On learning history-based policies for controlling Markov decision processes

Anonymous Authors[1]

Abstract

Reinforcement learning (RL) folklore suggests that history-based function approximation methods, such as recurrent neural nets or history-based state abstraction, perform better than their memory-less counterparts, due to the fact that function approximation in Markov decision processes (MDP) can be viewed as inducing a Partially observable MDP. However, there has been little formal analysis of such history-based algorithms, as most existing frameworks focus exclusively on memory-less features. In this paper, we introduce a theoretical framework for studying the behaviour of RL algorithms that learn to control an MDP using history-based feature abstraction mappings. Furthermore, we use this framework to design a practical RL algorithm and we numerically evaluate its effectiveness on a set of continuous control tasks.

1. Introduction

State abstraction and function approximation are vital components used by reinforcement learning (RL) algorithms to efficiently solve complex control problems when exact computations are intractable due to large state and action spaces. Over the past few decades, state abstraction in RL has evolved from the use of pre-determined and problem-specific features (Crites & Barto, 1995; Tsitsiklis & Roy, 1996; Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998; Singh et al., 2002; Kwok & Fox, 2004; Proper & Tadepalli, 2006) to the use of adaptive basis functions learnt by solving an isolated regression problem (Ormoneit & Sen, 2002; Menache et al., 2005; Keller et al., 2006; Petrik, 2007), and more recently to the use of neural network-based Deep-RL algorithms that embed state abstraction in successive layers of a neural network (Barto et al., 2004; Bellemare et al., 2019).

Feature abstraction results in information loss, and the resulting state features might not satisfy the controlled Markov property, even if this property is satisfied by the corresponding state (Sutton & Barto, 2018). One approach to counteract the loss of the Markov property is to generate the features using the history of state-action pairs, and empirical evidence suggests that using such history-based features are beneficial in practice (OpenAI et al., 2019). However, a theoretical characterisation of history-based Deep-RL algorithms for fully observed Markov Decision Processes (MDPs) is largely absent form the literature.

In this paper, we bridge this gap between theory and practise by providing a theoretical analysis of history-based RL agents acting in a MDP. Our approach adapts the notion of approximate information state (AIS) for POMDPs proposed in (Subramanian et al., 2020; Subramanian & Mahajan, 2019) to feature abstraction in MDPs, and we develop a theoretically grounded policy search algorithm for history-based feature abstractions and policies.

The rest of the paper is organised as follows: In Section 2, following a brief review of feature-based abstraction, we motivate the need for using history-based feature abstractions. In Section 3, we present a formal model for the co-design of the feature abstraction and control policy, derive a dynamic program using the AIS. We also derive bounds on the quality of approximate solutions to this dynamic program. In Section 4 we build on these approximation bounds to develop an RL algorithm for learning a history-based state representation and control policy. In Section 5, we present an empirical evaluation of our proposed algorithm on continuous control tasks. Finally, we discuss related work in Section 6 and conclude with future research directions in Section 7.

2. Background and Motivation

Consider an MDP $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$ where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, P denotes the controlled transition matrix, $r \colon \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ denotes the per-step reward, and $\gamma \in (0, 1)$ denotes the discount factor.

The performance of a randomised (and possibly history-dependent) policy π starting from a start state s_0 is measured by the value function, defined as:

$$V^{\pi}(s_0) = \mathbb{E}^{\pi}\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(S_t, A_t) \middle| S_0 = s_0 \right]. \quad (1)$$

A policy maximising $V^{\pi}(s_0)$ over all (randomised and possibly history dependent) policies is called the optimal policy with respect to initial state s_0 and is denoted by $\pi^{\star}$.

In many applications, $\mathcal{S}$ and $\mathcal{A}$ are combinatorially large or uncountable, which makes it intractable to compute the optimal policy. Most practical RL algorithms overcome this hurdle by using function approximation where the state is mapped to a feature space $\mathcal{Z}$ using a state abstraction function $\phi : \mathcal{S} \to \mathcal{Z}$. In Deep-RL algorithms, the last layer of the network is often viewed as a feature vector. These feature vectors are then used as an approximate state for approximating the value function $\hat{V} : \mathcal{Z} \to \mathcal{R}$ and/or computing an approximately optimal policy $\mu : \mathcal{Z} \to \Delta(\mathcal{A})$ (Sutton & Barto, 1998) (where $\Delta(\mathcal{A})$ denotes the set of probability distribution over actions). Therefore, the mapping from state to distribution of actions is given by the "flattened" policy $\tilde{\mu} = \mu \circ \phi$ *i.e.*, $\tilde{\mu} = \mu(\phi(\cdot))$.

A well known fact about function approximation is that the features that are used as an approximate state may not satisfy the controlled Markov property *i.e.*, in general,

$$\mathbb{P}(Z_{t+1} \mid Z_{1:t}, A_{1:t}) \neq \mathbb{P}(Z_{t+1} \mid Z_t, A_t).$$

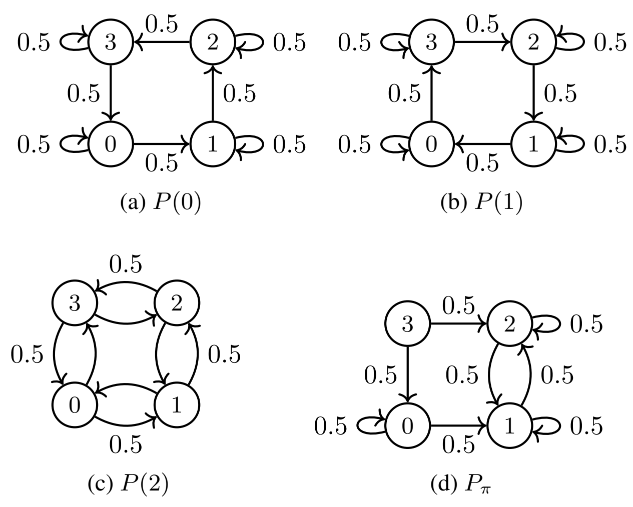

Figure 1. The transition probability for an example MDP

To see the implications of this fact, consider the toy MDP depicted in Figures 1a to 1c, with $\mathcal{S} = \{0, 1, 2, 3\}$, $\mathcal{A} = \{0, 1, 2\}$, $\{P_{s,s'}(a)\}_{a \in \mathcal{A}}$, and $r(0) = r(1) = -1$, $r(2) = 1$, $r(3) = -K$, where K is a large positive number. Given the reward structure the objective of the policy is to try to avoid state 3 and keep the agent at state 2 as much as possible. It is easy to see that the optimal policy is

$$\pi^\star(0) = 0, \quad \pi^\star(1) = 0, \quad \pi^\star(2) = 1, \text{ and } \quad \pi^\star(3) = 2.$$

Note that if the initial state is not state 3 then an agent will never visit that state under the optimal policy. Furthermore, any policy which cannot prevent the agent from visiting state 3 will have a large negative value and, therefore, cannot be optimal. Now suppose the feature space $\mathcal{Z} = \{0, 1\}$. It is easy to see that for any Markovian feature-abstraction $\phi \colon \mathcal{S} \to \mathcal{Z}$, no policy $\hat{\pi} \colon \mathcal{Z} \to \mathcal{A}$ can prevent the agent from visiting state 3. Thus, the best policy when using Markovian feature abstraction will perform significantly worse than the optimal policy (which has direct access to the state).

However, it is possible to construct a history-based feature-abstraction ϕ and a history-based control policy $\hat{\pi}$ that works with ϕ and is of the same quality as $\pi^\star$. For this, consider the following *codebooks* (where the entries denoted by a dot do not matter):

Now define

$$D(0) = \begin{bmatrix} 0 & 1 \\ 1 & 2 \\ 2 & 3 \\ 3 & 0 \end{bmatrix}, \qquad F(1) = \begin{bmatrix} 0 & 1 & \cdot & \cdot \\ \cdot & 0 & 1 & \cdot \\ \cdot & \cdot & 0 & 1 \\ 1 & \cdot & \cdot & 0 \end{bmatrix},$$

$$D(1) = \begin{bmatrix} 3 & 0 \\ 0 & 1 \\ 1 & 2 \\ 2 & 3 \end{bmatrix}, \qquad F(2) = \begin{bmatrix} 1 & \cdot & \cdot & 0 \\ 0 & 1 & \cdot & \cdot \\ \cdot & 0 & 1 & \cdot \\ \cdot & \cdot & 0 & 1 \end{bmatrix},$$

$$D(2) = \begin{bmatrix} 1 & 3 \\ 0 & 2 \\ 1 & 3 \\ 0 & 2 \end{bmatrix}, \qquad F(3) = \begin{bmatrix} \cdot & 0 & \cdot & 1 \\ 0 & \cdot & 1 & \cdot \\ \cdot & 0 & \cdot & 1 \\ 0 & \cdot & 1 & \cdot \end{bmatrix},$$

and consider the feature-abstraction policy $Z_t = F_{S_{t-1},S_t}(A_{t-1})$ and a control policy μ which is a finite state machine with memory, where the memory M_t that is updated as $M_t = D_{M_{t-1},Z_t}(A_{t-1})$ and the action A_t is chosen as $A_t = \pi(M_t)$, where $\pi \colon \mathcal{S} \to \Delta(\mathcal{A})$ is any pre-specified reference policy. It can be verified that if the system starts from a known initial state then $\mu \circ \phi = \pi$. Thus, if we choose the reference policy $\pi = \pi^\star$, then the agent will never visit state 3 under $\mu \circ \phi$, in contrast to Markovian feature-abstraction policies where (as we argued before) state 3 is always visited.

In the above example, we used the properties of the system dynamics and the reward function to design a history-based feature abstraction which outperforms memoryless feature abstractions. We are interested in developing such history-based feature abstractions using a learning framework when the system model is not known. We present such a construction in the next section.

3. Approximation bounds for history-based feature abstraction

The approximation results of our framework depend on the properties of metrics on probability spaces. We start with a brief overview of a general class of metrics known as

Integral Probability Measures (IPMs) (Müller, 1997); many of the commonly used metrics on probability spaces such as total variation (TV) distance, Wasserstein distance, and maximum-mean discrepency (MMD) are instances of IPMs. We then derive a general approximation bound that holds for general IPMs, and then specialize the bound to specific instances (TV, Wassserstein, and MMD).

3.1. Integral probability metrics (IPM)

Definition 3.1 ((Müller, 1997)). Let $(\mathcal{E}, \mathcal{G})$ be a measurable space and $\mathfrak{F}$ denote a class of uniformly bounded measurable functions on $(\mathcal{E}, \mathcal{G})$. The integral probability metric between two probability distributions $\nu_1, \nu_2 \in \mathcal{P}(\mathcal{E})$ with respect to the function class $\mathfrak{F}$ is defined as:

$$d_{\mathfrak{F}}(\nu_1, \nu_2) = \sup_{f \in \mathfrak{F}} \left| \int_{\mathcal{E}} f d\nu_1 - \int_{\mathcal{E}} f d\nu_2 \right|. \tag{2}$$

For any function f (not necessarily in $\mathfrak{F}$), the Minkowski functional $\rho_{\mathfrak{F}}$ associated with the metric $d_{\mathfrak{F}}$ is defined as:

$$\rho_{\mathfrak{F}}(f) \triangleq \inf\{\rho \in \mathcal{R}_{\geq 0} : \rho^{-1} f \in \mathfrak{F}\}. \tag{3}$$

Eq. (3), implies that that for any function f:

$$\left| \int_{\mathcal{E}} f d\nu_1 - \int_{\mathcal{E}} f d\nu_2 \right| \leq \rho_{\mathfrak{F}}(f) d_{\mathfrak{F}}(\nu_1, \nu_2). \tag{4}$$

In this paper, we use the following IPMs:

1. **Total Variation Distance**: If $\mathfrak{F}$ is chosen as $\mathfrak{F}^{\text{TV}} \triangleq \{\frac{1}{2} \operatorname{span}(f) = \frac{1}{2}(\max(f) - \min(f))\}$, then $d_{\mathfrak{F}}$ is the total variation distance, and its Minkowski functional is $\rho_{\mathfrak{F}^{\text{TV}}}(f) = \frac{1}{2} \operatorname{span}(f)$.
2. **Wasserstein/Kantorovich-Rubinstein Distance**: If $\mathcal{E}$ is a metric space and $\mathfrak{F}$ is chosen as $\mathfrak{F}^W \triangleq \{f : L_f \leq 1\}$ (where L_f denotes the Lipschitz constant of f with respect to the metric on $\mathcal{E}$), then $d_{\mathfrak{F}}$ is the Wasserstein or the Kantorovich distance. The Minkowski function for the Wasserstein distance is $\rho_{\mathfrak{F}^W}(f) = L_f$.
3. **Maximum Mean Discrepancy (MMD) Distance**: Let $\mathcal{U}$ be a reproducing kernel Hilbert space (RKHS) of real-valued functions on $\mathcal{E}$ and $\mathfrak{F}$ is choosen as $\mathfrak{F}^{MMD} \triangleq \{f \in \mathcal{U} : \|f\|_{\mathcal{U}} \leq 1\}$, (where $\|\cdot\|_{\mathcal{U}}$ denotes the RKHS norm), then $d_{\mathfrak{F}}$ is the Maximum Mean Discrepancy (MMD) distance and its Minkowski functional is $\rho_{\mathfrak{F}^{\text{MMD}}}(f) = \|f\|_{\mathcal{U}}$.

