

Reinforcement learning in partially observable environments using approximate information state

Amit Sinha

Center for Intelligent Machines
Electrical & Computer Engineering
McGill University
Montreal, Canada

December 2020

A thesis submitted to McGill University in partial fulfillment of the requirements for the
degree of Master of Science.

© 2020 Amit Sinha

Abstract

In this thesis, reinforcement learning (RL) for multi-stage decision making problems where the decision maker or the agent has partial observations of the state are considered. Many real world systems are partially observable—some examples are autonomous driving, robotics, smart grids, etc. However, most of the research on RL is centered around fully observable systems. Partially observable systems are considerably harder than fully observable systems because finding the optimal policy involves considering the entire trajectory of past actions and observations to decide on the current action. The exponential growth of history-dependent policies makes it much harder to find an optimal policy, whereas in fully observable systems the policy search space is much smaller. This further exacerbates issues related to high dimensionality of the observation, action and/or state spaces. The belief distribution over states can be used instead of the past actions and observations to circumvent the exponential growth of policies, but this requires model information about the system which may not always be available in a reinforcement learning setting.

An alternative to the history or belief distribution over states is the more general notion of the information state which is a representation that is sufficient for learning optimal agent behavior. The entire history and the belief state may be viewed as instances of information state. A desirable information state allows us to compress the history to a smaller size so that the policy search space is smaller. An approximate information state (AIS) can be learnt from data using function approximation without requiring any model information.

One of the most important steps in learning an effective AIS is being able to predict the probability distribution of the next AIS/observation given the current AIS and action. This is done by optimizing integral probability metrics (IPMs) which try to decrease the “distance” between the predicted distribution and the actual distribution. We demonstrate the versatility of AIS using two different choices of IPMs. One is the Wasserstein distance which is optimized in terms of a surrogate KL-divergence loss (KL IPM). The second is a distance-based maximum mean discrepancy loss (MMD IPM).

We show the scalability of the proposed methods through numerical experiments in environments of increasing difficulty. These methods are compared with approximate planning solutions for low and moderate-dimensional environments for a model-based baseline. For a model-free RL baseline, a recently proposed method using proximal policy optimization (PPO) with recurrent connections (LSTM) is used for all environments. We show that

the proposed RL methods based on AIS outperform the PPO with LSTM baseline in most environments. We also show that the performance achieved by our RL algorithms is close to the near optimal approximate planning solutions in the low and moderate-dimensional environments.

Résumé

Dans cette thèse, l'apprentissage par renforcement pour les problèmes de prise de décision à plusieurs étapes où le décideur ou l'agent a des observations partielles de l'état est considéré. De nombreux systèmes du monde réel sont partiellement observables—certains exemples sont la conduite autonome, la robotique, les réseaux intelligents, etc. Cependant, la plupart des recherches sur l'apprentissage par renforcement sont centrées sur des systèmes entièrement observables. Les systèmes partiellement observables sont considérablement plus difficiles que les systèmes entièrement observables parce que la recherche de la politique optimale implique la prise en compte de l'ensemble de la trajectoire des actions et des observations passées pour décider de l'action en cours. La croissance exponentielle des politiques dépendantes de l'histoire rend beaucoup plus difficile la recherche d'une politique optimale, alors que dans les systèmes entièrement observables, l'espace de recherche sur les politiques est beaucoup plus restreint. Cela exacerbe encore les problèmes liés à la haute dimensionnalité des espaces d'observation, d'action et/ou d'état. La répartition des croyances sur les états peut être utilisée à la place des actions et des observations passées pour contourner la croissance exponentielle des politiques, mais cela nécessite des informations modèles sur le système qui ne sont pas toujours disponibles dans un cadre d'apprentissage par renforcement.

Une alternative à la distribution de l'histoire ou des croyances sur les états est la notion plus générale d'état d'information qui est une représentation suffisante pour l'apprentissage du comportement optimal de l'agent. L'ensemble de l'histoire et l'état de croyance peuvent être considérés comme des exemples d'état d'information. Un état d'information souhaitable nous permet de comprimer l'histoire à une taille plus petite de sorte que l'espace de recherche des politiques est plus petit. Un état d'information approximatif peut être appris à partir de données en utilisant l'approximation de fonction sans avoir besoin d'informations de modèle.

L'une des étapes les plus importantes dans l'apprentissage d'un état d'information approximatif efficace est de pouvoir prédire la distribution de probabilité du prochain état d'information approximatif/observation en fonction d'état d'information approximatif et de l'action en cours. Cela se fait en optimisant les métriques de probabilité intégrales qui tentent de réduire la “distance” entre la distribution prévue et la distribution réelle. Nous démontrons la polyvalence de l'état d'information approximatif en utilisant deux choix

différents de métriques de probabilité intégrales. Le premier est la distance de Wasserstein qui est optimisée en termes de perte de divergence Kullback-Leibler de substitution. Le second est une perte de divergence moyenne maximale basée sur la distance.

Nous démontrons l’extensibilité des méthodes proposées par des expériences numériques dans des environnements de difficulté croissante. Ces méthodes sont comparées à des solutions de planification approximatives pour des environnements de faible et moyenne dimension pour une base de référence basée sur un modèle. Pour une ligne de base RL sans modèle, une méthode récemment proposée utilisant l’optimisation des politiques proximales avec connexions récurrentes (LSTM) est utilisée pour tous les environnements. Nous montrons que les méthodes d’apprentissage par renforcement proposées basées sur l’état d’information approximatif sont plus performantes que l’optimisation des politiques proximales avec ligne de base LSTM dans la plupart des environnements. Nous montrons également que les performances obtenues par nos algorithmes d’apprentissage par renforcement sont proches des solutions de planification approximatives presque optimales dans les environnements à faible et moyenne dimension.

Acknowledgments

First I would like to thank my supervisor, Prof. Aditya Mahajan for his support, encouragement and guidance during my time at McGill. His extensive involvement in the research has taught me a lot about the entire process.

I feel incredibly lucky to have had the opportunity to study at McGill University and for being awarded with the Graduate Excellence Fellowship and Top-Up awards throughout the years to help support me financially. I am also grateful to the *Fonds de recherche Québec* for awarding me with a FRQNT Master's scholarship which helped me to continue my studies in Québec. The numerical experiments were enabled in part by support provided by Calcul Québec and Compute Canada.

The atmosphere at McGill University and Montréal's cultural diversity was great to work in and it was helpful to provide perspective. The experience wouldn't have been the same without the good friends I've made here during my stay. I am particularly thankful to Jneid Jneid for his constant support and also for his help on the French translation of my thesis abstract.

I'd like to thank the other students in the Systems and Control lab for productive discussions as well as for their friendship. I had several interesting discussions with Jayakumar Subramanian and Raihan Seraj. Raihan was also primarily responsible for some of the baselines used in this thesis. The cheerful talks I used to have in the lab with Anirudha Jitani, Fatih Gurturk, Nima Akbarzadeh, Bornha Sayedana, Mohammad Afshari and Erfan SeyedSalehi would always help me to stay positive and productive.

Finally, it would not be possible for me to continue my studies without the constant support of my parents and other family members for which I am grateful.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Notation Used	3
1.3	Markov Decision Processes (MDP)	3
1.4	Partially Observable Markov Decision Processes (POMDP)	4
1.5	Overview of current techniques for planning and learning in POMDPs . . .	7
1.5.1	Planning in POMDPs	7
1.5.2	Learning in POMDPs	9
1.6	Contributions of this work	13
1.7	Claims of originality and publications	13
1.7.1	Claims of originality	13
1.7.2	Journal publication under review	14
1.7.3	Contributions of co-authors	14
2	Approximate information state	15
2.1	Input-output model considered	16
2.2	Information state	18
2.3	Approximate information state	21
2.3.1	Integral probability metrics (IPM)	21
2.3.2	Approximate information state (AIS) and approximate dynamic programming	24
2.3.3	AIS with observation compression	26
2.4	Infinite-horizon discounted reward setup	27
2.4.1	System model and problem formulation	28

2.4.2	A dynamic programming decomposition	28
2.4.3	Time-homogeneous information state and simplified dynamic program	30
2.4.4	Time-homogeneous AIS and approximate dynamic programming . .	32
3	Reinforcement learning for partially observed systems using AIS	34
3.1	The choice of an IPM	35
3.2	Learning an AIS for a fixed policy	39
3.3	AIS-based partially observed reinforcement learning (PORL)	40
4	Experimental Results	43
4.1	Low-dimensional environments	44
4.2	Moderate-dimensional environments	46
4.3	High-dimensional environments	47
5	Conclusions and Future Directions	60
5.1	Future Directions	61
A	Proofs	62
B	Implementation Details	72
B.1	Details about the network architecture, training, and hyperparameters . . .	72
B.1.1	Details for low dimensional environments:	73
B.1.2	Details for moderate dimensional environments:	74
B.1.3	Details for high dimensional environments:	75
B.1.4	Details for PPO with LSTM and Critic:	77
B.1.5	Details about hyperparameter tuning	79
	References	80

List of Figures

2.1	A stochastic input-output system	16
2.2	The timing diagram of the input-output system.	16
2.3	A stochastic input-output system with observation compression	27
4.1	Network architecture for PORL using AIS.	44
4.2	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for low-dimensional environments (for 10 random seeds).	51
4.2	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for low-dimensional environments (for 10 random seeds) (contd.). .	52
4.3	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for moderate-dimensional environments (for 10 random seeds). . .	53
4.4	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds).	54
4.4	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.). .	55
4.4	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.). .	56
4.4	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.). .	57
4.4	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.). .	58
4.4	Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.). .	59

List of Acronyms

MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
RL	Reinforcement Learning
PORL	Partially Observed Reinforcement Learning
AIS	Approximate Information State
IPM	Integral Probability Metric
MMD	Maximum Mean Discrepancy
PPO	Proximal Policy Optimization
LSTM	Long Short-Term Memory
RNN	Recurrent Neural Network
DQN	Deep Q-Networks

Chapter 1

Introduction

1.1 Motivation

Systems in which only a part of the state is observed are called partially observable systems. Many real world systems are partially observable—some examples are autonomous driving, robotics, smart grids, etc. The main goal of this thesis is to design learning algorithms that allow the decision maker or agent to make multi-stage decisions (or actions) in partially observable systems that lead to favourable outcomes. Reinforcement learning (RL) is one such approach which is based on the agent learning which actions to take based on rewards that it receives for performing certain actions in certain states. RL does not require model information and is referred to as model-free learning. This is in contrast to model-based planning methods like dynamic programming, which use model information to learn optimal decision making strategies. RL has a natural fit for settings where the system dynamics (model information) are unknown, but data can be generated from the system which can then be used for learning agent behaviour. It is often the case that we do not know the dynamics for real word systems and so it is desirable to apply RL to partially observable systems.

However, most of the RL literature assumes that all the necessary information about the system (the state) is known at every time step and there is a large body of literature dedicated towards this [1–5]. The literature on systems which are partially observable (only a part of the state is available) is relatively sparse [6–10]. In contrast to fully observable systems where the current state information is known at every time step, agents in partially observable systems need to factor in the entire trajectory of past actions and observations

to make optimal decisions, and so partially observable systems are much harder to solve. Developing learning algorithms for agents in partially observable environments would be helpful for several real world problems. Furthermore, solving partially observable problems can also help with solving certain types of multi-agent team problems which follow the partial history sharing paradigm [11].

An information state is a representation that is sufficient for learning optimal decision making. A desirable information state allows us to compress the history to a smaller size. An approximate information state (AIS) can be learnt from data using function approximation without requiring any model information. Integral probability metrics (IPMs) are used to measure the similarity between two different probability distributions. An AIS is learnt by optimizing the “distance” between the actual AIS/observation distribution and the predicted AIS/observation distribution based on the current AIS and action. It was previously proposed to use the Wasserstein distance to learn the AIS [12]. We show that general IPMs can be used and also obtain theoretical performance guarantees for a distance-based maximum mean discrepancy loss (MMD IPM). Our experiments are based on the Wasserstein distance which is optimized in terms of a surrogate KL-divergence loss (KL IPM) and the MMD IPM.

We show that one can use RL algorithms in conjunction with the theoretical framework of AIS towards solving partially observable systems. We demonstrate the extent to which these ideas can work for simulated environments of varying difficulties from low-dimensional environments like Tiger [13] and CheeseMaze [14] to high-dimensional MiniGrid environments [15]. We also develop alternate strategies for high-dimensional observation spaces by compressing the observations using an autoencoder before applying the learning algorithms.

The rest of this chapter is structured as follows. Sec. 1.3 and 1.4 establish the standard notation used for MDPs and POMDPs respectively and also describe the relevant fundamental dynamic programming equations that can be used to find optimal policies. In Sec. 1.5, some of the recent planning and learning methods for POMDPs are discussed. We also describe some of the details of the baseline algorithms used in this work. The planning baselines are based on recent approaches towards solving partially observable environments of a tractable size. We conclude this chapter with the details of the exact contributions of this work, claims of originality and the publication under review.

1.2 Notation Used

We use uppercase letters to denote random variables (e.g. X, Y), lowercase letters to denote their realizations (e.g. x, y). The space of probability distributions of \mathbf{X} is given by $\Delta(\mathbf{X})$. \mathbb{P} and \mathbb{E} denote the probability and expectation of a random variable. Letters in sans serif font denote sets (e.g. \mathbf{X}, \mathbf{Y}).

1.3 Markov Decision Processes (MDP)

We start with a review of MDPs, this is standard material from [16–18]. An MDP consists of the tuple $(\mathbf{S}, \mathbf{A}, \mathbf{P}, R, \gamma)$ where:

- \mathbf{S} is the state space. At each timestep t , the agent is in some state $S_t \in \mathbf{S}$.
- \mathbf{A} is the action space. At each timestep t , the agent observes the full state S_t and chooses an action based on some policy $A_t = \pi_t(S_t)$, where $A_t \in \Delta(\mathbf{A})$.
- \mathbf{P} is the set of transition kernels for the agent to go from one state to another on performing a certain action at a certain timestep t , $S_{t+1} = P(S_t, A_t)$, where $P \in \mathbf{P}$, $S_{t+1} \in \Delta(\mathbf{S})$ and $A_t \in \Delta(\mathbf{A})$.
- $R : \mathbf{S} \times \mathbf{A} \mapsto \mathbb{R}$ is the reward function. The reward at timestep t is a random variable $R(S_t, A_t)$, which we denote as R_t .
- $\gamma \in [0, 1)$ is the discount factor used to give more importance to present rewards and less importance to future rewards. However, for finite horizon problems it can be omitted.

The performance of a policy $\pi = (\pi_1, \pi_2, \dots, \pi_{\mathbf{T}})$ over a time horizon \mathbf{T} is given by:

$$J(\pi) = \mathbb{E}^{\pi} \left[\sum_{t=1}^{\mathbf{T}} R_t \right]. \quad (1.1)$$

The objective is to find a policy π for the agent that maximizes the expected total reward. This can be done by dynamic programming as follows.

Proposition 1 (Policy evaluation for MDPs). For any given (state-dependent) policy π , define the *reward-to-go* function for any time t and realization s_t of state S_t as

$$V_t^\pi(s_t) := \mathbb{E}^\pi \left[\sum_{i=t}^T R_i \mid S_t = s_t \right]. \quad (1.2)$$

The reward-to-go functions defined above satisfy the following recursion. Define $V_{T+1}^\pi(S_{T+1}) = 0$ and for any $t \in \{T, \dots, 1\}$,

$$V_t^\pi(s_t) = \mathbb{E}^\pi [R_t + V_{t+1}^\pi(S_{t+1}) \mid S_t = s_t]. \quad (1.3)$$

The reward-to-go function $V_t^\pi(s_t)$ denotes the expected cumulative rewards obtained in the future when starting from state s_t at time t and following policy π . Note that $V_t^\pi(s_t)$ depends on the policy π only through the choice of the future policy (π_t, \dots, π_T) and therefore can be computed without the knowledge of the past policy $(\pi_1, \dots, \pi_{t-1})$.

Proposition 2 (Dynamic programming for MDPs). Recursively define *value functions* $\{V_t : \mathbf{S}_t \rightarrow \mathbb{R}\}_{t=1}^{T+1}$ as follows. $V_{T+1}(S_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$,

$$V_t(s_t) := \max_{a_t \in \mathbf{A}} \mathbb{E} [R_t + V_{t+1}(S_{t+1}) \mid S_t = s_t, A_t = a_t]. \quad (1.4)$$

Then, a policy $\pi = (\pi_1, \dots, \pi_T)$ is optimal if and only if for all $t \in \{1, \dots, T\}$ it satisfies

$$\text{Supp}(\pi_t(s_t)) \subseteq \arg \max_{a_t \in \mathbf{A}} \mathbb{E} [R_t + V_{t+1}(S_{t+1}) \mid S_t = s_t, A_t = a_t]. \quad (1.5)$$

Note that the expectation in (1.4) can be computed without the knowledge of the policy π .

1.4 Partially Observable Markov Decision Processes (POMDP)

We start with a review of POMDPs, this is standard material from [19]. A POMDP consists of the tuple $(\mathbf{S}, \mathbf{A}, \mathbf{P}, R, \gamma, \mathbf{O}, \mathbf{Y})$ where the initial elements are the same as in the MDP problem (except for the actions A_t). The actions are now a function of the history H_t of all past actions and observations and are represented by $A_t = \pi_t(H_t)$, where $A_t \in \Delta(\mathbf{A})$. The observational elements are described as follows:

- Instead of seeing the actual state S_{t+1} , we see observation $Y_t \in \Delta(\mathcal{Y})$ after taking action A_t in state S_t . \mathcal{Y} is the space of observations.
- O is the conditional observation probability which governs the observational dynamics of the system as $Y_t = O(S_{t+1}, A_t)$, where $Y_t \in \Delta(\mathcal{Y})$

The objective of the agent remains the same as in the MDP case described in Eq. (1.1). The only major difference is that the policies defined in this case are not defined over the states S_t , but over trajectories of actions and observations, i.e., the history H_t . A standard approach is to consider the belief over the state space as a “state” (usually called belief state) instead of the history H_t itself. Given a history H_t with realization h_t , the belief state at time t is defined as,

$$b_t(s_t) := \mathbb{P}(S_t = s_t | H_t = h_t).$$

Suppose we are given an initial belief state $b_0 \in \Delta(\mathcal{S})$, which represents the probability distribution over states. For $t \in \{\mathsf{T} - 1, \dots, 0\}$, the updated belief state b_{t+1} can be written in terms of the current belief state b_t and the POMDP observation and transition dynamics as follows,

$$b_{t+1}(s_{t+1}) = \frac{\mathbb{P}(y_t | s_{t+1}, a_t)}{\mathbb{P}(y_t | b_t, a_t)} \sum_{s_t \in \mathcal{S}} \mathbb{P}(s_{t+1} | s_t, a_t) b_t(s_t),$$

$$\mathbb{P}(y_t | b_t, a_t) = \sum_{s_{t+1} \in \mathcal{S}} \mathbb{P}(y_t | s_{t+1}, a_t) \sum_{s_t \in \mathcal{S}} \mathbb{P}(s_{t+1} | s_t, a_t) b_t(s_t).$$

A dynamic program can be written down to solve for the optimal policy using the belief state as follows (this can be done because the belief state is a sufficient statistic for optimal policies in POMDPs).

Proposition 3 (Policy evaluation for POMDPs using belief state). For any given (belief state-dependent) policy π , define the *reward-to-go* function for any time t and realization b_t of belief B_t as

$$V_t^\pi(b_t) := \mathbb{E}^\pi \left[\sum_{i=t}^{\mathsf{T}} R_i \mid B_t = b_t \right]. \quad (1.6)$$

The reward-to-go functions defined above satisfy the following recursion. Define $V_{T+1}^\pi(B_{T+1}) = 0$ and for any $t \in \{T, \dots, 1\}$,

$$V_t^\pi(b_t) = \mathbb{E}^\pi[R_t + V_{t+1}^\pi(B_{t+1}) \mid B_t = b_t]. \quad (1.7)$$

The reward-to-go function $V_t^\pi(b_t)$ denotes the expected cumulative rewards obtained in the future when starting from belief b_t at time t and following policy π . Note that $V_t^\pi(b_t)$ only depends on the policy π only through the choice of the future policy (π_t, \dots, π_T) and therefore can be computed without the knowledge of the past policy $(\pi_1, \dots, \pi_{t-1})$.

Proposition 4 (Dynamic programming for POMDPs using belief state). Recursively define *value functions* $\{V_t: \mathbf{B}_t \rightarrow \mathbb{R}\}_{t=1}^{T+1}$ as follows. $V_{T+1}(B_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$,

$$V_t(b_t) := \max_{a_t \in \mathbf{A}} \mathbb{E}[R_t + V_{t+1}(B_{t+1}) \mid B_t = b_t, A_t = a_t]. \quad (1.8)$$

Then, a stochastic policy $\pi = (\pi_1, \dots, \pi_T)$ is optimal if and only if for all $t \in \{1, \dots, T\}$ it satisfies

$$\text{Supp}(\pi_t(b_t)) \subseteq \arg \max_{a_t \in \mathbf{A}} \mathbb{E}[R_t + V_{t+1}(B_{t+1}) \mid B_t = b_t, A_t = a_t]. \quad (1.9)$$

Note that the expectation in (1.8) can be computed without the knowledge of the policy π .

The belief space $\Delta(\mathbf{S})$ is the space of probability distributions over the state space. Although the belief space is continuous, it does not increase in size over time (as history does) and it has the following useful properties:

1. It is a sufficient statistic for optimal policies (it can be used to determine an optimal policy just as well as if we were using H_t instead).
2. The value function is piecewise linear and convex (PWLC). This means that it can be represented by a set of hyperplanes in the belief space which are generally referred to as α -vectors. For a belief b_t at time t , and a finite collection of hyperplanes Γ , this can be written as

$$V(b_t) = \max_{\alpha \in \Gamma} (\alpha \cdot b_t). \quad (1.10)$$

The PWLC property is preserved even after dynamic programming updates are made to the value function.

1.5 Overview of current techniques for planning and learning in POMDPs

Most approaches to solve MDP/POMDP problems can be categorized as model-based (planning) or model-free (learning) methods (although sometimes a hybrid approach is also followed). Model-based methods use model information such as the observation and transition probabilities and reward function used in the problem, whereas model-free methods rely on the agent acting in the environment and learning from the data observed by the agent. Model-based methods require model information which is not always available and they work extremely well for smaller problems, but for larger problems they become computationally intractable and model-free methods tend to be more effective.

1.5.1 Planning in POMDPs

Model-based methods are categorized as exact or approximate methods based on whether they solve the dynamic program for the belief state exactly or approximately. The PWLC property of the value function of the belief state allows it to be represented by a number of hyperplanes in the belief space denoted as α -vectors.

Incremental pruning [20] is an exact method that takes advantage of these properties to find the set of α -vectors that represents the optimal value function without having any redundant α -vectors that are dominated by others, i.e., referred to as a set having unique representation of minimum size in [20]. At the initial step there are a certain number of α -vectors, which become much larger after every backup operation (dynamic programming update), but many of the α -vectors obtained after performing a backup are dominated by other α -vectors because of the max operation in Eq. (1.10) and can be discarded. Dominated α -vectors can be found by solving a linear program and pruned accordingly. One of the main drawbacks of this method is that it considers the entire belief space, even though many of these points would never be reached for certain problems.

To reduce the number of belief points considered in the backup operations, point-based value iteration (PBVI) [8] can be used to only consider the reachable belief space rather than the entire belief space. This is an approximate method and works by considering the initial belief point and expanding it based on transition and observation probabilities for all possible actions. This limits the exponential growth (the growth becomes a polynomial

function of the factors involved) of α -vectors at every time step and thus the pruning step takes much lesser time. PBVI also has theoretical guarantees on the error estimates of the value function. This allows pruning based methods which take advantage of the PWLC property to be applied to much larger problems.

Another considerable reduction can be achieved by the method of successive approximations of the reachable space under optimal policies (SARSOP) [21], which further tries to consider the reachable space of belief points under an optimal policy. This is done by considering an upper and lower bound to the value function and following some heuristics to increase the likelihood of selecting belief points that would only be reachable under an optimal policy. This is also an approximate method and is applicable to larger problems than PBVI.

In contrast, a method which follows a completely different approach is an approximate method called QMDP [22]. This method uses the transition dynamics to solve the MDP problem as if the full state information was available and arrives at a value function based on state. It then combines the belief state information with these values to arrive at a value function for the partial observation case. This method is capable of working in certain problems, but a major disadvantage is that it is unable to find policies which need to take an action to gain information so that it can make a more informed decision. This method can be combined with learning approaches to overcome some of the limitations.

The planning methods described above are some of the standard methods to solve POMDPs that do not have very high dimensionality in their state, observation and action spaces. There is also standardized code available [23] to compare these algorithms with each other. This allows us to easily use these algorithms as a planning baseline for comparison with our approach. It should be noted that several other approaches to planning in POMDPs exist, however, we do not present these as we do not use them in the numerical experiments of this work. Some such algorithms are partially observable Monte Carlo planning (POMCP) [24] which uses a Monte Carlo update of the agent’s belief state with a Monte Carlo tree search from the current belief state, anytime regularized determinized sparse partially observable tree (ARDESPOT) [25], Monte Carlo value iteration (MCVI) [26] which presents an algorithm for continuous-state and continuous-observation POMDPs, anytime error minimization search (AEMS) [27] which aims to make more efficient computations by combining offline and online computations effectively. For a survey of point-based POMDP solvers see [28].

An alternative to these approaches is to consider the general idea of information state, which is a subset of all the information available to us (history of observations and actions) and is sufficient to find the optimal policy through dynamic programming. This allows us to reduce the policy search space considerably and allows more flexibility for planning solutions. Since the PWLC properties no longer necessarily hold, the solution methods for an information state are based on approximate dynamic programming [29]. But even in these cases, the planning solutions may become intractable for larger observation, state and action spaces.

1.5.2 Learning in POMDPs

In RL, the policy can be directly optimized to maximize returns. These methods are called policy gradient methods. REINFORCE [1] and recurrent policy gradients (RPG) [30] can be used for fully and partially observed systems respectively. A more recent policy gradient method is proximal policy optimization (PPO) [31]. These algorithms are described in detail at the end of this section.

