\infty}E_{\Pi,\lceil\Pi\rceil}(b_S,b_A)||\mathbf{T}_{(3)}||_{S_2}\boldsymbol{r}^{\pi}_2}_{Cor.\ 9}+\underbrace{\frac{2r_M}{1-\gamma}\sqrt{\frac{3|\mathcal{S}||\mathcal{A}|+4\log\frac{1}{\delta}}{2N}}}_{Thm.\ 7}
\end{aligned}
$$

Colored variables highlight hyperparameters of each part of the algorithm. Thm. 4 and Cor. 9 hold with probability 1 and Thm. 7 with probability $1-2\delta$; therefore the joint bound holds w.p. $1-2\delta$. While tighter bounds of Thm. 4 can be found in the literature [25, 46] for specific basis functions, our bound is general enough to hold for any suitable basis, without assumptions other than those already mentioned.

In the next section, we propose a practical algorithm which integrates all three steps.

4. Practical Algorithm

We first discuss the construction of $\lceil\pi\rceil$ via quantile discretization. The idea consists in grouping together similar state-action pairs (in term of visitation frequency). To this end, we use the quantile state visitation function $\mathcal{Q}_S$ and the state visitation distribution function Π_S, as well as their empirical counterparts, $\hat{\mathcal{Q}}_S$ and $\hat{\Pi}_S$. Quantile and distribution functions for the action space are defined analogously.

Algorithm 1: Quantile discretization

Input: Policy π, number of state bins b_S, number of action bins b_A
Result: Discrete policy $\lceil\pi\rceil$
Collect a set of states $S\subseteq\mathcal{S}$ and set of actions $A\subseteq\mathcal{A}$ via rollouts of π;
Build the empirical c.d.f. $\hat{\Pi}_s$ from set S;
Build the empirical c.d.f. $\hat{\Pi}_a$ from set A;
Find numbers $i_1,..,i_{b_S}$ s.t. $\hat{\Pi}_s(i_l)-\hat{\Pi}_s(i_{l-1})=\frac{1}{b_S}$ using $\hat{\mathcal{Q}}_s$ for all $s\in S, l=1,..,b_S$;
Find numbers $j_1,..,j_{b_A}$ s.t. $\hat{\Pi}_a(j_l)-\hat{\Pi}_a(j_{l-1})=\frac{1}{b_A}$ using $\hat{\mathcal{Q}}_a$ for all $a\in S, l=1,..,b_A$;
if $i_{l-1}\leq s\leq i_l$ *and* $j_{l'-1}\leq a\leq j_{l'}$ **then**
  Assign (s,a) to (l,l');
end
Set $\lceil\pi\rceil_{ll'}$ to $\pi(\frac{j_{l'-1}+j_{l'}}{2},\frac{i_{l-1}+i_l}{2})$

Algorithm 1 outlines the proposed discretization process allowing one to approximate any continuous policy (e.g., computed by a neural network) by a 2-dimensional table indexed by discrete states and actions. We use quantile discretization in order to have maximal resolution in frequently visited areas. It also allows for slightly faster sampling during rollouts, since the probability of falling in each bin is uniform.

Algorithm 2: RKHS policy embedding

Input: Policy π, num. components K, basis $\omega = \{\omega_k\}_{k=1}^{\infty}$
Result: A set of coefficients $\xi_1, ..., \xi_K$
1. Rollout π in the environment to estimate $\hat{\mathcal{Q}}_s, \hat{\mathcal{Q}}_a$ (empirical quantile functions);
2. Project π onto lattice to obtain $\lceil \pi \rceil$ using Alg. 1;
3. Prune $\lceil \pi \rceil$ with $\hat{\mathcal{Q}}_s, \hat{\mathcal{Q}}_a$ using Eq. 15 ;
4. Project $\lceil \pi \rceil$ onto first K elements of ω using one of the projection algorithms described in Section 2.2;

Since in practice working with continuous RKHS projections is cumbersome, we use Algorithm 1 in order to project the continuous policy π onto a discrete lattice, which is then projected onto the Hilbert space[3]. A natural consequence of working with embedded policies means that "taking actions according to $\hat{\pi}_K$" reduces to importance sampling of actions conditional on state bins (discussed in Appendix A.4). While importance sampling can handle unnormalized weights, it is easy to transform the discretized lattice into a proper distribution function by row-wise application of the softmax operator.

5. Experimental Results

In this section, we consider a range of control tasks: bandit turntable, Pendulum and Continuous Mountain Car (CMC) from OpenAI Gym [13]. All experimental details can be found in Appendix A.9.3.

In Pendulum and CMC, we omit GMM and fixed basis GMM methods, since the expectation-maximization algorithm runs into stability problems when dealing with a high number of components. Fixed-basis GMM serves as a baseline to benchmark optimization issues in the Expectation-Maximization (EM) algorithm for GMMs. Moreover, we use SVD for low-rank matrix factorization and the fourth order Daubechies wavelet [16] (DB4) as an orthonormal basis alternative to Fourier (DFT). In Pendulum and Mountain Car, all policy representation methods receive a pre-trained SAC policy [29], which they then project onto a lower-dimensional space; the original SAC policy's parameter count is shown as a vertical dotted line. In all experiments, we qualitatively compare various policy representation according to two criteria: ratio of returns to number of parameters, and variance of the returns collected by various policies.

5.1 Bandit Turntable

We evaluate our framework in the multi-armed bandit [55] inspired from **(author?)** [22] (see Figure 2a). Results are reported in Figure 2b via the average Wasserstein-1 metric, defined as $\mathbb{E}_s[W_1(\pi(\cdot|s), \hat{\pi}(\cdot|s))]$. From the figure, we can observe that GMM has more

[3]Python pseudocode can be found in Appendix A.9.1

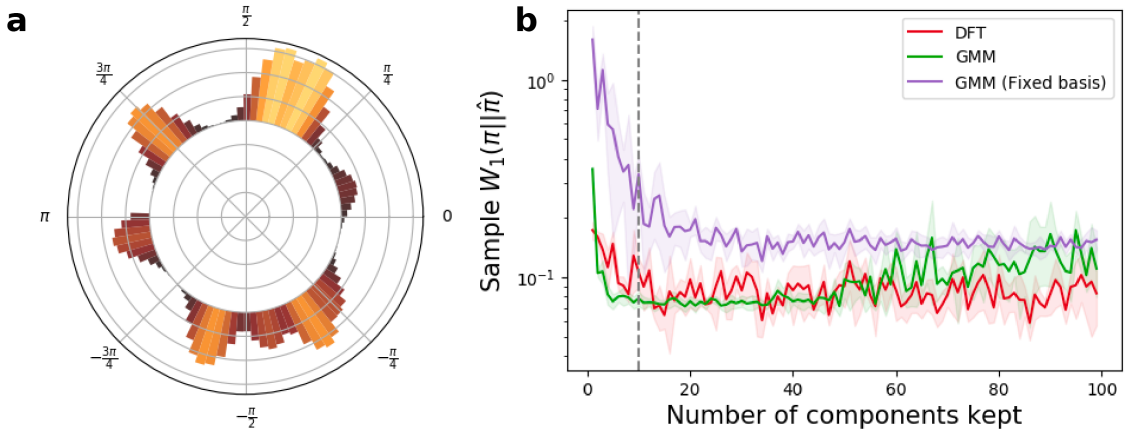

Figure 2: Bandit turntable environment in **(a)** polar coordinates. Magnitude of modes is proportional to the reward (lighter is higher). Actions are absolute angles in interval $[-\pi, \pi]$. **(b)** shows the W_1 distance between ground truth and order K approximations. Dotted line indicates the number of modes in the ground truth density.

trouble converging when we keep more than 60 components. This is due to the EM's stability issue when dealing with large number of components. While fixed basis GMM struggles to accurately approximate the policy, DFT shows a stable performance after a threshold of 10 components. The appendix contains additional results in the MAB setting motivated by recent advances [19, 23], and a simplified truncation lemma for average rewards. Note how the Wasserstein-1 error for the DFT policy decreases slowly, at a rate which heavily depends on the localisation of the original policy, since an infinite number of components is required to represent a degenerate distribution. In the above experiment, bandit policies were randomly sampled from a Gaussian mixture with varying standard deviation, which resulted in some quasi-deterministic policies being generated and leading to a large embedding error. On the other hand, GMM approximations are notoriously hard to fit exactly in multiple dimensions due to local convergence properties of the expectation-maximization algorithm.

5.2 Continuous Mountain Car

We compare all embedding methods with a maximum likelihood estimate (MLE) of π, which consists of approximating the discretized policy with samples from the continuous density $\mathbb{E}[\pi_K(\cdot|s)] \approx \frac{1}{K}\sum_{k=1}^{K} A_k,\ A_k \sim \pi(\cdot|s)$. The MLE rate of convergence to π is $O(\sqrt{K})$ with i.i.d. data, which grounds the convergence rate of other methods. Note that, for large values of K, the MLE policy is not sample-efficient, as it requires K action samples from *each state*, which cannot be obtained in practice from a fixed dataset. As shown in Figure 3 over 10 runs, DFT, SVD and DB4 reach same returns as the MLE method. Note that DB4 shows slightly better performance than DFT.

Due to space constraints, Appendix Figure 15 contains insights on the pruning and discretization errors.