3.2. Approximate information state

Given an MDP $\mathcal{M}$ and a feature space $\mathcal{Z}$, let $\mathcal{H}_t = \mathcal{S} \times \mathcal{A}$ denote the space of all histories $(S_{1:t}, A_{1:t-1})$ up to time t, where $S_{1:t}$ is a shorthand notation for the history of states $(S_1, \ldots, S_t)$, and similar interpretation holds for $A_{1:t}$. We are interested in learning history-based feature abstraction functions $\{\sigma_t : \mathcal{H}_t \to \mathcal{Z}\}_{t \geq 1}$ and a time homogenous policy $\mu : \mathcal{Z} \to \Delta(\mathcal{A})$ such that the flattened policy $\pi = \{\pi_t\}_{t \geq 1}$, where $\pi_t = \mu \circ \sigma_t$, is approximately optimal.

3.3. Approximate information state

Given an MDP $\mathcal{M}$ and a feature space $\mathcal{Z}$, let $\mathcal{H}_t = \mathcal{S} \times \mathcal{A}$ denote the space of all histories $(S_{1:t}, A_{1:t-1})$ up to time t, where $S_{1:t}$ is a shorthand notation for the history of states $(S_1, \ldots, S_t)$, and similar interpretation holds for $A_{1:t}$. We are interested in learning history-based feature abstraction functions $\{\sigma_t : \mathcal{H}_t \to \mathcal{Z}\}_{t \geq 1}$ and a time homogenous policy $\mu : \mathcal{Z} \to \Delta(\mathcal{A})$ such that the flattened policy $\pi = \{\pi_t\}_{t \geq 1}$, where $\pi_t = \mu \circ \sigma_t$, is approximately optimal.

Since the feature abstraction approximates the state, its quality depends on how well it can be used to approximate the per step reward and predict the next state. We formalise this intuition in definition below.

Definition 3.2. A family of history-based feature abstraction functions $\{\sigma_t : \mathcal{H}_t \to \mathcal{Z}\}_{t \geq 1}$ are said to be *recursively updatable* if there exists an update function $\hat{f} : \mathcal{Z} \times \mathcal{S} \times \mathcal{A} \to \mathcal{Z}$ such that the process $\{Z_t\}_{t \geq 1}$, where $Z_t = \sigma_t(S_{1:t}, A_{1:t-1})$, satisfies:

$$Z_{t+1} = \hat{f}(Z_t, S_{t+1}, A_t). \quad t \geq 1 \tag{5}$$

Definition 3.3. Given a family of history based recursively updatable feature abstraction functions $\{\sigma_t : \mathcal{H}_t \to \mathcal{Z}\}_{t \geq 1}$, the features $Z_t = \sigma_t(S_{1:t}, A_{1:t-1})$ are said to be (ϵ, δ)-*approximate information state* (AIS) with respect to a function space $\mathfrak{F}$ if there exist: (i) a reward approximation function $\hat{r} : \mathcal{Z} \times \mathcal{A} \to \mathcal{R}$, and (ii) an approximate transition kernel $\hat{P} : \mathcal{Z} \times \mathcal{A} \to \Delta(\mathcal{S})$ such that Z satisfies the following properties:

(P1) Sufficient for approximate performance evaluation: for all t,

$$|r(S_t, A_t) - \hat{r}(Z_t, A_t)| \leq \epsilon. \tag{6}$$

(P2) Sufficient for predicting future states approximately: for all t

$$d_{\mathfrak{F}}(P(\cdot|S_t, A_t), \hat{P}(\cdot|Z_t, A_t)) \leq \delta. \tag{7}$$

We call the tuple $(\hat{r}, \hat{P})$ as an (ϵ, δ)-AIS approximator. Note that similar definitions have appeared in other works *e.g.*, latent state (Gelada et al., 2019), and approximate information state for for POMDPs (Subramanian et al., 2020; Subramanian & Mahajan, 2019). However, in (Gelada et al., 2019) it is assumed that the feature abstractions are memory-less and the discussion is restricted to Wasserstein distance. The key difference from the POMDP model in (Subramanian et al., 2020; Subramanian & Mahajan, 2019) is that the in POMDPs the observation Z_t is a pre-specified function of the state while in the proposed model Z_t depends on our choice of feature abstraction.

As such, our key insight is that an AIS-approximator of a recursively updatable history-based feature abstraction can be used to define a dynamic program. In particular, given a history-based abstraction function $\{\sigma_t : \mathcal{H}_t \to \mathcal{Z}\}_{t\geq 1}$ which is recursively updatable using $\hat{f}$ and an (ϵ, δ) AIS-approximator $(\hat{P}, \hat{r})$, we can define the following dynamic programming decomposition:

For any $z_t \in \mathcal{Z},\ a_t \in \mathcal{A}$

$$\hat{Q}(z_t, a_t) = \hat{r}(z_t, a_t) + \gamma \sum_{s_{t+1}\in\mathcal{S}} \hat{P}(s_{t+1}|z_t, a_t)\hat{V}(\hat{f}(z_t, s_{t+1}, a_t)) \tag{8}$$

$$\hat{V}(z_t) = \max_{a_t\in\mathcal{A}} \hat{Q}(z_t, a_t), \quad \forall z_t \in \mathcal{Z} \tag{9}$$

Definition 3.4. Define $\mu\colon \mathcal{Z} \to \Delta(\mathcal{A})$ be any policy such that for any $z \in \mathcal{Z}$,

$$\mathrm{Supp}(\mu(z)) \subseteq \arg\max_{a\in\mathcal{A}} \hat{Q}(z, a). \tag{10}$$

Since μ is a policy from the feature space to actions, we can use it to define a policy from the history of the state action pairs to actions as:

$$\pi_t(s_{1:t}, a_{1:t-1}) \triangleq \mu(\sigma_t(s_{1:t}, a_{1:t-1})) \tag{11}$$

Therefore, the dynamic program defined in (9) indirectly defines a history-based policy π. The performance of any such history-based policy is given by the following dynamic program:

For any $h_t \in \mathcal{H},\ a_t \in \mathcal{A}$

$$Q^{\pi}_t(h_t, a_t)^{1} = r(s_t, a_t) + \gamma \sum_{s_{t+1}\in\mathcal{S}} P(s_{t+1}|s_t, a_t)V^{\pi}_{t+1}(h_{t+1}),$$
$$V^{\pi}_t(h_t) = \sum_{a_t\in\mathcal{A}} \pi(a_t|h_t)Q^{\pi}_t(h_t, a_t), \quad \forall h_t \in \mathcal{H} \tag{12}$$

We want to quantify the loss in performance when using the history based policy π. Note that since V^{π}_t is not time-homogeneous, we need to compute the worst-case difference between $V^{\star}$ and V^{π}_t, which is given by:

$$\Delta \triangleq \sup_{t\geq 0} \sup_{h_t=(s_{1:t},a_{1:t})\in\mathcal{H}_t} |V^{\star}(s_t) - V^{\pi}_t(h_t)|, \tag{13}$$

Our main approximation result is the following:

[1]We have dropped the time index from the policy to reduce clutter

Theorem 3.5. *The worst case difference between $V^{\star}$ and V^{π}_t is bounded by*

$$\Delta \leq 2\frac{\varepsilon + \gamma\delta\kappa_{\mathfrak{F}}(\hat{V}, \hat{f})}{1-\gamma}, \tag{14}$$

where $\kappa_{\mathfrak{F}}(\hat{V}, \hat{f}) = \sup_{z,a} \rho_{\mathfrak{F}}(\hat{V}(\hat{f}(\cdot, z, a)))$, $\rho_{\mathfrak{F}}(\cdot)$ is the Minkowski functional associated with the IPM $d_{\mathfrak{F}}$ as defined in (3).

Proof in Appendix A

Some salient features of the bound are as follows: First, the bound depends on the choice of metric on probability spaces. Different IPMs will result in a different value of δ and also a different value of $\kappa_{\mathfrak{F}}(\hat{V}, \hat{f})$. Second, the bound depends on the properties of $\hat{V}$. For this reason we call it an instance dependent bound. Sometimes, it is desirable to have bounds which do not require solving the dynamic program in (9). We present such bounds as below, note that these "instance independent" bounds are the derived by upper bounding $\kappa_{\mathfrak{F}}(\hat{V}, \hat{f})$. Therefore, these are looser than the upper bound in Theorem 3.5

Corollary 3.6. *If the function class $\mathfrak{F}$ is $\mathfrak{F}^{TV}$, then Δ as defined in (13) is upper bounded as:*

$$\Delta \leq \frac{2\epsilon}{(1-\gamma)} + \frac{\gamma\delta\,\mathrm{span}(\hat{r})}{(1-\gamma)^2}. \tag{15}$$

Proof in Appendix B

Corollary 3.7. *Let $L_{\hat{r}}$ and $L_{\hat{P}}$ denote the Lipschitz constants of the approximate reward function $\hat{r}$ and approximate transition function $\hat{P}$ respectively, and $L_{\hat{f}}$ is the uniform bound on the Lipschitz constant of $\hat{f}$ with respect to the state S_t. If $\gamma L_{\hat{P}} L_{\hat{f}} \leq 1$ and the function class $\mathfrak{F}$ is $\mathfrak{F}^{W}$, then Δ as defined in (13) is upper bounded as:*

$$\Delta \leq \frac{2\epsilon}{(1-\gamma)} + \frac{2\gamma\delta L_{\hat{r}}}{(1-\gamma)(1-\gamma L_{\hat{f}} L_{\hat{P}})}. \tag{16}$$

Proof in Appendix C

Corollary 3.8. *If the function class $\mathfrak{F}$ is $\mathfrak{F}^{MMD}$, then Δ as defined in (13) is upper bounded as:*

$$\Delta \leq 2\frac{\epsilon + \gamma\delta\kappa_{\mathcal{U}}(\hat{V}, \hat{f})}{(1-\gamma)}, \tag{17}$$

where $\mathcal{U}$ is a RKHS space, $\|\cdot\|_{\mathcal{U}}$ its associated norm and $\kappa_{\mathcal{U}}(\hat{V}, \hat{f}) = \sup_{z,a} \|(\hat{V}(\hat{f}(\cdot, z, a)))\|_{\mathcal{U}}$.

Proof. The proof follows from the properties of MMD described previously. □

In the following section we will show how one can use these theoretical insights to design a policy search algorithm.