Since there has been more work done on fully observable systems than on partially observable systems in RL, one direction has been to try to convert partially observable systems to fully observable systems by adding together past observations like in Atari games [32], the frames are stacked together in an attempt to incorporate more information in a decision making step. But it is not always computationally feasible to catch longer temporal dependencies in this manner and it is desirable for a learnt representation to develop notions of these dependencies via a learning process rather than setting it as a design parameter for a network. The closest work that is similar to DQN [3] for the partially observable case is deep recurrent Q-Networks (DRQN) [33] which involves creating a replay buffer of the agent’s experiences and sampling truncated trajectories (instead of transition samples as in DQN) from it in order to train recurrent network policies (up to a maximum of 10 recent frames in their experiments). The limitation of this work is that the horizons that are set are not very large, so long temporal dependencies cannot be captured.

More recent state-of-the-art work using recurrent policy networks with replay buffers is recurrent replay distributed DQN (R2D2) [10] which uses sequences up to a length of 80 frames in some experiments and improves upon several existing baselines in the Atari [32]

and DMLab-30 [34] environments. R2D2 incorporates some features from Rainbow DQN [4] and trains with an asynchronous multi-actor decentralized architecture. Some ways of initializing the internal cell states of the recurrent cells are introduced to facilitate effective learning.

Another approach is to learn a representation that is capable of predicting the next observation/next latent state as is done in world models [6], learning causal state representations [7] and approximate information state (AIS) [12]. In AIS, using this main idea makes it possible to apply approximate dynamic programming and RL algorithms with theoretical guarantees. More recent work on world models is PlaNet [35] and learning behaviors by latent imagination [36]. These ideas involve learning a model of the dynamics of the environment and then acting in the environment based on this environment dynamics model. PlaNet applies a planning solution to the learnt dynamics, whereas learning behaviors by latent imagination learns a policy by RL on artificial data that is generated by the learnt model. The main difference between papers based on world models and AIS is that there are provable theoretical guarantees on optimal performance with AIS. The world model based papers are based on several heuristics and empirical results. Another important difference is the method by which a representation is learnt, the techniques proposed in AIS could be directly applicable to learning effective environment models. In addition to this, the work presented in this thesis proposes more general ways to learn such environment models.

State aggregation techniques based on bisimulation metrics have been proposed in [37, 38] for MDPs and [39] for POMDPs. The key insight of these papers is to define a semi-metric called bisimulation metric on the state space of an MDP or the belief space of a POMDP as the unique fixed point of an operator on the space of semi-metrics on the state space of the MDP or the belief space of the POMDP. It is then shown that the value function is Lipschitz with respect to this metric. Then, they propose state aggregation based on the bisimulation metric.

A completely different class of model-based RL algorithms are methods using predictive state representations (PSRs) [40, 41]. PSRs are constructed only based on observational data so they can easily be adapted to the RL setup. There have been a number of papers which use PSRs to propose model based RL algorithms [42–48]. Various methods for learning low dimensional approximations of PSRs have been proposed in the literature, including approaches which use spectral learning algorithms [43–48], and stochastic gradient

descent [48]. Error bounds for using an approximate PSR were derived in [45, 49].

Bisimulation metrics and PSRs are conceptually different from the style of arguments used in this thesis, so we do not make any comparisons with these learning algorithms.

Model-based methods require information about the model that may generally be unknown. Additionally, although model-based solutions work well for low to moderate-dimensional problems, they become much harder for high-dimensional problems and the computations involved become intractable. A more attractive alternative is to consider model-free solutions which are based on learning from the agent’s past experience with the environment. In the remainder of this section, we describe the details of the RL background used in this work.

Policy gradient optimization - REINFORCE/RPG

These methods work on the basis of a parameterized policy, which is updated based on the policy gradient to improve the performance of the current policy via stochastic gradient ascent methods. From [1], the policy gradient for the full state case based on the REINFORCE algorithm is given as

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [G_t \nabla_{\theta} \log \pi_{\theta}(A_t | S_t)] . \quad (1.11)$$

Here G_t is the discounted return obtained from timestep t and is given as $\sum_{k=t}^T \gamma^{k-t} R_k$. The expectation is generally estimated using Monte Carlo samples of trajectories, following which updates to the parameters θ are made using any stochastic gradient ascent based method.

For the case of POMDPs, the same algorithm can be followed but using histories H_t rather than states S_t and one approach is to use network architectures which have recurrent connections (like RNNs or LSTMs) like in the recurrent policy gradients (RPG) algorithm [30]. The gradients in this case must be back-propagated through time (BPTT). One of the issues with this is that it requires entire trajectories to be fed to the network to get the output actions for that step. This can be somewhat alleviated by using truncated BPTT which only requires trajectories up to a certain fixed length to be fed in. However, doing so results in the loss of sequence information beyond that length in the decision making process. Based on [30] the policy gradients are given by:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=1}^T G_t \nabla_{\theta} \log \pi_{\theta}(A_t | H_t) \right]. \quad (1.12)$$

However, this equation requires episodes to have completed before the returns (and therefore the gradients) can be calculated. Thus Eq. (1.12) takes a backward view because it has to look back at past rewards after episode completion. In contrast, a forward view can also be taken and the same equation can be re-written using the G(PO)MDP gradient [50]. Eq. (1.13) allows us to compute the gradients online instead of computing gradients after episodes are completed.

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\sum_{t=1}^T \left(\sum_{\tau=1}^t \nabla_{\theta} \log \pi_{\theta}(A_{\tau} | H_{\tau}) \right) \gamma^{t-1} R_t \right]. \quad (1.13)$$

Proximal policy optimization with LSTMs

A more recent algorithm than RPG which is used in MiniGrid environments [51] as a baseline (for training from scratch) uses proximal policy optimization (PPO) [31] with a long short-term memory (LSTM) layer to capture the effect of the history of observations and actions.

An interesting way to look at the PPO update is by thinking of it as an update which does not allow the parameters θ to change significantly from the old parameters θ_{old} . A useful term to express this in the objective is the ratio between the policy probabilities of the new and old policies, i.e., $r_t^{PPO}(\theta) = \frac{\pi_{\theta}(a_t, h_t)}{\pi_{\theta_{old}}(a_t, h_t)}$. Another useful concept to define the PPO loss function is the advantage function which we denote here by A_t^{PPO} which is a general term which represents the value function minus an action independent baseline. The clipped surrogate objective for PPO is given by

$$L^{PPO}(\theta) = \mathbb{E}_{\pi_{\theta}} \left[\min(r_t^{PPO}(\theta) A_t^{PPO}, \text{clip}(r_t^{PPO}(\theta), 1 - \epsilon^{PPO}, 1 + \epsilon^{PPO}) A_t^{PPO}) \right], \quad (1.14)$$

where ϵ^{PPO} is a hyperparameter which controls how much the ratio $r_t^{PPO}(\theta)$ is allowed to deviate from the old policy by restricting it to lie within the $[1 - \epsilon^{PPO}, 1 + \epsilon^{PPO}]$ interval. The expectation is estimated using Monte Carlo samples of trajectories.

The difference between the original PPO equation in [31] and Eq. (1.14) is that the state s_t is replaced by the history h_t . This loss can be easily implemented through any Autograd

library (like TensorFlow or PyTorch). The losses at a time step t can be back-propagated throughout the entire history h_t . Since this might be computationally intensive, something similar to truncated BPTT can be used by only back-propagating gradients up to a finite number of time steps (as is done in [51]). Following this, the gradients obtained can be used in any gradient descent based algorithm (like ADAM).

1.6 Contributions of this work

The focus of this work is to extend the theory of AIS [12] and to evaluate the empirical performance of the theoretical framework of AIS. The theory of AIS is described in Chapter 2. The details on how reinforcement learning algorithms are designed in conjunction with AIS are given in Chapter 3. Experimentation on the network architecture and hyperparameters are carried out extensively in Chapter 4. The results show improvement in the performance of the algorithm for the environments considered in [12], as well as positive results for larger environments like those in the JuliaPOMDP repository [23] and MiniGrid environments [15]. In addition to improving the experimental results, further improvements are obtained for the new theory introduced in this work in some cases, although there are still some limitations to the theory introduced here.

We extend the recently proposed work on AIS [12] to include environments of moderate-dimensionality and the MiniGrid environments [15] which are of high-dimensionality. The idea of observation compression is used here to reduce the large observation space to something more tractable for high-dimensional environments. We introduce the MMD IPM, develop the corresponding theory and then apply this IPM to the same low, moderate and high-dimensional environments as the KL IPM (with similar observation compression for high-dimensional environments). The results based on the KL IPM for environments considered in [12] are also improved upon.

1.7 Claims of originality and publications

1.7.1 Claims of originality

1. Development of the theory for kernel-based IPM for optimizing AIS.

2. Improvement in experimental results (low-dimensional environments) of original methodology of optimizing AIS using the KL IPM in [12]. Furthermore, produced results for new environments not considered previously (moderate-dimensional environments).
3. Experimental results (low and moderate-dimensional environments) achieved using kernel-based MMD IPM.
4. Application of AIS to more difficult (high-dimensional) environments than that originally proposed in [12]. This includes results for the observation compression paradigm for both KL and MMD IPMs to environments with a large observation space.

1.7.2 Journal publication under review

J. Subramanian, A. Sinha, R. Seraj, and A. Mahajan, “Approximate information state for approximate planning and reinforcement learning in partially observed systems,” 2020 [52].

1.7.3 Contributions of co-authors

J. Subramaniam and A. Mahajan contributed equally to the basic ideas involved in AIS. A. Sinha and A. Mahajan contributed equally to the development of ideas involved in the kernel based IPMs. A. Sinha was primarily responsible for the numerical results in this work. R. Seraj was primarily responsible for the low-dimensional results for the planning baselines, and for the low and moderate-dimensional results for the PPO with LSTM baseline.

Chapter 2

Approximate information state

The recently proposed work on approximate information state (AIS) [12] requires that the predicted AIS/observation distribution be similar to the actual AIS/observation distribution given the current AIS and the action. To achieve this, they propose using the Wasserstein distance between the two distributions, but since it is expensive to compute, a surrogate KL-divergence loss between the two distributions is used which upper-bounds the Wasserstein distance.

In this work, we develop the notion of similarity between the predicted AIS/observation distribution and the actual AIS/observation distribution in terms of general integral probability metrics (IPMs), which can be used to measure the “distance” between the two distributions. In addition to the already existing approach of using the KL-divergence loss which upper-bounds the Wasserstein distance (which we denote as KL IPM), we introduce a distance-based maximum mean discrepancy (MMD IPM) which is based on a reduced kernel hilbert space (RKHS). We also introduce the idea of learning an AIS with compressed observations to deal with large observation spaces and show how the theoretical results on AIS naturally to extend to this case.

We begin this chapter by introducing the input-output model considered for POMDPs in Sec. 2.1. We choose this input-output model because it facilitates our style of arguments easily. The standard approach for POMDPs is to use the belief state dynamic program (Prop. 3 and 4). However, this requires model information which is not always available in the RL setting. Thus, we start by writing down the dynamic program based on history (all past actions and observations) in Prop. 5 and 6. The history tends to be large because of

the exponential growth of trajectories and therefore may not be efficient for identifying an optimal policy. An information state is a compressed representation of the history that is also a sufficient statistic for finding optimal policies. By using an information state instead of belief state, we lose the useful PWLC property which makes it easier to find optimal policies as described in Sec. 1.5.1. Even so, it is sometimes advantageous to do so because an information state can be significantly smaller than the belief state and so it may be easier to apply dynamic programming updates. The details are described in Sec. 2.2. Following this, we show how an AIS can be obtained from data in Sec. 2.3 using approximate dynamic programming. We also show that performance bounds can be obtained for certain choices of IPMs for the finite horizon case in Sec. 2.3 and for the infinite horizon case in Sec 2.4.

2.1 Input-output model considered

We view a partially observed system as a black-box input-output system shown in Fig. 2.1 instead of the standard POMDP model described in 1.4. At each time t , the system has two inputs and generates two outputs. The inputs to the system are a control input (also called an action) $A_t \in \mathbf{A}$ and a disturbance $W_t \in \mathbf{W}$. The outputs of the system are an observation $Y_t \in \mathbf{Y}$ and a reward $R_t \in \mathbb{R}$. For the ease of exposition, we assume that \mathbf{A} , \mathbf{W} , and \mathbf{Y} are finite sets. The analysis extends to general spaces under appropriate technical conditions. The history is given by past actions and observations, i.e., $h_t = (a_{1:t-1}, y_{1:t-1})$. Note that this is slightly atypical notation and the order in which the input and output variables are generated is shown in Fig. 2.2.

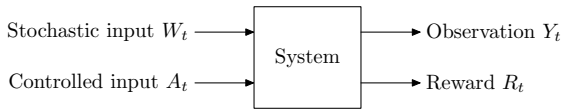


Fig. 2.1 A stochastic input-output system

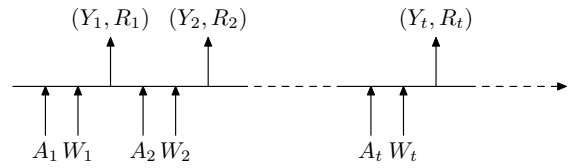


Fig. 2.2 The timing diagram of the input-output system.

We can say that the outputs (Y_t, R_t) at time t are some function of all the inputs

$(A_{1:t}, W_{1:t})$ up to time t , i.e.,

$$Y_t = f_t(A_{1:t}, W_{1:t}) \quad \text{and} \quad R_t = r_t(A_{1:t}, W_{1:t}),$$

where $\{f_t: \mathbf{A}^t \times \mathbf{W}^t \rightarrow \mathbf{Y}\}_{t=1}^T$ are called the system output functions and $\{r_t: \mathbf{A}^t \times \mathbf{W}^t \rightarrow \mathbb{R}\}_{t=1}^T$ are called the system reward functions.

There is an agent which observes the output Y_t and generates a control input or the action A_t as a (possibly stochastic) function of the history $H_t = (Y_{1:t-1}, A_{1:t-1})$ of the past observations and actions, i.e.,

$$A_t \sim \pi_t(H_t),$$

where $\pi := (\pi_t)_{t \geq 1}$ is a (history-dependent and possibly stochastic) policy. We use \mathbf{H}_t to denote the space of all histories up to time t . Then the policy π_t is a mapping from \mathbf{H}_t to $\Delta(\mathbf{A})$ (which denotes the space of probability measures on \mathbf{A}). We will use $\pi_t(a_t|h_t)$ to denote the probability of choosing action a_t at time t given history h_t and use $\text{Supp}(\pi_t(h_t))$ to denote the support of π_t (i.e., the set of actions chosen with positive probability).

We assume that the disturbance $\{W_t\}_{t \geq 1}$ is a sequence of independent random variables defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Thus, if the control input process $\{A_t\}_{t \geq 1}$ is specified, then the output processes $\{Y_t, R_t\}_{t \geq 1}$ are random variables on $(\Omega, \mathcal{F}, \mathbb{P})$. Specifying a policy π for the agent induces a probability measure on the output processes $\{Y_t, R_t\}_{t \geq 1}$, which we denote by \mathbb{P}^π .

Exact planning solutions are easy to compute in the case of MDPs with a finite number of states and actions through dynamic programming (provided that the number of states and actions is not too large). Value iteration and policy iteration use the Bellman Optimality Equation [1] to give exact planning solutions through dynamic programming. For the case of POMDPs, we can view the history H_t as a “state” of a Markov decision process (MDP) with transition probability

$$\mathbb{P}(H_{t+1} = (h'_t, a'_t, y_t) \mid H_t = h_t, A_t = a_t) = \begin{cases} \mathbb{P}(Y_t = y_t \mid H_t = h_t, A_t = a_t), & \text{if } h'_t = h_t \text{ \& } a'_t = a_t \\ 0, & \text{otherwise} \end{cases}$$

and per-step reward $\mathbb{E}[R_t \mid H_t, A_t]$. The same procedure for MDPs can be followed after that.

Proposition 5 (Policy evaluation for POMDPs using history as state). For any given

(history-dependent) policy π , define the *reward-to-go* function for any time t and realization h_t of history H_t as

$$V_t^\pi(h_t) := \mathbb{E}^\pi \left[\sum_{i=t}^T R_i \mid H_t = h_t \right]. \quad (2.1)$$

The reward-to-go functions defined above satisfy the following recursion. Define $V_{T+1}^\pi(H_{T+1}) = 0$ and for any $t \in \{T, \dots, 1\}$,

$$V_t^\pi(h_t) = \mathbb{E}^\pi [R_t + V_{t+1}^\pi(H_{t+1}) \mid H_t = h_t]. \quad (2.2)$$

The reward-to-go function $V_t^\pi(h_t)$ denotes the expected cumulative rewards obtained in the future when starting from history h_t at time t and following policy π . Note that $V_t^\pi(h_t)$ only depends on the policy π only through the choice of the future policy (π_t, \dots, π_T) and therefore can be computed without the knowledge of the past policy $(\pi_1, \dots, \pi_{t-1})$.

Proposition 6 (Dynamic programming for POMDPs using history as state). Recursively define *value functions* $\{V_t: \mathbf{H}_t \rightarrow \mathbb{R}\}_{t=1}^{T+1}$ as follows. $V_{T+1}(H_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$,

$$V_t(h_t) := \max_{a_t \in \mathbf{A}} \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t]. \quad (2.3)$$

Then, a stochastic policy $\pi = (\pi_1, \dots, \pi_T)$ is optimal if and only if for all $t \in \{1, \dots, T\}$ it satisfies

$$\text{Supp}(\pi_t(h_t)) \subseteq \arg \max_{a_t \in \mathbf{A}} \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t]. \quad (2.4)$$

Note that the expectation in (2.3) is with respect to the probability measure \mathbb{P} on (Ω, \mathcal{F}) and can be computed without the knowledge of the policy π .

However, for POMDPs with a large horizon there is a large number of possible “states” because of the exponential growth of trajectories and so this can only be used for small sized problems with a short horizon. It is shown in [53] that solving POMDPs optimally is NP-hard. Worst case POMDPs are undecidable. Finite-horizon POMDPs are PSPACE-complete [54].

2.2 Information state

Definition 1. Let $\{Z_t\}_{t=1}^T$ be a pre-specified collection of Banach spaces. A collection $\{\sigma_t: \mathbf{H}_t \rightarrow Z_t\}_{t=1}^T$ of history compression functions is called an *information state generator*

if the process $\{Z_t\}_{t=1}^T$, where $Z_t = \sigma_t(H_t)$, satisfies the following properties:

(P1) Sufficient for performance evaluation, i.e., for any time t , any realization h_t of H_t and any choice a_t of A_t , we have

$$\mathbb{E}[R_t \mid H_t = h_t, A_t = a_t] = \mathbb{E}[R_t \mid Z_t = \sigma_t(h_t), A_t = a_t].$$

(P2) Sufficient to predict itself, i.e., for any time t , any realization h_t of H_t and any choice a_t of A_t , we have that for any Borel subset B of Z_{t+1} ,

$$\mathbb{P}(Z_{t+1} \in B \mid H_t = h_t, A_t = a_t) = \mathbb{P}(Z_{t+1} \in B \mid Z_t = \sigma_t(h_t), A_t = a_t).$$

In the sequel, we will sometimes use the phrase “let $\{Z_t\}_{t=1}^T$ be an information state” to specify an information state and will implicitly assume that the corresponding information state spaces are $\{Z_t\}_{t=1}^T$ and the corresponding compression functions are $\{\sigma_t\}_{t=1}^T$.

Note that both the probabilities in Property (P2) can be computed without the knowledge of the policy π . Furthermore, there are no restrictions on the spaces $\{Z_t\}_{t=1}^T$ although in practice an information state is useful only when these spaces are “small” in an appropriate sense.

Condition (P1) is easy to verify but condition (P2) can be a bit abstract. For some models, instead of (P2), it is easier to verify the following stronger conditions:

(P2a) Evolves in a state-like manner, i.e., there exist measurable functions $\{\varphi_t\}_{t=1}^T$ such that for any time t and any realization h_{t+1} of H_{t+1} , we have

$$\sigma_{t+1}(h_{t+1}) = \varphi_t(\sigma_t(h_t), y_t, a_t).$$

Informally, the above condition may be written as $Z_{t+1} = \varphi_t(Z_t, Y_t, A_t)$.

(P2b) Is sufficient for predicting future observations, i.e., for any time t , any realization h_t of H_t and any choice a_t of A_t , we have that for any subset D of Y ,

$$\mathbb{P}(Y_t \in D \mid H_t = h_t, A_t = a_t) = \mathbb{P}(Y_t \in D \mid Z_t = \sigma_t(h_t), A_t = a_t).$$

Proposition 7. (P2a) and (P2b) imply (P2).

Proof. See Appendix A. □

Next, we show that an information state is useful because it is always possible to write a dynamic program based on the information state. To explain this dynamic programming decomposition, we first write the history-based dynamic programs of Proposition 5 and 6 in a more compact manner as follows: Let $V_{T+1}(h_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$, define

$$Q_t(h_t, a_t) := \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t], \quad (2.5a)$$

$$V_t(h_t) := \arg \max_{a_t \in \mathbf{A}} Q_t(h_t, a_t). \quad (2.5b)$$

The function $Q_t(h_t, a_t)$ is called the action-value function. Moreover, for a given stochastic policy $\pi = (\pi_1, \dots, \pi_T)$, where $\pi_t: \mathbf{H}_t \rightarrow \Delta(\mathbf{A}_t)$, let $V_{T+1}^\pi(h_{T+1}) = 0$ and for $t \in \{T, \dots, 1\}$, define

$$Q_t^\pi(h_t, a_t) := \mathbb{E}[R_t + V_{t+1}^\pi(H_{t+1}) \mid H_t = h_t, A_t = a_t], \quad (2.6a)$$

$$V_t^\pi(h_t) := \sum_{a_t \in \mathbf{A}} \pi_t(a_t \mid h_t) \cdot Q_t^\pi(h_t, a_t). \quad (2.6b)$$

Theorem 1. *Let $\{Z_t\}_{t=1}^T$ be an information state. Recursively define value functions $\{\bar{V}_t: \mathbf{Z}_t \rightarrow \mathbb{R}\}_{t=1}^{T+1}$, as follows: $\bar{V}_{T+1}(z_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$:*

$$\bar{Q}_t(z_t, a_t) := \mathbb{E}[R_t + \bar{V}_{t+1}(Z_{t+1}) \mid Z_t = z_t, A_t = a_t], \quad (2.7a)$$

$$\bar{V}_t(z_t) := \max_{a_t \in \mathbf{A}} \bar{Q}_t(z_t, a_t). \quad (2.7b)$$

Then, we have the following:

1. *For any time t , history h_t , and action a_t , we have that*

$$Q_t(h_t, a_t) = \bar{Q}_t(\sigma_t(h_t), a_t) \text{ and } V_t(h_t) = \bar{V}_t(\sigma_t(h_t)). \quad (2.8)$$

2. *Let $\bar{\pi} = (\bar{\pi}_1, \dots, \bar{\pi}_T)$, where $\bar{\pi}_t: \mathbf{Z}_t \rightarrow \Delta(\mathbf{A})$, be a stochastic policy. Then, the policy $\pi = (\pi_1, \dots, \pi_T)$ given by $\pi_t = \bar{\pi}_t \circ \sigma_t$ is optimal if and only if for all t and all realizations z_t of information states Z_t , $\text{Supp}(\bar{\pi}_t(z_t)) \subseteq \arg \max_{a_t \in \mathbf{A}} \bar{Q}_t(z_t, a_t)$.*

Proof. See Appendix A. □

2.3 Approximate information state

Given the benefits of a good information state, it is natural to consider a data-driven approach to identify an information state. An information state identified from data will not be exact and it is important to understand what is the loss in performance when using an approximate information state. Theorem 1 shows that a compression of the history which satisfies properties (P1) and (P2) is sufficient to identify a dynamic programming decomposition. Would a compression of history that approximately satisfied properties (P1) and (P2) lead to an approximate dynamic program? In this section, we show that the answer to this question is yes. First, we need to precisely define what we mean by “approximately satisfy properties (P1) and (P2)”. For that matter, we need to fix a distance metric on probability spaces. There are various metrics on probability space and it turns out that the appropriate distance metric for our purposes is the integral probability metric (IPM) [55].

2.3.1 Integral probability metrics (IPM)

Definition 2. Let (X, \mathcal{G}) be a measurable space and \mathfrak{F} denote a class of uniformly bounded measurable functions on (X, \mathcal{G}) . The integral probability metric (IPM) between two probability distributions $\mu, \nu \in \Delta(X)$ with respect to the function class \mathfrak{F} is defined as

$$d_{\mathfrak{F}}(\mu, \nu) := \sup_{f \in \mathfrak{F}} \left| \int_X f d\mu - \int_X f d\nu \right|.$$

In the literature, IPMs are also known as probability metrics with a ζ -structure; see e.g., [56, 57]. They are useful to establish weak convergence of probability measures. Methods for estimating IPM from samples are discussed in [58].

Examples of integral probability metrics (IPMs)

When (X, \mathcal{G}) is a metric space, then various commonly used distance metrics on (X, \mathcal{G}) lead to specific instances of IPM for a particular choice of function space \mathfrak{F} . We provide some examples below:

1. **TOTAL VARIATION DISTANCE:** If \mathfrak{F} is chosen as $\{f : \|f\|_{\infty} \leq 1\}$, then $d_{\mathfrak{F}}$ is the total

variation distance.¹

2. KOLMOGOROV DISTANCE: If $\mathsf{X} = \mathbb{R}^m$ and \mathfrak{F} is chosen as $\{\mathbf{1}_{(-\infty, t]} : t \in \mathbb{R}^m\}$, then $d_{\mathfrak{F}}$ is the Kolmogorov distance.
3. KANTOROVICH METRIC OR WASSERSTEIN DISTANCE: Let $\|f\|_{\text{Lip}}$ denote the Lipschitz semi-norm of a function. If \mathfrak{F} is chosen as $\{f : \|f\|_{\text{Lip}} \leq 1\}$, then $d_{\mathfrak{F}}$ is the Kantorovich metric. When X is separable, the Kantorovich metric is the dual representation of the Wasserstein distance via the Kantorovich-Rubinstein duality [59].
4. BOUNDED-LIPSCHITZ METRIC: If \mathfrak{F} is chosen as $\{f : \|f\|_{\infty} + \|f\|_{\text{Lip}} \leq 1\}$, then $d_{\mathfrak{F}}$ is the bounded-Lipschitz (or Dudley) metric.
5. MAXIMUM MEAN DISCREPANCY (MMD): Let \mathcal{H} be a reproducing kernel Hilbert space (RKHS) of real valued functions on X and let $\mathfrak{F} = \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq 1\}$, then $d_{\mathfrak{F}}$ is the maximum mean discrepancy² [61]. The energy distance studied in statistics [62] is a special case of maximum mean discrepancy; see [63] for a discussion.