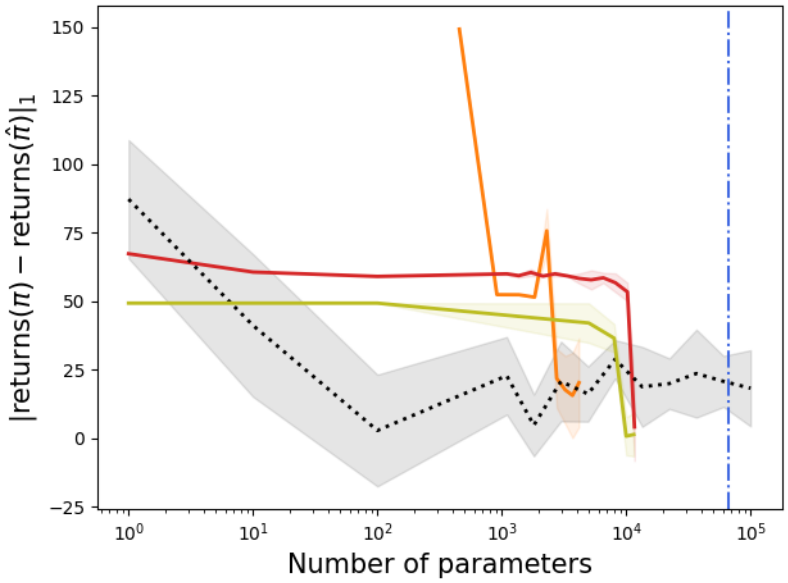

Figure 3: Difference in returns between embedded and ground truth policies for the Mountain Car task with respect of the number of parameters used.

5.3 Pendulum

The same experimental setup as for Mountain Car applies to this environment. As shown in Figure 4 over 10 runs, when the true policy has converged to a degenerate (optimal) distribution (5,000 steps), all methods show comparable performance (in terms of convergence). DFT shows better convergence at early stages of training (2,000 steps), that is when the true policy has a large variance. Refer to Appendix Figure 9-11 for visualizations of the true policies.

Task	$\sqrt{\mathbb{V}[\eta(\pi)]}$	$\sqrt{\mathbb{V}[\eta(\lceil\pi\rceil)]}$	$\sqrt{\mathbb{V}[\eta(\pi_{pruned})]}$	$\sqrt{\mathbb{V}[\eta(\hat{\pi}_K)]}$
Pendulum (2k)	114.51	110.21	107.82	84.75
Pendulum (3k)	333.06	317.73	301.46	191.83
Pendulum (4k)	363.6	351.15	312.4	273.1
Pendulum (5k)	182.2	181.9	179.10	163.3
CMC	28.1	29.2	26.1	11.0

Table 1: Variance of the collected returns in Pendulum-v0 by the original, discretized, pruned and truncated policies over 100 trials, with K set to half of its maximum value (i.e. $K = b_S b_A/2$).

Table 1 shows the standard deviation in returns collected by true and truncated policies over 100 trials. The truncated policies systematically have standard deviation of returns at

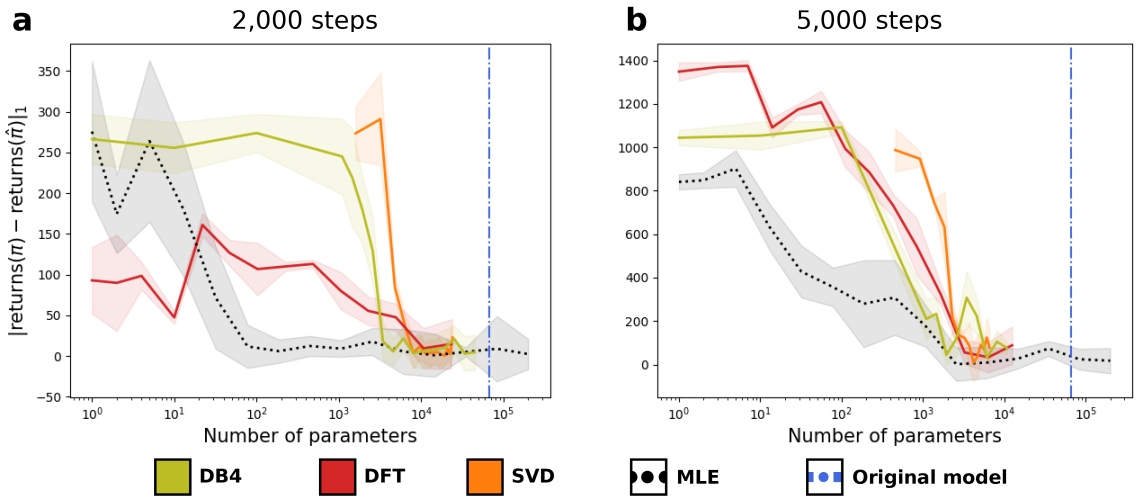

Figure 4: Absolute difference in returns collected by discretized and reconstructed Gaussian policies for **(a)** 2,000 steps and **(b)** 5,000 steps in the Pendulum task, averaged over 10 trials. Blue dots represent the number of parameters of the neural network policy. SVD, DFT and DB4 projections need an order of magnitude less in term of parameters to reconstruct the original policy.

most as large as the ones collected by the true, discretized and pruned policies. These empirical findings hint at the fact that projecting the policy onto a finite number of components has the most effect on reducing the variance of returns, which agrees with the theoretical result of Lemma 3. While the MLE estimator does have a better convergence rate in low parameter regimes, it does not scale well to higher dimensions, as there is need to perform an additional state aggregation step. It can also be noted that the variance of the returns from an MLE policy is much larger than that of policies represented within an RKHS.

6. Conclusion

In this work, we introduced a general framework for representing policies for reinforcement learning tasks. It allows to represent any continuous policy density by a projection onto the K first basis functions of a reproducing kernel Hilbert space (in practice, L^2). We showed theoretically that the performance of this embedded policy depends on the discretization, pruning and projection algorithms, and we provide upper bounds for these errors. In addition, by projecting to a lower dimensional space, our algorithm outputs a more stable policy which, under some assumptions, has at most the same variance as the original policy. Finally, we conducted experiments to demonstrate the behaviour of SVD, DFT, DB4 and GMM basis functions on a set of classical control tasks, supporting our performance guarantees. Moreover, our experiments show a reduction in the variance of the returns of the reconstructed policies, resulting in a more stable policy. A natural extension to our framework would be to directly operate in the continuous space, avoiding the discretization step procedure, which is left for future work.

Acknowledgements

This research is supported by the Canadian Institute for Advanced Research (CIFAR AI chair program). Bogdan Mazoure and Thang Doan contributed equally.

Appendix A. Appendix

Reproducibility Checklist

We follow the reproducibility checklist *"The machine learning reproducibility checklist"* and point to relevant sections explaining them here.
For all algorithms presented, check if you include:

- **A clear description of the algorithm, see main paper and included codebase.** The proposed approach is completely described by Alg. 2.
- **An analysis of the complexity (time, space, sample size) of the algorithm.** The space complexity of our algorithm depends on the number of desired basis. It is $4K$ for a 1-dimension $K-$component GMM with diagonal covariance, $mK + K^2 + nK$ for a $K-$truncated SVD of an $m \times n$ matrix, and only K for the wavelet and Fourier bases. Note that a simple implementation of SVD requires to find the three matrices $\mathbf{U}, \mathbf{D}, \mathbf{V}^\top$ before truncation.

  The time complexity depends on the reconstruction, pruning and discretization algorithms, which involves conducting rollouts w.r.t. policy π, training a quantile encoder on the state space trajectories, and finally embedding the the pruned matrix using the desired algorithm. We found that the Gaussian mixture is the slowest to train, due to shortcomings of the expectation-maximization algorithm.
- **A link to a downloadable source code, including all dependencies.** The code is included with Supplemental Files as a zip file; all dependencies can be installed using Python's package manager. Upon publication, the code would be available on Github. Additionally, we include the model's weights as well as the discretized policy for `Pendulum-v0` environment.

For all figures and tables that present empirical results, check if you include:

- **A complete description of the data collection process, including sample size.** We use standard benchmarks provided in OpenAI Gym (Brockman et al., 2016).
- **A link to downloadable version of the dataset or simulation environment.** Not applicable.
- **An explanation of how samples were allocated for training / validation / testing.** We do not use a training-validation-test split, but instead report the mean performance (and one standard deviation) of the policy at evaluation time across 10 trials.
- **An explanation of any data that were excluded.** The only data exclusion was done during policy pruning, as outlined in the main paper.

- **The exact number of evaluation runs.** 10 trials to obtain all figures, and 200 rollouts to determine ρ_π.
- **A description of how experiments were run.** See Section Experimental Results in the main paper and didactic example details in Appendix.
- **A clear definition of the specific measure or statistics used to report results.** Undiscounted returns across the whole episode are reported, and in turn averaged across 10 seeds. Confidence intervals shown in Fig. 3 were obtained using the pooled variance formula from a difference of means $t-$test.
- **Clearly defined error bars.** Confidence intervals are always mean$\pm$ 1 standard deviation over 10 trials.
- **A description of results with central tendency (e.g. mean) and variation (e.g. stddev)**. All results use the mean and standard deviation.
- **A description of the computing infrastructure used.** All runs used 1 CPU for all experiments with 8Gb of memory.

A.1 Bandit Example

An N-armed bandit has a reward distribution such that $r(i) = \frac{1}{\sqrt{2\pi}} \exp(-(i - N/2)^2/2) + U_i$ for $i = 1, .., N$, and stationary i.i.d. measurement noise U, with the additional restriction that $\mathbb{V}(U) = \lambda^2$, i.e. the signal-to-noise ratio (SNR) simplifies to SNR $= \frac{N}{2\lambda}$.

Running any suitable policy optimization method [19, 23] then produces a policy π which regresses on both the signal and the noise (blue curve in Figure 7). However, if the class of regressor functions is overparameterized, then the policy will also capture the noise, which in turn will lead to suboptimal collected rewards. If regularization is not an option (e.g. the training data was deleted due to privacy requirements as discussed in [53]), then separating signal from noise becomes more challenging.

Our approach involves projecting the policy into a Hilbert space in which noise is captured via fine-grained basis functions and therefore can be eliminated via truncation of the basis. Figure 7 illustrates the hypothetical bandit scenario, and the behavior of our approach. Executing the policy with 2 components will yield a larger reward signal than executing the original policy, because the measurement noise is removed via truncation.