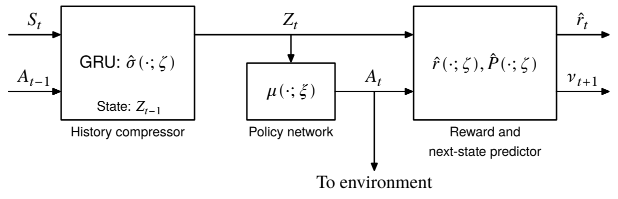

Figure 2. AIS approximator block

4. Reinforcement learning with history-based feature abstraction

The previous section helps us establish two results, the first result tells that a history-based representation can be called an AIS if it is able to evolve like a state and approximately predict the instantaneous reward and state transition. The second result tells us that if a policy is obtained using an AIS, then its performance loss is bounded in terms of the approximation error as in (14). In this section we will show how one can design a theoretically grounded computational framework that uses an RL algorithm to simultaneously learn an AIS and a policy. The key idea is to represent the AIS generator and the policy using a parametric family of functions/distributions and training them using a multi-timescale optimisation algorithm.

According to Definition 3.3, the AIS generator consists of four components, a compression function σ_t, the update function $\hat{f}$, an approximate reward predictor $\hat{r}$, and transition kernel $\hat{P}$. We can represent the history compression function using any time series approximators such as LSTMs or GRUs. An advantage of such memory based neural networks is that their internal layers are updated in a state-like manner. Therefore, we can satisfy Definition 3.2 since Z_t evolves according to the RNN's state update function such that $\hat{f}\colon \mathcal{Z}\times\mathcal{S}\times\mathcal{A}\to\mathcal{Z}$.

The main function of $\hat{r}$ and $\hat{P}$ is ensure that Z_t satisfies properties P1 and P2 (in Definition 3.3), *i.e.*, prediction of the instantaneous reward with a bounded error ϵ and approximation of the ground MDP's transition function with a bounded error δ. One way in which the computational framework can satisfy these is conditions is by explicitly optimising the AIS generator for the constants ϵ and δ. We can achieve this by modelling the reward predictor $\hat{r}$ using a multilayered perception (MLP) layer which uses the representation Z_t and action A_t to approximately predict the reward $\hat{r}_t$. In the same way, we can model the approximate transition kernel $\hat{P}$ using an appropriate class of stochastic kernel approximators *e.g.*, a softmax function or a mixture of Gaussian's to learn a parametric approximation of P. We can then train the AIS generator by minimising an appropriate objective function.

To make things more concrete, let us denote the AIS generator as the following collection: $\{\sigma_t(\cdot;\zeta), \hat{f}(\cdot;\zeta), f_{\hat{r}}(\cdot;\zeta), f_{\hat{P}}(\cdot\ :\ \zeta)\}$ where $f_{\hat{r}}(\cdot;\zeta) : \mathcal{Z}\times\mathcal{A}\to\mathcal{R}$ and $f_{\hat{P}}(\cdot;\zeta) : \mathcal{Z}\times\mathcal{A}\to\Delta(\mathcal{S})$ are the reward and transition approximators, and ζ are the parameters of the respective sub-components. Instead of separately optimising the reward prediction loss $|r(S_t,A_t)-\hat{r}(Z_t,A_t)|$, and the transition loss $d_{\mathfrak{F}}(P,\hat{P})$ we can combine them in a single objective function objective function as:

$$\mathcal{L}_{\text{AIS}}(\zeta) = \frac{1}{T}\sum_{t=0}^{T}\Bigg(\lambda\,\underbrace{(\hat{r}(Z_t,A_t;\zeta)-r(S_t,A_t))^2}_{\mathcal{L}_{\hat{R}(\cdot;\zeta)}} + (1-\lambda)\cdot\underbrace{d_{\mathfrak{F}}(\hat{P}(Z_t,A_t\,;\zeta),P)^2}_{\mathcal{L}_{\hat{P}}(\cdot;\zeta)}\Bigg), \tag{18}$$

where, T is the length of the episode or the rollout length, $\lambda\in[0,1]$ is a hyper-parameter, reward prediction loss $\mathcal{L}_{\hat{R}}(;\zeta)$ is simply the mean-squared error between the predicted and the observed reward, whereas the transition prediction loss $\mathcal{L}_{\hat{P}}(\cdot;\zeta)$ is the distance between predicted and observed transition distributions $\hat{P}$ and P. To compute $\mathcal{L}_{\hat{P}}(\cdot;\zeta)$, we need to choose an IPM. In principle we can pick any IPM, but we would want to use an IPM using which the distance $d_{\mathfrak{F}}$ can be efficiently computed.

4.1. Choice of an IPM

To compute the IPM $d_{\mathfrak{F}}$ we need to know the probability density functions $\hat{P}$ and P. As we assume $\hat{P}$ to belongs to a parametric family, we know its density function in closed form. However, since we are in the learning setup, we can only access samples from P. For a function a $f\in\mathfrak{F}$, and probability density functions P and $\hat{P}$ such that, $\nu_1 = P$, and $\nu_2=\hat{P}$, we can estimate the IPM $d_{\mathfrak{F}}$ between a distribution and samples using the duality $|\int_{\mathcal{Z}} f d\nu_1 - \int_{\mathcal{Z}} f d\nu_2|$. In this paper, we use two from of IPMs, the MMD distance and the Wasserstein/Kantorovich–Rubinstein distance.

4.1.1. MMD Distance:

Let m_ζ denote the mean of the distribution $\hat{P}(\cdot;\zeta)$. Then, the AIS-loss when MMD is used as an IPM is given by

$$\mathcal{L}_{\text{AIS}}(\zeta) = \frac{1}{T}\sum_{t=0}^{T}\Bigg(\lambda(\hat{r}(Z_t,A_t;\zeta)-r(S_t,A_t))^2 + (1-\lambda)(m_\zeta^{S_t}-2S_t)^\top m_\zeta^{S_t}\Bigg), \tag{19}$$

where $m_\zeta^{S_t}$ is obtained using the from the transition approximator, *i.e.*, the mapping $\hat{P}(\zeta) : \mathcal{Z}\times\mathcal{A}\to\mathcal{R}$. For the detailed derivation of the above loss see Appendix D.1.1

4.1.2. Wasserstein/Kantorovich–Rubinstein distance:

In principle, the Wasserstein/Kantorovich distance can be computed by solving a linear program (Sriperumbudur et al., 2012), but doing at every episode can be computationally expensive. Therefore, we propose to approximate the Wasserstein distance using a KL-divergence (Kullback & Leibler, 1951) based upper-bound. The simplified-KL divergence based AIS loss is given as:

$$\mathcal{L}_{\text{AIS}}(\zeta) = \frac{1}{T}\sum_{t=0}^{T}\Bigg(\lambda(\hat{r}(Z_t, A_t; \zeta) - r(S_t, A_t))^2 + (1-\lambda)\log(\hat{P}(S_t; \zeta))\Bigg), \tag{20}$$

where after dropping the terms which do not depend on ζ, we get $d^2_{\mathfrak{F}^{\text{W}}}(P, \hat{P}) \leq \log(\hat{P}(S_t; \zeta))$ is the simplified-KL-divergence based upper bound. For the details of this derivation see Appendix D.1.2.

4.2. Policy gradient algorithm

Algorithm 1: Policy Search with AIS

```
Input  : $\iota_0$: Initial state distribution,
         $\zeta_0$: Ais parameters,
         $\xi_0$: Actor parameters,
         $a_0$: Initial action,
         $\mathcal{D} = \emptyset$: Replay buffer,
         $N_{\text{comp}}$: Computation budget,
         $N_{\text{ep}}$: Episode length,
         $N_{\text{grad}}$: Gradient steps
1  for iterations $i = 0 : N_{comp}$ do
2      Sample start state $s_0 \sim \iota_0$;
3      for iterations $j = 0 : N_{ep}$ do
4          $z_j = \sigma_\zeta(s_{1:j}, a_{1:j-1})$;
5          $a_j = \mu_\xi(z_j)$;
6          $s_{j+1} = P(s_j, a_j)$;
7          $\mathcal{D} \leftarrow \{z_j, a_j, s_j, s_{j+1}\}$;
8          $a_{j-1} = a_j$;
9          $s_j = s_{j+1}$;
10     end
11     for every batch $b \in \mathcal{D}$ do
12         for gradient step $t = 0 : N_{grad}$ do
13             $\zeta_{t+1,b} = \zeta_{t,b} + b\nabla_\zeta \mathcal{L}_{\text{AIS}}(\zeta_{t,b})$;
14             $\xi_{t+1,b} = \xi_{t,b} + d\hat{\nabla}_\xi J(\xi_{t,b}, \zeta_{t,b})$
15         end
16     end
17 end
```

Following the design of the AIS block, we now provide a policy-gradient algorithm to learning both the AIS and policy. The schematic of our agent architecture is given in Figure 2, and pseudo-code is given in Algorithm 1. Given a feature space $\mathcal{Z}$, we can simultaneously learn the AIS-generator and the policy using a multi-timescale stochastic gradient ascent algorithm (Borkar, 2008). Let $\mu(\cdot; \xi) : \mathcal{Z} \to \Delta(\mathcal{A})$ be a parameterised stochastic policy with parameters ξ. Let $J(\xi, \zeta)$ denote the performance of the policy $\mu(\cdot;\ \xi)$. The policy gradient theorem (Sutton et al., 1999; Williams, 2004; Baxter & Bartlett, 2001) states that: For a rollout horizon T, we can estimate $\nabla_\xi J$ as:

$$\hat{\nabla}_\xi J(\xi_t, \zeta_t) = \sum_{t=1}^{T}\gamma^{t-1} r_t \Bigg(\sum_{\tau=1}^{t}\nabla_\xi \log(\mu(A_t|Z_t;\ \xi_t))\Bigg).$$

Following a rollout of length T, we can then update the parameters $\{(\zeta_i, \xi_i)\}_{i\geq 1}$ as follows:

$$\zeta_{i+1} = \zeta_i + b_i \nabla_\zeta \mathcal{L}_{\text{AIS}}(\zeta_i), \tag{21}$$

$$\xi_{i+1} = \xi_i + d_i \hat{\nabla}_\xi J(\xi_i, \zeta_i) \tag{22}$$

where the step-size $\{b_i\}_{i\geq 0}$ and $\{d_i\}_{i\geq 0}$ satisfy the standard conditions $\sum_i b_i = \infty$, $\sum_i b_i^2 < \infty$, $\sum_i d_i = \infty$ and $\sum_i d_i^2 < \infty$ respectively. Moreover, one can ensure that the AIS generator converges faster by choosing an appropriate learning rates such that, $\lim_{i\to\infty} \frac{d_i}{b_i} = 0$. It is also possible to design an Actor-Critic algorithm using similar arguments, we elaborate more on this and the convergence of these methods in Appendix E.1, and Appendix E.2.

5. Empirical evaluation

Through our experiments, we seek to answer the following questions: (1) Can history-based feature representations policies help improve the quality of solution found by a memory-less RL algorithms? (2) In terms of the solution quality how does the proposed method compare with other methods which use memory augmented policies as well as reward and transition predictors? (3) How does the choice of IPM affect the algorithms performance?

We answer question (1) by comparing our approach with the proximal policy gradient (PPO) (Schulman et al., 2017) and the policy-gradient version of DeepMDP framework (Gelada et al., 2019). For question (2) we compare our approach with modified versions of PlaNet (Hafner et al., 2019), Dreamer (Hafner et al., 2020), and VariBAD (Zintgraf et al., 2020).