We say that \mathfrak{F} is a closed set if it is closed under the topology of pointwise convergence. We say that \mathfrak{F} is a convex set if $f_1, f_2 \in \mathfrak{F}$ implies that for any $\lambda \in (0, 1)$, $\lambda f_1 + (1 - \lambda)f_2 \in \mathfrak{F}$. Note that all the above function classes are convex and all except Kolmogorov distance are closed.

We now list some useful properties of IPMs, which immediately follow from definition.

¹In particular, if μ and ν are absolutely continuous with respect to some measure λ and let $p = d\mu/d\lambda$ and $q = d\nu/d\lambda$, then

$$\left| \int_{\mathsf{X}} f d\mu - \int_{\mathsf{X}} f d\nu \right| = \left| \int_{\mathsf{X}} f(x)p(x)\lambda(dx) - \int_{\mathsf{X}} f(x)q(x)\lambda(dx) \right| \leq \|f\|_{\infty} \int_{\mathsf{X}} |p(x) - q(x)|\lambda(dx).$$

In this paper, we are defining total variation distance as $\int_{\mathsf{X}} |p(x) - q(x)|\lambda(dx)$. Typically, it is defined as half of that quantity. Note that it is possible to get a tighter bound than above where $\|f\|_{\infty}$ is replaced by $\frac{1}{2} \text{span}(f) = \frac{1}{2}(\max(f) - \min(f))$.

²One of features of MMD is that the optimizing f can be identified in closed form. In particular, if k is the kernel of the RKHS, then (see [58, 60] for details)

$$\begin{aligned} d_{\mathfrak{F}}(\mu, \nu) &= \left\| \int_{\mathsf{X}} k(\cdot, x) d\mu(x) - \int_{\mathsf{X}} k(\cdot, x) d\nu(x) \right\|_{\mathcal{H}} \\ &= \left[\int_{\mathsf{X}} \int_{\mathsf{X}} k(x, y) \mu(dx) \mu(dy) + \int_{\mathsf{X}} \int_{\mathsf{X}} k(x, y) \nu(dx) \nu(dy) - 2 \int_{\mathsf{X}} \int_{\mathsf{X}} k(x, y) \mu(dx) \nu(dy) \right]^{1/2}. \end{aligned}$$

We use an MMD as a IPM in the PORL algorithms proposed in Sec. 3, where we exploit this property.

1. Given a function class \mathfrak{F} and a function f (not necessarily in \mathfrak{F}),

$$\left| \int_{\mathbf{X}} f d\mu - \int_{\mathbf{X}} f d\nu \right| \leq \rho_{\mathfrak{F}}(f) \cdot d_{\mathfrak{F}}(\mu, \nu), \quad (2.9)$$

where $\rho_{\mathfrak{F}}(f)$ is the Minkowski functional with respect to \mathfrak{F} given by

$$\rho_{\mathfrak{F}}(f) := \inf\{\rho \in \mathbb{R}_{>0} : \rho^{-1}f \in \mathfrak{F}\}. \quad (2.10)$$

For the total variation distance, $|\int_{\mathbf{X}} f d\mu - \int_{\mathbf{X}} f d\nu| \leq \frac{1}{2} \text{span}(f) d_{\mathfrak{F}}(\mu, \nu)$. Thus, for total variation, $\rho_{\mathfrak{F}}(f) = \frac{1}{2} \text{span}(f)$. For the Kantorovich metric, $|\int_{\mathbf{X}} f d\mu - \int_{\mathbf{X}} f d\nu| \leq \|f\|_{\text{Lip}} d_{\mathfrak{F}}(\mu, \nu)$. Thus, for Kantorovich metric, $\rho_{\mathfrak{F}}(f) = \|f\|_{\text{Lip}}$. For the maximum mean discrepancy, $|\int_{\mathbf{X}} f d\mu - \int_{\mathbf{X}} f d\nu| \leq \|f\|_{\mathcal{H}} d_{\mathfrak{F}}(\mu, \nu)$. Thus, for maximum mean discrepancy, $\rho_{\mathfrak{F}}(f) = \|f\|_{\mathcal{H}}$.

2. Let \mathbf{X} and \mathbf{Y} be Banach spaces and let $\mathfrak{F}_{\mathbf{X}}$ and $\mathfrak{F}_{\mathbf{Y}}$ denote the function class for $d_{\mathfrak{F}}$ with domain \mathbf{X} and \mathbf{Y} , respectively. Then, for any $\ell: \mathbf{X} \rightarrow \mathbf{Y}$, any real-valued function $f \in \mathfrak{F}_{\mathbf{Y}}$ and any measures μ and ν on $\Delta(\mathbf{X})$, we have

$$\left| \int_{\mathbf{X}} f(\ell(x)) \mu(dx) - \int_{\mathbf{X}} f(\ell(x)) \nu(dx) \right| \leq \rho_{\mathfrak{F}_{\mathbf{X}}}(f \circ \ell) d_{\mathfrak{F}_{\mathbf{X}}}(\mu, \nu).$$

We define the contraction factor of the function ℓ as

$$\kappa_{\mathfrak{F}_{\mathbf{X}}, \mathfrak{F}_{\mathbf{Y}}}(\ell) = \sup_{f \in \mathfrak{F}_{\mathbf{Y}}} \rho_{\mathfrak{F}_{\mathbf{X}}}(f \circ \ell). \quad (2.11)$$

Therefore, we can say that for any $f \in \mathfrak{F}_{\mathbf{Y}}$,

$$\left| \int_{\mathbf{X}} f(\ell(x)) \mu(dx) - \int_{\mathbf{X}} f(\ell(x)) \nu(dx) \right| \leq \kappa_{\mathfrak{F}_{\mathbf{X}}, \mathfrak{F}_{\mathbf{Y}}}(\ell) d_{\mathfrak{F}_{\mathbf{X}}}(\mu, \nu). \quad (2.12)$$

For the total variation distance, $\frac{1}{2} \text{span}(f \circ \ell) \leq \|f \circ \ell\|_{\infty} \leq \|f\|_{\infty} \leq 1$. Thus, $\kappa_{\mathfrak{F}}(\ell) \leq 1$. For the Kantorovich metric, $\|f \circ \ell\|_{\text{Lip}} \leq \|f\|_{\text{Lip}} \|\ell\|_{\text{Lip}}$. Thus, $\kappa_{\mathfrak{F}}(\ell) \leq \|\ell\|_{\text{Lip}}$.

2.3.2 Approximate information state (AIS) and approximate dynamic programming

Now we define a notion of approximate information state (AIS) as a compression of the history of observations and actions which approximately satisfies properties (P1) and (P2).

Definition 3. Let $\{\hat{Z}_t\}_{t=1}^T$ be a pre-specified collection of Banach spaces, \mathfrak{F} be a function class for IPMs, and $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ be pre-specified positive real numbers. A collection $\{\hat{\sigma}_t: \mathbf{H}_t \rightarrow \hat{Z}_t\}_{t=1}^T$ of history compression functions, along with approximate update kernels $\{\hat{P}_t: \hat{Z}_t \times \mathbf{A} \rightarrow \Delta(\hat{Z}_{t+1})\}_{t=1}^T$ and reward approximation functions $\{\hat{r}_t: \hat{Z}_t \times \mathbf{A} \rightarrow \mathbb{R}\}_{t=1}^T$, is called an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ -AIS *generator* if the process $\{\hat{Z}_t\}_{t=1}^T$, where $\hat{Z}_t = \hat{\sigma}_t(H_t)$, satisfies the following properties:

(AP1) Sufficient for approximate performance evaluation, i.e., for any time t , any realization h_t of H_t and any choice a_t of A_t , we have

$$|\mathbb{E}[R_t \mid H_t = h_t, A_t = a_t] - \hat{r}_t(\hat{\sigma}_t(h_t), a_t)| \leq \varepsilon_t.$$

(AP2) Sufficient to predict itself approximately. i.e., for any time t , any realization h_t of H_t , any choice a_t of A_t , and for any Borel subset B of \hat{Z}_{t+1} , define $\mu_t(B) := \mathbb{P}(\hat{Z}_{t+1} \in B \mid H_t = h_t, A_t = a_t)$ and $\nu_t(B) := \hat{P}_t(B \mid \hat{\sigma}_t(h_t), a_t)$; then,

$$d_{\mathfrak{F}}(\mu_t, \nu_t) \leq \delta_t.$$

We use the phrase “ (ε, δ) -AIS” when ε_t and δ_t do not depend on time.

Similar to Proposition 7, we can provide an alternative characterization of an AIS where we replace (AP2) with the following approximations of (P2a) and (P2b).

(AP2a) Evolves in a state-like manner, i.e., there exist measurable update functions $\{\hat{\varphi}_t: \hat{Z}_t \times \mathbf{Y} \times \mathbf{A}\}_{t=1}^T$ such that for any realization h_{t+1} of H_{t+1} , we have

$$\hat{\sigma}_{t+1}(h_{t+1}) = \hat{\varphi}_t(\hat{\sigma}_t(h_t), y_t, a_t).$$

(AP2b) Is sufficient for predicting future observations approximately, i.e., there exist measurable observation prediction kernels $\{\hat{P}_t^y: \hat{Z}_t \times \mathbf{A} \rightarrow \Delta(\mathbf{Y})\}_{t=1}^T$ such that

for any time t , any realization h_t of H_t , any choice a_t of A_t , and for any Borel subset \mathbf{B} of \mathbf{Y} define, $\mu_t^y(\mathbf{B}) := \mathbb{P}(Y_t \in \mathbf{B} \mid H_t = h_t, A_t = a_t)$ and $\nu_t^y(\mathbf{B}) = \hat{P}_t^y(\mathbf{B} \mid \hat{\sigma}_t(h_t), a_t)$; then,

$$d_{\mathfrak{F}}(\mu_t^y, \nu_t^y) \leq \delta / \kappa_{\mathfrak{F}}(\hat{\varphi}_t),$$

where $\kappa_{\mathfrak{F}}(\hat{\varphi}_t)$ is defined as $\sup_{h_t \in \mathbf{H}_t, a_t \in \mathbf{A}_t} \kappa_{\mathfrak{F}}(\hat{\varphi}_t(\hat{\sigma}_t(h_t), \cdot, a_t))$. Note that for the total variation distance $\kappa_{\mathfrak{F}}(\hat{\varphi}_t) = 1$; for the Kantorovich distance $\kappa_{\mathfrak{F}}(\hat{\varphi}_t)$ is equal to the Lipschitz uniform bound on the Lipschitz constant of $\hat{\varphi}_t$ with respect to y_t .

Proposition 8. (AP2a) and (AP2b) imply (AP2) holds with transition kernels $\{\hat{P}_t^y\}_{t=1}^T$ defined as follows: for any Borel subset \mathbf{B} of $\hat{\mathbf{Z}}$,

$$\hat{P}_t(\mathbf{B} \mid \hat{\sigma}_t(h_t), a_t) = \int_{\mathbf{Y}} \mathbb{1}_{\mathbf{B}}(\hat{\varphi}_t(\hat{\sigma}_t(h_t), y_t, a_t)) \hat{P}_t^y(dy_t \mid \hat{\sigma}_t(h_t), a_t).$$

Therefore, we can alternatively define an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ -AIS generator as a tuple $\{(\hat{\sigma}_t, \hat{r}_t, \hat{\varphi}_t, \hat{P}_t^y)\}_{t=1}^T$ which satisfies (AP1), (AP2a), and (AP2b).

Proof. See Appendix A. □

Our main result is to establish that any AIS gives rise to an approximate dynamic program.

Theorem 2. Suppose $\{\hat{\sigma}_t, \hat{P}_t, \hat{r}_t\}_{t=1}^T$ is an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ -AIS generator. Recursively define approximate action-value functions $\{\hat{Q}_t: \hat{\mathbf{Z}}_t \times \mathbf{A} \rightarrow \mathbb{R}\}_{t=1}^T$ and value functions $\{\hat{V}_t: \hat{\mathbf{Z}}_t \rightarrow \mathbb{R}\}_{t=1}^T$ as follows: $\hat{V}_{T+1}(\hat{z}_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$:

$$\hat{Q}_t(\hat{z}_t, a_t) := \hat{r}_t(\hat{z}_t, a_t) + \int_{\hat{\mathbf{Z}}_t} \hat{V}_{t+1}(\hat{z}_{t+1}) \hat{P}_t(d\hat{z}_{t+1} \mid \hat{z}_t, a_t), \quad (2.13a)$$

$$\hat{V}_t(\hat{z}_t) := \max_{a_t \in \mathbf{A}} \hat{Q}_t(\hat{z}_t, a_t). \quad (2.13b)$$

Then, we have the following:

1. **Value function approximation:** For any time t , realization h_t of H_t , and choice a_t of A_t , we have

$$|Q_t(h_t, a_t) - \hat{Q}_t(\hat{\sigma}_t(h_t), a_t)| \leq \alpha_t \quad \text{and} \quad |V_t(h_t) - \hat{V}_t(\hat{\sigma}_t(h_t))| \leq \alpha_t, \quad (2.14)$$

where α_t satisfies the following recursion: $\alpha_{T+1} = 0$ and for $t \in \{T, \dots, 1\}$,

$$\alpha_t = \varepsilon_t + \rho_{\mathfrak{F}}(\hat{V}_{t+1})\delta_t + \alpha_{t+1}.$$

Therefore,

$$\alpha_t = \varepsilon_t + \sum_{\tau=t+1}^T [\rho_{\mathfrak{F}}(\hat{V}_{\tau})\delta_{\tau-1} + \varepsilon_{\tau}].$$

2. Approximately optimal policy: Let $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_T)$, where $\hat{\pi}_t: \hat{Z}_t \rightarrow \Delta(\mathbf{A})$, be a stochastic policy that satisfies

$$\text{Supp}(\hat{\pi}(\hat{z}_t)) \subseteq \arg \max_{a_t \in \mathbf{A}} \hat{Q}_t(\hat{z}_t, a_t). \quad (2.15)$$

Define policy $\pi = (\pi_1, \dots, \pi_T)$, where $\pi_t: \mathbf{H}_t \rightarrow \Delta(\mathbf{A})$ by $\pi_t := \hat{\pi}_t \circ \hat{\sigma}_t$. Then, for any time t , realization h_t of H_t , and choice a_t of A_t , we have

$$|Q_t(h_t, a_t) - Q_t^{\pi}(h_t, a_t)| \leq 2\alpha_t \quad \text{and} \quad |V_t(h_t) - V_t^{\pi}(h_t)| \leq 2\alpha_t. \quad (2.16)$$

Proof. See Appendix A. □

An immediate implication of Theorems 1 and 2 is the following.

Corollary 1. Let $\{\sigma_t\}_{t=1}^T$ be an information state generator and $\{(\hat{\sigma}_t, \hat{P}_t, \hat{r}_t)\}_{t=1}^T$ be an AIS generator. Then, for any time t , realization h_t of history H_t , and choice a_t of action A_t , we have

$$|\bar{Q}_t(\sigma_t(h_t), a_t) - \hat{Q}_t(\hat{\sigma}_t(h_t), a_t)| \leq \alpha_t \quad \text{and} \quad |\bar{V}_t(\sigma_t(h_t)) - \hat{V}_t(\hat{\sigma}_t(h_t))| \leq \alpha_t.$$

2.3.3 AIS with observation compression

In applications with high-dimensional observations such as video input, it is desirable to pre-process the video frames into a low-dimensional representation before passing them on to a planning or learning algorithm. In this section, we generalize the notion of AIS to account for such observation compression. Observation compression is used in the form of an autoencoder for our experiments involving high-dimensional environments.

Definition 4. As in the definition of AIS, suppose $\{\hat{Z}_t\}_{t=1}^T$ are a pre-specified collection of Banach spaces, \mathfrak{F} be a function class for IPMs, and $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ be pre-specified positive real numbers. In addition, suppose we have a set \hat{Y} of compressed observations and a compression function $q: Y \rightarrow \hat{Y}$. Let \hat{H}_t denote the history $(\hat{Y}_{1:t-1}, A_{1:t-1})$ of compressed observations and actions and \hat{H}_t denote the space of realizations of such compressed histories. Then, a collection $\{\hat{\sigma}_t: \hat{H}_t \rightarrow \hat{Z}_t\}_{t=1}^T$ of history compression functions, along with observation compression function $q: Y \rightarrow \hat{Y}$, approximate update kernels $\{\hat{P}_t: \hat{Z}_t \times A \rightarrow \Delta(\hat{Z}_{t+1})\}_{t=1}^T$ and reward approximation functions $\{\hat{r}_t: \hat{Z}_t \times A \rightarrow \mathbb{R}\}_{t=1}^T$, is called an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ -*observation-compressed AIS generator* if the process $\{\hat{Z}_t\}_{t=1}^T$, where $\hat{Z}_t = \hat{\sigma}_t(\hat{H}_t)$, satisfies properties (AP1) and (AP2).

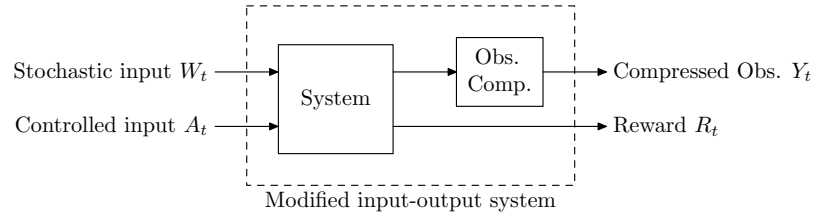


Fig. 2.3 A stochastic input-output system with observation compression

In essence, we can view observation compression as a new input-output system whose outputs are (\hat{Y}_t, R_t) instead of (Y_t, R_t) as shown in Fig. 2.3. A construction similar to observation-compressed AIS is proposed in [6], where it is shown that such a construction performs well empirically, but there was no analysis of the approximation guarantees of such a construction.

An immediate implication of the above definition is the following:

Corollary 2. Let $\{\hat{\sigma}_t, q, \hat{P}_t, \hat{r}_t\}_{t \geq 1}$ be an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ -observation-compression AIS. Then, the bounds of Theorem 2 hold.

2.4 Infinite-horizon discounted reward setup

So far, we have restricted attention to the finite horizon setup. In this section, we show how to generalize the notions of information state and approximate information state to the infinite horizon discounted reward setup.

2.4.1 System model and problem formulation

We consider the same model as described in Sec. 2.1 but assume that the system runs for an infinite horizon. The performance of any (history dependent and possibly stochastic) policy $\pi := (\pi_1, \pi_2, \dots)$, where $\pi_t: \mathbf{H}_t \rightarrow \Delta(\mathbf{A})$, is given by

$$J(\pi) := \liminf_{T \rightarrow \infty} \mathbb{E}^\pi \left[\sum_{t=1}^T \gamma^{t-1} R_t \right],$$

where $\gamma \in (0, 1)$ is the discount factor. As before, we assume that the agent knows the system dynamics $\{f_t\}_{t \geq 1}$, the reward functions $\{r_t\}_{t \geq 1}$, and the probability measure \mathbb{P} on the primitive random variables $\{W_t\}_{t \geq 1}$. The objective of the agent is to choose a policy π that maximizes the expected discounted total reward $J(\pi)$.

Note that we use \liminf rather than \lim in the above definition because in general the limit might not exist. We later assume that the rewards are uniformly bounded (see Assumption 1) which, together with the finiteness of the action space, implies that the limit is well defined. When the action space is uncountable, we need to impose appropriate technical conditions on the model to ensure that an appropriate measurable section condition holds [5].

2.4.2 A dynamic programming decomposition

In the finite-horizon setup, we started with a dynamic program to evaluate the performance $\{V_t^\pi\}_{t=1}^T$ for any history dependent policy π . We then identified an upper-bound $\{V_t\}_{t=1}^T$ on $\{V_t^\pi\}_{t=1}^T$ and showed that this upper bound is tight and achieved by any optimal policy. The subsequent analysis of the information state and the approximate information state based dynamic programs was based on comparison with $\{V_t\}_{t=1}^T$.

One conceptual difficulty with the infinite horizon setup is that we cannot write a general dynamic program to evaluate the performance $\{V_t^\pi\}_{t \geq 1}$ of an arbitrary history dependent policy π and therefore identify a tight upper-bound $\{V_t\}_{t \geq 1}$. In traditional MDP models, this conceptual difficulty is resolved by restricting attention to Markov strategies and then establishing that the performance of a Markov strategy can be evaluated by solving a fixed point equation. For partially observed MDPs, a similar resolution works because one can view the belief state as an information state. However, for general partially observed models

as considered in this paper, there is no general methodology to identify a time-homogeneous information state. So, we follow a different approach and identify a dynamic program which bounds the performance of a general history dependent policy. We impose the following mild assumption on the model.

Assumption 1. The reward process $\{R_t\}_{t \geq 1}$ is uniformly bounded and takes values inside a finite interval $[R_{\min}, R_{\max}]$.

Given any (history dependent) policy π , we define the *reward-to-go* function for any time t and any realization h_t of H_t as

$$V_t^\pi(h_t) := \mathbb{E}^\pi \left[\sum_{s=t}^{\infty} \gamma^{s-t} R_s \mid H_t = h_t \right]. \quad (2.17)$$

Define the corresponding action value function as:

$$Q_t^\pi(h_t, a_t) := \mathbb{E}^\pi [R_t + \gamma V_{t+1}^\pi(H_{t+1}) \mid H_t = h_t, A_t = a_t]. \quad (2.18)$$

As stated above, we cannot identify a dynamic program to recursively compute $\{V_t^\pi\}_{t \geq 1}$. Nonetheless, we show that under Assumption 1 we can identify arbitrarily precise upper and lower bounds for $\{V_t^\pi\}_{t \geq 1}$ which can be recursively computed.

Proposition 9. Arbitrarily pick a horizon T and define $\{J_{t,T}^\pi: \mathbf{H}_t \rightarrow \mathbb{R}\}_{t=1}^T$ as follows: $J_{T,T}^\pi(h_T) = 0$ and for $t \in \{T-2, \dots, 1\}$,

$$J_{t,T}^\pi(h_t) := \mathbb{E}^\pi [R_t + \gamma J_{t+1,T}^\pi(H_{t+1}) \mid H_t = h_t]. \quad (2.19)$$

Then, for any time $t \in \{1, \dots, T\}$ and realization h_t of H_t , we have

$$J_{t,T}^\pi(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\min} \leq V_t^\pi(h_t) \leq J_{t,T}^\pi(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}. \quad (2.20)$$

Proof. See Appendix A. □

Note that Proposition 9 gives a recursive method to approximately evaluate the performance of any history dependent policy π . We can modify the recursion in (2.19) to obtain policy independent upper bound on performance of an arbitrary policy. For that matter,

define value functions $\{V_t: \mathbf{H}_t \rightarrow \mathbb{R}\}_{t \geq 1}$ as follows:

$$V_t(h_t) = \sup_{\pi} V_t^{\pi}(h_t), \quad (2.21)$$

where the supremum is over all history dependent policies. Furthermore, define action-value functions $\{Q_t: \mathbf{H}_t \times \mathbf{A} \rightarrow \mathbb{R}\}_{t \geq 1}$ as follows:

$$Q_t(h_t, a_t) = \mathbb{E}[R_t + \gamma V_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t]. \quad (2.22)$$

Then, we have the following.

Proposition 10. Arbitrarily pick a horizon T and define $\{J_{t,T}: \mathbf{H}_t \rightarrow \mathbb{R}\}$ as follows: $J_{T,T}(h_T) = 0$ and for $t \in \{T-2, \dots, 1\}$,

$$J_{t,T}(h_t) := \max_{a_t \in \mathbf{A}} \mathbb{E}[R_t + \gamma J_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t]. \quad (2.23)$$

Then, for any time $t \in \{1, \dots, T\}$ and realization h_t of H_t ,

$$V_t^{\pi}(h_t) \leq J_{t,T}(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}. \quad (2.24)$$

Therefore,

$$J_{t,T}(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\min} \leq V_t(h_t) \leq J_{t,T}(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}. \quad (2.25)$$

Note that $J_{t,T}(h_t)$ is the optimal value function for a finite horizon system with the discounted reward criterion that runs for horizon $T-1$.

Proof. See Appendix A. □

2.4.3 Time-homogeneous information state and simplified dynamic program

Definition 5. Given a Banach space Z , an information state generator $\{\sigma_t: \mathbf{H}_t \rightarrow Z\}$ is said to be *time-homogeneous* if, in addition to (P1) and (P2), it satisfies the following:

- (S) The expectation $\mathbb{E}[R_t \mid Z_t = \sigma_t(H_t), A_t = a_t]$ and the transition kernel $\mathbb{P}(Z_{t+1} \in B \mid Z_t = \sigma_t(H_t), A_t = a_t)$ are time-homogeneous.

In general, a time-homogeneous information state may not exist for all partially observed models and it is important to understand conditions under which such an information state exists. However, we do not pursue that direction in this work.

For any time-homogeneous information state, define the Bellman operator $\mathcal{B}: [\mathcal{Z} \rightarrow \mathbb{R}] \rightarrow [\mathcal{Z} \rightarrow \mathbb{R}]$ as follows: for any uniformly bounded function $\bar{V}: \mathcal{Z} \rightarrow \mathbb{R}$

$$[\mathcal{B}\bar{V}](z) = \max_{a \in \mathcal{A}} \mathbb{E}[R_t + \gamma \bar{V}(Z_{t+1}) \mid Z_t = z, A_t = a], \quad (2.26)$$

where $\gamma \in (0, 1)$ is the discount factor. Because of (S), the expectation on the right hand side does not depend on time. Due to discounting, the operator \mathcal{B} is a contraction and therefore, under Assumption 1, the fixed point equation

$$\bar{V} = \mathcal{B}\bar{V} \quad (2.27)$$

has a unique bounded solution (due to the Banach fixed point theorem). Let \bar{V}^* be the fixed point and π^* be any policy such that $\pi^*(z)$ achieves the arg max in the right hand side of (2.26) for $[\mathcal{B}\bar{V}^*](z)$. It is easy to see that \bar{V}^* is the performance of the time homogeneous policy (π^*, π^*, \dots) . However, it is not obvious that \bar{V}^* equals to the optimal performance V_1 defined in (2.21), because the proof of Theorem 1 relies on backward induction and is not applicable to infinite horizon models. So, we present an alternative proof below which uses the performance bounds of Proposition 10.