A.2 Convergence Results for Fourier Approximation and GMM Mixture

Rate of convergence of Fourier approximation Let π_K^{Fourier} denote the density approximated by the first K^{th} Fourier partial sums [57], then a result from [31] shows that

$$||\pi - \pi_K^{\text{Fourier}}||_1 = O(K^{-1}) \; . \tag{18}$$

More recent results [25] provide even tighter bounds for continuous and periodic functions with $m - 1$ continuous derivatives and bounded m^{th} derivative. In such case,

$$||\pi - \pi_K^{\text{Fourier}}||_\infty = O(K^{-(2m+1)}) \; . \tag{19}$$

Rate of convergence of mixture approximation Let π_K^{MM} denote a (finite) $K-$component mixture model. A result from [46] shows the following result for π_K^{MM} in the class of mixtures with K marginally-independent scaled density functions and π in the class of lower-bounded probability density functions:

$$KL(\pi||\pi_K^{\text{MM}}) = O(K^{-1} + n^{-1}), \tag{20}$$

where n is the number of i.i.d. samples used in the learning of π_K^{MM}.

The constants hidden inside the big-O notation depend on the nature of the function and can become quite large, which can explain differences in empirical evaluation.

A.3 Discretization of Continuous Policies

Consider the case when f is a continuous, positive and decreasing function with a discrete sequence $a_n = f(n)$. For example, the reconstruction error $W_1(\Pi, \hat{\Pi})$ as well as difference in returns $|\eta(\pi) - \eta(\hat{\pi})|$ fall under this family. Then, $\int_0^\infty f(t)dt < \infty$ implies that $\sum_{i=0}^\infty a_n < \infty$. Under these assumptions, convergence guarantees (e.g. on monotonically decreasing reconstruction error) in continuous space imply convergence in the discrete (empirical) setting. Hence, we operate on discrete spaces rather than continuous ones.

All further computations of distance between two discretized policies will have to be computed over the corresponding discrete grid $(i,j), i = 1,..,b_S\ ,\ j = 1,..,b_A$. One can look at the average action taken by two MDP policies at state s:

$$\left|\mathbb{E}_{a\sim\pi}[a|s] - \mathbb{E}_{a\sim\tilde{\pi}}[a|s]\right| \leq W_1(\Pi_s, \tilde{\Pi}_s) \tag{21}$$

This relationship is one of the motivations behind using W_1 to assess goodness-of-fit in the experiments section.

Naively discretizing some function f over a fixed interval $[a,b]$ can be done by n equally spaced bins. To pass from a continuous value $f(x)$ to discrete, one would find $i \in 1,..,n$ such that $\frac{b-a}{n}i \leq x \leq \frac{b-a}{n}(i+1)$. However, using a uniform grid for a non-uniform f wastes representation power. To re-allocate bins in low density areas, we use the quantile binning method, which first computes the cumulative distribution function of f, called F. Then, it finds n points such that the probability of falling in each bin is equal. Quantile binning can be see as uniform binning in probability space, and exactly corresponds to uniform discretization if f is constant (uniform distribution).
Below, we present an example of discretizing 4,000 sample points taken from four $\mathcal{N}(\mu_i, 0.3)$ distributions, using the quantile binning.

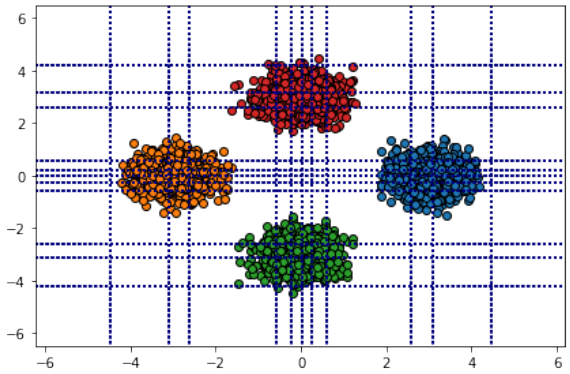

Figure 5: Quantile binning for the four Gaussians problem, using 10 bins per dimension.

In Figure 5, we are only allowed to allocate 10 bins per dimension. Note how the grid is denser around high-probability regions, while the furthermost bins are the widest. This uneven discretization, if applied to the policy density function, allows the agent to have higher detail around high-probability regions.
In Python, the function `sklearn.preprocessing.KBinsDiscretizer` can be used to easily discretize samples from the target density.

Since policy densities are expected to be more complex than the previous example, we analyze quantile binning in the discrete Mountain Car environment. To do so, we train a DQN policy until convergence (i.e. collected rewards at least -110), and perform 500 rollouts using the greedy policy. Trajectories are used to construct a 2-dimensional state visitation histogram. At the same time, all visited states are given to the quantile binning algorithm, which is used to assign observations to bins.

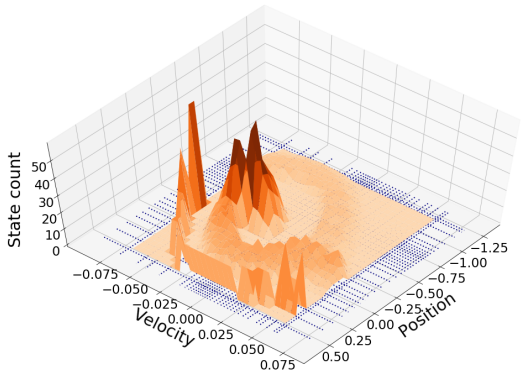

Figure 6: Quantile binning of unnormalized state visitation counts of DQN policy in the discrete Mountain Car task. Bins closer to the starting states are visited more often than those at the end of the trajectory. Blue dots represents the bins limits.

Figure 6 presents the state visitation histogram with $n_S = 30$ bins, where color represents the state visitation count. Bins are shown by dotted lines.

A.4 Acting with Tabular Policies

In order to execute the learned policy, one has to be able to (1) generate samples conditioned on a given state and (2) evaluate the log-probability of state-action pairs. Acting with a tabular density can be done via various sampling techniques. For example, importance sampling of the atoms corresponding to each bin is possible when b_A is small, while the inverse c.d.f. method is expected to be faster in larger dimensions. The process can be further optimized on a case-per-case basis for each method. For example, sampling directly from the real part of a characteristic function can be done with the algorithm defined in [17]

Furthermore, it is possible to use any of the above algorithms to jointly sample (s, a) pairs under the assumption that states were discretized by quantiles. First, uniformly sample a state bin and then use any of the conditional sampling algorithms to sample action a. Optionally, one can add uniform noise (clipped to the range of the bin) to the sampled action. This naive trick would transform discrete actions into continuous approximations of the policy network.

A.5 Bandit Motivating Example

We observe that truncating the embedded policy can have a denoising effect, i.e. boosting the signal-to-noise ratio (SNR). As a motivation, consider a multi-armed bandit setting [55] with reward distribution $r(i) = \frac{1}{\sqrt{2\pi}} \exp(-(i - N/2)^2/2) + U_i$ for $i = 1, .., N$, and stationary measurement noise U_i. By running any suitable policy optimization method [19, 23], we obtain a policy π which regresses on both the signal and the noise. Figure 7 illustrates the hypothetical bandit scenario. Applying DFT followed by truncation produces π_K, some of which are shown in Figure 7a. Sampling actions from π_K yields a lower average regret up to K=10, because the measurement noise is truncated and the correct signal mode is recovered.

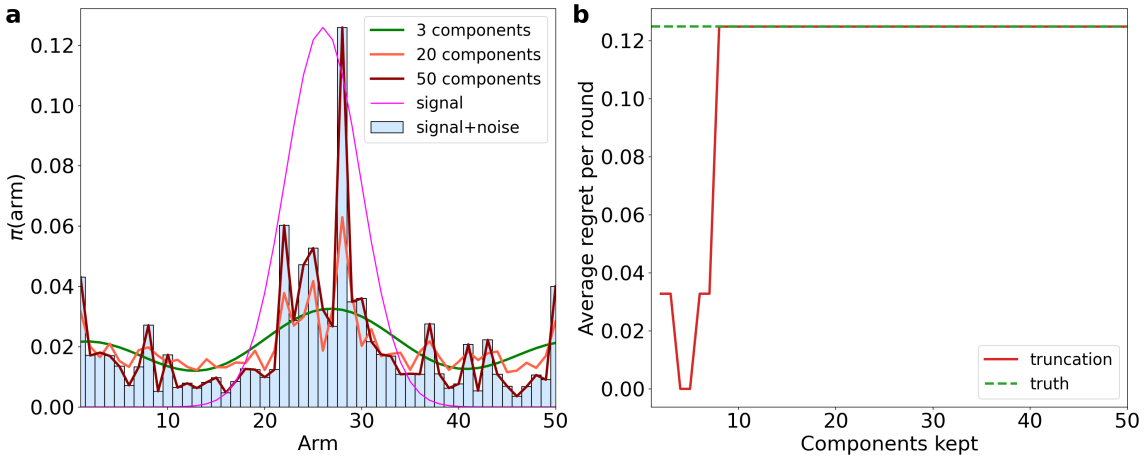

Figure 7: **(a)** Motivating bandit example with ground truth signal, noisy signal and truncated policy, and **(b)** rewards collected by greedy (i.e. the mode) truncated and true policies

The bandit performance bound can be stated as follows:

Lemma 11. *Let $\mathcal{H}$ be an RKHS, $\pi_1, \pi_2 \in \mathcal{H}$ be represented by $\{\xi_k^{\pi_1}\}_{k=1}^{\infty}$ and $\{\xi_k^{\pi_2}\}_{k=1}^{\infty}$ and $r : \mathcal{A} \to \mathbb{R}$ be the bandit's reward function. Let $M > 0$ be such that $|\pi_1(a) - \pi_2(a)| \leq M||\pi_1 - \pi_2||_{\mathcal{H}}$ for all $a \in \mathcal{A}$ and let $p, q > 0$ s.t. $\frac{1}{p} + \frac{1}{q} = 1$. Then*

$$|\eta(\pi_1) - \eta(\pi_2)| \leq |\mathcal{A}|^{1/q} M ||r||_p \sum_{k=1}^{\infty} \frac{(\xi_k^{\pi_1} - \xi_k^{\pi_2})^2}{\lambda_k}$$

Lemma 11 guarantees that rewards collected by π_2 will depend on the distance between π_1 and π_2. The proof can be found in the next section.