For question (3) we compare the performance of our method using different MMD kernels and KL-divergence based approximation of Wasserstein distance. All the approaches are evaluated on six continuous control tasks from the MuJoCo (Todorov et al., 2012) OpenAI-Gym suite. To ensure a fair comparison, the baselines and their respective hyper-parameter settings are taken from well tested stand-alone implementations provided by Dhariwal et al. (2017). From

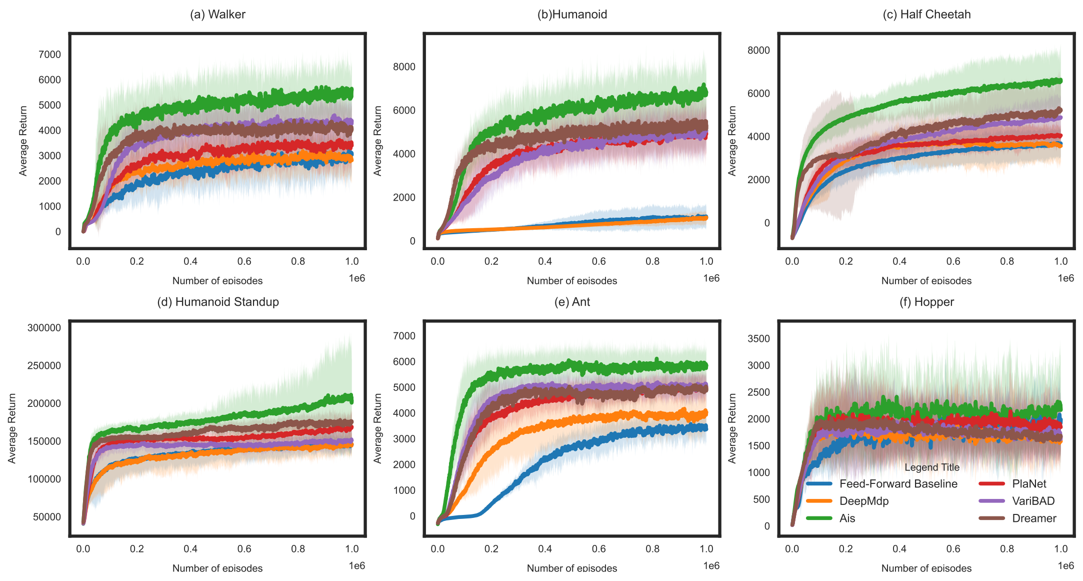

Figure 3. Empirical results averaged over 50 Monte Carlo runs with shaded regions representing the quantiles.

an implementation perspective, our framework can be used to modify any off-the-shelf policy-gradient algorithm by simply replacing (or augmenting) the feature abstraction layers of the policy and/or value networks with recurrent neural networks (RNNs), trained with the appropriate losses, as outlined previously. In these experiments, we replace the fully connected layers in PPO's architecture with a Gated Recurrent Unit (GRU). For all the implementations, we initialise the hidden state of the GRU to zero at the beginning of the trajectory. This strategy simplifies the implementation and also allows for independent decorrelated sampling of sequences, therefore ensuring robust optimisation of the networks (Hausknecht & Stone, 2015). It is important to note that we can extend our framework to other policy gradient methods such as SAC (Haarnoja et al., 2018), TD3 (Fujimoto et al., 2018) or DDPG (Lillicrap et al., 2016), after satisfying certain technical conditions. However, we leave these extensions for future work. Additional experimental details and results on different IPMs can be found in Appendix F.

6. Related Work

The development of RL algorithms with memory-based feature abstractions has been an active area of research, and most existing algorithms have tackled this problem using non-parametric methods like Nearest neighbour (Bentley, 1975; Friedman et al., 1977; Peng, 1995), Locally-weighted regression (Baird & Klopf, 1993; Atkeson et al., 1997; Moore et al., 1997), and Kernel-based regression (Connell & Utgoff, 1987; Dietterich & Wang, 2001; Ormoneit & Sen, 2002; Xu et al., 2006; Bhat et al., 2012; Barreto et al., 2016). Despite their solid theoretical footing, these methods, have limited applicability as they are difficult to scale to high-dimensional state and action spaces. More recently, several methods that propose using recurrent neural networks for learning history-based abstractions have enjoyed considerable success in complex computer games (Hausknecht & Stone, 2015; Jaderberg et al., 2017; Espeholt et al., 2018; Gruslys et al., 2018; Ha & Schmidhuber, 2018) however most of these methods have been designed for partially observable environments where use of history-based methods is often necessary. To the best of our knowledge, the only other work where a history-based RL algorithm is used for controlling a MDP is presented by OpenAI et al. (2019). In this work the authors show that using an LSTM-based agent architecture results in superior performance for the object reorientation using robotic arms. However, the authors do not provide a theoretical analysis of their method.

6.1. Bisimulation metrics

On the theoretical front, our work is closely related to state aggregation techniques based on bisimulation metrics proposed by Givan et al. (2003); Ferns et al. (2004; 2011). The bisimulation metric is the fixed point of an operator on the space of semi-metrics defined over the state space of an MDP with Lipschitz value functions. Apart from state aggregation, bisimulation metrics have been used for feature discovery (Comanici & Precup, 2011; Ruan et al., 2015), and transfer learning (Castro & Precup, 2010). However, computational impediments have prevented their broad adoption. Our work can be viewed as an alternative to bisimulation for the analysis of history-based state abstractions and deep RL methods. Our work can also be thought of as extension of the DeepMDP framework (Gelada et al., 2019) to history-based policies and direct policy search methods.

6.2. AIS and Agent state

The notion of AIS is closely related to the epistemic state recently proposed by Lu et al. (2021). An epistemic state is a bounded representation of the history. It is updated recursively as the agent collects more information, and is represented as an environment proxy Υ which is learnt by optimising a target/objective function χ. Since Υ is a random variable, its entropy $\mathbb{H}(\Upsilon)$ is used to represent system's uncertainty about the environment. The framework proposed in this paper can considered as a practical way of constructing the system epistemic state where, the AIS Z_t represents both the epistemic state and the environment proxy Υ, $\mathcal{L}_{\text{AIS}}$ represents χ, and instead of entropy, the constants ϵ, and δ represent the systems uncertainty about the environment. The study of the AIS framework in the regret minimisation paradigm could help establish a relationship between the ϵ, δ, and $\mathbb{H}(\Upsilon)$, thereby helping designers develop principled algorithms which synthesise ideas like information directed sampling for direct policy search algorithms.

6.3. Analysis of RL algorithms with attention mechanism

Recently, there has been considerable interest in developing RL algorithms which use attention mechanism/transformer architectures (Bahdanau et al., 2015; Xu et al., 2015) for learning feature abstractions (Zambaldi et al., 2019; Mott et al., 2019; Sorokin et al., 2015; Oh & Kaneko, 2018; Ritter et al., 2021; Parisotto et al., 2020; Chen et al., 2021; Loynd et al., 2020; Tang et al., 2020; Pritzel et al., 2017). Attention mechanism extract task relevant information from historical observations and can be used instead of RNNs for processing sequential data (Vaswani et al., 2017). As we do not impose a functional from on the history compression function $\sigma_t(\cdot)$ in Definition 3.3, any attention mechanism can be interpreted as history compression function, and one can construct a valid information state by ensuring that the output of the attention mechanism satisfies (P1) and (P2). That being said, even without optimising $\mathcal{L}_{\text{AIS}}$, the approximation bound in Theorem 3.5 still applies for RL algorithms with attention mechanisms, with the caveat that the constants ϵ, and δ may be arbitrarily large. A thorough empirical analysis of the effect of different attention mechanisms, and the AIS loss on the on the error constants ϵ, and δ could help us gain a better understanding of the way in which such design choices could influence the learning process.

6.4. AIS for POMDPs

The concept of an AIS used in this paper is similar to the idea of AIS for POMDPs (Subramanian & Mahajan, 2019; Subramanian et al., 2020). Moreover, the literature also contains several other methods which have enjoyed empirical success in using history-based policies for controlling POMDPs (Holmes & Jr., 2006; Daswani et al., 2013; Hutter, 2014; Schaefer et al., 2007; Hafner et al., 2020; 2019). In principle, one can use any of these methods for controlling MDPs. However, this does not immediately provide a tight bound for the approximation error. The MDP model has more structure than POMDPs, and our goal in this paper is to use this fact to present a tighter analysis of the approximation error.

7. Conclusion and future work

This paper presents the design and analysis of a principled approach for learning history-based policies for controlling MDPs. We believe that our approximation bounds can be helpful for practitioners to study the effect of some of their design choices on the solution quality. On the practical side, the proposed algorithm shows favourable results on high-dimensional control tasks. Note that one can also use the bounds in Theorem 3.5 to analyse the approximation error of other history-based methods. However, since some of these algorithms do not satisfy Definition 3.3, the resulting approximation error might be arbitrarily large. Such blow-ups in the approximation error could be because the bound itself is loose or the optimality gap is large. This would depend on the specifics of the methods and remains to be investigated. As such, a sharper analysis of the approximation error by factoring in the specific design choices of other methods is an interesting direction for future research. Another interesting direction would be to conduct a thorough empirical evaluation exploring the design choices of history compression functions.

References

Atkeson, C. G., Moore, A. W., and Schaal, S. *Locally Weighted Learning*, pp. 11–73. Springer Netherlands, Dordrecht, 1997. ISBN 978-94-017-2053-3. doi: 10.1007/978-94-017-2053-3_2. URL https://doi.org/10.1007/978-94-017-2053-3_2.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.0473.

Baird, L. C. and Klopf, A. H. Reinforcement learning with high-dimensional, continuous actions. Technical report, Wright Laboratory, 1993.

Barreto, A., Doina, P., and Joelle, P. Practical kernel-based reinforcement learning. *Journal of Machine Learning Research*, 2016.

Barto, A., Anderson, C., and Sutton, R. Synthesis of nonlinear control surfaces by a layered associative search network. *Biological Cybernetics*, 43:175–185, 2004.

Baxter, J. and Bartlett, P. L. Infinite-horizon policy-gradient estimation. *J. Artif. Intell. Res.*, 15:319–350, 2001. doi: 10.1613/jair.806. URL https://doi.org/10.1613/jair.806.

Bellemare, M. G., Dabney, W., Dadashi, R., Taïga, A. A., Castro, P. S., Roux, N. L., Schuurmans, D., Lattimore, T., and Lyle, C. A geometric perspective on optimal representations for reinforcement learning. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 4360–4371, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/3cf2559725a9fdfa602ec8c887440f32-Abstract.html.

Bentley, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18:509–517, 1975.

Bertsekas, D. P. and Tsitsiklis, J. N. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996. ISBN 1886529108.

Bhat, N., Moallemi, C. C., and Farias, V. F. Non-parametric approximate dynamic programming via the kernel method. In *NIPS*, 2012.

Borkar, V. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008. ISBN 9780521515924. URL https://books.google.ca/books?id=QLxIvgAACAAJ.

Borkar, V. S. Stochastic approximation with two time scales. *Systems & Control Letters*, 29:291–294, 1997.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym, 2016.

Castro, P. S. and Precup, D. Using bisimulation for policy transfer in mdps. In *AAAI*, 2010.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *CoRR*, abs/2106.01345, 2021. URL https://arxiv.org/abs/2106.01345.

Comanici, G. and Precup, D. Basis function discovery using spectral clustering and bisimulation metrics. In *AAAI*, 2011.

Connell, M. E. and Utgoff, P. Learning to control a dynamic physical system. *Computational Intelligence*, 3, 1987.

Crites, R. H. and Barto, A. G. Improving elevator performance using reinforcement learning. In Touretzky, D. S., Mozer, M., and Hasselmo, M. E. (eds.), *Advances in Neural Information Processing Systems 8, NIPS, Denver, CO, USA, November 27-30, 1995*, pp. 1017–1023. MIT Press, 1995. URL https://proceedings.neurips.cc/paper/1995/file/390e982518a50e280d8e2b535462ec1f-Paper.pdf.

Daswani, M., Sunehag, P., and Hutter, M. Q-learning for history-based reinforcement learning. In Ong, C. S. and Ho, T. B. (eds.), *Asian Conference on Machine Learning, ACML 2013, Canberra, ACT, Australia, November 13-15, 2013*, volume 29 of *JMLR Workshop and Conference Proceedings*, pp. 213–228. JMLR.org, 2013. URL http://proceedings.mlr.press/v29/Daswani13.html.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. https://github.com/openai/baselines, 2017.

Dietterich, T. G. and Wang, X. Batch value function approximation via support vectors. In *NIPS*, 2001.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1407–1416. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/espeholt18a.html.

Ferns, N., Panangaden, P., and Precup, D. Metrics for finite markov decision processes. In *Conferrence on Uncertainty in Artificial Intelligence*, 2004.