Theorem 3. *Let $\{Z_t\}_{t \geq 1}$ be a time-homogeneous information state process with generator $\{\sigma_t: H_t \rightarrow \mathcal{Z}\}_{t \geq 1}$. Suppose Assumption 1 holds and let \bar{V}^* be the unique bounded fixed point of (2.26). Then, for any time t and realization h_t of H_t , we have*

$$V_t(h_t) = \bar{V}^*(\sigma_t(h_t)).$$

Furthermore, let $\pi^: \mathcal{Z} \rightarrow \Delta(\mathcal{A})$ be a time-homogeneous (stochastic) policy such that $\text{Supp}(\pi^*(z))$ is a subset of the arg max of the right hand side of (2.26). Then, the time-homogeneous policy $\pi^* := (\pi^*, \pi^*, \dots)$ is optimal.*

Proof. See Appendix A. □

2.4.4 Time-homogeneous AIS and approximate dynamic programming

Definition 6. Given a Banach space $\hat{\mathbf{Z}}$, a function class \mathfrak{F} for IPMs, and positive real numbers (ε, δ) , we say that a collection $\{\hat{\sigma}_t: \mathbf{H}_t \rightarrow \hat{\mathbf{Z}}\}_{t \geq 1}$ along with a time-homogeneous update kernel $\hat{P}: \hat{\mathbf{Z}} \times \mathbf{A} \rightarrow \Delta(\hat{\mathbf{Z}})$ and a time-homogeneous reward approximation function $\hat{r}: \hat{\mathbf{Z}} \times \mathbf{A} \rightarrow \mathbb{R}$ is a (ε, δ) *time homogeneous AIS generator* if the process $\{\hat{Z}_t\}_{t \geq 1}$, where $\hat{Z}_t = \hat{\sigma}_t(H_t)$, satisfies (AP1) and (AP2) where \hat{r}_t , \hat{P}_t , ε_t and δ_t in the definition of (AP1) and (AP2) are replaced by their time-homogeneous counterparts.

For any time-homogeneous AIS, define the approximate Bellman operator $\hat{\mathcal{B}}: [\hat{\mathbf{Z}} \rightarrow \mathbb{R}] \rightarrow [\hat{\mathbf{Z}} \rightarrow \mathbb{R}]$ as follows: for any uniformly bounded function $\hat{V}: \hat{\mathbf{Z}} \rightarrow \mathbb{R}$,

$$[\hat{\mathcal{B}}\hat{V}](\hat{z}) = \max_{a \in \mathbf{A}} \left\{ \hat{r}(\hat{z}, a) + \gamma \int_{\hat{\mathbf{Z}}} \hat{V}(\hat{z}') \hat{P}(d\hat{z}' | \hat{z}, a) \right\}. \quad (2.28)$$

Note that the expectation on the right hand side does not depend on time. Due to discounting, the operator $\hat{\mathcal{B}}$ is a contraction, and therefore, under Assumption 1, the fixed point equation

$$\hat{V} = \hat{\mathcal{B}}\hat{V} \quad (2.29)$$

has a unique bounded solution (due to the Banach fixed point theorem). Let \hat{V}^* be the fixed point and $\hat{\pi}^*$ be any policy such that $\hat{\pi}^*(\hat{z})$ achieves the arg max in the right hand side of (2.28) for $[\hat{\mathcal{B}}\hat{V}^*](\hat{z})$. It is not immediately clear if \hat{V}^* is close to the performance of policy $\pi = (\pi_1, \pi_2, \dots)$, where $\pi_t = \pi^* \circ \hat{\sigma}_t$, or if \hat{V}^* is close to the optimal performance. The proof of Theorem 2 relies on backward induction and is not immediately applicable to the infinite horizon setup. Nonetheless, we establish results similar to Theorem 2 by following the proof idea of Theorem 3.

Theorem 4. Suppose $(\{\hat{\sigma}_t\}_{t \geq 1}, \hat{P}, \hat{r})$ is a time-homogeneous (ε, δ) -AIS generator. Consider the fixed point equation (2.29), which we rewrite as follows:

$$\hat{Q}(\hat{z}, a) := \hat{r}(\hat{z}, a) + \int_{\hat{\mathbf{Z}}} \hat{V}_{t+1}(\hat{z}_{t+1}) \hat{P}(d\hat{z}_{t+1} | \hat{z}, a), \quad (2.30a)$$

$$\hat{V}(\hat{z}) := \max_{a \in \mathbf{A}} \hat{Q}(\hat{z}, a). \quad (2.30b)$$

Let \hat{V}^* denote the fixed point of (2.30) and \hat{Q}^* denote the corresponding action-value function. Then, we have the following:

1. **Value function approximation:** For any time t , realization h_t of H_t , and choice a_t of A_t , we have

$$|Q_t(h_t, a_t) - \hat{Q}^*(\hat{\sigma}_t(h_t), a_t)| \leq \alpha \quad \text{and} \quad |V_t(h_t) - \hat{V}^*(\hat{\sigma}_t(h_t))| \leq \alpha, \quad (2.31)$$

where

$$\alpha = \frac{\varepsilon + \gamma \rho_{\mathfrak{F}}(\hat{V}^*)\delta}{1 - \gamma}$$

2. **Approximately optimal policy:** Let $\hat{\pi}^*: \hat{Z} \rightarrow \Delta(\mathbf{A})$ be a stochastic policy that satisfies

$$\text{Supp}(\hat{\pi}^*(\hat{z})) \subseteq \arg \max_{a \in \mathbf{A}} \hat{Q}^*(\hat{z}, a). \quad (2.32)$$

Define policy $\pi = (\pi_1, \pi_2, \dots)$, where $\pi_t: \mathbf{H}_t \rightarrow \Delta(\mathbf{A})$ is defined by $\pi_t := \hat{\pi}^* \circ \hat{\sigma}_t$. Then, for any time t , realization h_t of H_t , and choice a_t of A_t , we have

$$|Q_t(h_t, a_t) - Q_t^\pi(h_t, a_t)| \leq 2\alpha \quad \text{and} \quad |V_t(h_t) - V_t^\pi(h_t)| \leq 2\alpha. \quad (2.33)$$

Proof. See Appendix A. □

Chapter 3

Reinforcement learning for partially observed systems using AIS

Previous work in RL on AIS [12] proposed using a two-time scale algorithm for learning the policy alongside the AIS using just the KL-divergence as a surrogate for the Wasserstein distance. In this work, we present a policy gradient based reinforcement learning (RL) algorithm for infinite horizon partially observed systems for a general IPM. Particularly, we derive AIS loss equations for the KL IPM in Eq. (3.7) for (AP2) and Eq. (3.9) for (AP2a), (AP2b) and for the MMD IPM in Eq. (3.8) for (AP2) and Eq. (3.10) for (AP2a), (AP2b). The derivations can be extended to other IPMs in a straightforward manner. The algorithm learns a time-homogeneous AIS generator $(\hat{\sigma}_t, \hat{r}, \hat{P})$ which satisfies (AP1) and (AP2) or a time-homogeneous AIS generator $(\hat{\sigma}_t, \hat{r}, \hat{\varphi}, \hat{P}^y)$ which satisfies (AP1), (AP2a), and (AP2b). In our approach, the key idea is to represent each component of the AIS generator using a parametric family of functions/distributions and use a multi-time scale stochastic gradient descent algorithm [64] which learns the AIS generator at a faster time-scale than the policy and/or the action-value function.

For the ease of exposition, we first assume that the policy is fixed and describe how to learn the AIS generator using stochastic gradient descent. To specify an AIS, we must pick an IPM \mathfrak{F} as well. Although, in principle, we can choose any IPM, in practice, we want to choose an IPM such that the distance $d_{\mathfrak{F}}(\mu_t, \nu_t)$ in (AP2) or (AP2b) can be computed efficiently. We discuss the choice of IPMs in Sec. 3.1 and then discuss the stochastic gradient descent algorithm to learn the AIS-generator for a fixed policy in Sec. 3.2. Then we

describe how to simultaneously learn the AIS generator and the policy using a multi-time scale algorithm, first for an actor only framework and then for an actor-critic framework in Sec. 3.3 under the setting of partially observed reinforcement learning (PORL).

3.1 The choice of an IPM

As we will explain in the next section in detail, our general *modus operandi* is to assume that the stochastic kernel \hat{P} or \hat{P}^y that we are trying to learn belongs to a parametric family and then update the parameters of the distribution to either minimize $d_{\mathfrak{F}}(\mu, \nu)$ defined in (AP2) or minimize $d_{\mathfrak{F}}(\mu^y, \nu^y)$ defined in (AP2b). Just to keep the discussion concrete, we focus on (AP2). Similar arguments apply to (AP2b) as well. First note that for a particular choice of parameters, we know the distribution ν in closed form, but we do not know the distribution μ in closed form and only have samples from that distribution. One way to estimate the IPM between a distribution and samples from another distribution is to use duality and minimize $|\int_{\mathbb{Z}} f d\mu - \int_{\mathbb{Z}} f d\nu|$ over the choice of function f such that $f \in \mathfrak{F}$. When $d_{\mathfrak{F}}$ is equal to the total variation distance or the Wasserstein distance, this optimization problem may be solved using a linear program [58]. However, solving a linear program at each step of the stochastic gradient descent algorithm can become a computational bottleneck. We propose two alternatives here. The first is to use the total variation distance or the Wasserstein distance but instead of directly working with them, we use a KL divergence based upper bound as a surrogate loss. The other alternative is to work with RKHS-based MMD (maximum mean discrepancy) distance, which can be computed from samples without solving an optimization problem [58]. It turns out that for the AIS-setup, a specific form of MMD known as distance-based MMD is particularly convenient as we explain below.

KL-divergence based upper bound for total variation or Wasserstein distance.

The surrogate loss considered here is very similar to what was derived in [12]. Recall that the KL-divergence between two densities μ and ν on $\Delta(\mathbf{X})$ is defined as

$$D_{\text{KL}}(\mu \parallel \nu) = \int_{\mathbf{X}} \log \mu(x) \mu(dx) - \int_{\mathbf{X}} \log \nu(x) \mu(dx).$$

The total variation distance can be upper bounded by the KL-divergence using Pinsker's inequality [65] (see footnote 1 for the difference in constant factor from the standard Pinsker's inequality):

$$d_{\text{TV}}(\mu, \nu) \leq \sqrt{2D_{\text{KL}}(\mu \parallel \nu)}. \quad (3.1)$$

As we will explain in the next section, we consider the setup where we know the distribution ν but only obtain samples from the distribution μ . Since there are two losses—the reward prediction loss ε and the AIS/observation prediction loss δ , we work with minimizing the weighted square average $\lambda\varepsilon^2 + (1 - \lambda)\delta^2$, where $\lambda \in [0, 1]$ is a hyper-parameter. Pinsker's inequality (3.1) suggests that instead of $d_{\text{TV}}(\mu, \nu)^2$, we can use the surrogate loss function

$$\int_{\mathbf{X}} \log \nu(x) \mu(dx)$$

where we have dropped the term that does not depend on ν . Note that the above expression is the same as the cross-entropy between μ and ν which can be efficiently computed from samples. In particular, if we get T i.i.d. samples X_1, \dots, X_T from μ , then

$$\frac{1}{T} \sum_{t=1}^T \log \nu(X_t) \quad (3.2)$$

is an unbiased estimator of $\int_{\mathbf{X}} \log \nu(x) \mu(dx)$.

Finally, if \mathbf{X} is a bounded space with diameter D , then

$$d_{\text{Wass}}(\mu, \nu) \leq D d_{\text{TV}}(\mu, \nu).$$

So, using cross-entropy as a surrogate loss also works for Wasserstein distance.

Distance-based MMD. The key idea behind using a distance-based MMD is the following result.

Proposition 11 (Theorem 22 of [63]). Let $\mathbf{X} \subseteq \mathbb{R}^m$ and $d_{\mathbf{X},p}: \mathbf{X} \times \mathbf{X} \rightarrow \mathbb{R}_{\geq 0}$ be a metric given by $d_{\mathbf{X},p}(x, x') = \|x - x'\|_2^p$, for $p \in (0, 2]$. Let $k_p: \mathbf{X} \times \mathbf{X} \rightarrow \mathbb{R}$ be any kernel given

$$k_p(x, x') = \frac{1}{2} [d_{\mathbf{X},p}(x, x_0) + d_{\mathbf{X},p}(x', x_0) - d_{\mathbf{X},p}(x, x')],$$

where $x_0 \in \mathbf{X}$ is arbitrary, and let \mathcal{H}_p be a RKHS with kernel k_p and $\mathfrak{F}_p = \{f \in \mathcal{H}_p :$

$\|f\|_{\mathcal{H}_p} \leq 1\}$. Then, for any distributions $\mu, \nu \in \Delta(X)$, the IPM $d_{\mathfrak{F}_p}(\mu, \nu)$ can be expressed as follows:

$$d_{\mathfrak{F}_p}(\mu, \nu) = \sqrt{\mathbb{E}[d_{X,p}(X, W)] - \frac{1}{2}\mathbb{E}[d_{X,p}(X, X')] - \frac{1}{2}\mathbb{E}[d_{X,p}(W, W')]}, \quad (3.3)$$

where $X, X' \sim \mu$, $W, W' \sim \nu$ and (X, X', W, W') are all independent.

We call d_p defined above as a *distance-based* MMD. For $p = 1$ (for which d_X corresponds to the L_2 distance), the expression inside the square root in (3.3) is called the Energy distance in the statistics literature [62]. In [63], the above result is stated for a general semimetric of a negative type. Our statement of the above result is specialized to the semimetric $d_{X,p}$. See Proposition 3 and Example 15 of [63] for details.

As explained in the previous section, we work with minimizing the weighted square average $\lambda\varepsilon^2 + (1 - \lambda)\delta^2$, where λ is a hyper-parameter. Proposition 11 suggests that instead of $d_{\mathfrak{F}_p}(\mu, \nu)^2$, we can use a surrogate loss function

$$\int_X \int_X \|x - w\|_2^p \mu(dx) \nu(dw) - \frac{1}{2} \int_X \int_X \|w - w'\|_2^p \nu(dw) \nu(dw') \quad (3.4)$$

for $p \in (0, 2]$, where we have dropped the term that does not depend on ν . It is possible to compute the surrogate loss efficiently from samples as described in [58]. In particular, if we get T i.i.d. samples X_1, \dots, X_T from μ , then

$$\frac{1}{T} \sum_{t=1}^T \int_X \|X_t - w\|_2^p \nu(dw) - \frac{1}{2} \int_X \int_X \|w - w'\|_2^p \nu(dw) \nu(dw') \quad (3.5)$$

is an unbiased estimator of (3.4).

In our numerical experiments, we use the surrogate loss (3.5) for $p = 2$, which simplifies as follows.

Proposition 12. Consider the setup of Proposition 11 for $p = 2$. Suppose ν_ξ is a known parameterized distribution with mean M_ξ and X is a sample from μ . Then, the gradient of

$$(M_\xi - 2X)^\top M_\xi \quad (3.6)$$

with respect to ξ is an unbiased estimator of $\nabla_\xi d_{\mathfrak{F}_2}(\mu, \nu_\xi)^2$.

Proof. See Appendix A. □

The implication of Proposition 12 is if we use MMD with the RKHS \mathcal{H}_2 defined in Proposition 11, then we can use the expression in (3.6) as a surrogate loss function for $d_{\mathfrak{F}_2}(\mu, \nu_\xi)^2$.

Now we show how to compute the surrogate loss (3.6) for two types of parameterized distributions ν_ξ .

1. **SURROGATE LOSS FOR PREDICTING DISCRETE VARIABLES:** When predicting a discrete-valued random variable, say a discrete-valued AIS \hat{Z}_{t+1} in (AP2) or a discrete-valued observation Y_t in (AP2b), we view the discrete random variable as a continuous-valued random vector by representing it as a one-hot encoded vector. In particular, if the discrete random variable, which we denote by V , takes m values, then its one-hot encoded representation, which we denote by X , takes values in the corner points of the simplex on \mathbb{R}^m . Now, suppose ν_ξ is any parameterized distribution on the discrete set $\{1, \dots, m\}$ (e.g., the softmax distribution). Then, in the one-hot encoded representation, the mean M_ξ is given by

$$M_\xi = \sum_{i=1}^m \nu_\xi(i) e_i = \begin{bmatrix} \nu_\xi(1) \\ \vdots \\ \nu_\xi(m) \end{bmatrix},$$

where e_i denotes the m -dimensional unit vector with 1 in the i -th location. Thus, when we one-hot encode discrete AIS or discrete observations, the “mean” M_ξ is same as the probability mass function (PMF) ν_ξ . Thus, effectively, $d_{\mathfrak{F}_2}(\mu, \nu)^2$ is equivalent to $\|\mu - \nu\|_2^2$ and (3.6) is an unbiased estimator where we have removed the terms that do not depend on ν .

2. **SURROGATE LOSS FOR PREDICTING CONTINUOUS VARIABLES:** When predicting a continuous-valued random variable, say a continuous-valued AIS \hat{Z}_{t+1} in (AP2) or a continuous-valued observation Y_t in (AP2b), we can immediately use the surrogate loss (3.6) as long as the parameterized distribution ν_ξ is such that its mean M_ξ is given in closed form. Note that the surrogate loss (3.6) only depends on the mean of the distribution and not one any other moment. So, any two distributions ν and ν' that have the same mean, the surrogate loss between any distribution μ and ν is

same as the surrogate loss between μ and ν' . Thus, using the surrogate loss (3.6) for predicting continuous variables only makes sense when we expect the true distribution to be close to a deterministic function.

3.2 Learning an AIS for a fixed policy

The definition of AIS suggests that there are two ways to construct an information state from data: we either learn a time-homogeneous AIS-generator $(\hat{\sigma}, \hat{r}, \hat{P})$ that satisfies (AP1) and (AP2) or we learn a time-homogeneous AIS-generator $(\hat{\sigma}, \hat{r}, \hat{\phi}, \hat{P}^y)$ that satisfies (AP1), (AP2a), and (AP2b). In either case, there are three types of components of AIS-generators: (i) regular functions such as \hat{r} and $\hat{\phi}$; (ii) history compression functions $\{\hat{\sigma}_t\}_{t \geq 1}$; and (iii) stochastic kernels \hat{P} and \hat{P}^y . To learn these components from data, we must choose parametric class of functions for all of these. In this section, we do not make any assumption about how these components are chosen. In particular, \hat{r} and $\hat{\phi}$ could be represented by any class of function approximators (such as a multi-layer perceptron); $\hat{\sigma}$ could be represented by any class of time-series approximators (such as a RNN or its refinements such as LSTM or GRU); and \hat{P} and \hat{P}^y could be represented by any class of stochastic kernel approximators (such as softmax distribution or mixture of Gaussians). We use ξ_t to denote the corresponding parameters.

There are two losses in the definition of an AIS: the reward loss $|R_t - \hat{r}(\hat{z}_t, a_t)|$ and the prediction loss $d_{\mathfrak{F}}(\mu_t, \nu_t)$ or $d_{\mathfrak{F}}(\mu_t^y, \nu_t^y)$. We combine these into a single criterion and minimize the combined loss function

$$\frac{1}{T} \sum_{t=1}^T \left[\lambda |R_t - \hat{r}(\hat{z}_t, a_t)|^2 + (1 - \lambda) d_{\mathfrak{F}}(\mu_t, \nu_t)^2 \right]$$

where T is the length of the episode or the rollout horizon and $\lambda \in [0, 1]$ may be viewed as a hyper-parameter.

As described in Section 3.1, there are two possibilities to efficiently compute $d_{\mathfrak{F}}(\mu_t, \nu_t)^2$: use total-variation distance or Wasserstein distance as the IPM and use surrogate loss (3.2); or use distance-based MMD as the IPM and use the surrogate loss (3.6).

In particular, to choose an AIS that satisfies (AP1) and (AP2), we either minimize the

surrogate loss

$$\frac{1}{T} \sum_{t=1}^T [\lambda |R_t - \hat{r}(\hat{z}_t, a_t)|^2 + (1 - \lambda) \log(\nu_t(\hat{z}_{t+1}))] \quad (3.7)$$

or we minimize the surrogate loss (specialized for $p = 2$)

$$\frac{1}{T} \sum_{t=1}^T [\lambda |R_t - \hat{r}(\hat{z}_t, a_t)|^2 + (1 - \lambda)(M_t - 2\hat{z}_{t+1})^\top M_t] \quad (3.8)$$

where M_t is the mean of the distribution ν_t .

Similarly, in order to choose an AIS that satisfies (AP1), (AP2a) and (AP2b), we minimize the surrogate loss

$$\frac{1}{T} \sum_{t=1}^T [\lambda |R_t - \hat{r}(\hat{z}_t, a_t)|^2 + (1 - \lambda) \log(\nu_t^y(y_t))] \quad (3.9)$$

or we minimize the surrogate loss (specialized for $p = 2$)

$$\frac{1}{T} \sum_{t=1}^T [\lambda |R_t - \hat{r}(\hat{z}_t, a_t)|^2 + (1 - \lambda)(M_t^y - 2y_t)^\top M_t^y] \quad (3.10)$$

where M_t^y is the mean of the distribution ν_t^y .

We use $\bar{\xi}$ to denote the parameters of the AIS-generator, i.e., the parameters of $(\hat{\sigma}, \hat{P}, \hat{r})$ when using (AP1) and (AP2) or the parameters of $(\hat{\sigma}, \hat{\varphi}, \hat{P}^y, \hat{r})$ when using (AP1), (AP2a), (AP2b). We then use $\mathcal{L}(\bar{\xi})$ to denote the corresponding loss (3.7), (3.8), (3.9), or (3.10). Then, we can learn the parameters $\bar{\xi}$ using stochastic gradient descent:

$$\bar{\xi}_{k+1} = \bar{\xi}_k - a_k \nabla_{\bar{\xi}} \mathcal{L}(\bar{\xi}_k), \quad (3.11)$$

where the learning rates $\{a_k\}_{k \geq 0}$ satisfy the standard conditions $\sum a_k = \infty$ and $\sum a_k^2 < \infty$.

3.3 AIS-based partially observed reinforcement learning (PORL)

Given the stochastic gradient descent algorithm to learn an AIS-generator for a fixed policy, we can simultaneously learn a policy and AIS-generator by following a multi-time scale stochastic gradient descent [64], where we learn the AIS-generator at a faster learning rate

than the policy.

In particular, let $\pi_\theta: \hat{Z} \rightarrow \Delta(\mathbf{A})$ be a parameterized stochastic policy with parameters θ . Let $J(\bar{\xi}, \theta)$ denote the performance of policy π_θ . From the policy gradient theorem [2, 50], we know that

$$\nabla_\theta J(\bar{\xi}, \theta) = \mathbb{E} \left[\sum_{t=1}^{\infty} \left(\sum_{\tau=1}^t \nabla_\theta \log \pi_\theta(A_t \mid \hat{Z}_t) \right) \gamma^{t-1} R_t \right] \quad (3.12)$$

which can be estimated from a sampled trajectory with a rollout horizon of T using the G(PO)MDP gradient [50]

$$\hat{\nabla}_\theta J(\bar{\xi}, \theta) = \sum_{t=1}^T \left(\sum_{\tau=1}^t \nabla_\theta \log \pi_\theta(a_t \mid \hat{z}_t) \right) \gamma^{t-1} R_t. \quad (3.13)$$

We can iteratively update the parameters $\{(\bar{\xi}_k, \theta_k)\}_{k \geq 1}$ of both the AIS-generator and policy as follows. We start with an initial choice $(\bar{\xi}_1, \theta_1)$, update both parameters after a rollout of T as follows

$$\bar{\xi}_{k+1} = \bar{\xi}_k - a_k \nabla_{\bar{\xi}} \mathcal{L}(\bar{\xi}_k) \quad \text{and} \quad \theta_{k+1} = \theta_k + b_k \hat{\nabla}_\theta J(\bar{\xi}_k, \theta_k) \quad (3.14)$$

where the learning rates $\{a_k\}_{k \geq 1}$ and $\{b_k\}_{k \geq 1}$ satisfy the standard conditions on multi-time scale learning: $\sum_k a_k = \infty$, $\sum_k b_k = \infty$, $\sum_k a_k^2 < \infty$, $\sum_k b_k^2 < \infty$, and $\lim_{k \rightarrow \infty} b_k/a_k = 0$, which ensures that AIS-generator learns at a faster rate than the policy.

A similar idea can be used for an actor-critic algorithm. Suppose we have a parameterized policy $\pi_\theta: \hat{Z} \rightarrow \Delta(\mathbf{A})$ and a parameterized critic $\hat{Q}_\zeta: \hat{Z} \times \mathbf{A} \rightarrow \mathbb{R}$, where θ denotes the parameters of the policy and ζ denotes the parameters of the critic. Let $J(\bar{\xi}, \theta, \zeta)$ denote the performance of the policy. From the policy gradient theorem [2, 66], we know that

$$\nabla_\theta J(\bar{\xi}, \theta, \zeta) = \frac{1}{1-\gamma} \mathbb{E} [\nabla_\theta \log \pi_\theta(A_t \mid \hat{Z}_t) Q_\zeta(\hat{Z}_t, A_t)] \quad (3.15)$$

which can be estimated from a sampled trajectory with a rollout horizon of T by

$$\hat{\nabla}_\theta J(\bar{\xi}, \theta, \zeta) = \frac{1}{(1-\gamma)T} \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t \mid \hat{z}_t) \hat{Q}_\zeta(\hat{z}_t, a_t). \quad (3.16)$$

For the critic, we use the temporal difference loss

$$\mathcal{L}_{\text{TD}}(\bar{\xi}, \theta, \zeta) = \frac{1}{T} \sum_{t=1}^T \text{smoothL1}(\hat{Q}_{\zeta}(\hat{z}_t, a_t) - R_t - \gamma \hat{Q}_{\zeta}(\hat{z}_{t+1}, a_{t+1})) \quad (3.17)$$

where the `smoothL1` loss (a specific form of the Huber loss) is given by the smooth L_1 distance:

$$\text{smoothL1}(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| < 1 \\ |x| - \frac{1}{2} & \text{otherwise.} \end{cases}$$

We can iteratively update the parameters $\{(\bar{\xi}_k, \theta_k, \zeta_k)\}_{k \geq 1}$ of the AIS-generator, policy, and critic as follows. We start with an initial choice $(\bar{\xi}_1, \theta_1, \zeta_1)$, and update all the parameters after a rollout of T as follows

$$\bar{\xi}_{k+1} = \bar{\xi}_k - a_k \nabla_{\bar{\xi}} \mathcal{L}(\bar{\xi}_k), \quad \theta_{k+1} = \theta_k + b_k \widehat{\nabla}_{\theta} J(\bar{\xi}_k, \theta_k, \zeta_k) \quad \text{and} \quad \zeta_{k+1} = \zeta_k - c_k \nabla_{\zeta} \mathcal{L}_{\text{TD}}(\bar{\xi}_k, \theta_k, \zeta_k) \quad (3.18)$$

where the learning rates $\{a_k\}_{k \geq 1}$, $\{b_k\}_{k \geq 1}$, $\{c_k\}_{k \geq 1}$ satisfy the standard conditions on multi-time scale learning: $\sum_k a_k = \infty$, $\sum_k b_k = \infty$, $\sum_k c_k = \infty$, $\sum_k a_k^2 < \infty$, $\sum_k b_k^2 < \infty$, $\sum_k c_k^2 < \infty$, $\lim_{k \rightarrow \infty} c_k/a_k = 0$, and $\lim_{k \rightarrow \infty} b_k/c_k = 0$, which ensures that AIS-generator learns at a faster rate than the critic, and the critic learns at a faster rate than the policy.