A.6 Proofs

Proof. (Lemma 3) Throughout the proof, we will be using the expectation $\mathbb{E}_\beta$ which can be seen as the classical inner product weighted by the initial state-action distribution

$$\langle f, g \rangle_\beta = \int_{s,a} f(s,a) g(s,a) \beta(s,a) d(sa)$$

over the domain of s, a.

If we take a uniform initialization density, i.e. $\beta(s,a) \propto (|\mathcal{S}||\mathcal{A}|)^{-1}$, then the inner product simplifies to

$$\langle f, g \rangle_\beta = (|\mathcal{S}||\mathcal{A}|)^{-1} \langle f, g \rangle,$$

which will be used further on to simplify the covariance term between orthonormal basis functions.

After a first-order Talor expansion of variances around $\eta(\pi)$ and $\eta(\hat{\pi}_K)$ described in [9], we obtain

$$\begin{aligned}\mathbb{V}_\beta[\pi(a|s)] &= \{\sigma'(\eta(\pi))\}^2\mathbb{V}_\beta[Q^\pi(s,a)] + R_2^\pi \\ \mathbb{V}_\beta[\hat{\pi}_K(a|s)] &= \{\sigma'(\eta(\hat{\pi}_K))\}^2\mathbb{V}_\beta[Q^{\hat{\pi}_K}(s,a)] + R_2^{\hat{\pi}_K},\end{aligned} \tag{22}$$

where R_2^π is the Taylor residual of order 2 for policy π. This identity is very widely used in statistics, especially for approximating the variance of functions of statistics through the *delta method* [65].

Expanding the variance of sum of correlated random variables, we obtain

$$\mathbb{V}_\beta[\pi(a|s)] = \mathbb{V}_\beta[\hat{\pi}_K(a|s)] + \mathbb{V}_\beta[\varepsilon_K(a|s)] + 2\mathbb{V}_\beta[\hat{\pi}_K(a|s), \varepsilon_K(a|s)] \tag{23}$$

The covariance term $\mathbb{V}_\beta[\hat{\pi}_K(a|s), \varepsilon_K(a|s)]$ can be decomposed as $\mathbb{E}_\beta[\hat{\pi}_K(a|s)\varepsilon_K(a|s)] - \mathbb{E}_\beta[\hat{\pi}_K(a|s)]\mathbb{E}_\beta[\varepsilon_K(a|s)]$.

Then, the second moment expression of the covariance becomes zero with respect to this inner product, due to the use of orthonormal basis functions. We use the dominated convergence theorem which lets the order of summation be changed for $\sum_{j=K}^{\infty}\xi_j\omega_j(s,a)$, which greatly simplifies the calculations.

$$\begin{aligned}\mathbb{E}_\beta[\hat{\pi}_K(a|s)\varepsilon_K(a|s)] &= \mathbb{E}_\beta[\sum_{k=1}^{K}\xi_k\omega_k(s,a)\sum_{j=K}^{\infty}\xi_j\omega_j(s,a)] \\ &= \sum_{k=1}^{K}\sum_{j=K}^{\infty}\xi_k\xi_j\mathbb{E}_\beta[\omega_k(s,a)\omega_j(s,a)] \\ &= \sum_{k=1}^{K}\sum_{j=K}^{\infty}\xi_k\xi_j\langle\omega_k,\omega_j\rangle_\beta \\ &= 0 \quad (\omega_k(s,a) \text{ and } \omega_j(s,a) \text{ are orthogonal })\end{aligned} \tag{24}$$

Combining both expressions yields the relationship between the variance of the truncated policy and that of the expected returns.

$$\begin{aligned}\mathbb{V}_\beta[Q^\pi] &= \left[\frac{\{\sigma'(\eta(\hat{\pi}_K))\}^2\mathbb{V}_\beta[Q^{\hat{\pi}_K}] - 2\mathbb{E}_\beta[\hat{\pi}_K]\mathbb{E}_\beta[\varepsilon_K] + \mathbb{V}_\beta[\varepsilon_K] + E_2^{\hat{\pi}_K}}{\{\sigma'(\eta(\pi))\}^2}\right] + R_2^\pi \\ &= \left[\frac{\{\sigma'(\eta(\hat{\pi}_K))\}^2\mathbb{V}_\beta[Q^{\hat{\pi}_K}] - 2\mathbb{E}_\beta[\hat{\pi}_K]\mathbb{E}_\beta[\varepsilon_K] + \mathbb{V}_\beta[\varepsilon_K]}{\{\sigma'(\eta(\pi))\}^2}\right] + R_2^\pi + \frac{R_2^{\hat{\pi}_K}}{\{\sigma'(\eta(\pi))\}^2}\end{aligned} \tag{25}$$

If the Taylor expansion is performed in a neighborhood of $\eta(\pi)$ and $\eta(\hat{\pi}_K)$, we can then consider the error terms as sufficiently small and neglect them, as is often the case in classical statistics [9].

$$\mathbb{V}_\beta[Q^\pi] \approx \left(\frac{\sigma'(\eta(\hat{\pi}_K))}{\sigma'(\eta(\pi))}\right)^2\mathbb{V}_\beta[Q^{\hat{\pi}_K}] + \frac{\mathbb{V}_\beta[\varepsilon_K] - 2\mathbb{E}_\beta[\hat{\pi}_K]\mathbb{E}_\beta[\varepsilon_K]}{\{\sigma'(\eta(\pi))\}^2} \tag{26}$$

For the variance of the returns of the true policy to be not lower than the variance of returns of the truncated policy, the coefficient of the first term on the right hand side must be at most 1, and the second term must be positive. Note that these are sufficient but not necessary conditions.

Precisely, the following conditions must be satisfied:

1. $\sigma'(\eta(\hat{\pi}_K)) \leq \sigma'(\eta(\pi))$;

2. $\sqrt{\frac{\mathbb{E}_\beta[\varepsilon_K^2]}{3}} \geq \mathbb{E}_\beta[\varepsilon_K]$. This is an assumption on the second moment behavior of the policy and is satisfied by, for example, the Student's t distribution with degrees of freedom $\nu > 1$.

3. The Taylor error terms R_2^π and $R_2^{\hat{\pi}_K}$ are small since the expansion is performed around $\eta(\pi)$ and $\eta(\hat{\pi}_K)$, respectively.

That is, under various initializations, truncation of policy coefficients can be beneficial by reducing the variance.

□

Proof. (Theorem 6) We want to quantify the difference between the continuous cdf Π and the step-function defined by the quantile binning strategy, $\lfloor \Pi \rfloor$, with b bins. Notice that the error between Π and $\lfloor \Pi \rfloor$ in bin i can be written as

$$\delta_i = \int_{q_{i-1}}^{q_i} \Pi(x)dx - \frac{q_i - q_{i-1}}{b}, \tag{27}$$

where $q_i = \hat{\mathcal{Q}}\big(\frac{i}{b}\big)$.

The total error across all bins is therefore

$$\delta = \sum_{i=1}^{b} \Big[\int_{q_{i-1}}^{q_i} \Pi(x)dx - \frac{q_i - q_{i-1}}{b} \Big], \tag{28}$$

The error δ is computed across both states and actions, meaning that the total volume of the error between the policies can be thought of as error in $\mathcal{A}$ made for a single state group, which leads to the state approximation error being considered b_S times, akin to conditional probabilities.

The state-action total error can be computed as follows

$$\delta_{a,s} = \sum_{i=1}^{b_S} \Big[\int_{q_{i-1}^S}^{q_i^S} \Pi_S(s)ds - \frac{i(q_i^S - q_{i-1}^S)}{b_S} \Big] \sum_{j=1}^{b_A} \Big[\int_{q_{i,j-1}^A}^{q_{i,j}^A} \Pi_A(a)da - \frac{j(q_{i,j}^A - q_{i,j-1}^A)}{b_A} \Big], \tag{29}$$

where superscripts and subscripts A and S denotes dependence of the quantity on either action or state, respectively. The notation $q_{i,j}$ refers to the quantile $\hat{\mathcal{Q}}(\frac{i}{b})$ computed conditional on the state falling into bin i. □

Proof. (Lemma 11)

$$\begin{aligned}\Big|\eta(\pi_1) - \eta(\pi_2)\Big| &= \Big|\sum_{a\in\mathcal{A}} r(a)\Big[\pi_1(a) - \pi_2(a)\Big]\Big| \\ &= \Big|\sum_{a\in\mathcal{A}} r(a)\Delta(a)\Big| \\ &\leq \sum_{a\in\mathcal{A}}\Big|r(a)\Delta(a)\Big| \\ &= ||r\Delta||_1 \\ &\leq ||r||_p||\Delta||_q \\ &\leq |\mathcal{A}|^{1/q}M||r||_p\sum_{k=1}^{\infty}\frac{(\xi_k^{\pi_1} - \xi_k^{\pi_2})^2}{\lambda_k}\end{aligned}$$

where the before last line is a direct application of Holder inequality and the last line happens because evaluation is a bounded linear functional such that $|f(x)| \leq M||f||_{\mathcal{H}}$ for all $f \in \mathcal{H}$. □

Proof. (Theorem 4) Recall the following result from [32, 51]:

$$|\eta(\pi_1) - L_{\pi_2}(\pi_1)| \leq \frac{4\alpha^2\gamma\bar{\epsilon}}{(1-\gamma)^2}, \tag{30}$$

where $\bar{\epsilon} = \max_{s,a}|Q_{\pi_2}(s,a) - V_{\pi_2}(s)|$ and $\alpha = \max_s TV(\pi_1(\cdot|s), \pi_2(\cdot|s))$.