Ferns, N., Panangaden, P., and Precup, D. Bisimulation metrics for continuous markov decision processes. *SIAM J. Comput.*, 40:1662–1714, 2011.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Trans. Math. Softw.*, 3:209–226, 1977.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1582–1591. PMLR, 2018. URL http://proceedings.mlr.press/v80/fujimoto18a.html.

Fukumizu, K., Gretton, A., Lanckriet, G., Schölkopf, B., and Sriperumbudur, B. K. Kernel choice and classifiability for rkhs embeddings of probability distributions. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/file/685ac8cadc1be5ac98da9556bc1c8d9e-Paper.pdf.

Gelada, C., Kumar, S., Buckman, J., Nachum, O., and Bellemare, M. G. Deepmdp: Learning continuous latent space models for representation learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2170–2179. PMLR, 2019. URL http://proceedings.mlr.press/v97/gelada19a.html.

Givan, R., Dean, T. L., and Greig, M. Equivalence notions and model minimization in markov decision processes. *Artif. Intell.*, 147:163–223, 2003.

Gruslys, A., Dabney, W., Azar, M. G., Piot, B., Bellemare, M. G., and Munos, R. The reactor: A fast and sample-efficient actor-critic agent for reinforcement learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rkHVZWZAZ.

Ha, D. and Schmidhuber, J. World models. *CoRR*, abs/1803.10122, 2018. URL http://arxiv.org/abs/1803.10122.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1856–1865. PMLR, 2018. URL http://proceedings.mlr.press/v80/haarnoja18b.html.

Hafner, D., Lillicrap, T. P., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565. PMLR, 2019. URL http://proceedings.mlr.press/v97/hafner19a.html.

Hafner, D., Lillicrap, T. P., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Hausknecht, M. J. and Stone, P. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposia, Arlington, Virginia, USA, November 12-14, 2015*, pp. 29–37. AAAI Press, 2015. URL http://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11673.

Holmes, M. P. and Jr., C. L. I. Looping suffix tree-based inference of partially observable hidden state. In Cohen, W. W. and Moore, A. W. (eds.), *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pp. 409–416. ACM, 2006. doi: 10.1145/1143844.1143896. URL https://doi.org/10.1145/1143844.1143896.

Hutter, M. Extreme state aggregation beyond mdps. In Auer, P., Clark, A., Zeugmann, T., and Zilles, S. (eds.), *Algorithmic Learning Theory - 25th International Conference, ALT 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8776 of *Lecture Notes in Computer Science*, pp. 185–199. Springer, 2014. doi: 10.1007/978-3-319-11662-4_14. URL https://doi.org/10.1007/978-3-319-11662-4_14.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=SJ6yPD5xg.

Keller, P. W., Mannor, S., and Precup, D. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pp. 449–456, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143901. URL `https://doi.org/10.1145/1143844.1143901`.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Kullback, S. and Leibler, R. A. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. doi: 10.1214/aoms/1177729694. URL `https://doi.org/10.1214/aoms/1177729694`.

Kwok, C. and Fox, D. Reinforcement learning for sensing strategies. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 4, pp. 3158–3163 vol.4, 2004. doi: 10.1109/IROS.2004.1389903.

Leslie, D. Reinforcement learning in games. 2004.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y. (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL `http://arxiv.org/abs/1509.02971`.

Loynd, R., Fernandez, R., Celikyilmaz, A., Swaminathan, A., and Hausknecht, M. J. Working memory graphs. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6404–6414. PMLR, 2020. URL `http://proceedings.mlr.press/v119/loynd20a.html`.

Lu, X., Roy, B. V., Dwaracherla, V., Ibrahimi, M., Osband, I., and Wen, Z. Reinforcement learning, bit by bit. *CoRR*, abs/2103.04047, 2021. URL `https://arxiv.org/abs/2103.04047`.

Menache, I., Mannor, S., and Shimkin, N. Basis function adaptation in temporal difference reinforcement learning. *Ann. Oper. Res.*, 134(1):215–238, 2005. doi: 10.1007/s10479-005-5732-z. URL `https://doi.org/10.1007/s10479-005-5732-z`.

Moore, A. W., Schneider, J. G., and Deng, K. Efficient locally weighted polynomial regression predictions. In *ICML*, 1997.

Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., and Rezende, D. J. Towards interpretable reinforcement learning using attention augmented agents. In *NeurIPS*, 2019.

Müller, A. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997. ISSN 00018678. URL `http://www.jstor.org/stable/1428011`.

Oh, H. and Kaneko, T. Deep recurrent q-network with truncated history. In *2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 34–39, 2018. doi: 10.1109/TAAI.2018.00017.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation, 2019.

Ormoneit, D. and Sen, S. Kernel-based reinforcement learning. *Mach. Learn.*, 49(2-3):161–178, 2002. doi: 10.1023/A:1017928328829. URL `https://doi.org/10.1023/A:1017928328829`.

Parisotto, E., Song, H. F., Rae, J. W., Pascanu, R., Gülçehre, Ç., Jayakumar, S. M., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M., Heess, N., and Hadsell, R. Stabilizing transformers for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7487–7498. PMLR, 2020. URL `http://proceedings.mlr.press/v119/parisotto20a.html`.

Peng, J. Efficient memory-based dynamic programming. In Prieditis, A. and Russell, S. (eds.), *Machine Learning Proceedings 1995*, pp. 438–446. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: https://doi.org/10.1016/B978-1-55860-377-6.50061-X. URL `https://www.sciencedirect.com/science/article/pii/B978155860377650061X`.

Petrik, M. An analysis of laplacian methods for value function approximation in mdps. In Veloso, M. M. (ed.), *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pp. 2574–2579, 2007. URL `http://ijcai.org/Proceedings/07/Papers/414.pdf`.

Pritzel, A., Uria, B., Srinivasan, S., Badia, A. P., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. Neural episodic control. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on*

Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017, volume 70 of *Proceedings of Machine Learning Research*, pp. 2827–2836. PMLR, 2017. URL http://proceedings.mlr.press/v70/pritzel17a.html.

Proper, S. and Tadepalli, P. Scaling model-based average-reward reinforcement learning for product delivery. In Fürnkranz, J., Scheffer, T., and Spiliopoulou, M. (eds.), *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pp. 735–742. Springer, 2006. doi: 10.1007/11871842_74. URL https://doi.org/10.1007/11871842_74.

Raichuk, A., Stanczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., and Gelly, S. What matters for on-policy deep actor-critic methods? a large-scale study. In *ICLR*, 2021.

Ritter, S., Faulkner, R., Sartran, L., Santoro, A., Botvinick, M., and Raposo, D. Rapid task-solving in novel environments. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=F-mvpFpn_0q.

Ruan, S. S., Comanici, G., Panangaden, P., and Precup, D. Representation discovery for mdps using bisimulation metrics. In *AAAI*, 2015.

Schaefer, A., Udluft, S., and Zimmermann, H.-G. A recurrent control neural network for data efficient reinforcement learning. *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, pp. 151–157, 2007.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Sejdinovic, D., Sriperumbudur, B., Gretton, A., and Fukumizu, K. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics*, 41 (5):2263–2291, 2013. ISSN 00905364, 21688966. URL http://www.jstor.org/stable/23566550.

Singh, S. P., Litman, D. J., Kearns, M. J., and Walker, M. A. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *J. Artif. Intell. Res.*, 16:105–133, 2002. doi: 10.1613/jair.859. URL https://doi.org/10.1613/jair.859.

Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., and Ignateva, A. Deep attention recurrent q-network. *ArXiv*, abs/1512.01693, 2015.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., and Lanckriet, G. R. G. On the empirical estimation of integral probability metrics. *Electronic Journal of Statistics*, 6(none):1550 – 1599, 2012. doi: 10.1214/12-EJS722. URL https://doi.org/10.1214/12-EJS722.

Subramanian, J. and Mahajan, A. Approximate information state for partially observed systems. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pp. 1629–1636, 2019. doi: 10.1109/CDC40024.2019.9029898.

Subramanian, J., Sinha, A., Seraj, R., and Mahajan, A. Approximate information state for approximate planning and reinforcement learning in partially observed systems. *CoRR*, abs/2010.08843, 2020. URL https://arxiv.org/abs/2010.08843.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998. ISBN 0-262-19398-1. URL http://www.cs.ualberta.ca/%7Esutton/book/ebook/the-book.html.

Sutton, R. S. and Barto, A. G. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 2018.

Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In Solla, S. A., Leen, T. K., and Müller, K. (eds.), *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*, pp. 1057–1063. The MIT Press, 1999. URL https://proceedings.neurips.cc/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf.

Tang, Y., Nguyen, D., and Ha, D. Neuroevolution of self-interpretable agents. In Coello, C. A. C. (ed.), *GECCO '20: Genetic and Evolutionary Computation Conference, Cancún Mexico, July 8-12, 2020*, pp. 414–424. ACM, 2020. doi: 10.1145/3377930.3389847. URL https://doi.org/10.1145/3377930.3389847.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, 2012.

Tsitsiklis, J. N. and Roy, B. V. Feature-based methods for large scale dynamic programming. *Mach. Learn.*, 22(1-3):59–94, 1996. doi: 10.1023/A:1018008221616. URL https://doi.org/10.1023/A:1018008221616.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 2004.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015.

Xu, X., Xie, T., Hu, D., and Lu, X. Kernel least-squares temporal difference learning. In *Computational intelligence and neuroscience*, 2006.

Zambaldi, V. F., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D. P., Lillicrap, T. P., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M. M., Vinyals, O., and Battaglia, P. W. Deep reinforcement learning with relational inductive biases. In *ICLR*, 2019.

Zintgraf, L. M., Shiarlis, K., Igl, M., Schulze, S., Gal, Y., Hofmann, K., and Whiteson, S. Varibad: A very good method for bayes-adaptive deep RL via meta-learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Hkl9JlBYvr.

Appendix

A. Proof for Theorem 3.5

For readability we will restate the theorem statement

Theorem A.1. *For any time t, any realisation s_t of S_t, a_t of A_t, let $h_t = (s_{1:t}, a_{1:t-1})$, and $z_t = \sigma_t(h_t)$. The worst case difference between $V^\star$ and V_t^π is bounded as:*

$$\Delta \leq 2\frac{\varepsilon + \gamma\delta\kappa_{\mathfrak{F}}(\hat{V}, \hat{f})}{1-\gamma}, \tag{23}$$

where, $\kappa_{\mathfrak{F}}(\hat{V}, \hat{f}) = \sup_{z,a} \rho_{\mathfrak{F}}(\hat{V}(\hat{f}(\cdot, z, a)))$. and $\rho_{\mathfrak{F}}(\cdot)$ is the Minkowski functional associated with the IPM $d_{\mathfrak{F}}$ as defined in (3).

Proof. For this proof we will use the following convention: For a generic history $h_t \in \mathcal{H}_t$, we assume that $h_t = (s_{1:t}, a_{1:t-1})$, moreover, note that $z_t = \sigma_t(h_t)$.

Now from (3.1), and Definition 3.3 for any a_t, s_t, z_t:

$$\max_{h \in \mathcal{H}_t, a_t \in \mathcal{A}} \Big| r(s_t, a_t) - \hat{r}(z_t, a_t) \Big| \leq \epsilon.$$

$$\max_{h \in \mathcal{H}_t, a_t \in \mathcal{A}} \Bigg| \sum_{s_{t+1} \in \mathcal{S}} \Bigg(P(s_{t+1}|s_t, a_t)\hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) - \hat{P}(s_{t+1}|z_t, a_t)\hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) \Bigg) \Bigg| \leq \delta\rho_{\mathfrak{F}}(\hat{V}(\hat{f}(\cdot, z_t, a_t))). \tag{24}$$

Now using triangle inequality we get:

$$\|V^\star(s_t) - V_t^\pi(h_t)\|_\infty \overset{(a)}{\leq} \underbrace{\|V^\star(s_t) - \hat{V}(z_t)\|_\infty}_{\text{term 1}} + \underbrace{\|V_t^\pi(h_t) - \hat{V}(z_t)\|_\infty}_{\text{term 2}}, \tag{25}$$

where (a) follows from triangle inequality.