Under standard technical conditions (see Theorem 23 of [64] or Page 35 of [67]), we can show that iterations (3.14) and (3.18) will converge to a stationary point of the corresponding ODE limits. At convergence, depending on ε and δ for the quality of AIS approximation, we can obtain approximation guarantees corresponding to Theorem 4.

We conclude this discussion by mentioning that algorithms similar to the AIS-based PORL have been proposed in the literature including [6, 7, 9, 35, 36, 68–74]. However, these papers only discuss the empirical performance of the proposed algorithms but do not derive performance bounds.

Chapter 4

Experimental Results

We perform numerical experiments to check the effectiveness of AIS-based PORL algorithms proposed in the previous section. The code for all AIS experiments is available in [75]. We consider three classes of POMDP environments, which have increasing difficulty in terms of the dimension of their state and observation spaces:

1. Low-dimensional environments (Tiger, Voicemail, and Cheese Maze)
2. Moderate-dimensional environments (Rock Sampling and Drone Surveillance)
3. High-dimensional environments (different variations of MiniGrid)

Previous experimental work on AIS in [12] was based on (AP1) and (AP2) and the KL-divergence loss as a surrogate using an actor-critic framework and only a few low-dimensional environments mentioned here were used. In this work, we use the actor only framework and learn an AIS based on (AP1), (AP2a) and (AP2b) and the KL IPM and MMD IPM. Apart from the newer and harder environments considered in this work, we also present results on learning an AIS which has compressed observations. Also, experimental results for the corresponding environments in [12] were improved upon in this work.

There are four components of the corresponding AIS-generator: the history compression function $\hat{\sigma}$, the AIS update function $\hat{\varphi}$, the reward prediction function \hat{r} , and the observation prediction kernel \hat{P}^y . We model $\hat{\sigma}$ as an LSTM, where the memory update unit of LSTM acts as $\hat{\varphi}$. We model \hat{r} , \hat{P}^y , and the policy $\hat{\pi}$ as feed-forward neural networks. A block diagram of the network architecture is shown in Fig. 4.1 and the details of the networks and the hyperparameters are presented in Appendix B.1. To avoid over-fitting, we use the

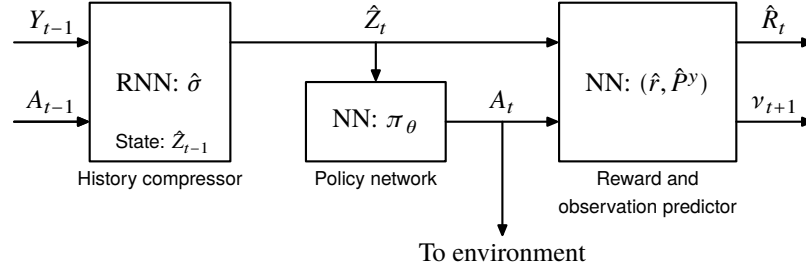


Fig. 4.1 Network architecture for PORL using AIS.

same network architecture and hyperparameters for all environments in the same difficulty class.

We repeat each experiment for multiple random seeds and plot the median value along with the uncertainty band from the first to the third quartile. For all environments, we compare our performance with a baseline which uses an actor-critic algorithm where both the actor and critic are modeled using LSTM and the policy parameters are updated using PPO. This architecture was proposed as a baseline for the MiniGrid environments in [51]. The details of the baseline architecture are presented in Appendix B.1.

To evaluate the performance of the policy while training for AIS-based PORL, a separate set of rollouts is carried out at fixed intervals of time steps and the mean of these rollouts is considered. For the PPO baseline a number of parallel actors are used during training, and once the episodes are completed, their returns are stored in a list. A fixed number (based on the number of parallel actors) of past episodes are considered to evaluate the mean performance of the current policy during training. See Appendix B.1 for details.

For the low and moderate-dimensional environments, we compare the performance with the best performing planning solution obtained from the JuliaPOMDP repository [23]. For the high-dimensional environments, finding a planning solution is intractable, so we only compare with the PPO baseline mentioned previously.

4.1 Low-dimensional environments

In these POMDP environments, the size of the unobserved state space is less than about 10 and the planning solution can be easily obtained using standard POMDP solvers.

1. **TIGER:** The Tiger environment is a sequential hypothesis testing task proposed in

[13]. The environment consists of two doors, with a tiger behind one door and a treasure behind the other. The agent can either perform a LISTEN action, which has a small negative reward of -1 and gives a noisy observation about the location of the tiger, or the agent can open one of the doors. Opening the door with the treasure gives a reward of $+10$ while opening the door with a tiger gives a large negative reward of -100 . After opening a door, the environment is reset.

2. VOICEMAIL: The Voicemail environment is also a sequential hypothesis testing task proposed in [76]. This environment models a dialog system for managing voicemails. The agent can either perform an ASK action, which has a small negative reward of -1 and gives a noisy observation about the intent of the user, or the agent can execute SAVE or DELETE. Choosing a SAVE/DELETE action which matches the intent of the user gives a reward of $+5$. The agent receives a negative reward of -20 for action DELETE when the user intent is SAVE, while choosing action SAVE when the user intent is DELETE gives a smaller but still significant negative reward of -10 . Since the user prefers SAVE more than DELETE, the initial belief is given by $[0.65, 0.35]$ for SAVE and DELETE respectively. After taking a SAVE/DELETE action, the agent moves on to the next voicemail message.

3. CHEESEMAZE: The CheeseMaze environment is a POMDP with masked states proposed in [14]. The environment consists of 11 states and 7 observations as shown on the right. The objective is to reach the goal state, which is indicated by observation 7. The agent only receives a reward of $+1$, when the goal state is reached.

1	2	3	2	4
5		5		5
6		7		6

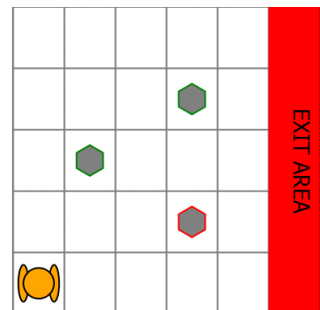
For all three environments, we compare the performance of AIS-based PORL with the LSTM+PPO baseline, described earlier. We also compare with the best performing planning solution from the JuliaPOMDP repository [23]. The results are presented in Fig. 4.2, which shows both AIS-based PORL and LSTM+PPO converge close to the planning solutions relatively quickly.¹

¹The performance of all learning algorithms for CHEESEMAZE are better than the best planning solution. We solved the CHEESEMAZE model with other solvers available in the JuliaPOMDP [23], and all these solutions performed worse than the solution obtained by incremental pruning presented here.

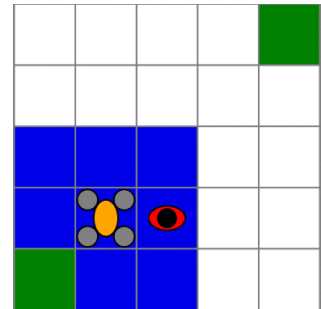
4.2 Moderate-dimensional environments

In these environments, the size of the unobserved state is moderately large (of the order of 10^2 to 10^3 unobserved states) and the optimal planning solution cannot be easily obtained using standard POMDP solvers. However, an approximate planning solution can be easily obtained using standard approximation algorithms for POMDPs.

1. **ROCKSAMPLE:** RockSample is a scalable POMDP environment introduced in [77] which models the rover science exploration. The $\text{RockSample}(n, k)$ environment consists of a $n \times n$ grid with k rocks. The rocks are at known positions. Some of the rocks which are labeled as GOOD rocks have scientific values; other rocks which are labeled as BAD rocks do not. Sampling a rock is expensive and the agent has a noisy long-range sensor to help determine if a rock is GOOD before choosing to approach and sample it. At each stage, the agent can choose from $k + 5$ actions: NORTH, SOUTH, EAST, WEST, SAMPLE, $\text{CHECK}_1, \dots, \text{CHECK}_k$. The first four are deterministic single-step motion actions. The SAMPLE action samples the rock at the current location; if the rock is GOOD, there is a reward of +20 and the rock becomes BAD (so that no further reward can be gained from sampling it); if the rock is BAD, there is a negative reward of -10. The right edge of the map is a terminal state and reaching it gives a reward of +10. In our experiments, we use a $\text{RockSample}(5, 3)$ environment.



2. **DRONESURVEILLANCE:** DroneSurveillance is a POMDP model of deploying an autonomous aerial vehicle in a partially observed, dynamic, indoor environment introduced in [78]. The environment is a 5×5 grid with two agents: a ground agent which moves randomly and an aerial agent, whose motion has to be controlled. The aerial agent starts at the bottom-left cell and has to reach the upper-right cell (the goal state) without being in the same location as the ground agent. The ground agent cannot enter the start or goal states. The aerial agent has a downward facing camera which can view a 3×3 grid centered at its current location and it can perfectly see the location of the ground agent if it is in this view. At each stage, the aerial agent may choose from 5 actions: NORTH, SOUTH, EAST, WEST, HOVER. The first four are deterministic single-step motion actions and the HOVER action keeps the aerial vehicle at its current position. Reaching the goal gives a reward of +1 and ends the episode. If both agents are in the same cell, there is a negative reward of -1 and the episode ends.



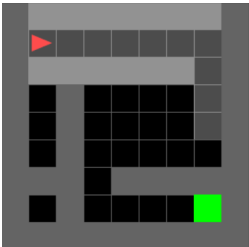
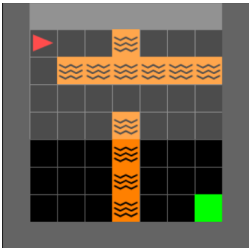
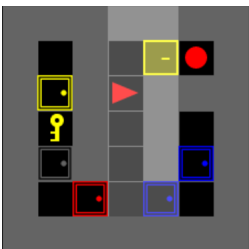
The visualizations above are taken from the JuliaPOMDP environments [23]. For both environments, we compare the performance of AIS-based PORL with the LSTM+PPO baseline described earlier. We also compare with the best performing planning solution from the JuliaPOMDP repository [23]. The results are shown in Fig. 4.3 which shows that both AIS-based PORL algorithms converge close to the best planning solution in both environments. The performance of LSTM+PPO is similar in DRONESURVEILLANCE but LSTM+PPO gets stuck in a local minima in ROCKSAMPLE.

4.3 High-dimensional environments

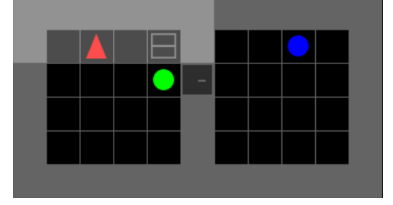
We use the MiniGrid environments from the BabyAI platform [15], which are partially observable 2D grid environments which has tasks of increasing complexity level. The environment has multiple entities (agent, walls, lava, boxes, doors, and keys); objects can be picked up, dropped, and moved around by the agent; doors can be unlocked via keys of the same color (which might be hidden inside boxes). The agents can see a 7×7 view in front of it but it cannot see past walls and closed doors. At each time, it can choose from the following actions: {MOVE FORWARD, TURN LEFT, TURN RIGHT, OPEN DOOR/BOX,

PICK UP ITEM, DROP ITEM, DONE}. The agent can only hold one item at a time. The objective is to reach a goal state in the quickest amount of time (which is captured by assigning to the goal state a reward which decays over time).

Most of the environments have a certain theme, and we cluster the environments accordingly. The visualizations below are taken from the Gym MiniGrid environments [15].

1. **SIMPLE CROSSING:** A simple crossing environment is a 2D grid with columns of walls with an opening (or a crossing). The agent can traverse the wall only through the openings and needs to find a path from the start to the goal state. There are four such environments (MGSCS9N1, MGSCS9N2, MGSCS9N3, and MGSCS11N5) where the label S_nN_m means that the size of the environment is $n \times n$ and there are m columns of walls. 
2. **LAVA CROSSING:** The lava crossing environments are similar to the simple crossing environments, but the walls are replaced by lava. If the agent steps on to the lava block then it dies and the episode ends. Therefore, exploration is more difficult in lava crossing as compared to simple crossing. There are two such environments (MGLCS9N1 and MGLCS9N2) where the label S_nN_m has the same interpretation as simple crossing. 
3. **KEY CORRIDOR:** The key corridor environments consist of a central corridor which has rooms on the left and right sides which can be accessed through doors. When the door is locked it can be accessed through doors. When the door is locked it can be opened using a key of the same color. The agent has to move to the location of the key, pick it up, move to the location of the correct door, open the door, drop the key, and pick up the colored ball. There are three such environments (MGKCS3R1, MGKCS3R2, and MGKCS3R3), where the label S_nR_m means that the size of the grid is proportional to n and the number of rooms present is $2m$. 

4. OBSTRUCTED MAZE: The obstructed maze environments are similar to key corridor environments but the key is inside a box and the box has to be opened to find the key. We consider two such environments (MGOM1Dl and MGOM1Dlh). In MGOM1Dl box is already open while in MGOM1Dlh the box is closed.

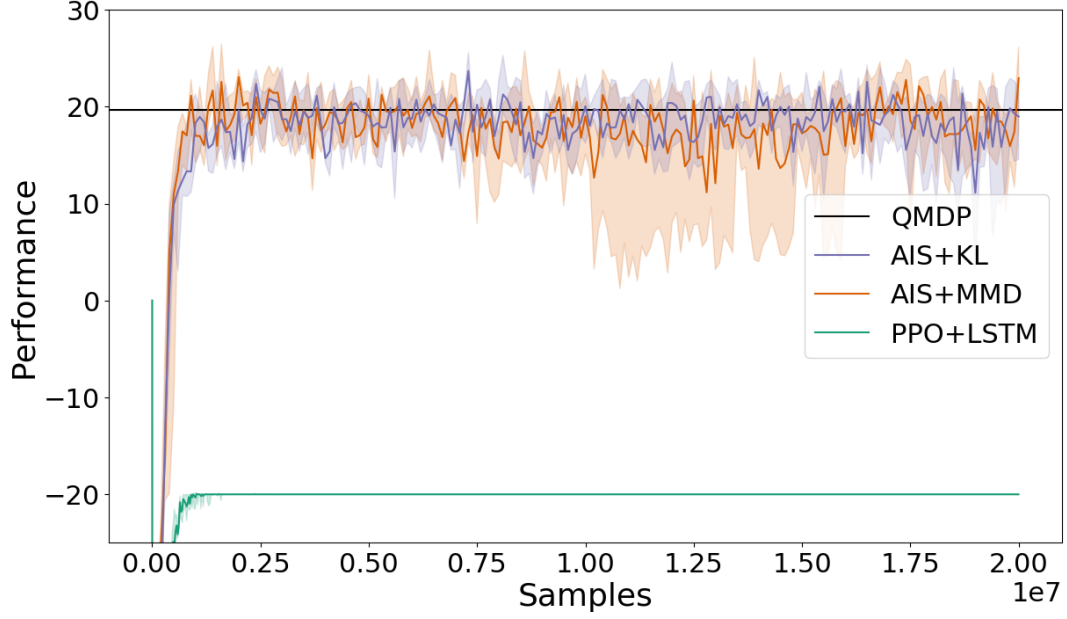


There is an additional such environment in the BabyAI platform (MGOM1Dlhb), which is more suitable for continual learning algorithms so we exclude it here.

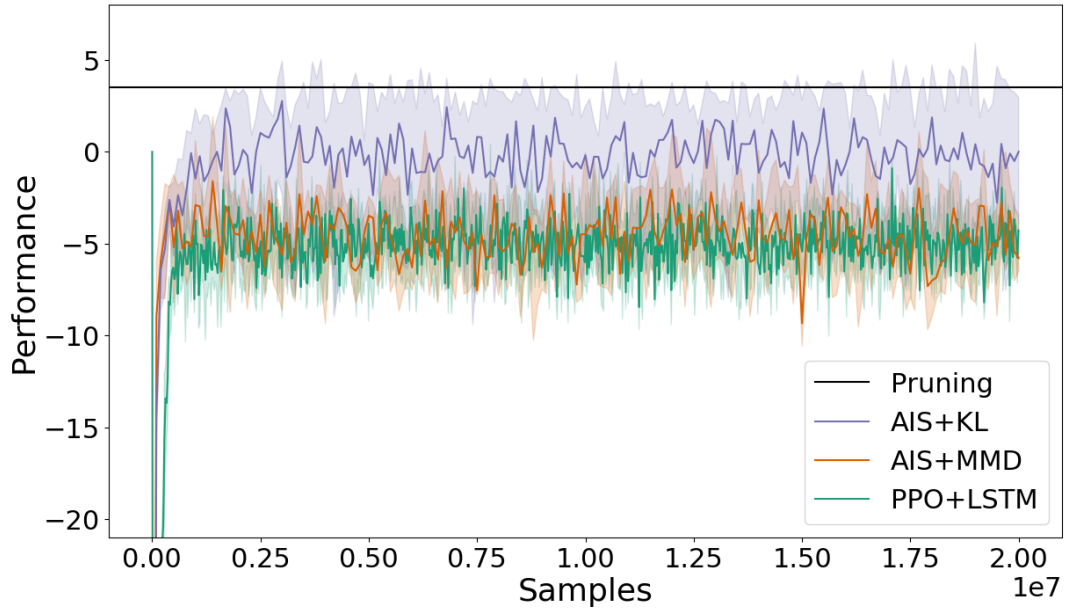
The number of observations in a given MiniGrid environment is discrete but is too large to model it as a one-hot encoded discrete observation as done in the previous environments. Instead we compress the observations as described in Section 2.3.3 by using an autoencoder to convert a large discrete space to a continuous space with a tractable size. A separate autoencoder is trained for each environment using a dataset that is created by performing random rollouts. Once the autoencoder is trained over the fixed dataset for several epochs, it is fixed and used to generate the observations for learning the AIS. This is very similar to [6], where they learn the autoencoder in a similar fashion and then fix it, following which their training procedure for the next observation distribution prediction and policy takes place. Alternatively, the autoencoder can also be trained parallel to the AIS and policy so that it can represent the entire observation space sufficiently (at least the parts that are important) by using (AP2) instead of (AP2a) and (AP2b). This is more similar to the recent work based on world models [35, 36].

Note that the output of the autoencoder is a continuous variable and we are using MMD with $p = 2$ as an IPM. As explained in Section 3.1, $d_{\mathfrak{F}_2}(\mu, \nu)^2$ only depends on the mean of μ and ν . So, to simplify the computations, we assume that ν is a Dirac delta distribution centered at its mean. Thus, effectively, we are predicting the mean of the next observation. In general, simply predicting the mean of the observations may not lead to a good representation, but in the MiniGrid environments, the transitions are deterministic and the only source of stochasticity in the observations is due to the initial configuration of the environment. So, in practice, simply predicting the mean of the next observation works reasonably well. We emphasize that for other more general environments with truly stochastic observations, such a choice of IPM may not work well and it may be better to choose the MMD $d_{\mathfrak{F}_p}$ defined in Proposition 11 for a different value of p , say $p = 1$ (which corresponds to the energy distance [62]).

For all MiniGrid environments, we compare the performance of AIS-based PORL with the LSTM+PPO baseline proposed in [51]. The results are shown in Fig. 4.4 which shows that for most environments AIS-based PORL converges to better performance values. Note that AIS-based PORL fails to learn in the LAVA CROSSING environments (MGLCS9N1 and MGLCS9N2) while LSTM+PPO fails to learn in the larger KEY CROSSING environments (MGKCS3R2 and MGKCS3R3) and in the OBSTRUCTED MAZE environments (MGOM1Dl and MGOM1Dlh).

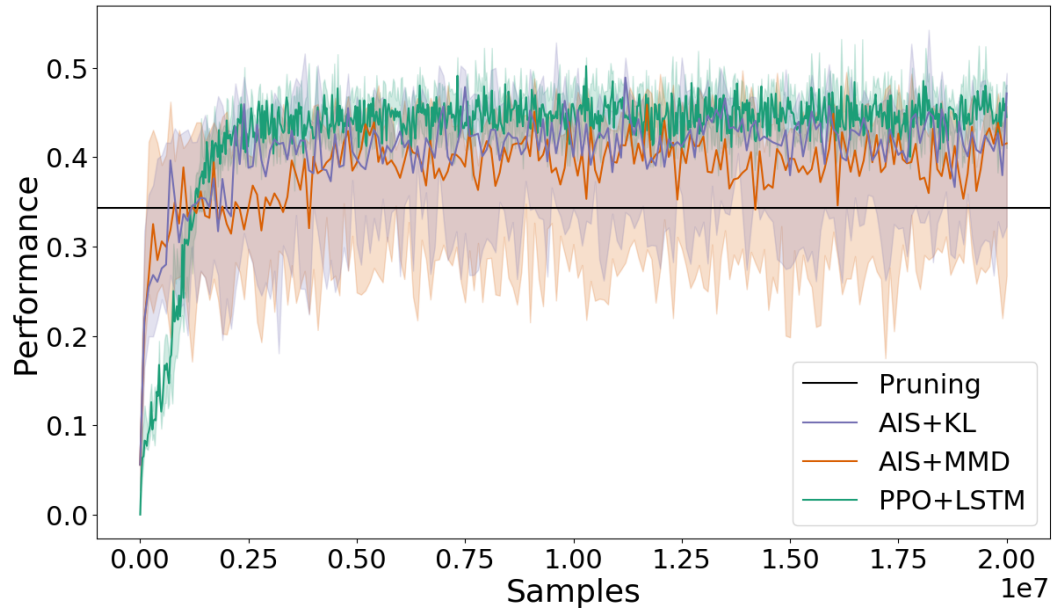


(a) Tiger



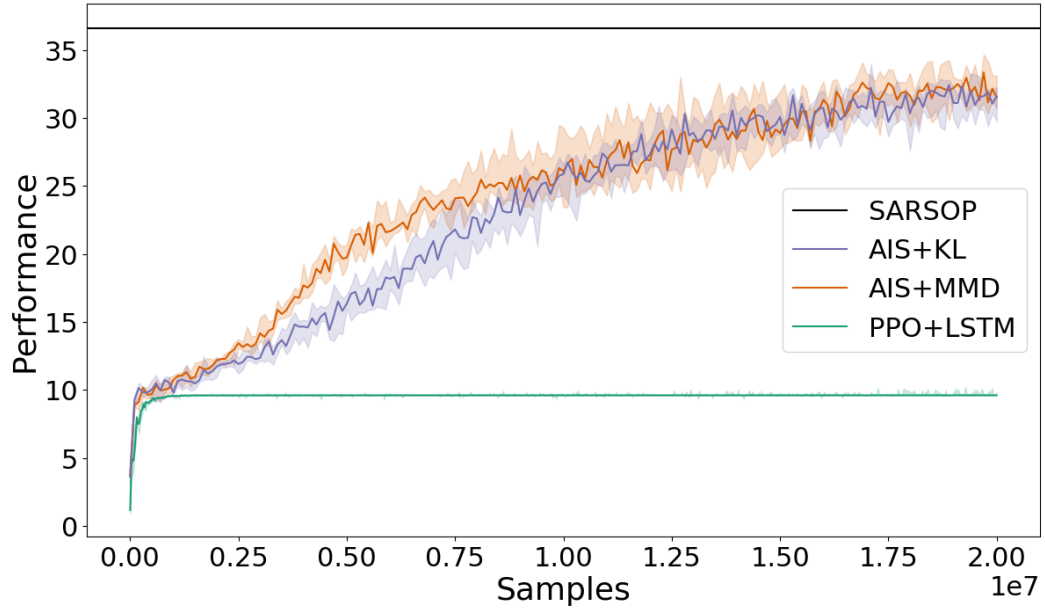
(b) Voicemail

Fig. 4.2 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for low-dimensional environments (for 10 random seeds).

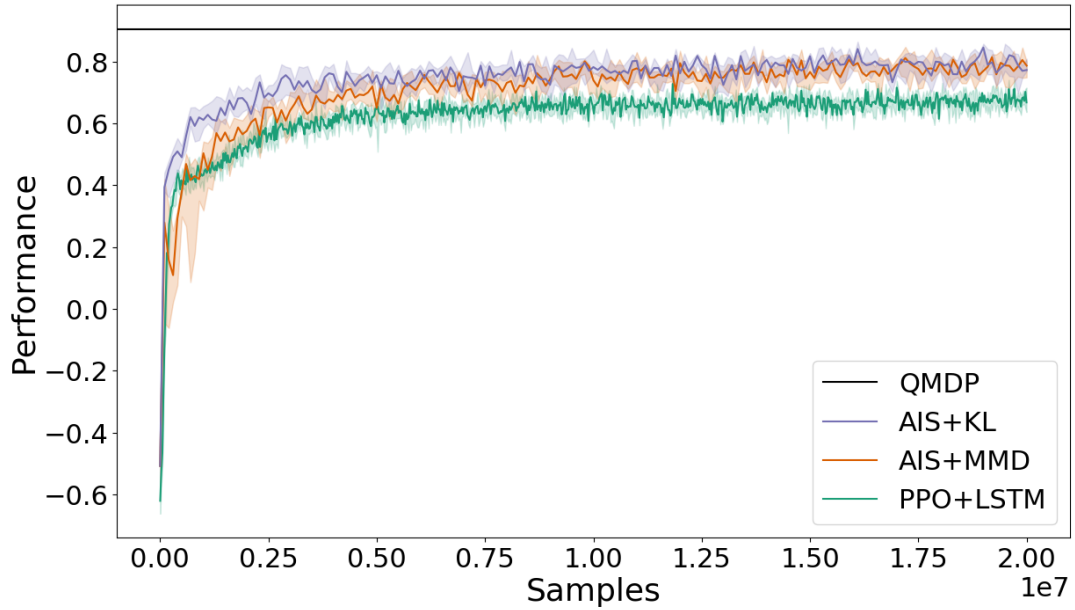


(c) Cheese Maze

Fig. 4.2 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for low-dimensional environments (for 10 random seeds) (contd.).