Instead, suppose that for every fixed state $s \in \mathcal{S}$, there exists an RKHS of all square-integrable conditional densities of the form $\pi(\cdot|s)$. We examine the difference term between policies in this $\mathcal{H}_s$. Since $\pi_1, \pi_2 \in \mathcal{H}_s$, $\delta_{1,2} = \pi_1 - \pi_2 \in \mathcal{H}_s$ and hence there exist $M_s > 0$ such that $|\delta_{1,2}(a)| \leq M_s||\delta_{1,2}||_{\mathcal{H}_s}$ for all $a \in \mathcal{A}$. Then,

$$\begin{aligned}\sum_{a\in\mathcal{A}}|\pi_1(a) - \pi_2(a)| &\leq \sum_{a\in\mathcal{A}}|\delta_{1,2}(a)| \\ &\leq M_s\sum_{a\in\mathcal{A}}||\delta_{1,2}||_{\mathcal{H}_s} \\ &\leq |\mathcal{A}|M_s\sum_{k=1}^{\infty}\frac{(\xi_k^{\pi_1} - \xi_k^{\pi_2})^2}{\lambda_k} \\ &= |\mathcal{A}|M_s\Delta_{\mathcal{H}_s}^{\infty},\end{aligned} \tag{31}$$

where we let $\Delta_{\mathcal{H}_s}^{\infty} = \sum_{k=1}^{\infty}\frac{(\xi_k^{\pi_1} - \xi_k^{\pi_2})^2}{\lambda_k}$ for shorter notation.

Then,

$$|\eta(\pi_1) - L_{\pi_2}(\pi_1)| \leq \frac{4|\mathcal{A}|^2\gamma\bar{\epsilon}}{(1-\gamma)^2}(\max_{s\in\mathcal{S}}M_s\Delta_{\mathcal{H}_s}^{\infty})^2, \tag{32}$$

We can obtain a performance bound by first expanding the left hand side

$$L_{\pi_2}(\pi_1) = \eta(\pi_2) + \sum_s \rho_{\pi_2}(s)\sum_a \pi_1(a|s)A_{\pi_2}(s,a)$$

Using Eq. (30), and using reverse triangle inequality, we have

$$\left|\eta(\pi_2)-\eta(\pi_1)\right| - \left|\sum_s \rho_{\pi_2}(s)\sum_a \pi_1(a|s)A_{\pi_2}(s,a)\right| \leq \left|\eta(\pi_2) - \left\{\eta(\pi_1)+\sum_s \rho_{\pi_2}(s)\sum_a \pi_1(a|s)A_{\pi_2}(s,a)\right\}\right| \leq \frac{4\alpha^2\gamma\bar{\varepsilon}}{(1-\gamma)^2}$$

Therefore,

$$\left|\eta(\pi_2)-\eta(\pi_1)\right| \leq \frac{4\alpha^2\gamma\bar{\epsilon}}{(1-\gamma)^2} + \left|\sum_s \rho_{\pi_2}(s)\sum_a \pi_1(a|s)A_{\pi_2}(s,a)\right| \tag{33}$$

We now proceed to bound the second term on the RHS:

$$\begin{aligned}
\mathbb{E}_{s\sim\rho_{\pi_2}}[\mathbb{E}_{a\sim\pi_1}[A^{\pi_2}(s,a)]] &= \mathbb{E}_{s\sim\rho_{\pi_2}}[\sum_a \pi_1(a|s)A^{\pi_2}(s,a)] \\
&= \mathbb{E}_{s\sim\rho_{\pi_2}}[\sum_a (\pi_2(a|s)+\delta_{1,2}(s,a))A^{\pi_2}(s,a)] \\
&= \underbrace{\mathbb{E}_{s\sim\rho_{\pi_2}}[\sum_a \pi_2(a|s)A^{\pi_2}(s,a)]}_{=0} + \mathbb{E}_s[\sum_a \delta_{1,2}(s,a)A^{\pi_2}(s,a)] \\
&= \mathbb{E}_{s\sim\rho_{\pi_2}}[\sum_a \delta_{1,2}(s,a)A^{\pi_2}(s,a)] \\
&\leq \mathbb{E}_{s\sim\rho_{\pi_2}}[\sum_{a\in\mathcal{A}} |\delta_{1,2}(s,a)|A^{\pi_2}(s,a)] \\
&\leq \mathbb{E}_{s\sim\rho_{\pi_2}}[M_s\Delta^{\infty}_{\mathcal{H}_s}\sum_{a\in\mathcal{A}} A^{\pi_2}(s,a)]
\end{aligned} \tag{34}$$

We combine Eq. (31) with Eq. (33) to obtain the following RKHS form on the improvement lower bound.

$$|\eta(\pi_2)-\eta(\pi_1)| \leq \frac{4|\mathcal{A}|^2\bar{\epsilon}\gamma}{(1-\gamma)^2}(\max_{s\in\mathcal{S}} M_s\Delta^{\infty}_{\mathcal{H}_s})^2 + \mathbb{E}_{\rho_{\pi_2}}[M_s\Delta^{\infty}_{\mathcal{H}_s}\sum_{a\in\mathcal{A}} A^{\pi_2}(s,a)] \tag{35}$$

If we suppose that weights after K are small enough, we can split the approximation error using Lemma 2 for $R_K = \frac{\varepsilon^{2(K+1)}}{1-\varepsilon^2}$:

$$M_s\Delta^{\infty}_{\mathcal{H}_s} \leq M_s\Big(\Delta^{K}_{\mathcal{H}_s} + R_K\Big). \tag{36}$$

Using this in Eq. (35) yields the final result:

$$|\eta(\pi_2)-\eta(\pi_1)| \leq \frac{4|\mathcal{A}|^2\bar{\epsilon}\gamma}{(1-\gamma)^2}\Big(\max_{s\in\mathcal{S}}(M_s\Delta^{K}_{\mathcal{H}_s})^2 + 2R_K\max_{s\in\mathcal{S}} M_s\Delta^{K}_{\mathcal{H}_s} + R_K^2\max_{s\in\mathcal{S}} M_s^2\Big) + \mathbb{E}_{\rho_{\pi_2}}[M_s\Delta^{\infty}_{\mathcal{H}_s}\sum_{a\in\mathcal{A}} A^{\pi_2}(s,a)] \tag{37}$$

$$|\eta(\pi_2) - \eta(\pi_1)| \leq \frac{4|\mathcal{A}|^2\bar{\epsilon}\gamma}{(1-\gamma)^2}\left\{\max_{s\in\mathcal{S}}(M_s\Delta_{\mathcal{H}_s}^K)^2 + O(\varepsilon^{2K}\max_{s\in\mathcal{S}}M_s\Delta_{\mathcal{H}_s}^K)\right\} + \mathbb{E}_{\rho_{\pi_2}}[M_s\Delta_{\mathcal{H}_s}^\infty\sum_{a\in\mathcal{A}}A^{\pi_2}(s,a)] \tag{38}$$

$$\leq \mathbb{E}_{\rho_{\pi_2}}[M_s\Delta_{\mathcal{H}_s}^\infty\sum_{a\in\mathcal{A}}A^{\pi_2}(s,a)]\ . \tag{39}$$

Where the last line comes from the fact that π_2 is the projection of π_1 onto the first top K bases of $\mathcal{H}_s$ hence $\Delta_{\mathcal{H}_s}^K = 0$

□

Remark: If ε is close to zero (i.e. π_1 and π_2 have very similar $\xi_K, \xi_{K+1}..$), then the expression simplifies to

$$|\eta(\pi_2) - \eta(\pi_1)| \leq \frac{4|\mathcal{A}|^2\bar{\epsilon}\gamma}{(1-\gamma)^2}\max_{s\in\mathcal{S}}(M_s\Delta_{\mathcal{H}_s}^K)^2 + \mathbb{E}_{\rho_{\pi_2}}[M_s\Delta_{\mathcal{H}_s}^\infty\sum_{a\in\mathcal{A}}A^{\pi_2}(s,a)]\ . \tag{40}$$

Proof. (Theorem 8) We first represent the approximate transition matrix $\mathbf{P}_{\tilde{\pi}}$ induced by the policy $\tilde{\pi}$ as a perturbation of the true transition:

$$\begin{aligned}\mathbf{P}_{\tilde{\pi}} &= \mathbf{P}_\pi + (\mathbf{P}_{\tilde{\pi}} - \mathbf{P}_\pi)\\ &= \mathbf{P}_\pi + \mathbf{E}\end{aligned} \tag{41}$$

Then, the difference between stationary distributions $\rho_{\tilde{\pi}}$ and ρ_π is equal to [52, 15]:

$$\rho_\pi^\top - \rho_{\tilde{\pi}}^\top = \rho_{\tilde{\pi}}^\top\mathbf{E}\mathbf{Z}, \tag{42}$$

where $\mathbf{Z} = (\mathbf{I} - \mathbf{P}_\pi + \mathbf{1}_{|\mathcal{S}|}\rho_\pi^\top)^{-1}$ is the *fundamental matrix* of the Markov chain induced by $\mathbf{P}_\pi$ and $\mathbf{1}_{|\mathcal{S}|}$ is a vector of ones.

In particular, the above result holds for Schatten norms [7]:

$$||\rho_\pi - \rho_{\tilde{\pi}}||_2 \leq ||\rho_{\tilde{\pi}}||_2||\mathbf{Z}||_{S_\infty}||\mathbf{E}||_{S_\infty} \leq ||\mathbf{Z}||_{S_\infty}||\mathbf{E}||_{S_1} \tag{43}$$

So far, this result is known for irreducible, homogeneous Markov chains and has no decision component.