We will now proceed by bounding terms 1 and 2 separately

Bounding term 1:

$$\|V^\star(s_t) - \hat{V}(z_t)\|_\infty \leq \max_{h \in \mathcal{H}_t} \Bigg| \max_{a_t \in \mathcal{A}} \Big[Q^\star(s_t, a_t) - \hat{Q}(z_t, a_t) \Big] \Bigg|, \tag{26}$$

Therefore, for any action a_t

$$\begin{aligned}
\max_{h \in \mathcal{H}_t} \Bigg| \max_{a_t \in \mathcal{A}} \Big[Q^\star(s_t, a_t) - \hat{Q}(z_t, a_t) \Big] \Bigg| &= \max_{h \in \mathcal{H}_t} \Bigg| \max_{a_t \in \mathcal{A}} \Bigg[r(s_t, a_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t)V^\star(s_{t+1}) \\
&\quad - \hat{r}(z_t, a_t) - \gamma \sum_{s_{t+1} \in \mathcal{S}} \hat{P}(s_{t+1}|z_t, a_t)\hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) \Bigg] \Bigg| \\
&\overset{(a)}{\leq} \epsilon + \max_{h \in \mathcal{H}_t, a_t \in \mathcal{A}} \Bigg| \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t)V^\star(s_{t+1}) - \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t)\hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) \Bigg| \\
&\quad + \max_{h \in \mathcal{H}_t, a_t \in \mathcal{A}} \Bigg| \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t)\hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) - \gamma \sum_{s_{t+1} \in \mathcal{S}} \hat{P}(s_{t+1}|z_t, a_t)\hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) \Bigg| \\
&\overset{(b)}{\leq} \epsilon + \gamma\|(V^\star(s_t) - \hat{V}(z_t))\|_\infty + \gamma\delta\rho_{\mathfrak{F}}(\hat{V}(\hat{f}(\cdot, z_t, a_t))),
\end{aligned}$$

where (a) from triangle inequality and (b) is due to (24). Now defining $\kappa_{\mathfrak{F}}(\hat{V}, \hat{f}) = \sup_{z,a} \rho_{\mathfrak{F}}(\hat{V}(\hat{f}(\cdot, z, a)))$, and substituting the above result in (26) we get

$$|V^\star(s_t) - \hat{V}(z_t)| \leq \frac{\varepsilon + \gamma\delta\kappa_{\mathfrak{F}}(\hat{V}, \hat{f})}{1-\gamma}. \tag{27}$$

Bounding term 2:

$$\|V_t^\pi(h_t) - \hat{V}(z_t)\|_\infty \leq \max_{h \in \mathcal{H}_t} \left| \max_{a_t \in \mathcal{A}} \left[Q_t^\pi(h_t, a_t) - \hat{Q}(z_t, a_t) \right] \right|, \tag{28}$$

Therefore, for any action a_t

$$\begin{aligned}
\max_{h \in \mathcal{H}_t} \left| \max_{a_t \in \mathcal{A}} \left[Q^\pi(h_t, a_t) - \hat{Q}(z_t, a_t) \right] \right| &= \max_{h \in \mathcal{H}_t} \left| \max_{a_t \in \mathcal{A}} \left[r(s_t, a_t) + \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) V_{t+1}^\pi(h_{t+1}) - (\hat{r}(z_t, a_t) + \right.\right. \\
&\left.\left. \gamma \sum_{s_{t+1} \in \mathcal{S}} \hat{P}(s_{t+1}|z_t, a_t) \hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) \right] \right|, \\
&\overset{(a)}{\leq} \epsilon + \max_{h \in \mathcal{H}_t, a_t \in \mathcal{A}} \left| \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) V_{t+1}^\pi(h_{t+1}) - \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) \hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) \right| \\
&+ \max_{h \in \mathcal{H}_t, a_t \in \mathcal{A}} \left| \gamma \sum_{s_{t+1} \in \mathcal{S}} P(s_{t+1}|s_t, a_t) \hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) - \gamma \sum_{s_{t+1} \in \mathcal{S}} \hat{P}(s_{t+1}|z_t, a_t) \hat{V}(\hat{f}(s_{t+1}, z_t, a_t)) \right|, \\
&\overset{(b)}{\leq} \epsilon + \gamma \|(V^\pi(h_t) - \hat{V}(z_t))\|_\infty + \gamma\delta\rho_{\mathfrak{F}}(\hat{V}(\hat{f}(\cdot, z_t, a_t))),
\end{aligned}$$

where (a) is from triangle inequality, (b) is due to (24), with $\kappa_{\mathfrak{F}}(\hat{V}, \hat{f}) = \sup_{z,a} \rho_{\mathfrak{F}}(\hat{V}(\hat{f}(\cdot, z, a)))$, and substituting the above result in (28) we get

$$\|V_t^\pi(h_t) - \hat{V}(z_t)\|_\infty \leq \frac{\varepsilon + \gamma\delta\kappa_{\mathfrak{F}}(\hat{V}, \hat{f})}{1-\gamma}. \tag{29}$$

The final result then follows by adding (27) and (29). □

B. Proof for Corollary 3.6

Lemma B.1. *If $\hat{V}$ is the optimal value function of the MDP $\hat{\mathcal{M}}$ induced by the process $\{Z_t\}_{t \geq 0}$, then*

$$\operatorname{span}(\hat{V}) \leq \frac{\operatorname{span}(\hat{r})}{1-\gamma}. \tag{30}$$

Proof. The result follows by observing that the per-step reward $\hat{r}(Z_t, A_t) \in [\min(\hat{r}), \max(\hat{r})]$. Therefore $\max(\hat{V}) \leq \max(\hat{r})$ and $\min(\hat{V}) \geq \min(\hat{r})$. □

Corollary B.2. *If the function class $\mathfrak{F}$ is $\mathfrak{F}^{TV}$, then Δ defined in (13) is upper bounded as:*

$$\Delta \leq \frac{2\epsilon}{1-\gamma} + \frac{\gamma\delta \operatorname{span}(\hat{r})}{(1-\gamma)^2}, \tag{31}$$

Proof. From Section 3.1 we know that for the Total variation distance $\rho_{\mathfrak{F}^{\mathrm{TV}}}(\hat{V}) = \operatorname{span}(\hat{V})$ and $\kappa(\hat{f}) = 1$. The result in the corollary then follows from Lemma B.1. □

C. Proof for Corollary 3.7

Definition C.1. For any Lipschitz function $f : (\mathcal{Z}, d_Z) \to (\mathcal{R}, |\cdot|)$, and probability measures ν_1, and ν_2 on $(\mathcal{Z}, d_Z)$

$$\left| \int_{\mathcal{Z}} f d\nu_1 - \int_{\mathcal{Z}} f d\nu_2 \right| \leq \|f\|_L . d_{\mathfrak{F}^{\mathrm{W}}}(\nu_1, \nu_2) \leq L_f d_{\mathfrak{F}^{\mathrm{W}}}(\nu_1, \nu_2), \tag{32}$$

where L_f is the Lipschitz constant of f and $d_{\mathfrak{F}^{\mathrm{W}}}$ is the Wasserstein distance.

Definition C.2. Let d be a metric on the AIS/Feature space $\mathcal{Z}$. The MDP $\hat{\mathcal{M}}$ induced by the process $\{Z_t\}_{t\geq 0}$ is said to be $(L_{\hat{r}}, L_{\hat{P}})$ - Lipschitz if for any $Z_1, Z_2 \in \mathcal{Z}$, the reward $\hat{r}$ and transition $\hat{P}$ of $\hat{\mathcal{M}}$ satisfy the following:

$$\left| r(Z_1, A) - r(Z_2, A) \right| \leq L_{\hat{r}} d(Z_1, Z_2), \tag{33}$$

$$d_{\mathfrak{F}^{\mathrm{W}}}(\hat{P}(\cdot|Z_1, A), \hat{P}(\cdot|Z_2, A) \leq L_{\hat{P}} d(Z_1, Z_2), \tag{34}$$

where $d_{\mathfrak{F}^{\mathrm{W}}}$ is the Wasserstein or the Kantorovitch-Rubinstein distance.

Lemma C.3. *Let $\hat{V} : \mathcal{Z} \to \mathcal{R}$ be $L_{\hat{V}}$ continuous. Define:*

$$\hat{Q}(z, a) = \hat{r}(z, a) + \gamma \sum_{s'} \hat{P}(s'|z, a)\hat{V}(\hat{f}(s', z, a).$$

Then $\hat{Q}$ is $(L_{\hat{r}} + \gamma L_{\hat{V}} L_{\hat{f}} L_{\hat{P}})$-Lipschitz continuous.

Proof. For any action a

$$\left|\hat{Q}(z_1, a) - \hat{Q}(z_2, a)\right| \overset{(a)}{\leq} \left|\hat{r}(z_1, a) - \hat{r}(z_2, a)\right| + \gamma \left| \sum_{s'} \hat{P}(s'|z_1, a)\hat{V}(\hat{f}(s', z_1, a)) - \hat{P}(s'|z_2, a)\hat{V}(\hat{f}(s', z_2, a)) \right|, \tag{35}$$

$$\overset{(b)}{\leq} (L_{\hat{r}} + \gamma L_{\hat{V}} L_{\hat{f}} L_{\hat{P}}) d(z_1, z_2), \tag{36}$$

where (a) due to triangle inequality, and (b) follows form Definition C.1, Definition C.2, and because $\|a \circ b\|_L \leq \|a\|_L \cdot \|b\|_L$. □

Lemma C.4. *Let $\hat{Q} : \mathcal{Z} \times \mathcal{A} \to \mathcal{R}$ be $L_{\hat{Q}}$- Lipschitz continuous, Define*

$$\hat{V}(z) = \max_{a_t \in \mathcal{A}} \hat{Q}(z, a).$$

Then $\hat{V}$ is $L_{\hat{Q}}$ Lipschitz

Proof. Consider $z_1, z_2 \in \mathcal{Z}$, and let a_1 and a_2 denote the corresponding optimal action. Then,

$$\hat{V}(z_1) - \hat{V}(z_2) = \hat{Q}(z_1, a_1) - \hat{Q}(z_2, a_2), \tag{37}$$

$$\overset{(a)}{\leq} \hat{Q}(z_1, a_2) - \hat{Q}(z_2, a_2), \tag{38}$$

$$\overset{(b)}{\leq} L_{\hat{Q}} d(z_1, z_2), \tag{39}$$

By symmetry,

$$\hat{V}(z_2) - \hat{V}(z_1) \leq L_{\hat{Q}} d(z_1, z_2).$$

Therefore,

$$\left|\hat{V}(z_1) - \hat{V}(z_2)\right| \leq L_{\hat{Q}} d(z_1, z_2).$$

□

Lemma C.5. *Consider the following dynamic program defined in (9):*[2]

$$\hat{Q}_t(z_t, a_t) = \hat{r}(z_t, a_t) + \gamma \sum_{s_t \in \mathcal{S}} \hat{P}(s_t | z_t, a_t) \hat{V}(\hat{f}(z_t, s_t, a_t)), \; \forall z \in \mathcal{Z}, a \in \mathcal{A}$$
$$\hat{V}_t(z_t) = \max_{a \in \mathcal{A}} \hat{Q}_t(z_t, a_t), \; \forall z \in \mathcal{Z}$$

Then at any time t, we have:

$$L_{\hat{V}_{t+1}} = L_{\hat{r}} + \gamma L_{\hat{P}} L_{\hat{f}} L_{\hat{V}_t}.$$

Proof. We prove this by induction. At time $t = 1$ $\hat{Q}_1(z, a) = \hat{r}(z, a)$, therefore $L_{\hat{Q}_1} = L_{\hat{r}}$. Then according to Lemma C.4, $\hat{V}_1$ is Lipschitz with Lipschitz constant $L_{\hat{V}_1} = L_{\hat{Q}_1} = L_{\hat{r}}$. This forms the basis of induction. Now assume that at time t, $\hat{V}_t$ is $L_{\hat{V}_t}$- Lipschitz. By Lemma C.3 $\hat{Q}_{t+1}$ is $L_{\hat{r}} + \gamma L_{\hat{f}}, L_{\hat{P}} L_{\hat{V}_t}$. Therefore by Lemma C.4, $\hat{V}_{(t+1)}$ is Lipschitz with constant:

$$L_{\hat{V}_{t+1}} = L_{\hat{r}} + \gamma L_{\hat{f}} L_{\hat{P}} L_{\hat{V}_t}.$$