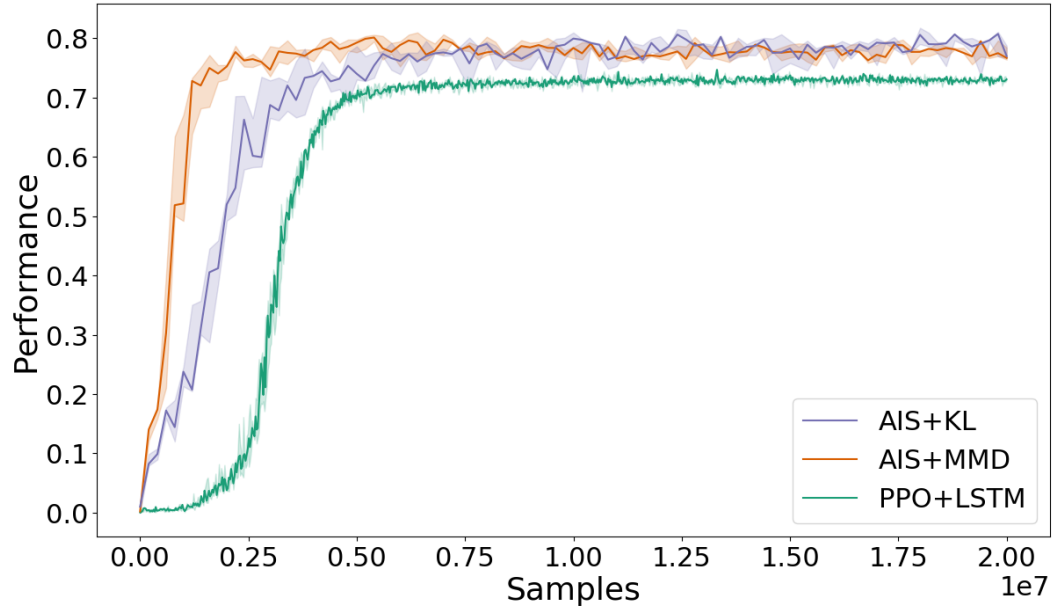


(a) Rock Sampling

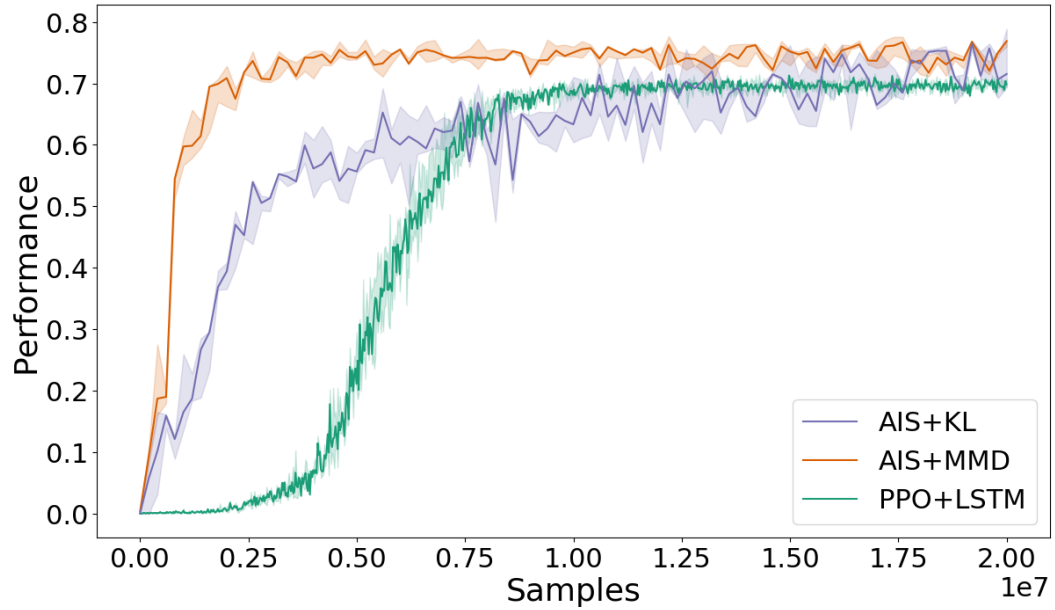


(b) Drone Surveillance

Fig. 4.3 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for moderate-dimensional environments (for 10 random seeds).

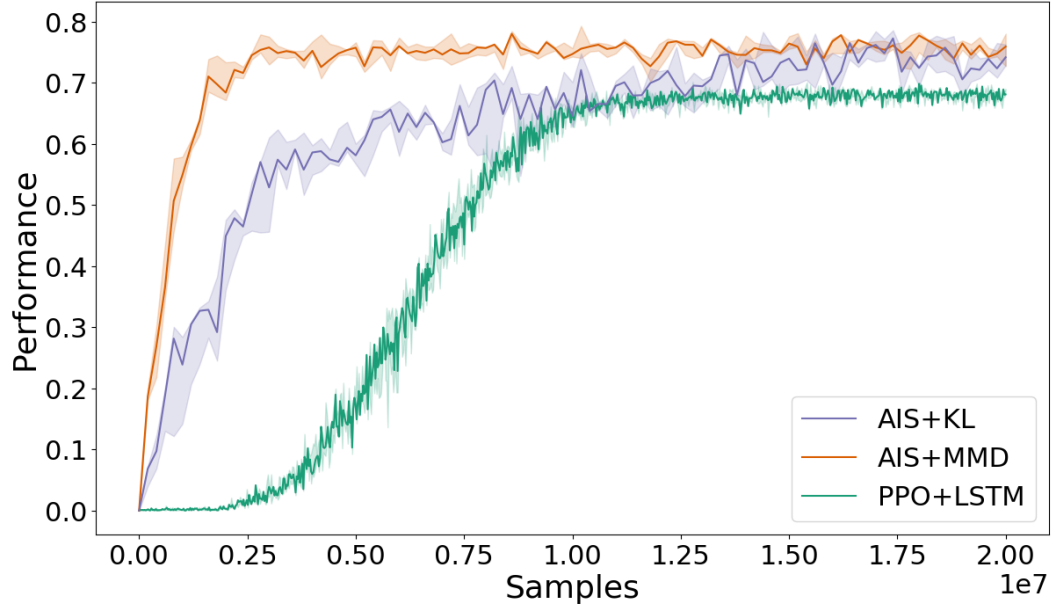


(a) MGSCS9N1

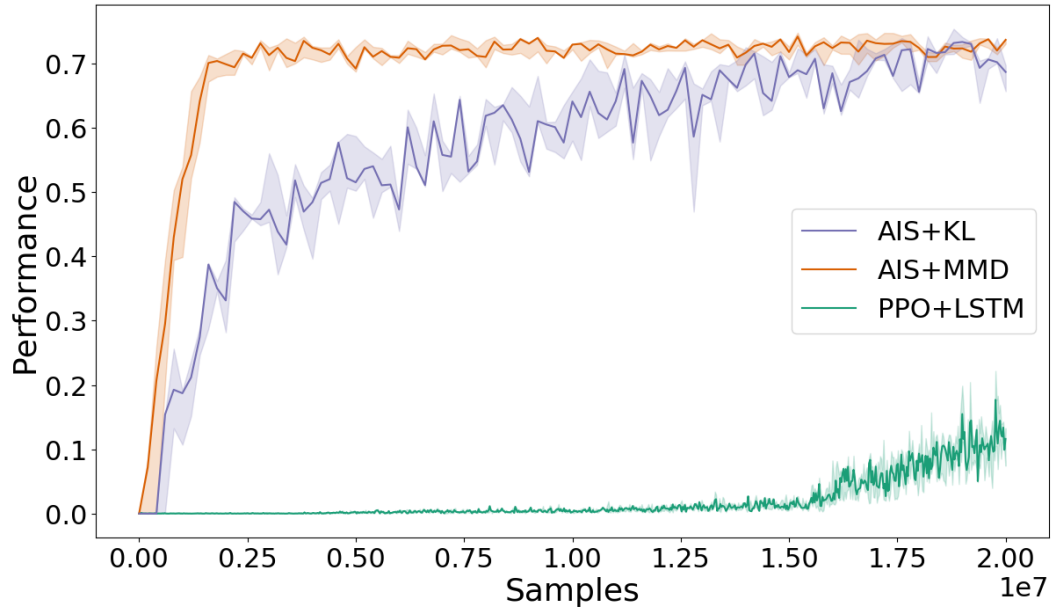


(b) MGSCS9N2

Fig. 4.4 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds).

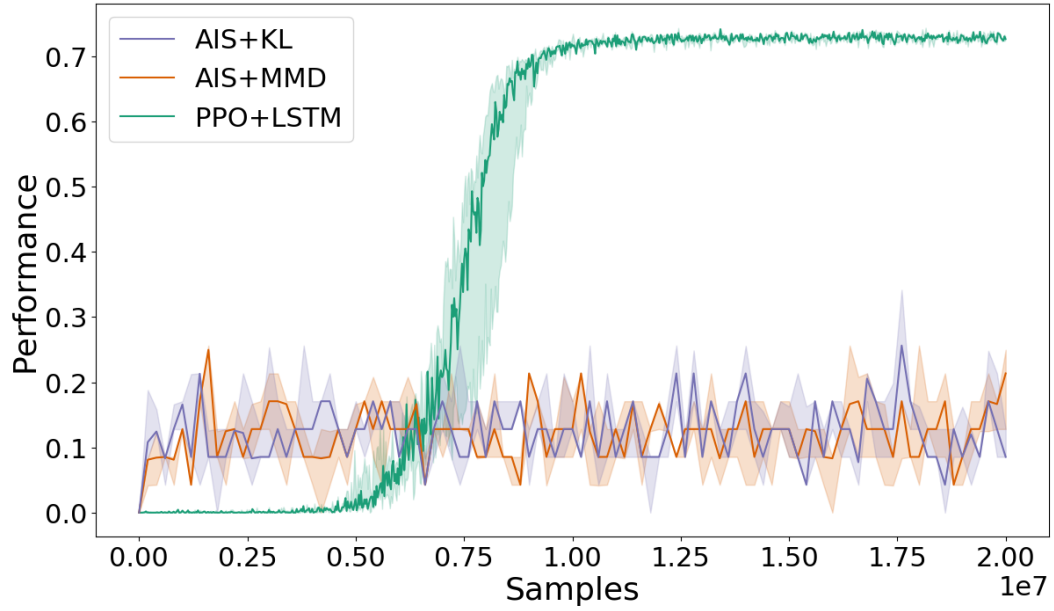


(c) MGSCS9N3

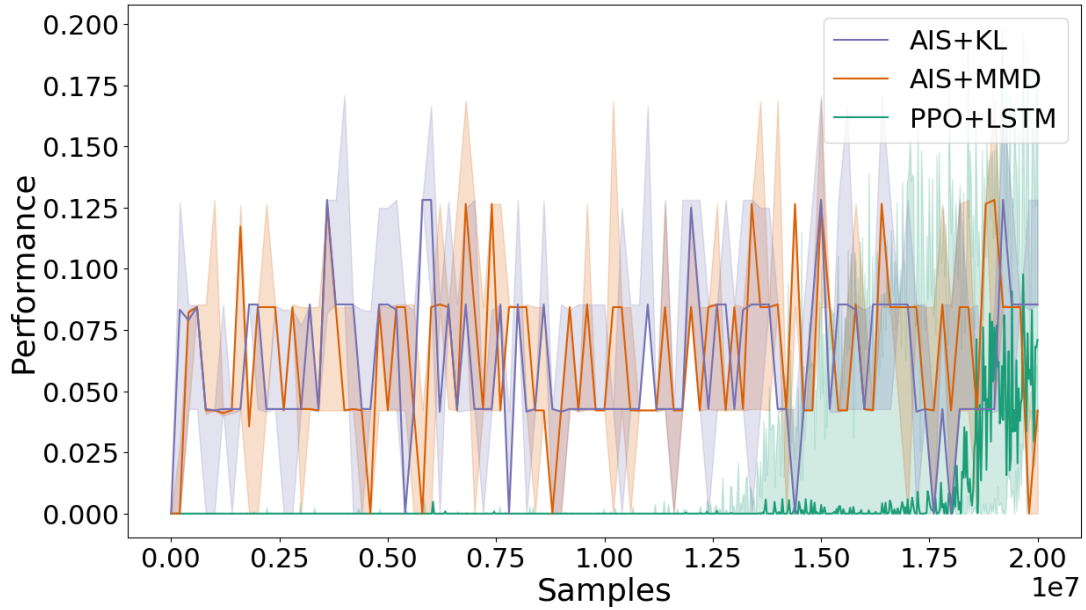


(d) MGSCS11N5

Fig. 4.4 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.).

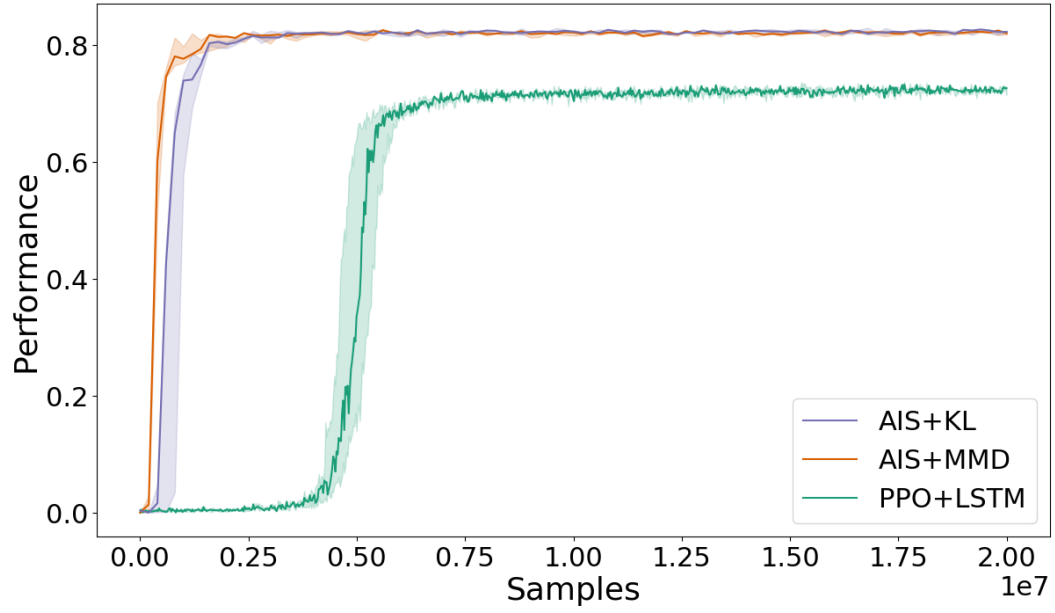


(e) MGLCS9N1

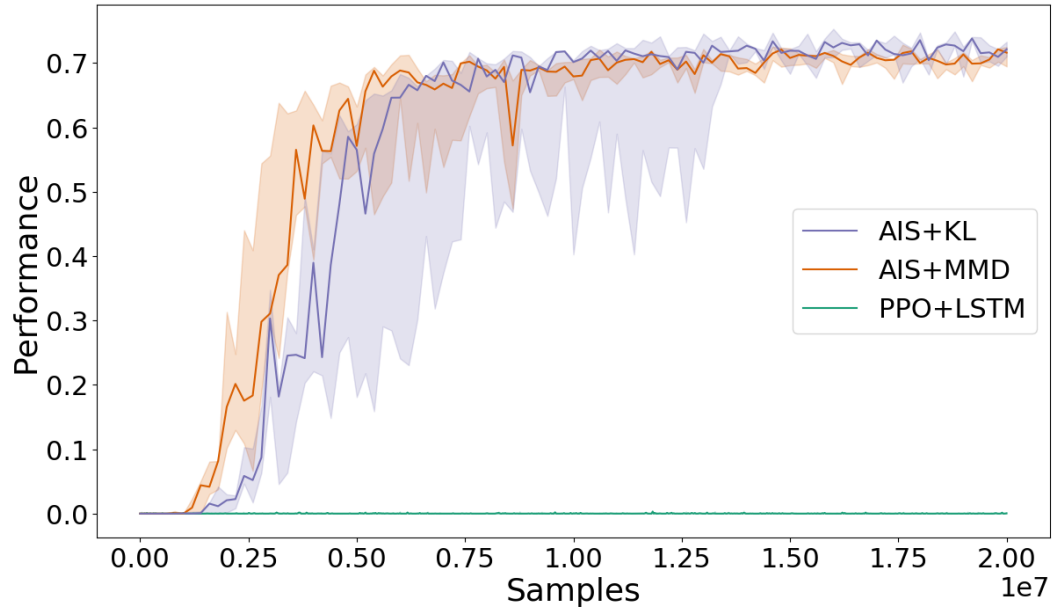


(f) MGLCS9N2

Fig. 4.4 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.).

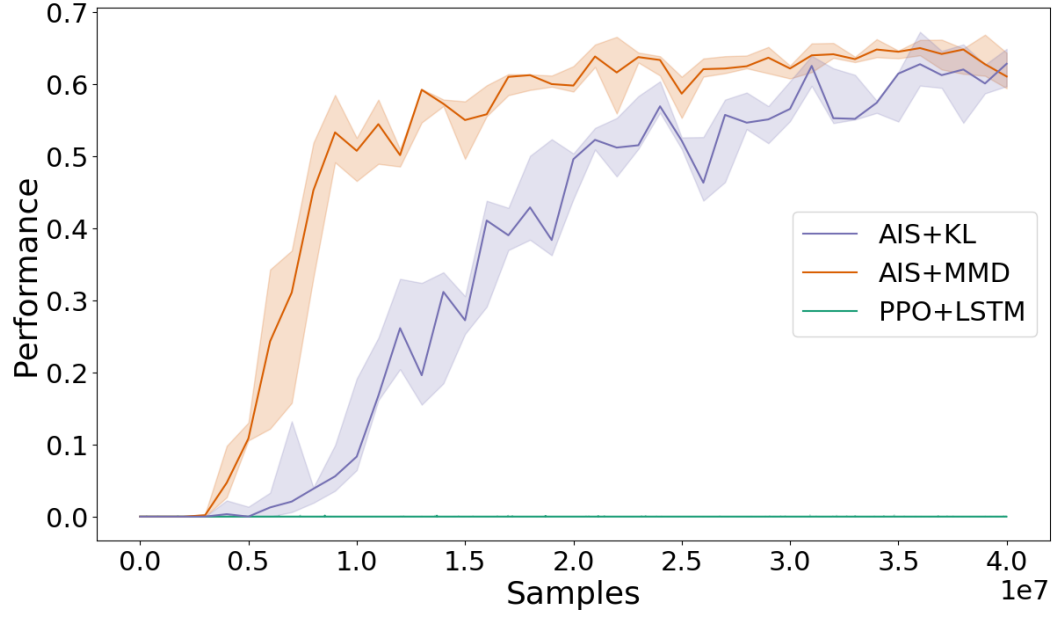


(g) MGKCS3R1

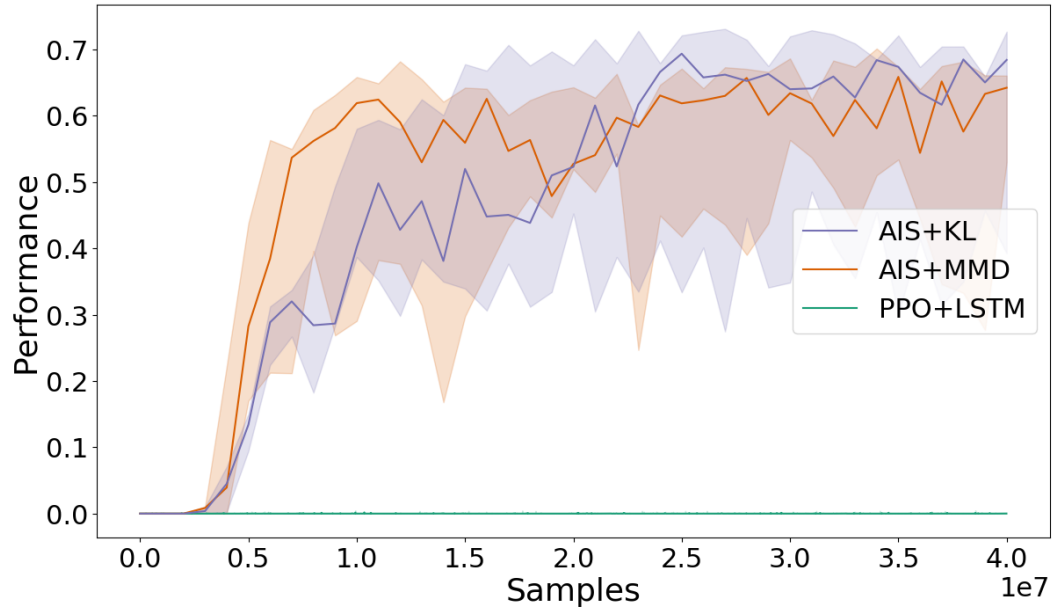


(h) MGKCS3R2

Fig. 4.4 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.).

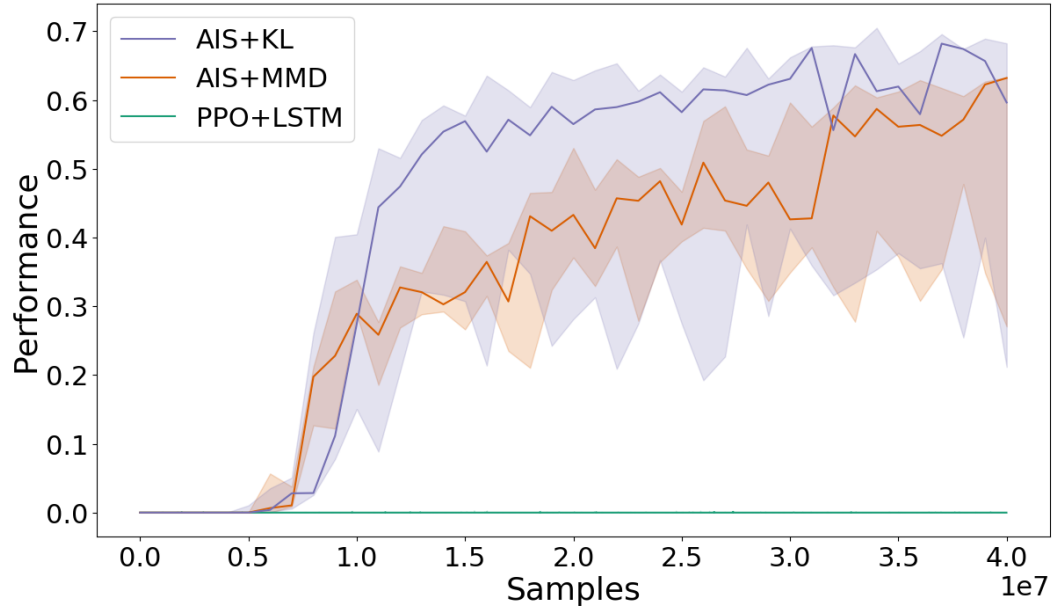


(i) MGKCS3R3



(j) MGOM1D1

Fig. 4.4 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.).



(k) MGOM1Dlh

Fig. 4.4 Comparison of AIS-based actor only PORL algorithm with LSTM+PPO baseline for high-dimensional environments (for 5 random seeds) (contd.).

Chapter 5

Conclusions and Future Directions

In this work, we build upon the recently proposed idea of AIS [12] for learning in partially observable environments. We extend the theoretical results on performance bounds for approximate planning and learning for a general IPM similar to the bounds in [12] for the KL-divergence loss as a surrogate for the Wasserstein distance. In particular, we present results on the KL IPM and the MMD IPM accompanied with PORL algorithms and we show that other general IPMs can also be considered. We also explore the idea of observation compression to deal with environments having high-dimensional observation spaces. We mainly look at two different paradigms to learn such a representation from observational data. Either we can try to learn an AIS that predicts the distribution of the next AIS or the next observation, given the current AIS and action. Predicting the next AIS was generally found to be harder in experiments, so the results presented in this work rely on predicting the distribution of the next observation instead. This makes sense since an AIS is non-stationary and keeps changing as it is being learnt and so it is more difficult to learn.

A multi-time scale PORL algorithm is introduced which learns the policy and AIS compression functions side-by-side, but at different time scales (the AIS should be learnt at a faster time scale than the policy). These methods are end-to-end and do not require any intervention so they are easy to deploy.

We show that the ideas involved in AIS work on several environments of low, moderate and high-dimensionality and we obtain a deeper understanding on how to set the practical parameters of the networks involved. The two IPMs explored in numerical experiments are

the KL IPM and the MMD IPM, which exhibit similar performance in most environments. Both these variations also generally outperform the PPO with LSTM connections baseline in our experiments. Their performance is also close to the planning solutions (whenever such a planning solution is available).

5.1 Future Directions

The results presented here were mainly achieved just by incorporating the simple REINFORCE algorithm without much of the existing RL machinery in the current state-of-the-art, like actor-critic methods, $TD(\lambda)$ returns, experience replay, DQN [3] or a version of DQN like Rainbow DQN [4] are interesting directions to explore. It would also be interesting to see these ideas applied to larger partially observable problems such as robotics problems in the Mujoco [79] platform.

Appendix A

Proofs

Proposition 7. (P2a) and (P2b) imply (P2).