Consider the matrix $\mathbf{E}$, which is the difference between expected transition models of true and approximate policies. It can be expanded into products of matricized tensors:

$$\begin{aligned}\mathbf{E} &= \mathbf{P}_{\tilde{\pi}} - \mathbf{P}_\pi\\ &= \mathbf{A}(\tilde{\mathbf{\Pi}}\otimes\mathbf{I})\mathbf{T}_{(3)} - \mathbf{A}(\mathbf{\Pi}\otimes\mathbf{I})\mathbf{T}_{(3)}^\top\\ &= \mathbf{A}((\tilde{\mathbf{\Pi}} - \mathbf{\Pi})\otimes\mathbf{I})\mathbf{T}_{(3)}^\top\end{aligned} \tag{44}$$

The norm of $\mathbf{E}$ can also be upper bounded as follows:

$$\begin{aligned}||\mathbf{E}||_{S_1} &\leq ||\mathbf{A}||_{S_1}||(\tilde{\mathbf{\Pi}} - \mathbf{\Pi})\otimes\mathbf{I}\mathbf{T}_{(3)}^\top||_{S_\infty}\\ &\leq ||\mathbf{A}||_{S_1}||(\tilde{\mathbf{\Pi}} - \mathbf{\Pi})\otimes\mathbf{I}\mathbf{T}_{(3)}^\top||_{S_1}\\ &\leq ||\mathbf{A}||_{S_1}||(\tilde{\mathbf{\Pi}} - \mathbf{\Pi})\otimes\mathbf{I}||_{S_2}||\mathbf{T}_{(3)}^\top||_{S_2}\\ &\leq ||\mathbf{A}||_{S_1}||(\tilde{\mathbf{\Pi}} - \mathbf{\Pi})\otimes\mathbf{I}||_{S_1}||\mathbf{T}_{(3)}^\top||_{S_2}\\ &\leq ||\mathbf{A}||_{S_1}||\tilde{\mathbf{\Pi}} - \mathbf{\Pi}||_{S_1}||\mathbf{I}||_{S_\infty}||\mathbf{T}_{(3)}^\top||_{S_2}\\ &= ||\tilde{\mathbf{\Pi}} - \mathbf{\Pi}||_{S_1}||\mathbf{T}_{(3)}||_{S_2}\end{aligned} \tag{45}$$

Combining this result from that of [52] yields

$$||\rho_\pi - \rho_{\tilde{\pi}}||_2 \leq \underbrace{||\mathbf{Z}||_{S_\infty}}_{\substack{\text{Depends on} \\ \text{MDP} + \pi}} \underbrace{||\tilde{\mathbf{\Pi}} - \mathbf{\Pi}||_{S_1}}_{\text{Depends on } \pi} \underbrace{||\mathbf{T}_{(3)}||_{S_2}}_{\text{Depends on MDP}} \tag{46}$$

□

Proof. (Corollary 9)

Take two policies π, π'. Let $\boldsymbol{r}^{\pi'} = \mathbb{E}_\pi[r'] \leq \mathbb{E}_\pi[r] = \boldsymbol{r}^\pi$ without loss of generality. Denote $\mathbf{r}_2^\pi = ||\mathbf{r}^\pi||_2$. Observe that

$$|\rho_\pi \boldsymbol{r}^\pi - \rho_{\pi'} \boldsymbol{r}^{\pi'}| = |(\rho_\pi - \rho_{\pi'})\boldsymbol{r}^\pi + \rho_{\pi'}(\boldsymbol{r}^\pi - \boldsymbol{r}^{\pi'})| \leq ||\rho_\pi - \rho_{\pi'}||_2 ||\boldsymbol{r}^\pi||_2 + ||\rho_{\pi'}||_2 ||\boldsymbol{r}^\pi - \boldsymbol{r}^{\pi'}||_2$$

Note that when b_S, b_A are large, then $||\boldsymbol{r}^\pi - \boldsymbol{r}^{\pi'}||_2$ is small, and the equation simplifies to

$$|\eta(\pi) - \eta(\pi')| \leq ||\rho_\pi - \rho_{\pi'}||_2 \boldsymbol{r}_2^\pi \tag{47}$$

and similarly, we get the following bound from Schatten norm definition

$$||\rho_\pi - \rho_{\pi'}||_2 \leq ||\mathbf{Z}||_{S_\infty} E_{\Pi, \lceil \Pi' \rceil}(b_S, b_A) ||\mathbf{T}_{(3)}||_{S_2} \tag{48}$$

Plugging the last expression into the previous one yields the result. □

A.7 Time Complexity Comparison for Different Approximation Methods

The time complexity for a k-truncated SVD decomposition of a matrix $A \in \mathbb{R}^{n \times m}$ is $O(mnk)$ as discussed in [18]. Fast Fourier transform can be done in $O(mn \log(mn))$ for the 2-dimensional case [38]. Hence, SVD is expected to be faster whenever $k < \log(nm)$. The discrete wavelet transform's (DWT) time complexity depends on the choice of mother wavelets. When using Mallat's algorithm, the 2-dimensional DWT is known to have complexities ranging from $O(n^2 \log n)$ to as low as $O(n^2)$, for a square matrix of size $n \times n$ and depending on the choice of mother wavelet [47]. Finally, we expect FFT and Daubechies wavelets to have less parameters than SVD, since the former use pre-defined orthonormal bases, while the later must store the left and right singular vectors.

A.8 Rate of Convergence of Stationary Distribution Under Random Policy

We validate the rate of convergence of Thm. 8 in the toy experiment described below. Consider a deterministic chain MDP with N states. A fixed policy in state i transitions to $i+1$ with probability α and to $i-1$ with probability $1-\alpha$. If in states 1 or N, the agent remains there with probability $1-\alpha$ and α, respectively. We consider the expected transition model $\mathbf{P}_\pi$ obtained using the policy reconstructed through discrete Fourier transform. A visualization of Theorem 8 is shown in Fig. 8.

A.9 Experiment Details

This subsection covers all aspects of the conducted experiments, including pseudocode, additional results and details about the setup.

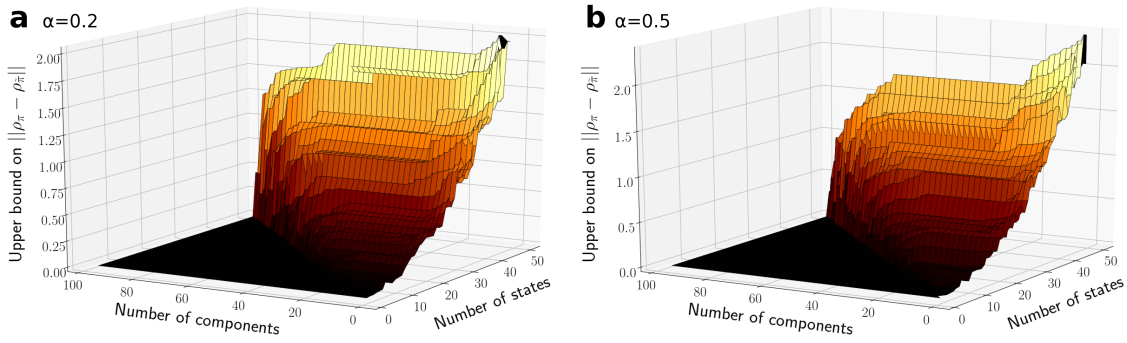

Figure 8: Upper bound on $||\rho_\pi - \rho_{\tilde{\pi}}||$ in the chain MDP task as a function of number of Fourier components and of the number of states.

A.9.1 Python Pseudocode

Below is the Python-like pseudocode for the case of Fourier bases (which can be easily adapted to all bases examined in this paper). This pseudocode is heavily simplified Python, and skips some bookkeeping steps such as edge cases, but convey the overall flow of the algorithm.

```
def RKHS_policy_step_1_and_2(cts_pi,env,n_trajectories,n_episode_steps,b_S,b_A):
    """
    cts_pi: Object with 2 methods:
        - sample(state): samples an action from pi(.|state);
        - log_prob(state,action): returns pi(a|s);
    env: OpenAI Gym environment;
    n_trajectories: Number of trajectories in rollouts (positive integer);
    n_episode_steps: Number of maximum steps in each episode (positive integer);
    b_S: number of state bins per dimension (positive integer);
    b_A: number of action bins per dimension (positive integer).
    """
    from sklearn.preprocessing import KBinsDiscretier
    states,actions,rewards = rollout(cts_pi,env.n_trajectories,n_episode_steps)
    # We use quantile in the paper but KMeans can also be used to find the bins
    enc_A = KBinsDiscretizer(n_bins=b_A,encode='ordinal',strategy='quantile')
    enc_S = KBinsDiscretizer(n_bins=b_S,encode='ordinal',strategy='quantile')
    bins_A = enc_A.fit_transform(actions)
    bins_S = enc_S.fit_transform(states)

    dsc_pi = np.zeros(shape=(b_S,b_A))
    for i in range(b_S):
        for j in range(b_A):
            s,a = enc_S.inverse_transform([i]), enc_A.inverse_transform([j])
```

```
            dsc_pi[i,j] = np.exp(cts_pi.log_prob(s,a))
        dsc_pi[i,:] /= np.sum(dsc_pi[i,:) # re-normalize every conditional pmf

    rho_pi = np.zeros(shape=(b_S,))
    for bin in bins_S:
        rho_pi[bin] += 1
    rho_pi /= np.sum(rho_pi)

    return dsc_pi, rho_pi
```

```
def RKHS_policy_step_3(dsc_pi,rho_pi):
    """
    dsc_pi: np.array of size b_S x b_A;
    rho_pi: np.array of length b_S.
    """
    idx_to_keep = np.where(rho_pi>0)

    pruned_dsc_pi = dsc_pi[idx_to_keep]

    def pruned_agent(state_bin):
        actions = np.arange(0,b_A)
            if state_bin in idx_to_keep:
                # probabilities from the lattice
                probs = pruned_dsc_pi[idx_to_keep==state_bin]
            else:
                # uniform probabilities
                probs = actions*0+1/actions.size
            return np.random.choice(actions,1,p=probs)

    return pruned_agent
```

```
def RKHS_policy_step_4(dsc_pi,K):
    """
    dsc_pi: np.array of size b_S x b_A;
    K: number of components to keep (positive integer).
    """
    FS = np.fft.fftn(dsc_pi) # transform to Fourier domain
    fshift = np.real(np.fft.fftshift(FS)).flatten() # vector of size b_S x b_A
    # find threshold s.t. exactly K components are not zeroed out
    threshold = sorted(flat_fshift,reverse=True)[K]
    mask = np.real(fshift >= th).astype(int)
    fshift *= mask # apply the binary mask to zero out \xi_K,\xi_K+1,...
```

```
        f_ishift = np.fft.ifftshift(fshift) # inverse shift of truncated spectrum
        f_dft = np.real(np.fft.ifftn(f_ishift))
```

A.9.2 Bandit turntable

We consider a bandit problem in which rewards are spread in a circle as shown in Figure 2 a). Actions consist of angles in the interval $[-\pi, \pi]$. Given the corresponding Boltzmann policy, we compare the reconstruction quality of the discrete Fourier transform and GMM. In addition to Fourier and GMM, we also compare with fixed basis GMM, where the means are sampled uniformly over $[-\pi, \pi]$ and variance is drawn uniformly from $\{0.1, 0.5, 1, 2\}$; only mixing coefficients are learned. This provides insight into whether the reconstruction algorithms are learning the policy, or if π is so simple that a random set of Gaussian can already approximate it well.