□

Theorem C.6. *Given any $(L_{\hat{r}}, L_{\hat{P}})$- Lipschitz MDP, if $\gamma L_{\hat{P}} L_{\hat{f}} \leq 1$, then the infinite horizon γ-discounted value function $\hat{V}$ is Lipschitz continuous with Lipschitz constant*

$$L_{\hat{V}} = \frac{L_{\hat{r}}}{1 - \gamma L_{\hat{f}} L_{\hat{P}}}.$$

Proof. Consider the sequence of $L_t = L_{\hat{V}_t}$ values. For simplicity write $\alpha = \gamma L_{\hat{P}} L_{\hat{f}}$. Then the sequence $\{L_t\}_{t \geq 1}$ is given by : $L_1 = L_{\hat{r}}$ and for $t \geq 1$,

$$L_{t+1} = L_{\hat{r}} + \alpha L_t,$$

Therefore,

$$L_t = L_{\hat{r}} + \alpha L_{\hat{r}} + \ldots + \alpha_{t+1} = \frac{1 - \alpha^t}{1 - \alpha} L_{\hat{r}}.$$

This sequence converges if $|\alpha| \leq 1$. Since α is non-negative, this is equivalent to $\alpha \leq 1$, which is true by hypothesis. Hence L_t is a convergent sequence. At convergence, the limit $L_{\hat{V}}$ must satisfy the fixed point of the recursion relationship introduced in Lemma C.5, hence,

$$L_{\hat{V}} = L_{\hat{r}} + \gamma L_{\hat{f}} L_{\hat{P}} L_{\hat{V}}.$$

Consequently, the limit is equal to,

$$L_{\hat{V}} = \frac{L_{\hat{r}}}{1 - \gamma L_{\hat{f}} L_{\hat{P}}}.$$

□

Corollary C.7. *If $\gamma L_{\hat{P}} L_{\hat{f}} \leq 1$ and the function class $\mathfrak{F}$ is $\mathfrak{F}^W$, then Δ as defined in (13) is upper bounded as:*

$$\Delta \leq \frac{2\epsilon}{(1 - \gamma)} + \frac{2\gamma\delta L_{\hat{r}}}{(1 - \gamma)(1 - \gamma L_{\hat{f}} L_{\hat{P}})}, \tag{40}$$

Proof. The proof follows from the observation that for $d_{\mathfrak{F}^W}$, $\rho_{\mathfrak{F}^W} = L_{\hat{V}}$, and then using the result from Theorem C.6. □

[2]We have added t as a subscript to denote the computation time *i.e.*, the time at which the respective function is updated.

D. Algorithmic Details

D.1. Choice of an IPM:

D.1.1. MMD

One advantage of choosing $d_{\mathfrak{F}}$ as the MMD distance is that unlike the Wasserstein distance, its computation does not require solving an optimisation problem. Another advantage is that we can leverage some of their properties to further simplify our computation, as follows:

Proposition D.1 (Theorem 22 (Sejdinovic et al., 2013)). *Let $\mathcal{X} \subseteq \mathcal{R}^m$, and $d_{\mathcal{X},p} : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ be a metric given by $d_{\mathcal{X},p}(x, x') = \|x - x'\|_2^p$, for $p \in (0, 2]$. Let $k_p : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ be any kernel given:*

$$k_p(x, x') = \frac{1}{2}(d_{\mathcal{X},p}(x, x_0) + d_{\mathcal{X},p}(x', x_0) - d_{\mathcal{X},p}(x, x')), \tag{41}$$

where $x_0 \in \mathcal{X}$ is arbitrary, and let $\mathcal{U}_p$ be a RKHS kernel with kernel k_p and $\mathfrak{F}_p = \{f \in \mathcal{U}_p : \|f\|_{\mathcal{U}_p} \geq 1\}$. Then for any distributions ν_1, $\nu_2 \in \Delta\mathcal{X}$, the IPM can be expressed as:

$$d_{\mathfrak{F}}(\nu_1, \nu_2) = \left(\mathbb{E}[d_{\mathcal{X},p}(X_1, W_1)] - \frac{1}{2}\mathbb{E}[d_{\mathcal{X},p}(X_1, X_2)] - \frac{1}{2}\mathbb{E}[d_{\mathcal{X},p}(W_1, W_2)]\right)^{\frac{1}{2}}, \tag{42}$$

where X_1, X_2, and W_1, W_2 are i.i.d. samples from ν_1 and ν_2 respectively.

The main implication of Proposition D.1 is that, instead of using (42), for $p \in (0, 2]$ we can use the following as a surrogate for $d_{\mathfrak{F}_p}$:

$$\int_{\mathcal{X}}\int_{\mathcal{X}} \|x_1 - w_1\|_2^p \nu_1(dx_1)\nu_2(dw_1) - \frac{1}{2}\int_{\mathcal{X}}\int_{\mathcal{X}} \|w_1 - w_2\|_2^p \nu_2(dw_1)\nu_2(dw_1). \tag{43}$$

Moreover, according to Sriperumbudur et al. (2012) for n identically and independently distributed (i.i.d) samples $\{X_i\}_{i=0}^n \sim \nu_1$ an unbiased estimator of (43) is given as:

$$\frac{1}{n}\sum_{i=1}^{n}\int_{\mathcal{X}} \|X_i - w_1\|_2^p \nu_1 d(w_1) - \frac{1}{2}\int_{\mathcal{X}}\int_{\mathcal{X}} \|w_1 - w_2\|_2^p \nu_1(dw_1)\nu_2(dw_2). \tag{44}$$

We implement a simplified version of the surrogate loss in (44) as follows:

Proposition D.2 ((Subramanian et al., 2020)). *Given the setup in Proposition D.1 and $p = 2$, Let $\nu_2(\zeta)$ be a parametric distribution with mean m and let $X \sim \nu_1$, then the gradient $\nabla_\zeta (m_\zeta - 2X)^\top m_\zeta$ is an unbiased estimator of $\nabla_\zeta d_{\mathfrak{F}_2}(\alpha, \nu_\zeta)^2$*

Proof. Let $X_1, X_2 \sim \nu_1$, and $W_1, W_2 \sim \nu_2(\zeta)$

$$\therefore \nabla_\zeta d_{\mathfrak{F}_2}(\nu_1, \nu_2(\zeta))^2 = \nabla_\zeta \left[\mathbb{E}\|X_1 - W_1\|_2^2 - \frac{1}{2}\mathbb{E}\|X_1 - X_2\|_2^2 - \frac{1}{2}\mathbb{E}\|W_1 - W_2\|_2^2\right], \tag{45}$$

$$\overset{(a)}{=} \nabla_\zeta \left[\mathbb{E}\|W_1\|_2^2 - 2\mathbb{E}\|X_1\|^\top \mathbb{E}\|W_1\|\right], \tag{46}$$

where (a) follows from the fact that X does not depend on ζ, which simplifies the implementation of the MMD distance. $\square$

In this way we can simplify the computation of $d_{\mathfrak{F}}$ using a parametric stochastic kernel approximator and MMD metric.

Note that when are trying to approximate a continuous distribution we can readily use the loss function (46) as long as the mean m_ζ of $\nu_2(\zeta)$ is given in closed form. The AIS loss is then given as:

$$\mathcal{L}_{\text{AIS}}(\zeta) = \frac{1}{T}\sum_{t=0}^{T}\left(\lambda(f_{\hat{r}}(Z_t, A_t; \zeta) - r(S_t, A_t))^2 + (1 - \lambda)(m_\zeta^{S_t} - 2S_t)^\top m_\zeta^{S_t}\right), \tag{47}$$

where $m_\zeta^{S_t}$ is obtained using the from the transition approximator, *i.e.*, the mapping $f_{\hat{P}}(\zeta) : \mathcal{Z} \times \mathcal{A} \to \mathcal{R}$.

D.1.2. Wasserstein Distance

The the KL-divergence between two densities ν_1 and ν_2 on for any $X \in \mathcal{X} \subset \mathcal{R}^m$ is defined as:

$$d_{\text{KL}}(\nu_1 \| \nu_2) = \int_{\mathcal{X}} \log(\nu_1(x))\nu_1(dx) - \int_{\mathcal{X}} \log(\nu_2(x))\nu_1(dx) \tag{48}$$

Moreover, if $\mathcal{X}$ is bounded space with diameter D, then the relation between the Wasserstein distance $d_{\mathfrak{F}^{\text{W}}}$, Total variation distance $d_{\mathfrak{F}^{\text{TV}}}$, and the KL divergence is given as :

$$d_{\mathfrak{F}^{\text{W}}}(\nu_1, \nu_2) \le D d_{\mathfrak{F}^{\text{TV}}}(\nu_1, \nu_2) \overset{(a)}{\le} \sqrt{2 d_{\text{KL}}(\nu_1 \| \nu_2)}, \tag{49}$$

where, (a) follows from the Pinsker's inequality. Note that in (18) we use $d^2_{\mathfrak{F}}$. Therefore, we can use a (simplified) KL-divergence based surrogate objective given as:

$$\int_{\mathcal{X}} \log(\nu_2(x;\zeta))\nu_1(dx), \tag{50}$$

where we have dropped the terms which do not depend on ζ. Note that the above expression is same as the cross entropy between ν_1 and ν_2 which can be effectively computed using samples. In particular, if we get T i.i.d samples from ν_1, then,

$$\frac{1}{T}\sum_{i=0}^{T} \log(\nu_2(x_i;\zeta)) \tag{51}$$

is an unbiased estimator of $\int_{\mathcal{X}} \log(\nu_2(x;\zeta))\nu_1(dx)$.

The KL divergence based AIS loss is then given as:

$$\mathcal{L}_{\text{AIS}}(\zeta) = \frac{1}{T}\sum_{t=0}^{T}\Bigg(\lambda(f_{\hat{r}}(Z_t, A_t;\zeta) - r(S_t, A_t))^2 + (1-\lambda)\log(\hat{P}(S_t;\zeta))\Bigg), \tag{52}$$

E. Extension to actor critic and convergence

E.1. Actor Critic Algorithm

As mentioned previously, similar to Section 4.2 we can also design an AIS-based Actor-Critic algorithm. As such, in addition to a parameterised policy $\pi(\cdot;\xi)$ and AIS generator $(\sigma_t(\cdot;\zeta), \hat{f}, \hat{r}, \hat{P})$ we can also a parameterised critic $\hat{V}(\cdot;\vartheta) : \mathcal{Z} \to \mathcal{R}$, where ϑ are the parameters for the critic. The performance of policy $\mu(\cdot;\xi)$ is then given by $J(\xi,\zeta,\vartheta)$. According to policy gradient theorem (Sutton et al., 1999; Baxter & Bartlett, 2001) the gradient of $J(\xi,\zeta,\vartheta)$, is given as:

$$\nabla_\xi J(\xi,\zeta,\vartheta) = \mathbb{E}\Bigg[\nabla_\xi \log(\mu(\cdot;\xi))\hat{V}(\cdot;\vartheta)\Bigg]. \tag{53}$$

And for a trajectory of length T, we approximate it as:

$$\hat{\nabla}_\xi J(\xi,\zeta,\vartheta) = \frac{1}{T}\sum_{t=1}^{T}\Bigg[\nabla_\xi \log(\mu(\cdot;\xi))\hat{V}(\cdot;\vartheta)\Bigg]. \tag{54}$$

The parameters ϑ can be learnt by optimising the temporal difference loss given as:

$$\mathcal{L}_{\text{TD}}(\xi,\zeta,\vartheta) = \frac{1}{T}\sum_{t=0}^{T} \texttt{smoothL1}(\hat{V}(Z_t;\vartheta) - r(Z_t, A_t) - \gamma\hat{V}(Z_{t+1};\vartheta)). \tag{55}$$

The parameters $\{(\zeta_i, \xi_i, \vartheta_i)\}_{i\ge 1}$ can then be updated using a multi-timescale stochastic approximation algorithm as follows:

$$\zeta_{i+1} = \zeta_i + b_i \nabla_\zeta \mathcal{L}_{\text{AIS}}(\zeta_i), \tag{56a}$$

$$\vartheta_{i+1} = \vartheta_i + c_i \nabla_\vartheta \mathcal{L}_{\text{TD}}(\xi_i, \zeta_i, \vartheta_i), \tag{56b}$$

$$\xi_{i+1} = \xi_i + d_i \hat{\nabla}_\xi J(\xi_i, \zeta_i, \vartheta), \tag{56c}$$

where the step-size $\{b_i\}_{i\geq 0}$, $\{c_i\}_{i\geq 0}$ and $\{d_i\}_{i\geq 0}$ satisfy the standard conditions $\sum_i b_i = \infty$, $\sum_i b_i^2 < \infty$, $\sum_i c_i = \infty$, $\sum_i c_i^2 < \infty$, $\sum_i d_i = \infty$ and $\sum_i d_i^2 < \infty$ respectively. Moreover, one can ensure that the AIS generator converges first, followed by the critic and the actor by choosing an appropriate step-sizes such that, $\lim_{i\to\infty} \frac{d_i}{b_i} = 0$ and $\lim_{i\to\infty} \frac{c_i}{d_i} = 0$.