Proof. For any Borel subset D of Z_{t+1} , we have

$$\begin{aligned}
 & \mathbb{P}(Z_{t+1} \in D \mid H_t = h_t, A_t = a_t) \\
 & \stackrel{(a)}{=} \sum_{y_t \in Y} \mathbb{P}(Y_t = y_t, Z_{t+1} \in D \mid H_t = h_t, A_t = a_t) \\
 & \stackrel{(b)}{=} \sum_{y_t \in Y} \mathbb{1}\{\varphi_t(\sigma_t(h_t), y_t, a_t) \in D\} \mathbb{P}(Y_t = y_t \mid H_t = h_t, A_t = a_t) \\
 & \stackrel{(c)}{=} \sum_{y_t \in Y} \mathbb{1}\{\varphi_t(\sigma_t(h_t), y_t, a_t) \in D\} \mathbb{P}(Y_t = y_t \mid Z_t = \sigma_t(h_t), A_t = a_t) \\
 & \stackrel{(d)}{=} \mathbb{P}(Z_{t+1} \in D \mid Z_t = \sigma_t(h_t), A_t = a_t)
 \end{aligned}$$

where (a) follows from the law of total probability, (b) follows from (P2a), (c) follows from (P2b) and (d) from the law of total probability. \square

Theorem 1. Let $\{Z_t\}_{t=1}^T$ be an information state. Recursively define value functions $\{\bar{V}_t: Z_t \rightarrow \mathbb{R}\}_{t=1}^{T+1}$, as follows: $\bar{V}_{T+1}(z_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$:

$$\bar{Q}_t(z_t, a_t) := \mathbb{E}[R_t + \bar{V}_{t+1}(Z_{t+1}) \mid Z_t = z_t, A_t = a_t], \quad (2.7a)$$

$$\bar{V}_t(z_t) := \max_{a_t \in A} \bar{Q}_t(z_t, a_t). \quad (2.7b)$$

Then, we have the following:

1. For any time t , history h_t , and action a_t , we have that

$$Q_t(h_t, a_t) = \bar{Q}_t(\sigma_t(h_t), a_t) \text{ and } V_t(h_t) = \bar{V}_t(\sigma_t(h_t)). \quad (2.8)$$

2. Let $\bar{\pi} = (\bar{\pi}_1, \dots, \bar{\pi}_T)$, where $\bar{\pi}_t: Z_t \rightarrow \Delta(\mathbf{A})$, be a stochastic policy. Then, the policy $\pi = (\pi_1, \dots, \pi_T)$ given by $\pi_t = \bar{\pi}_t \circ \sigma_t$ is optimal if and only if for all t and all realizations z_t of information states Z_t , $\text{Supp}(\bar{\pi}_t(z_t)) \subseteq \arg \max_{a_t \in \mathbf{A}} \bar{Q}_t(z_t, a_t)$.

Proof. We prove the result by backward induction. By construction, (2.8) is true at time $T + 1$. This forms the basis of induction. Assume that (2.8) is true at time $t + 1$ and consider the system at time t . Then,

$$\begin{aligned} Q_t(h_t, a_t) &= \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t] \\ &\stackrel{(a)}{=} \mathbb{E}[R_t + \bar{V}_{t+1}(\sigma_{t+1}(H_{t+1})) \mid H_t = h_t, A_t = a_t] \\ &\stackrel{(b)}{=} \mathbb{E}[R_t + \bar{V}_{t+1}(Z_{t+1}) \mid Z_t = \sigma_t(h_t), A_t = a_t] \\ &\stackrel{(c)}{=} \bar{Q}_t(\sigma_t(h_t), a_t), \end{aligned}$$

where (a) follows from the induction hypothesis, (b) follows from the properties (P1) and (P2) of information state, and (c) follows from the definition of \bar{Q} . This shows that the action-value functions are equal. By maximizing over the actions, we get that the value functions are also equal. The optimality of the policy follows immediately from (2.8). \square

Proposition 8. (AP2a) and (AP2b) imply (AP2) holds with transition kernels $\{\hat{P}_t^y\}_{t=1}^T$ defined as follows: for any Borel subset \mathbf{B} of $\hat{\mathbf{Z}}$,

$$\hat{P}_t(\mathbf{B} \mid \hat{\sigma}_t(h_t), a_t) = \int_{\mathbf{Y}} \mathbb{1}_{\mathbf{B}}(\hat{\varphi}_t(\hat{\sigma}_t(h_t), y_t, a_t)) \hat{P}_t^y(dy_t \mid \hat{\sigma}_t(h_t), a_t).$$

Therefore, we can alternatively define an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ -AIS generator as a tuple $\{(\hat{\sigma}_t, \hat{r}_t, \hat{\varphi}_t, \hat{P}_t^y)\}_{t=1}^T$ which satisfies (AP1), (AP2a), and (AP2b).

Proof. Note that by the law of total probability, μ_t and ν_t defined in (AP2) are

$$\begin{aligned}\mu_t(\mathbf{B}) &= \int_{\mathbf{Y}} \mathbb{1}_{\mathbf{B}}(\hat{\varphi}_t(\hat{\sigma}_t(h_t), y_t, a_t)) \mu_t^y(dy_t), \\ \nu_t(\mathbf{B}) &= \int_{\mathbf{Y}} \mathbb{1}_{\mathbf{B}}(\hat{\varphi}_t(\hat{\sigma}_t(h_t), y_t, a_t)) \nu_t^y(dy_t).\end{aligned}$$

Thus, for any function $f: \hat{\mathbf{Z}}_{t+1} \rightarrow \mathbb{R}$,

$$\begin{aligned}\int_{\hat{\mathbf{Z}}_{t+1}} f d\mu_t &= \int_{\mathbf{Y}_t} f(\hat{\varphi}_t(\hat{\sigma}_t(h_t), y_t, a_t)) \mu_t^y(dy_t), \\ \int_{\hat{\mathbf{Z}}_{t+1}} f d\nu_t &= \int_{\mathbf{Y}_t} f(\hat{\varphi}_t(\hat{\sigma}_t(h_t), y_t, a_t)) \nu_t^y(dy_t).\end{aligned}$$

The result then follows from (2.12). \square

Theorem 2. Suppose $\{\hat{\sigma}_t, \hat{P}_t, \hat{r}_t\}_{t=1}^T$ is an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ -AIS generator. Recursively define approximate action-value functions $\{\hat{Q}_t: \hat{\mathbf{Z}}_t \times \mathbf{A} \rightarrow \mathbb{R}\}_{t=1}^T$ and value functions $\{\hat{V}_t: \hat{\mathbf{Z}}_t \rightarrow \mathbb{R}\}_{t=1}^T$ as follows: $\hat{V}_{T+1}(\hat{z}_{T+1}) := 0$ and for $t \in \{T, \dots, 1\}$:

$$\hat{Q}_t(\hat{z}_t, a_t) := \hat{r}_t(\hat{z}_t, a_t) + \int_{\hat{\mathbf{Z}}_t} \hat{V}_{t+1}(\hat{z}_{t+1}) \hat{P}_t(d\hat{z}_{t+1} \mid \hat{z}_t, a_t), \quad (2.13a)$$

$$\hat{V}_t(\hat{z}_t) := \max_{a_t \in \mathbf{A}} \hat{Q}_t(\hat{z}_t, a_t). \quad (2.13b)$$

Then, we have the following:

1. **Value function approximation:** For any time t , realization h_t of H_t , and choice a_t of A_t , we have

$$|Q_t(h_t, a_t) - \hat{Q}_t(\hat{\sigma}_t(h_t), a_t)| \leq \alpha_t \quad \text{and} \quad |V_t(h_t) - \hat{V}_t(\hat{\sigma}_t(h_t))| \leq \alpha_t, \quad (2.14)$$

where α_t satisfies the following recursion: $\alpha_{T+1} = 0$ and for $t \in \{T, \dots, 1\}$,

$$\alpha_t = \varepsilon_t + \rho_{\mathfrak{F}}(\hat{V}_{t+1})\delta_t + \alpha_{t+1}.$$

Therefore,

$$\alpha_t = \varepsilon_t + \sum_{\tau=t+1}^T [\rho_{\mathfrak{F}}(\hat{V}_{\tau})\delta_{\tau-1} + \varepsilon_{\tau}].$$

2. **Approximately optimal policy:** Let $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_T)$, where $\hat{\pi}_t: \hat{\mathbf{Z}}_t \rightarrow \Delta(\mathbf{A})$, be a stochastic policy that satisfies

$$\text{Supp}(\hat{\pi}(\hat{z}_t)) \subseteq \arg \max_{a_t \in \mathbf{A}} \hat{Q}_t(\hat{z}_t, a_t). \quad (2.15)$$

Define policy $\pi = (\pi_1, \dots, \pi_T)$, where $\pi_t: \mathbf{H}_t \rightarrow \Delta(\mathbf{A})$ by $\pi_t := \hat{\pi}_t \circ \hat{\sigma}_t$. Then, for any time t , realization h_t of H_t , and choice a_t of A_t , we have

$$|Q_t(h_t, a_t) - Q_t^\pi(h_t, a_t)| \leq 2\alpha_t \quad \text{and} \quad |V_t(h_t) - V_t^\pi(h_t)| \leq 2\alpha_t. \quad (2.16)$$

Proof. We prove both parts by backward induction. We start with value function approximation. Eq. (2.14) holds at $T+1$ by definition. This forms the basis of induction. Assume that (2.14) holds at time $t+1$ and consider the system at time t . We have that

$$\begin{aligned} & |Q_t(h_t, a_t) - \hat{Q}_t(\hat{\sigma}_t(h_t), a_t)| \\ & \stackrel{(a)}{\leq} |\mathbb{E}[R_t \mid H_t = h_t, A_t = a_t] - \hat{r}_t(\hat{\sigma}_t(h_t), a_t)| \\ & \quad + \mathbb{E}[|V_{t+1}(H_{t+1}) - \hat{V}_{t+1}(\hat{\sigma}_{t+1}(H_{t+1}))| \mid H_t = h_t, A_t = a_t] \\ & \quad + \left| \mathbb{E}[\hat{V}_{t+1}(\hat{\sigma}_{t+1}(H_{t+1})) \mid H_t = h_t, A_t = a_t] - \int_{\hat{\mathbf{Z}}_t} \hat{V}_{t+1}(\hat{z}_{t+1}) \hat{P}_t(d\hat{z}_{t+1} \mid \hat{\sigma}_t(h_t), a_t) \right| \\ & \stackrel{(b)}{\leq} \varepsilon_t + \alpha_{t+1} + \rho_{\mathcal{F}}(\hat{V}_{t+1})\delta_t = \alpha_t \end{aligned}$$

where (a) follows from triangle inequality and (b) follows from (AP1), the induction hypothesis, (AP2) and (2.9). This proves the first part of (2.14). The second part follows from

$$|V_t(h_t) - \hat{V}_t(\hat{\sigma}_t(h_t))| \stackrel{(a)}{\leq} \max_{a_t \in \mathbf{A}} |Q_t(h_t, a_t) - \hat{Q}_t(\hat{\sigma}_t(h_t), a_t)| \leq \alpha_t,$$

where (a) follows from the inequality $\max f(x) \leq \max |f(x) - g(x)| + \max g(x)$.

To prove the policy approximation, we first prove an intermediate result. For policy $\hat{\pi}$ recursively define $\{\hat{Q}_t^{\hat{\pi}}: \hat{\mathbf{Z}} \times \mathbf{A} \rightarrow \mathbb{R}\}_{t=1}^T$ and $\{\hat{V}_t^{\hat{\pi}}: \hat{\mathbf{Z}} \rightarrow \mathbb{R}\}_{t=1}^{T+1}$ as follows: $\hat{V}_{T+1}^{\hat{\pi}}(\hat{z}_{T+1}) := 0$

and for $t \in \{T, \dots, 1\}$:

$$\hat{Q}_t^{\hat{\pi}}(\hat{z}_t, a_t) := \hat{r}_t(\hat{z}_t, a_t) + \int_{\hat{Z}_t} \hat{V}_t^{\hat{\pi}}(\hat{z}_{t+1}) \hat{P}_t(d\hat{z}_{t+1} \mid \hat{z}_t, a_t) \quad (\text{A.1a})$$

$$\hat{V}_t^{\hat{\pi}}(\hat{z}_t) := \sum_{a_t \in \mathbf{A}} \hat{\pi}_t(a_t \mid \hat{z}_t) \cdot \hat{Q}_t^{\hat{\pi}}(\hat{z}_t, a_t). \quad (\text{A.1b})$$

Note that (2.15) implies that

$$\hat{Q}_t^{\hat{\pi}}(\hat{z}_t, a_t) = \hat{Q}_t(\hat{z}_t, a_t) \quad \text{and} \quad \hat{V}_t^{\hat{\pi}}(\hat{z}_t) = \hat{V}(\hat{z}_t). \quad (\text{A.2})$$

Now, we prove that

$$|Q_t^{\pi}(h_t, a_t) - \hat{Q}_t^{\hat{\pi}}(\hat{\sigma}_t(h_t), a_t)| \leq \alpha_t \quad \text{and} \quad |V_t^{\pi}(h_t) - \hat{V}_t^{\hat{\pi}}(\hat{\sigma}_t(h_t))| \leq \alpha_t. \quad (\text{A.3})$$

We prove the result by backward induction. By construction, Eq. (A.3) holds at time $T+1$. This forms the basis of induction. Assume that (A.3) holds at time $t+1$ and consider the system at time t . We have

$$\begin{aligned} & |Q_t^{\pi}(h_t, a_t) - \hat{Q}_t^{\hat{\pi}}(\hat{\sigma}_t(h_t), a_t)| \\ & \stackrel{(a)}{\leq} |\mathbb{E}[R_t \mid H_t = h_t, A_t = a_t] - \hat{r}_t(\hat{\sigma}_t(h_t), a_t)| \\ & \quad + \mathbb{E}[|V_{t+1}^{\pi}(H_{t+1}) - \hat{V}_{t+1}^{\hat{\pi}}(\hat{\sigma}_{t+1}(H_{t+1}))| \mid H_t = h_t, A_t = a_t] \\ & \quad + \left| \mathbb{E}[\hat{V}_{t+1}^{\hat{\pi}}(\hat{\sigma}_{t+1}(H_{t+1})) \mid H_t = h_t, A_t = a_t] - \int_{\hat{Z}_t} \hat{V}_{t+1}^{\hat{\pi}}(\hat{z}_{t+1}) \hat{P}_t(d\hat{z}_{t+1} \mid \hat{\sigma}_t(h_t), a_t) \right| \\ & \stackrel{(b)}{\leq} \varepsilon_t + \alpha_{t+1} + \rho_{\mathfrak{F}}(\hat{V}_{t+1})\delta_t = \alpha_t \end{aligned}$$

where (a) follows from triangle inequality and (b) follows from (AP1), the induction hypothesis, (AP2) and (2.9). This proves the first part of (A.3). The second part follows from the triangle inequality:

$$|V_t^{\pi}(h_t) - \hat{V}_t^{\hat{\pi}}(\hat{\sigma}_t(h_t))| \leq \sum_{a_t \in \mathbf{A}} \hat{\pi}_t(a_t \mid \hat{\sigma}_t(h_t)) |Q_t^{\pi}(h_t, a_t) - \hat{Q}_t^{\hat{\pi}}(\hat{\sigma}_t(h_t), a_t)| \leq \alpha_t.$$

Now, to prove the policy approximation, we note that

$$|Q_t(h_t, a_t) - Q_t^\pi(h_t, a_t)| \leq |Q_t(h_t, a_t) - \hat{Q}_t^{\hat{\pi}}(\hat{\sigma}_t(h_t), a_t)| + |Q_t^\pi(h_t, a_t) - \hat{Q}_t^{\hat{\pi}}(\hat{\sigma}_t(h_t), a_t)| \leq \alpha_t + \alpha_t,$$

where the first inequality follows from the triangle inequality, the first part of the second inequality follows from (2.14) and (A.2) and the second part follows from (A.3). This proves the first part of (2.16). The second part of (2.16) follows from the same argument. \square

Proposition 9. Arbitrarily pick a horizon T and define $\{J_{t,T}^\pi: \mathbf{H}_t \rightarrow \mathbb{R}\}_{t=1}^T$ as follows: $J_{T,T}^\pi(h_T) = 0$ and for $t \in \{T-2, \dots, 1\}$,

$$J_{t,T}^\pi(h_t) := \mathbb{E}^\pi[R_t + \gamma J_{t+1,T}^\pi(H_{t+1}) \mid H_t = h_t]. \quad (2.19)$$

Then, for any time $t \in \{1, \dots, T\}$ and realization h_t of H_t , we have

$$J_{t,T}^\pi(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\min} \leq V_t^\pi(h_t) \leq J_{t,T}^\pi(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}. \quad (2.20)$$

Proof. The proof follows from backward induction. Note that for $t = T$, $R_t \in [R_{\min}, R_{\max}]$ implies that

$$\frac{R_{\min}}{1-\gamma} \leq V_T^\pi(h_T) \leq \frac{R_{\max}}{1-\gamma}.$$

This forms the basis of induction. Now assume that (2.20) holds for time $t+1$ and consider the model for time t :

$$\begin{aligned} V_t^\pi(h_t) &= \mathbb{E}^\pi \left[\sum_{s=t}^{\infty} \gamma^{s-t} R_s \mid H_t = h_t \right] \\ &\stackrel{(a)}{=} \mathbb{E}^\pi \left[R_t + \gamma \mathbb{E}^\pi \left[\sum_{s=t+1}^{\infty} \gamma^{s-(t+1)} R_s \mid H_{t+1} \right] \mid H_t = h_t \right] \\ &\stackrel{(b)}{\leq} \mathbb{E}^\pi \left[R_t + \gamma \mathbb{E}^\pi \left[J_{t+1,T}^\pi(H_{t+1}) + \frac{\gamma^{T-(t+1)}}{1-\gamma} R_{\max} \mid H_{t+1} \right] \mid H_t = h_t \right] \\ &\stackrel{(c)}{=} J_{t,T}^\pi(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}, \end{aligned}$$

where (a) follows from the smoothing property of conditional expectation, (b) follows from the induction hypothesis, and (c) follows from the definition of $J_{t,T}^\pi(\cdot)$. This establishes one

side of (2.20). The other side can be established in a similar manner. Therefore, the result holds by the principle of induction. \square

Proposition 10. Arbitrarily pick a horizon T and define $\{J_{t,T}: \mathbf{H}_t \rightarrow \mathbb{R}\}$ as follows: $J_{T,T}(h_T) = 0$ and for $t \in \{T-2, \dots, 1\}$,

$$J_{t,T}(h_t) := \max_{a_t \in \mathbf{A}} \mathbb{E}[R_t + \gamma J_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t]. \quad (2.23)$$

Then, for any time $t \in \{1, \dots, T\}$ and realization h_t of H_t ,

$$V_t^\pi(h_t) \leq J_{t,T}(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}. \quad (2.24)$$

Therefore,

$$J_{t,T}(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\min} \leq V_t(h_t) \leq J_{t,T}(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}. \quad (2.25)$$

Note that $J_{t,T}(h_t)$ is the optimal value function for a finite horizon system with the discounted reward criterion that runs for horizon $T-1$.

Proof. By following almost the same argument as Proposition 6, we can establish that for any history dependent policy π , $J_{t,T}^\pi(h_t) \leq J_{t,T}(h_t)$, which immediately implies (2.24).

Maximizing the left hand side of (2.24) gives us the upper bound in (2.25). For the lower bound in (2.25), observe that

$$\begin{aligned} V_t(h_t) &= \sup_{\pi} \mathbb{E}^\pi \left[\sum_{s=t}^{\infty} \gamma^{s-t} R_s \mid H_t = h_t \right] \\ &\stackrel{(a)}{\geq} \sup_{\pi} \mathbb{E}^\pi \left[\sum_{s=t}^{T-1} \gamma^{s-t} R_s + \sum_{s=T}^{\infty} \gamma^{s-t} R_{\min} \mid H_t = h_t \right] \\ &= \sup_{\pi} \mathbb{E}^\pi \left[\sum_{s=t}^{T-1} \gamma^{s-t} R_s \mid H_t = h_t \right] + \frac{\gamma^{T-t}}{1-\gamma} R_{\min} \\ &\stackrel{(b)}{=} J_{t,T}(h_t) + \frac{\gamma^{T-t}}{1-\gamma} R_{\min}. \end{aligned}$$

where (a) follows from the fact that $R_s \geq R_{\min}$ and (b) follows from the definition of $J_{t,T}(h_t)$. This complete the proof of (2.25). \square

Theorem 3. Let $\{Z_t\}_{t \geq 1}$ be a time-homogeneous information state process with generator $\{\sigma_t: H_t \rightarrow Z\}_{t \geq 1}$. Suppose Assumption 1 holds and let \bar{V}^* be the unique bounded fixed point of (2.26). Then, for any time t and realization h_t of H_t , we have

$$V_t(h_t) = \bar{V}^*(\sigma_t(h_t)).$$

Furthermore, let $\pi^*: Z \rightarrow \Delta(A)$ be a time-homogeneous (stochastic) policy such that $\text{Supp}(\pi^*(z))$ is a subset of the arg max of the right hand side of (2.26). Then, the time-homogeneous policy $\pi^* := (\pi^*, \pi^*, \dots)$ is optimal.

Proof. Consider the following sequence of value functions: $\bar{V}^{(0)}(z) = 0$ and for $n \geq 0$, define $\bar{V}^{(n+1)} = \mathcal{B}\bar{V}^{(n)}$. Now fix a horizon T and consider the finite-horizon discounted reward problem of horizon $T - 1$. As argued earlier, $J_{t,T}(h_t)$ is the optimal value-function for this finite horizon discounted problem. Moreover, note that $\{Z_t\}_{t=1}^T$ is an information state for this finite horizon discounted problem. Therefore, from using the result of Theorem 1, we get that for any time $t \in \{1, \dots, T\}$, and realization h_t of H_t ,

$$J_{t,T}(h_t) = \bar{V}^{(T-t)}(\sigma_t(h_t)).$$

Substituting (2.25) from Proposition 10 in the above, we get

$$\bar{V}^{(T-t)}(\sigma_t(h_t)) + \frac{\gamma^{T-t}}{1-\gamma} R_{\min} \leq V_t(h_t) \leq \bar{V}^{(T-t)}(\sigma_t(h_t)) + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}.$$

The result follows from taking limit $T \rightarrow \infty$ and observing that $\bar{V}^{(T-t)}(z)$ converges to $\bar{V}^*(z)$. \square

Theorem 4. Suppose $(\{\hat{\sigma}_t\}_{t \geq 1}, \hat{P}, \hat{r})$ is a time-homogeneous (ε, δ) -AIS generator. Consider the fixed point equation (2.29), which we rewrite as follows:

$$\hat{Q}(\hat{z}, a) := \hat{r}(\hat{z}, a) + \int_{\hat{Z}} \hat{V}_{t+1}(\hat{z}_{t+1}) \hat{P}(d\hat{z}_{t+1} \mid \hat{z}, a), \quad (2.30a)$$

$$\hat{V}(\hat{z}) := \max_{a \in A} \hat{Q}(\hat{z}, a). \quad (2.30b)$$

Let \hat{V}^* denote the fixed point of (2.30) and \hat{Q}^* denote the corresponding action-value function. Then, we have the following:

1. **Value function approximation:** For any time t , realization h_t of H_t , and choice a_t of A_t , we have

$$|Q_t(h_t, a_t) - \hat{Q}^*(\hat{\sigma}_t(h_t), a_t)| \leq \alpha \quad \text{and} \quad |V_t(h_t) - \hat{V}^*(\hat{\sigma}_t(h_t))| \leq \alpha, \quad (2.31)$$

where

$$\alpha = \frac{\varepsilon + \gamma \rho_{\mathfrak{F}}(\hat{V}^*)\delta}{1 - \gamma}$$

2. **Approximately optimal policy:** Let $\hat{\pi}^*: \hat{Z} \rightarrow \Delta(\mathbf{A})$ be a stochastic policy that satisfies

$$\text{Supp}(\hat{\pi}^*(\hat{z})) \subseteq \arg \max_{a \in \mathbf{A}} \hat{Q}^*(\hat{z}, a). \quad (2.32)$$

Define policy $\pi = (\pi_1, \pi_2, \dots)$, where $\pi_t: H_t \rightarrow \Delta(\mathbf{A})$ is defined by $\pi_t := \hat{\pi}^* \circ \hat{\sigma}_t$. Then, for any time t , realization h_t of H_t , and choice a_t of A_t , we have

$$|Q_t(h_t, a_t) - Q_t^\pi(h_t, a_t)| \leq 2\alpha \quad \text{and} \quad |V_t(h_t) - V_t^\pi(h_t)| \leq 2\alpha. \quad (2.33)$$

Proof. The proof follows by combining ideas from Theorem 2 and 3. We provide a detailed proof of the value approximation. The proof argument for policy approximation is similar.

Consider the following sequence of value functions: $\hat{V}^{(0)}(\hat{z}) = 0$ and for $n \geq 0$, define $\hat{V}^{(n+1)} = \hat{\mathcal{B}}\hat{V}^{(n)}$. Now fix a horizon T and consider the finite-horizon discounted reward problem of horizon $T - 1$. As argued earlier, $J_{t,T}(h_t)$ is the optimal value-function for this finite horizon discounted problem. Moreover, note that $\{\hat{Z}_t\}_{t=1}^T$ is an (ε, δ) -AIS for this finite horizon discounted problem. Therefore, from using the result of Theorem 2, we get that for any time $t \in \{1, \dots, T\}$, and realization h_t of H_t ,

$$|J_{t,T}(h_t) - \hat{V}^{(T-t)}(\hat{\sigma}_t(h_t))| \leq \alpha_t,$$

where

$$\alpha_t = \varepsilon + \sum_{\tau=t+1}^{T-1} \gamma^{\tau-t} [\rho_{\mathfrak{F}}(\hat{V}^{(T-\tau)})\delta + \varepsilon].$$

Substituting (2.25) from Proposition 10 in the above, we get that

$$\hat{V}^{(T-t)}(\hat{\sigma}_t(h_t)) - \alpha_t + \frac{\gamma^{T-t}}{1-\gamma} R_{\min} \leq V_t(h_t) \leq \hat{V}^{(T-t)}(\hat{\sigma}_t(h_t)) + \alpha_t + \frac{\gamma^{T-t}}{1-\gamma} R_{\max}.$$

Since $\hat{\mathcal{B}}$ is a contraction, from the Banach fixed point theorem we know that $\lim_{T \rightarrow \infty} \hat{V}^{(T-t)} = \hat{V}^*$. Therefore, by continuity of $\rho_{\mathfrak{F}}(\cdot)$, we have $\lim_{T \rightarrow \infty} \rho_{\mathfrak{F}}(\hat{V}^{T-t}) = \rho_{\mathfrak{F}}(\hat{V}^*)$. Consequently, $\lim_{T \rightarrow \infty} \alpha_t = \alpha$. Therefore, taking the limit $T \rightarrow \infty$ in the above equation, we get

$$\hat{V}^*(\hat{\sigma}_t(h_t)) - \alpha \leq V_t(h_t) \leq \hat{V}^*(\hat{\sigma}_t(h_t)) + \alpha,$$

which establishes the bound on the value function in (2.31). The bound on the action-value function in (2.31) follows from a similar argument. \square

Proposition 12. Consider the setup of Proposition 11 for $p = 2$. Suppose ν_ξ is a known parameterized distribution with mean M_ξ and X is a sample from μ . Then, the gradient of

$$(M_\xi - 2X)^\top M_\xi \tag{3.6}$$

with respect to ξ in an unbiased estimator of $\nabla_\xi d_{\mathfrak{F}_2}(\mu, \nu_\xi)^2$.