A.9.3 Experimental parameters

All parameters were chosen using a memory-performance trade-off heuristic. For example, for Continuous Mountain Car, we based ourselves off Figure 15 to select $b_S = 35$.

Hyperparameters	Turntable	Pendulum	Continuous Mountain Car
b_A	100	15	10
b_S	N/A	35	35
Optimizer	Adam		
Architecture	256		
Learning rate	1e-03		
Hidden dimension	256		
# rollouts	100		
Torch seeds	0 to 9		

Table 2: Hyperparameters used across experiments, N/A: not applicable

A.9.4 Pendulum

This classical mechanic task consists of a pendulum which needs to swing up in order to stay upright. The actions represent the joint effort (torque) between -2 and 2 units. We train SAC until convergence and save snapshots of the actor and critic after $2k$ and $5k$ steps. The reconstruction task is to recover both the Gaussian policy (actor) and the Boltzmann-Q policy (critic, temperature set to 1).

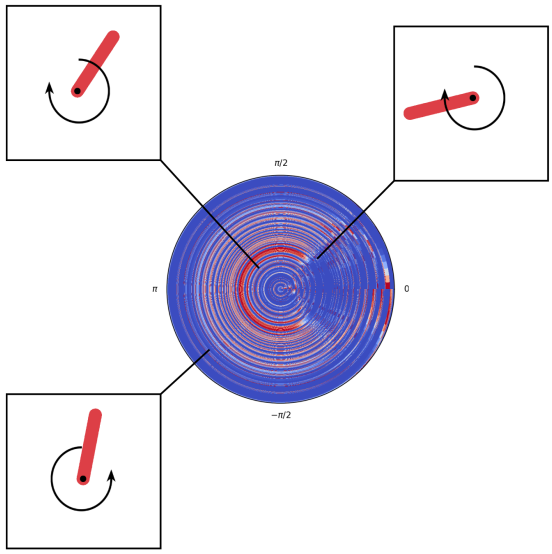

Figure 9: Plots of Gaussian Π after $5k$ training steps in polar coordinates. Each circle of different radius represents a states in the `Pendulum` environment as shown by the snapshots. Higher intensity colors (red) represent higher density mass on the given angular action.

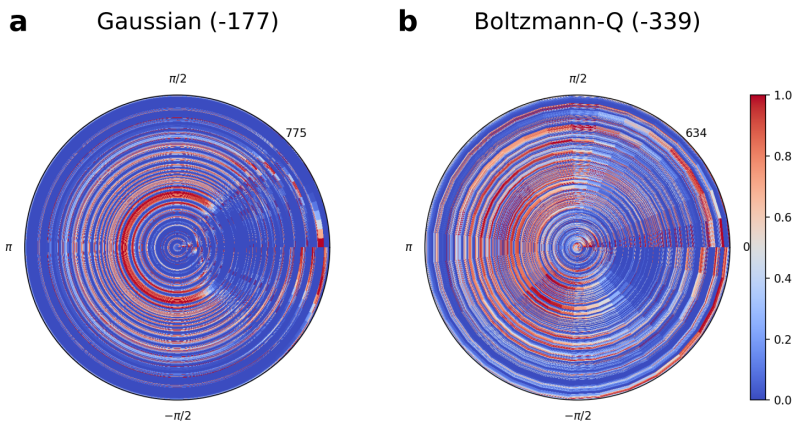

Figure 10: Plots of discretized and pruned (a) Gaussian policy and (b) Boltzmann-Q policy in polar coordinates for 5 thousands training steps of soft actor-critic. Rewards collected by their respective continuous networks are indicated in parentheses. Each circle of radius s corresponds to $\pi(\cdot|s)$ for a discrete $s = 1, 2, .., 750$. All densities are on the same scale.

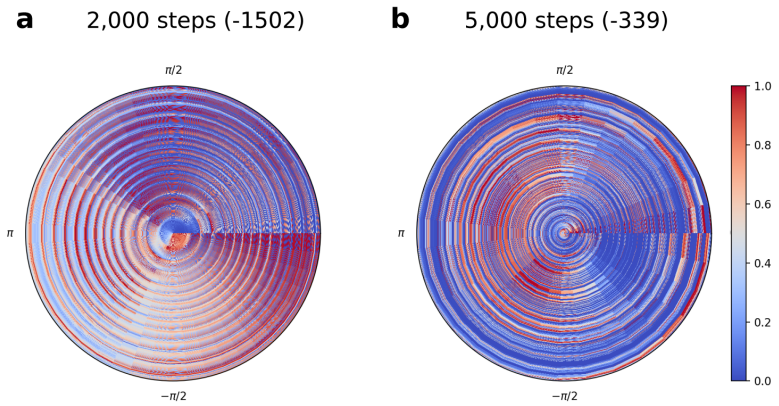

Figure 11: Plots of discretized and pruned Boltzmann policy in polar coordinates for 2 and 5 thousands training steps of soft actor-critic. Each circle of radius s corresponds to $\pi(\cdot|s)$ for a discrete $s = 1, 2, .., 750$.

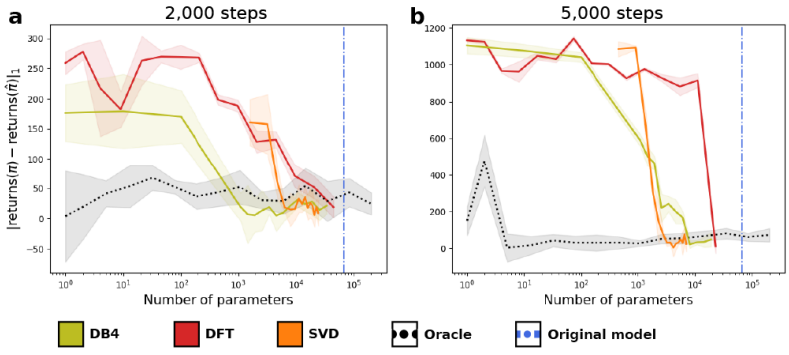

Figure 12: Absolute difference in returns collected by discretized and reconstructed Boltzmann-Q policies for *Pendulum-v0*, averaged over 10 trials. Blue dots represent the number of parameters of the neural network policy. SVD, DFT and DB4 projections need an order of magnitude less in term of parameters to reconstruct the original policy.

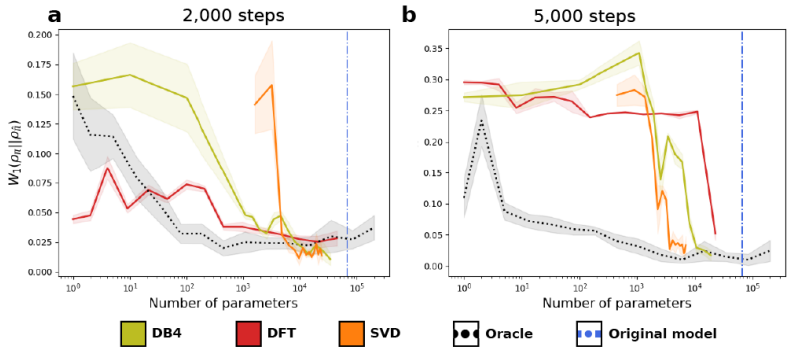

Figure 13: W_1 distance between the true and reconstructed stationary Boltzmann-Q distributions, averaged over 10 trials for *Pendulum-v0*.

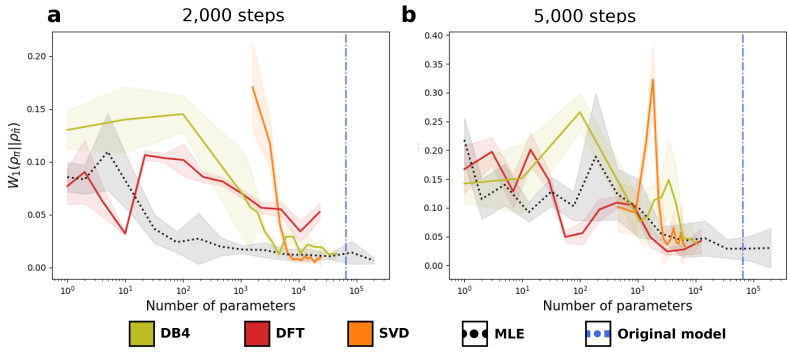

Figure 14: W_1 distance between the true and reconstructed stationary distributions, averaged over 10 trials. SVD, DFT and DB4 methods show a fast convergence to the oracle's stationary distribution.

A.9.5 Mountain Car

In this environment, the agent (a car) must reach the flag located at the top of a hill. It needs to go back and forth in order to generate sufficient momentum to reach the top. The agent is allowed to apply a speed motion in the interval $[-1, 1]$. We use Soft Actor-Critic (SAC, [30]) to train the agent until convergence ($25k$ steps).

Figure 15a shows that the error between the true and truncated policies steadily decreases as a function of projection weights kept. Note that SVD's parameters are computed as total entrie in the matrices U, D, V, and hence the smallest possible rank (rank 1) of D dictates the minimal number of parameters. Figure 15b-c show how the discretization and prunning performances behave as a function of trajectories (for pruning, since it relies on trajectories to estimate ρ_π), or bins per state dimension (for discretization).