E.2. Convergence analysis

In this section we will discuss the convergence of the AIS-based policy gradient in Section 4.2 as well as Actor-Critic algorithm presented in the previous subsection. The proof of convergence relies on multi-timescale stochastic approximation Borkar (2008) under conditions similar to the standard conditions for convergence of policy gradient algorithms with function approximation stated below, therefore it would suffice to provide a proof sketch.

Assumption E.1.

1. The values of step-size parameters b, d and c (for the actor critic algorithm) are set such that the timescales of the updates for ζ, ξ, and ϑ (for Actor-Critic algorithm) are separated, *i.e.*, $b_t \gg d_t$, and for the Actor-Critic algorithm $b_t \gg c_t \gg d_t$, $\sum_i b_i = \infty$, $\sum_i b_i^2 < \infty$, $\sum_i c_i = \infty$, $\sum_i c_i^2 < \infty$, $\sum_i d_i = \infty$ and $\sum_i d_i^2 < \infty$, $\lim_{i\to\infty} \frac{d_i}{b_i} = 0$ and $\lim_{i\to\infty} \frac{c_i}{d_i} = 0$,
2. The parameters ζ, ξ and ϑ (for Actor-Critic algorithm) lie in a convex, compact and closed subset of Euclidean spaces.
3. The gradient $\nabla_\zeta \mathcal{L}_{\text{AIS}}$ is Lipschitz in ζ_t, and $\hat{\nabla}_\xi J(\xi, \zeta)$ is Lipschitz in ξ_t, and ζ_t. Whereas for the Actor-Critic algorithm the gradient of the TD loss $\nabla_\vartheta \mathcal{L}_{\text{TD}}(\zeta, \xi, \vartheta)$ and the policy gradient $\hat{\nabla}_\xi J(\zeta, \xi, \vartheta)$ is Lipschitz in $(\zeta_t, \xi_t, \vartheta_t)$.
4. All the estimates of all the gradients $\nabla_\zeta \mathcal{L}_{\text{AIS}}$, $\nabla_\xi J(\xi, \zeta)$, $\nabla_\vartheta \mathcal{L}_{\text{TD}}(\zeta, \xi, \vartheta)$ and are unbiased with bounded variance[3].

Assumption E.2.

1. The ordinary differential equation (ODE) corresponding to (22) is locally asymptotically stable.
2. The ODEs corresponding to (21) is globally asymptotically stable.
3. For the Actor-Critic algorithm, the ODE corresponding to (56b) is globally asymptotically stable and has a fixed point which is Lipschitz in ξ.

Theorem E.3. *Under assumption E.1 and E.2, along any sample path, almost surely we have the following:*

1. *The iteration for ζ in (21) converges to an AIS generator that minimises the $\mathcal{L}_{AIS}$.*
2. *The iteration for ξ in (22) converges to a local maximum of the performance $J(\zeta^\star, \xi)$ where $\zeta^\star$, and $\vartheta^\star$ (for Actor Critic) are the converged value of ζ, ϑ.*
3. *For the Actor-Critic algorithm the iteration for ϑ in (56b) converges to critic that minimises the error with respect to the true value function.*

Proof. The proof for this theorem follows the technique used in (Leslie, 2004; Borkar, 2008). Due to the specific choice of learning rate the AIS-generator is updated at a faster time-scale than the actor, therefore it is "quasi static" with respect to to the actor while the actor observes a "nearly equilibriated" AIS generator. Similarly in the case of the Actor-Critic algorithm the AIS generator observes a stationary critic and actor, whereas the critic and actor see "nearly equilibriated" AIS generator. The Martingale difference condition (A3) of Borkar (2008) is satisfied due to Item 4 in assumption E.1. As such since our algorithm satisfies all the four conditions by (Leslie, 2004, page35), (Borkar, 1997, Theorem 23), the result then follows by combining the theorem on (Leslie, 2004, page 35)(Borkar, 2008, Theorem 23) and (Borkar, 1997, Theorem 2.2). □

F. Experimental Details

F.1. Environments

Our algorithms are evaluated on MuJoCo (Todorov et al., 2012, mujoco-py version 2.0.2.9) via OpenAI gym (Brockman et al., 2016, version 0.17.1) interface, using the v2 environments. The environment, state-space, action space, and reward function are not modified or pre-processed in any way for easy reproducibility and fair comparison with previous results. Each environment runs for a maximum of 2048 time steps or until some termination condition and has a multi-dimensional action space with values in the range of (-1, 1), except for Humanoid which uses the range of (-0.4, 0.4).

[3]This assumption is only satisfied in tabular MDPs.

Common	Optimiser	Adam
	Discount Factor γ	0.99
	Inital standard deviation for the policy	0.0
	PPO-Epochs	12
	Clipping Coefficient	0.2
	Entropy-Regulariser	0
	Batch Size	512
	Episode Length	2048
AIS generator	History Compressor	GRU
	Hidden layer dimension	256
	Step size	1.5e-3
	λ	0.3
Actor	Step size	3.5e-4
	No of hidden layers	1
	Hidden layer Dimension	32

Table 1. Hyperparameters

F.2. Hyper-parameters

Table 1 contains all the hyper-parameters used in our experiments. Both the policy and AIS networks are trained with Adam optimiser (Kingma & Ba, 2015), with a batch size of 512. We follow Raichuk et al. (2021)'s recommended protocol for training on-policy policy based methods, and perform 12 PPO updates after every policy evaluation subroutine. To ensure separation of time-scales the step size of the AIS generator and the policy network is set to 1.5e^{-3} and 3.5e^{-4} respectively. Hyper-parameters of our approach are searched over a grid of values, but an exhaustive grid search is not carried out due to prohibitive computational cost. We start with the recommended hyper-parameters for the baseline implementations and tune them further around promising values by an iterative process of performing experiments and observing results.

For the state-based RNN baseline we have tuned the learning rate over a grid of values starting from 1e-4 to 4e-4 and settled on 3.5e-4 as it achieved the best performance. Similarly the hidden layer size set to 256 as it is observed to achieve best performance. For the feed-forward baselines we use the implementation by OpenAI baselines (Dhariwal et al., 2017) with their default hyper-parameters.

F.3. Modifications to baselines and their hyper-parameters

Note that the methods in (Hafner et al., 2020; 2019) are designed for pixel-based control tasks and cannot be readily used for continuous control tasks in this paper. To help them process real-valued state vectors, we replace the convolutional and deconvolutional layers in their architectures by fully connected layers. We observed that feed-forward layers of size 256 for PlaNET (Hafner et al., 2019), and 300 for Dreamer (Hafner et al., 2020) produced the best results for both the methods. For PlaNET (Hafner et al., 2019) we used the default hyper-parameters and varied the learning rate over a grid of values starting from 1e-4 to 4e-4. We observed that this method achieved best performance at the learning rate of 3e-3. For Dreamer (Hafner et al., 2020) we used the default hyper-parameters and varied the learning rate over a grid of values starting from 6e-4, 6e-5 and 6e-5 respectively to 1e-3, 1e-4, 1e-4 respectively. We observed that this method achieved the best performance at 7.5e-4, 9e-5 and 8.5e-5 respectively.

The architecture proposed in this paper is similar to the VariBAD (Zintgraf et al., 2020) method proposed by Zintgraf et al.. The main motivation of Zintgraf et al. was evaluate VariBAD for multi-task RL, however we believe that the specifics of this method are relevant to both the ideas presented in this paper and the single-task RL setup. In our experiments we use the code base provided by Zintgraf et al. without any modifications and observe that the method performs on-par with several other baselines.

F.4. Type of MMDs

The MMD distance given by (46) in Appendix D.1.1, can be computed using different types of characteristic kernels (for a detailed review see (Sriperumbudur et al., 2012; Fukumizu et al., 2009; Sejdinovic et al., 2013)). In this paper we consider computing (46) using the Laplace, Gaussian and energy distance kernels. The performance of the proposed approach under different kernels is shown in Figure 4. It can be observed that for the continuous control tasks in the MuJoCo suite, the energy distance yields better performance, and therefore we implement Equation (46) using the energy distance for all the experiments.

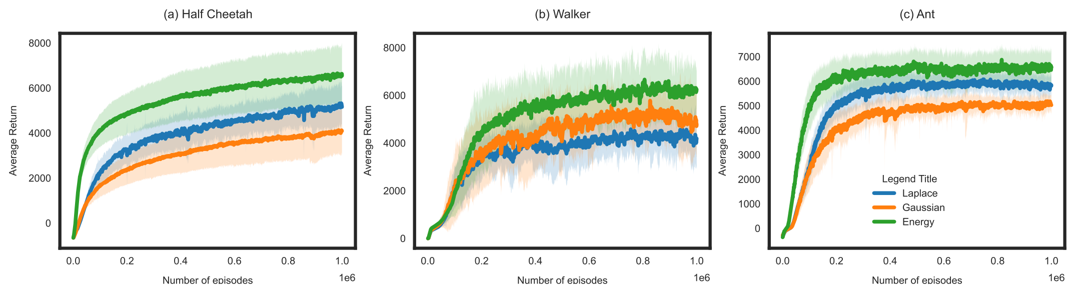

Figure 4. Comparison of different MMDs, averaged over 50 runs with ± 1 `std-dev`

F.5. MMD vs KL

Next, we compare the performance of our method under MMD (energy distance)-based AIS loss in (19) and KL-based AIS loss given in (20). From Figure 5, one can observe that for the Mujoco tasks, MMD-based loss leads to better performance.

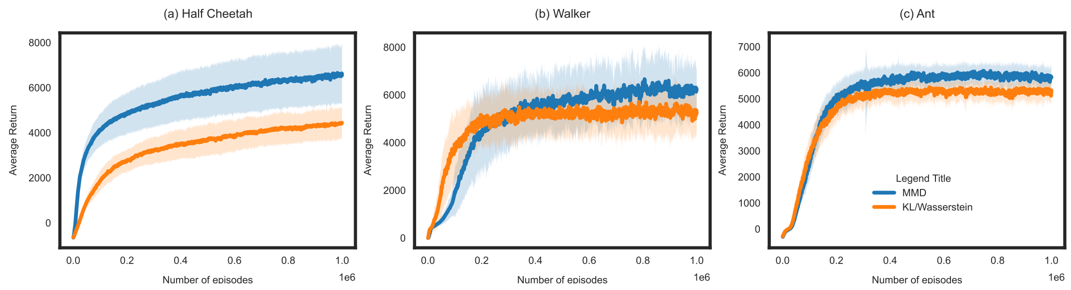

Figure 5. Comparison of Wasserstein vs MMDs, averaged over 50 runs with ± 1 `std-dev`