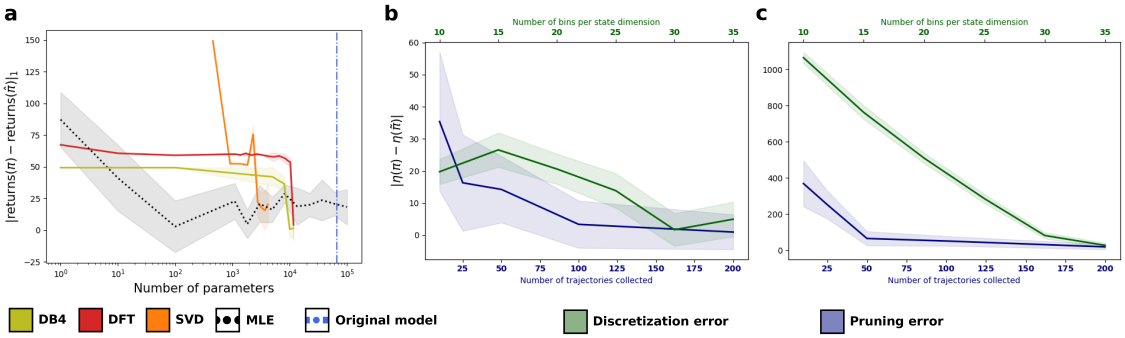

Figure 15: **(a)** Difference in returns between embedded and ground truth policies for the Mountain Car task with respect of the number of parameters used and the discretization and pruning errors for **(b)** the Gaussian policy of `ContinuousMountainCar-v0` and **(c)** the Boltzmann-Q policy of `Pendulum-v0`

A.10 Usefulness of the Pruning Step

In an environment where the state space is very large, or when policies are degenerate along a narrow path in the state space, the pruning step can turn out to be very useful.

Table 3 shows the proportion of parameters from the discretized policy that was pruned. Pruned states are ones which are visited with probability at most ε; in all our experiments, ε was set to 0, so no visited state was ever pruned.

Task	Pruned %
Pendulum (2K)	95.36
Pendulum (3K)	94.49
Pendulum (4K)	94.59
Pendulum (5K)	98.45

Table 3: Percentage of pruned parameters for the Pendulum environment policies

The pruning step is extremely efficient in the Pendulum task, allowing to reduce the memory complexity between the discretization and the truncation step by up to 98%.

Note that as policies get trained on more samples, they concentrate along the optimal policy. This in turn means that less exploration is needed, and hence the policy coverage gets reduced, allowing us to prune more states.

References

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. *arXiv preprint arXiv:1811.11214*, 2018.

Anayo K Akametalu, Jaime F Fisac, Jeremy H Gillula, Shahab Kaynama, Melanie N Zeilinger, and Claire J Tomlin. Reachability-based safe learning with gaussian processes. In *53rd IEEE Conference on Decision and Control*, pages 1424–1431. IEEE, 2014.

Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

Anil Aswani, Humberto Gonzalez, S Shankar Sastry, and Claire Tomlin. Provably safe and robust learning-based model predictive control. *Automatica*, 49(5):1216–1226, 2013.

André MS Barreto, Doina Precup, and Joelle Pineau. Practical kernel-based reinforcement learning. *The Journal of Machine Learning Research*, 17(1):2372–2441, 2016.

Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Advances in neural information processing systems*, pages 2494–2504, 2018.

Bernhard Baumgartner. An inequality for the trace of matrix products, using absolute values. *arXiv preprint arXiv:1106.6189*, 2011.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. *arXiv preprint arXiv:1707.06887*, 2017.

Haym Benaroya, Seon Mi Han, and Mark Nagurka. *Probability models in engineering and science*, volume 193. CRC press, 2005.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pages 908–918, 2017.

Byron Boots, Geoffrey Gordon, and Arthur Gretton. Hilbert space embeddings of predictive state representations. *arXiv preprint arXiv:1309.6819*, 2013.

Byron Boots, Sajid M Siddiqi, and Geoffrey J Gordon. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7):954–966, 2011.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

Xingguo Chen, Yang Gao, and Ruili Wang. Online selective kernel-based temporal difference learning. *IEEE transactions on neural networks and learning systems*, 24(12):1944–1956, 2013.

Grace E Cho and Carl D Meyer. Comparison of perturbation bounds for the stationary distribution of a markov chain. *Linear Algebra and its Applications*, 335(1-3):137–150, 2001.

Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on pure and applied mathematics*, 41(7):909–996, 1988.

Luc Devroye. An automatic method for generating random variates with a given characteristic function. In *SIAM journal on applied mathematics*, 1986.

Simon S Du, Yining Wang, and Aarti Singh. On the power of truncated svd for general high-rank matrix estimation problems. In *Advances in neural information processing systems*, pages 445–455, 2017.

Miroslav Dudik, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369*, 2011.

Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, pages 642–669, 1956.

Juan José Egozcue, José Luis Díaz-Barrero, and Vera Pawlowsky-Glahn. Hilbert space of probability density functions based on aitchison geometry. *Acta Mathematica Sinica*, 22(4):1175–1182, 2006.

Matthew Fellows, Kamil Ciosek, and Shimon Whiteson. Fourier Policy Gradients. In *ICML*, 2018.

Dylan J Foster, Alekh Agarwal, Miroslav Dudík, Haipeng Luo, and Robert E Schapire. Practical contextual bandits with regression oracles. *arXiv preprint arXiv:1803.01088*, 2018.

Javier Garcıa and Fernando Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

C Giardina and P Chirlian. Bounds on the truncation error of periodic signals. *IEEE Transactions on Circuit Theory*, 19(2):206–207, 1972.

Koen Goetschalckx, Bert Moons, Patrick Wambacq, and Marian Verhelst. Efficiently combining svd, pruning, clustering and retraining for enhanced neural network compression. In *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning*, pages 1–6, 2018.

Steffen Grunewalder, Guy Lever, Luca Baldassarre, Massi Pontil, and Arthur Gretton. Modelling transition dynamics in mdps with rkhs embeddings. *arXiv preprint arXiv:1206.4655*, 2012.

Alfred Haar. *Zur theorie der orthogonalen funktionensysteme*. Georg-August-Universitat, Gottingen., 1909.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

Dunham Jackson. *The theory of approximation*. The American mathematical society, 1930.

Sham M. Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *ICML*, 2002.

Motonobu Kanagawa and Kenji Fukumizu. Recovering distributions from gaussian rkhs embeddings. In *Artificial Intelligence and Statistics*, pages 457–465, 2014.

Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

Guy Lever, John Shawe-Taylor, Ronnie Stafford, and Csaba Szepesvári. Compressed conditional mean embeddings for model-based reinforcement learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Tianyu Li, Bogdan Mazoure, Doina Precup, and Guillaume Rabusseau. Efficient planning under partial observability with unnormalized q functions and spectral learning. In *International Conference on Artificial Intelligence and Statistics*, pages 2852–2862. PMLR, 2020.

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. *arXiv preprint arXiv:1509.02971*, 2015.

Mathias Lohne. The computational complexity of the fast fourier transform. 2017.

Yongxi Lu, Abhishek Kumar, Shuangfei Zhai, Yu Cheng, Tara Javidi, and Rogerio Feris. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5334–5343, 2017.

Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

J Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 209:415–446, 1909.

Mahdi Pakdaman Naeini, Gregory F Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, volume 2015, page 2901. NIH Public Access, 2015.

Yu Nishiyama, Abdeslam Boularias, Arthur Gretton, and Kenji Fukumizu. Hilbert space embeddings of pomdps. *arXiv preprint arXiv:1210.4887*, 2012.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *arXiv preprint arXiv:1703.02702*, 2017.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Alexander Rakhlin, Dmitry Panchenko, and Sayan Mukherjee. Risk bounds for mixture density estimation. *ESAIM: Probability and Statistics*, 9:220–229, 2005.

Howard L Resnikoff, O Raymond Jr, et al. *Wavelet analysis: the scalable structure of information*. Springer Science & Business Media, 2012.

Bengt Ringnér. Law of the unconscious statistician. *Unpublished Note*, 2009.

Matthew Robards, Peter Sunehag, Scott Sanner, and Bhaskara Marthi. Sparse kernel-sarsa (λ) with an eligibility trace. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 1–17. Springer, 2011.

Dorsa Sadigh, S Shankar Sastry, Sanjit A Seshia, and Anca Dragan. Information gathering actions over human internal state. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 66–73. IEEE, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR.

Paul J Schweitzer. Perturbation theory and finite markov chains. *Journal of Applied Probability*, 5(2):401–413, 1968.

Shayak Sen, Saikat Guha, Anupam Datta, Sriram K Rajamani, Janice Tsai, and Jeannette M Wing. Bootstrapping privacy compliance in big data systems. In *2014 IEEE Symposium on Security and Privacy*, pages 327–342. IEEE, 2014.

Thiago D. Simão, Romain Laroche, and Rémi Tachet des Combes. Safe policy improvement with an estimated baseline policy, 2019.

Aleksandrs Slivkins. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*, 2019.

Le Song, Xinhua Zhang, Alex Smola, Arthur Gretton, and Bernhard Schölkopf. Tailoring density estimation via reproducing kernel moment matching. In *Proceedings of the 25th international conference on Machine learning*, pages 992–999, 2008.

E.M. Stein and R. Shakarchi. *Fourier Analysis: An Introduction*. Princeton University Press, 2003.

Hongwei Sun and Qiang Wu. Application of integral operator for regularized least-square regression. *Mathematical and Computer Modelling*, 49(1-2):276–285, 2009.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, 2018.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. Learning the variance of the reward-to-go. *The Journal of Machine Learning Research*, 17(1):361–396, 2016.

Gavin Taylor and Ronald Parr. Kernelized value function approximation for reinforcement learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 1017–1024, 2009.

John N Tsitsiklis and Benjamin Van Roy. Optimal stopping of markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. *IEEE Transactions on Automatic Control*, 44(10):1840–1851, 1999.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Kirk Wolter. *Introduction to variance estimation*. Springer Science & Business Media, 2007.

Xin Xu, Zhongsheng Hou, Chuanqiang Lian, and Haibo He. Online learning control using adaptive critic designs with sparse kernel machines. *IEEE Transactions on Neural Networks and Learning Systems*, 24(5):762–775, 2013.

Zhuoran Yang, Yongxin Chen, Mingyi Hong, and Zhaoran Wang. Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost. *Advances in neural information processing systems*, 32, 2019.

Kehe Zhu. *Operator theory in function spaces*. Number 138. American Mathematical Soc., 2007.