Proof. For $p = 2$, we have that

$$d_{\mathfrak{F}_2}(\mu, \nu_\xi)^2 = \mathbb{E}[\|X - W\|_2^2] - \frac{1}{2}\mathbb{E}[\|X - X'\|_2^2] - \frac{1}{2}\mathbb{E}[\|W - W'\|_2^2],$$

where $X, X' \sim \mu$ and $W, W' \sim \nu_\xi$. Simplifying the right hand side, we get that

$$d_{\mathfrak{F}_2}(\mu, \nu_\xi)^2 = \|\mathbb{E}[X]\|_2^2 - 2\mathbb{E}[X]^\top \mathbb{E}[W] + \|\mathbb{E}[W]\|_2^2.$$

Note that the term $\|\mathbb{E}[X]\|_2^2$ does not depend on the distribution ν_ξ . Thus, the expression (3.6) captures all the terms which depend on ξ . \square

Appendix B

Implementation Details

B.1 Details about the network architecture, training, and hyperparameters

As explained in Sec. 4, the AIS-generator consists of four components: the history compression function $\hat{\sigma}$, the AIS update function $\hat{\varphi}$, the reward prediction function \hat{r} , and the observation prediction kernel \hat{P}^y . We model the first as an LSTM, where the memory update unit of LSTM acts as $\hat{\varphi}$. We model \hat{r} , \hat{P}^y , and the policy $\hat{\pi}$ as feed-forward neural networks. We describe the details for each difficulty class of environment separately. In the description below, we use $\text{Linear}(n, m)$ to denote a linear layer $\text{Tanh}(n, m)$ to denote a tanh layer, $\text{ReLU}(n, m)$ to denote a ReLU layer, and $\text{LSTM}(n, m)$ to denote an LSTM layer, where n denotes the number of inputs and m denotes the number of outputs of each layer. The size of the input of the outputs depend on the size of the observation and action spaces, which we denote by n_O and n_A , respectively as well as on the dimension of AIS and for the case of minigrid environments, the dimension of the latent space for observations, we denote by $d_{\hat{z}}$ and d_O . We also use $\text{Conv2d}(IC, OC, (FSx, FSy))$ to denote a 2D convolutional layer with IC , OC , (FSx, FSy) represent the number of input channels, output channels and kernel size (along x and y) respectively. Note that the strides are the same as the kernel size in this case. ELU represents Exponential Linear Unit and is used to model the prediction of variance. Finally, $\text{GMM}(n_{\text{comp}})$ represents a Gaussian Mixture Model with n_{comp} Gaussian components. Most of the details are common for both the AIS+KL and the AIS+MMD cases, we make a distinction whenever they are different.

B.1.1 Details for low dimensional environments:

• ENVIRONMENT DETAILS:

Environment	Discount	No. of actions	No. of obs.
	γ	n_A	n_O
VOICEMAIL	0.95	3	2
TIGER	0.95	3	2
CHEESEMAZE	0.7	4	7

The discount factor for CHEESEMAZE is chosen to match with standard value used in that environment [14].

• AIS AND NETWORK DETAILS:

- Dimensions of AIS ($d_{\hat{z}}$) : 40
- Weight in AIS loss (λ) (KL) : 0.0001
- Weight in AIS loss (λ) (MMD) : 0.001

$\hat{\sigma}$	\hat{r}	\hat{P}^y	$\hat{\pi}$
Linear($n_O + n_A + 1, d_{\hat{z}}$)	Linear($n_A + d_{\hat{z}}, \frac{1}{2}d_{\hat{z}}$)	Linear($n_A + d_{\hat{z}}, \frac{1}{2}d_{\hat{z}}$)	Linear($d_{\hat{z}}, d_{\hat{z}}$)
\Downarrow	\Downarrow	\Downarrow	\Downarrow
Tanh($d_{\hat{z}}, d_{\hat{z}}$)	Tanh($\frac{1}{2}d_{\hat{z}}, \frac{1}{2}d_{\hat{z}}$)	Tanh($\frac{1}{2}d_{\hat{z}}, \frac{1}{2}d_{\hat{z}}$)	Tanh($d_{\hat{z}}, d_{\hat{z}}$)
\Downarrow	\Downarrow	\Downarrow	\Downarrow
LSTM($d_{\hat{z}}, d_{\hat{z}}$)	Linear($\frac{1}{2}d_{\hat{z}}, 1$)	Linear($\frac{1}{2}d_{\hat{z}}, n_O$)	Linear($d_{\hat{z}}, n_A$)
		\Downarrow	\Downarrow
		Softmax	Softmax

- TRAINING DETAILS: As explained in Section 3.3, we update the parameters after a rollout of T , which we call a *training batch*. The choice of parameters for the training batch are as follows:

- Samples per training batch : 200
- Number of training batches : 10^5

In addition, we use the following learning rates:

- AIS learning rate : ADAM(0.003)
- Policy learning rate (KL) : ADAM(0.0006)
- Policy learning rate (MMD) : ADAM(0.0008)

In the above description, we use ADAM(α) to denote the choice of α parameter of ADAM. All other parameters have their default value.

- EVALUATION DETAILS:
 - No. of batches after which evaluation is done : 500
 - Number of rollouts per evaluation : 50

B.1.2 Details for moderate dimensional environments:

- ENVIRONMENT DETAILS:

Environment	Discount	No. of actions	No. of obs.
	γ	n_A	n_O
DRONE SURVEILLANCE	0.99	5	10
ROCK SAMPLING	0.99	8	3

- AIS AND NETWORK DETAILS:

- Dimensions of AIS ($d_{\hat{Z}}$) : 128
- Weight in AIS loss (λ) (KL) : 0.0001
- Weight in AIS loss (λ) (MMD) : 0.001

$\hat{\sigma}$	\hat{r}	\hat{P}^y	$\hat{\pi}$
LSTM($n_O + n_A + 1, d_{\hat{Z}}$)	Linear($n_A + d_{\hat{Z}}, \frac{1}{2}d_{\hat{Z}}$)	Linear($n_A + d_{\hat{Z}}, \frac{1}{2}d_{\hat{Z}}$)	Linear($d_{\hat{Z}}, n_A$)
	\Downarrow	\Downarrow	\Downarrow
	ReLU($\frac{1}{2}d_{\hat{Z}}, \frac{1}{2}d_{\hat{Z}}$)	ReLU($\frac{1}{2}d_{\hat{Z}}, \frac{1}{2}d_{\hat{Z}}$)	Softmax
	\Downarrow	\Downarrow	
	Linear($\frac{1}{2}d_{\hat{Z}}, 1$)	Linear($\frac{1}{2}d_{\hat{Z}}, n_O$)	
		\Downarrow	
		Softmax	

- **TRAINING DETAILS:** As explained in Section 3.3, we update the parameters after a rollout of T , which we call a *training batch*. The choice of parameters for the training batch are as follows:
 - Samples per training batch : 200
 - Number of training batches : 10^5

In addition, we use the following learning rates:

- AIS learning rate : ADAM(0.003)
- Policy learning rate : ADAM(0.0007)

In the above description, we use ADAM(α) to denote the choice of α parameter of ADAM. All other parameters have their default value.

- **EVALUATION DETAILS:**
 - No. of batches after which evaluation is done : 500
 - Number of rollouts per evaluation : 100

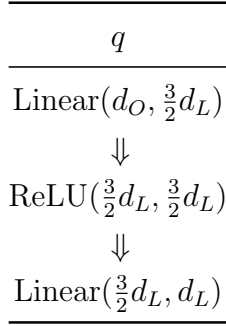
B.1.3 Details for high dimensional environments:

- **ENVIRONMENT DETAILS:**

Note that here n_O represents the number of possible observations that a general minigrid environment can have. With the actual rules of the environment plugged in, this number is smaller since some combinations of the encoded observation are not possible. The actual input that we get from the environment is a vector of size 147 (d_O) which is basically an observation grid of 7×7 with 3 channels containing characteristic information about the observation.

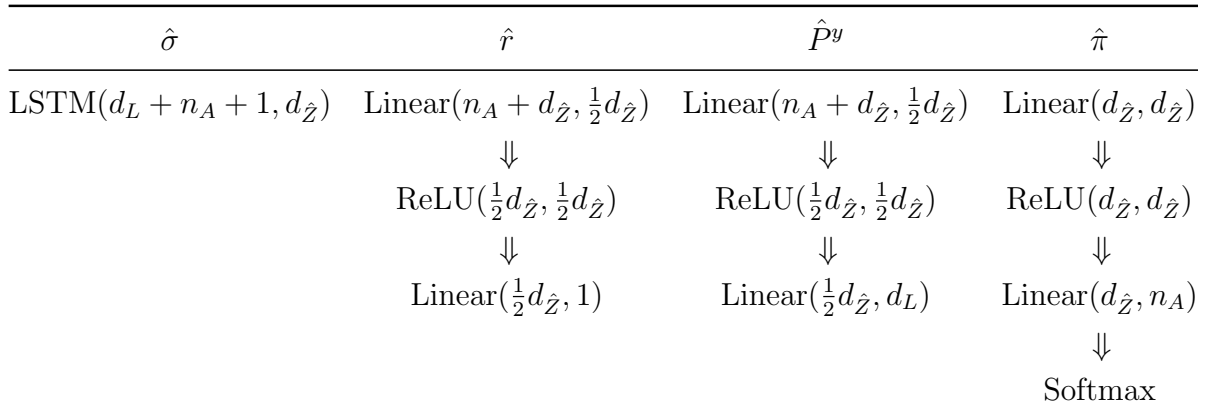
Environment	Discount	No. of actions	No. of obs.	Obs. dimen.
	γ	n_A	n_O	d_O
MINIGRID ENVS	0.99	7	$(6 \times 11 \times 3)^{7 \times 7}$	$7 \times 7 \times 3$

- **AUTOENCODER (q) DETAILS:**
 - Latent space dimensions (d_L) : 64
 - Type of autoencoder used : Basic autoencoder
 - Reconstruction Loss Criterion Used : Mean Square Error

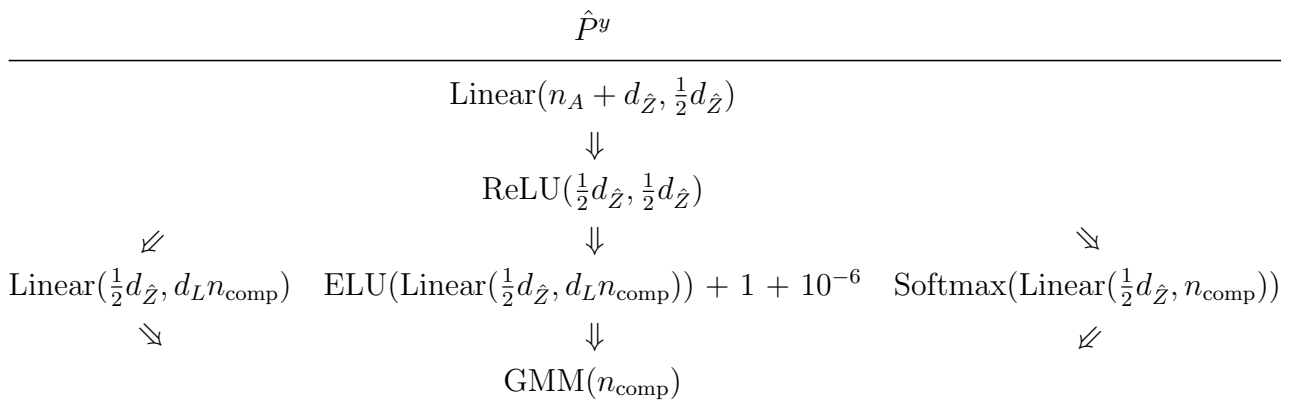


• AIS AND NETWORK DETAILS:

- Dimensions of AIS ($d_{\hat{z}}$) : 128
- Weight in AIS loss (λ) : 0.1
- Number of GMM components used (n_{comp}) (only for KL) : 5



For KL, \hat{P}^y is replaced by the following while other networks remain the same:



Note that the third layer here is for the mean vector of each component, the diagonal vector for variance of each component and the mixture weights of each component of the GMM respectively. This goes into the GMM probability model.

- **TRAINING DETAILS:** As explained in Section 3.3, we update the parameters after a rollout of T , which we call a *training batch*. The choice of parameters for the training batch are as follows:
 - Samples per training batch : 200
 - Number of training batches : 2×10^5 (MGKCS3R3, MGOM1Dl, MGOM1Dlh)
 10^5 (others)

In addition, we use the following learning rates:

- AIS learning rate : ADAM(0.001)
- Policy learning rate : ADAM(0.0007)

In the above description, we use ADAM(α) to denote the choice of α parameter of ADAM. All other parameters have their default value.

- **EVALUATION DETAILS:**
 - No. of batches after which evaluation is done : 5000 (MGKCS3R3, MGOM1Dl, MGOM1Dlh)
1000 (others)
 - Number of rollouts per evaluation : 20

B.1.4 Details for PPO with LSTM and Critic:

- **ENVIRONMENT DETAILS:**

The environment details are the same as mentioned previously.

- **NETWORK DETAILS:**

- Low and moderate dimensionality environments:

Feature Extractor	Actor Head	Critic Head
LSTM(n_O, n_O)	Linear($n_O, 64$)	Linear($n_O, 64$)
	↓	↓
	Tanh(64, 64)	Tanh(64, 64)
	↓	↓
	Linear(64, n_A)	Linear(64, 1)
	↓	
	Softmax	

– High dimensionality environments:

- Observation tensor : $7 \times 7 \times 3$
- Embedding size (d_E) : 64

Conv. Feature Extractor	Actor Head	Critic Head
Conv2d($3, \frac{1}{4}d_E, (2, 2)$)	Linear(d_E, d_E)	Linear(d_E, d_E)
↓	↓	↓
ReLU	Tanh(d_E, d_E)	Tanh(d_E, d_E)
↓	↓	↓
MaxPool2d	Linear(d_E, n_A)	Linear($d_E, 1$)
↓	↓	
Conv2d($\frac{1}{4}d_E, \frac{1}{2}d_E, (2, 2)$)	Softmax	
↓		
ReLU		
↓		
Conv2d($\frac{1}{2}d_E, d_E, (2, 2)$)		
↓		
ReLU		
↓		
LSTM(d_E, d_E)		

• TRAINING DETAILS:

- Number of parallel actors : 64
- Number of training batches : 4×10^7 (MGKCS3R3, MGOM1Dl, MGOM1Dlh)
 2×10^7 (others)
- Epochs per training batch : 4
- Samples per training batch : 1280
- Frames per parallel actor : 40
- GAE (λ_{GAE}) : 0.99
- Trajectory recurrence length : 20

In addition, we use ADAM with the following details:

- Learning rate α : 0.0001
- ADAM parameter ϵ : 0.00001

- EVALUATION DETAILS:
 - No. of batches after which evaluation is done : 200
 - Rollouts used for evaluation : All recent episodes completed by all actors

B.1.5 Details about hyperparameter tuning

Hyperparameter tuning was carried out by searching a grid of values, but exhaustive grid search was not carried out due to the prohibitive computational cost. Instead, coarse values were used initially as starting points and finer tuning was done around promising values, which was essentially an iterative process of performing experiments, observing results and trying similar parameters to the ones generating good results. Hyperparameters observed in each previous environment class (low, moderate, high dimensionality) were used as a starting point for the search in the new environment class.

Performance was quite sensitive to different learning rates used for the AIS and policy in most environments. Performance generally improved or remained the same when a larger AIS State Size was used (values considered were 128, 256, 512 for moderate/high-dimensional environments and 5, 10, 20, 40 for low-dimensional environments), although in some cases, it was more unstable during training. λ values considered were between 0 and 1 and generally only made a difference (in terms of performance results) when the rewards were very large. The choice of activation function between ReLU and Tanh did not seem to make a significant difference for the considered environments.

References

- [1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [2] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in *Neural Information Processing Systems (NIPS)*, pp. 1057–1063, Nov. 2000.
- [3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [4] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, “Rainbow: Combining improvements in deep reinforcement learning,” in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [5] O. Hernández-Lerma and J. B. Lasserre, *Discrete-time Markov control processes: basic optimality criteria*. Springer, 2012.
- [6] D. Ha and J. Schmidhuber, “World models,” *arXiv preprint arXiv:1803.10122*, 2018.
- [7] A. Zhang, Z. C. Lipton, L. Pineda, K. Azizzadenesheli, A. Anandkumar, L. Itti, J. Pineau, and T. Furlanello, “Learning causal state representations of partially observable environments,” *arXiv preprint arXiv:1906.10437*, 2019.
- [8] J. Pineau, G. Gordon, S. Thrun, *et al.*, “Point-based value iteration: An anytime algorithm for pomdps,” in *IJCAI*, vol. 3, pp. 1025–1032, 2003.
- [9] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, “Recurrent policy gradients,” *Logic Journal of the IGPL*, vol. 18, no. 5, pp. 620–634, 2010.
- [10] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, “Recurrent experience replay in distributed reinforcement learning,” in *International conference on learning representations*, 2018.

-
- [11] A. Nayyar, A. Mahajan, and D. Teneketzis, “Decentralized stochastic control with partial history sharing: A common information approach,” *IEEE Transactions on Automatic Control*, vol. 58, no. 7, pp. 1644–1658, 2013.
 - [12] J. Subramanian and A. Mahajan, “Approximate information state for partially observed systems,” in *Conference of Decision and Control (CDC), Nice, France*, 2019.
 - [13] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.
 - [14] R. A. McCallum, “Overcoming incomplete perception with utile distinction memory,” in *International Conference on Machine Learning (ICML)*, 1993.
 - [15] M. Chevalier-Boisvert, L. Willems, and S. Pal, “Minimalistic gridworld environment for openai gym.” <https://github.com/maximecb/gym-minigrid>, 2018.
 - [16] P. R. Kumar and P. Varaiya, *Stochastic systems: Estimation, identification, and adaptive control*. SIAM, 2015.
 - [17] M. L. Puterman, “Markov decision processes,” *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.
 - [18] D. P. Bertsekas, D. P. Bertsekas, D. P. Bertsekas, and D. P. Bertsekas, *Dynamic programming and optimal control*, vol. 1. Athena scientific Belmont, MA, 1995.
 - [19] V. Krishnamurthy, *Partially observed Markov decision processes*. Cambridge University Press, 2016.
 - [20] A. R. Cassandra, M. L. Littman, and N. L. Zhang, “Incremental pruning: A simple, fast, exact method for partially observable markov decision processes,” *arXiv preprint arXiv:1302.1525*, 2013.
 - [21] H. Kurniawati, D. Hsu, and W. S. Lee, “Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces,” in *Robotics: Science and systems*, vol. 2008, Zurich, Switzerland., 2008.
 - [22] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, “Learning policies for partially observable environments: Scaling up,” in *Machine Learning Proceedings 1995*, pp. 362–370, Elsevier, 1995.
 - [23] M. Egorov, Z. N. Sunberg, E. Balaban, T. A. Wheeler, J. K. Gupta, and M. J. Kochenderfer, “POMDPs.jl: A framework for sequential decision making under uncertainty,” *Journal of Machine Learning Research*, vol. 18, no. 26, pp. 1–5, 2017.

-
- [24] D. Silver and J. Veness, “Monte-carlo planning in large pomdps,” *Advances in neural information processing systems*, vol. 23, pp. 2164–2172, 2010.
 - [25] A. Somani, N. Ye, D. Hsu, and W. S. Lee, “Despot: Online pomdp planning with regularization,” *Advances in neural information processing systems*, vol. 26, pp. 1772–1780, 2013.
 - [26] H. Bai, D. Hsu, and W. S. Lee, “Integrated perception and planning in the continuous space: A pomdp approach,” *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1288–1302, 2014.
 - [27] S. Ross, B. Chaib-Draa, *et al.*, “Aems: An anytime online search algorithm for approximate policy refinement in large pomdps,” in *IJCAI*, pp. 2592–2598, 2007.
 - [28] G. Shani, J. Pineau, and R. Kaplow, “A survey of point-based pomdp solvers,” *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.
 - [29] D. P. Bertsekas, *Approximate dynamic programming*. Athena scientific Belmont, 2012.
 - [30] D. Wierstra, A. Förster, J. Peters, and J. Schmidhuber, “Recurrent policy gradients,” *Logic Journal of the IGPL*, vol. 18, no. 5, pp. 620–634, 2010.
 - [31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
 - [32] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, “The arcade learning environment: An evaluation platform for general agents,” *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.
 - [33] M. Hausknecht and P. Stone, “Deep recurrent q-learning for partially observable mdps,” in *2015 AAAI Fall Symposium Series*, 2015.
 - [34] C. Beattie, J. Z. Leibo, D. Teplyaev, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, *et al.*, “Deepmind lab,” *arXiv preprint arXiv:1612.03801*, 2016.
 - [35] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson, “Learning latent dynamics for planning from pixels,” in *International Conference on Machine Learning*, pp. 2555–2565, PMLR, 2019.
 - [36] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, “Dream to control: Learning behaviors by latent imagination,” *arXiv preprint arXiv:1912.01603*, 2019.
 - [37] N. Ferns, P. Panangaden, and D. Precup, “Metrics for finite Markov decision processes,” in *Conference on Uncertainty in Artificial Intelligence (UAI)*, pp. 162–169, AUAI Press, 2004.

- [38] N. Ferns, P. Panangaden, and D. Precup, “Bisimulation metrics for continuous Markov decision processes,” *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1662–1714, 2011.
- [39] P. S. Castro, P. Panangaden, and D. Precup, “Equivalence relations in fully and partially observable Markov decision processes,” in *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1653–1658, 2009.
- [40] M. L. Littman, R. S. Sutton, and S. P. Singh, “Predictive representations of state,” in *Neural Information Processing Systems (NIPS)*, 2002.
- [41] S. P. Singh, M. L. Littman, N. K. Jong, D. Pardoe, and P. Stone, “Learning predictive state representations,” in *International Conference on Machine Learning (ICML)*, 2003.
- [42] M. James, S. Singh, and M. Littman, “Planning with predictive state representations,” in *International Conference on Machine Learning and Applications (ICMLA)*, 2004.
- [43] M. Rosencrantz, G. Gordon, and S. Thrun, “Learning low dimensional predictive representations,” in *International Conference on Machine Learning (ICML)*, 2004.
- [44] B. Boots, S. M. Siddiqi, and G. J. Gordon, “Closing the learning-planning loop with predictive state representations,” *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 954–966, 2011.
- [45] W. Hamilton, M. M. Fard, and J. Pineau, “Efficient learning and planning with compressed predictive states,” *The Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 3395–3439, 2014.
- [46] A. Kulesza, N. Jiang, and S. P. Singh, “Spectral learning of predictive state representations with insufficient statistics,” in *AAAI Conference on Artificial Intelligence*, pp. 2715–2721, 2015.
- [47] A. Kulesza, N. Jiang, and S. Singh, “Low-rank spectral learning with weighted loss functions,” in *Artificial Intelligence and Statistics*, pp. 517–525, 2015.
- [48] N. Jiang, A. Kulesza, and S. P. Singh, “Improving predictive state representations via gradient descent,” in *AAAI Conference on Artificial Intelligence*, pp. 1709–1715, 2016.
- [49] B. Wolfe, M. R. James, and S. Singh, “Approximate predictive state representations,” in *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 363–370, 2008.
- [50] J. Baxter and P. L. Bartlett, “Infinite-horizon policy-gradient estimation,” *Journal of Artificial Intelligence Research*, vol. 15, pp. 319–350, 2001.

-
- [51] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio, “Babyai: First steps towards grounded language learning with a human in the loop,” *arXiv preprint arXiv:1810.08272*, 2018.
 - [52] J. Subramanian, A. Sinha, R. Seraj, and A. Mahajan, “Approximate information state for approximate planning and reinforcement learning in partially observed systems,” *arXiv preprint arXiv:2010.08843*, 2020.
 - [53] N. Vlassis, M. L. Littman, and D. Barber, “On the computational complexity of stochastic controller optimization in pomdps,” *ACM Transactions on Computation Theory (TOCT)*, vol. 4, no. 4, pp. 1–8, 2012.
 - [54] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender, “Complexity of finite-horizon markov decision process problems,” *Journal of the ACM (JACM)*, vol. 47, no. 4, pp. 681–720, 2000.
 - [55] A. Müller, “Integral probability metrics and their generating classes of functions,” *Advances in Applied Probability*, vol. 29, no. 2, pp. 429–443, 1997.
 - [56] V. M. Zolotarev, “Probability metrics,” *Theory of Probability & Its Applications*, vol. 28, pp. 278–302, Jan. 1983.
 - [57] S. T. Rachev, *Probability Metrics and the Stability of Stochastic Models*. Wiley, 1991.
 - [58] B. K. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. R. G. Lanckriet, “On the empirical estimation of integral probability metrics,” *Electronic Journal of Statistics*, vol. 6, no. 0, pp. 1550–1599, 2012.
 - [59] C. Villani, *Optimal transport: Old and New*. Springer, 2008.
 - [60] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, “A kernel method for the two-sample-problem,” in *Neural Information Processing Systems (NIPS)*, p. 513–520, 2006.
 - [61] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, G. R. G. Lanckriet, and B. Schölkopf, “Injective Hilbert space embeddings of probability measures,” in *Conference on Learning Theory*, 2008.
 - [62] G. J. Székely and M. L. Rizzo, “Testing for equal distributions in high dimensions,” *InterStat*, no. 5, 2004.
 - [63] D. Sejdinovic, B. Sriperumbudur, A. Gretton, and K. Fukumizu, “Equivalence of distance-based and RKHS-based statistics in hypothesis testing,” *The Annals of Statistics*, vol. 41, pp. 2263–2291, Oct. 2013.

-
- [64] V. S. Borkar, “Stochastic approximation with two time scales,” *Systems & Control Letters*, vol. 29, no. 5, pp. 291–294, 1997.
 - [65] I. Csiszar and J. Körner, *Information theory: coding theorems for discrete memoryless systems*. Cambridge University Press, 2011.
 - [66] V. R. Konda and J. N. Tsitsiklis, “On actor-critic algorithms,” *SIAM Journal on Control and Optimization*, vol. 42, no. 4, pp. 1143–1166, 2003.
 - [67] D. S. Leslie, *Reinforcement learning in games*. PhD thesis, The University of Bristol, 2004.
 - [68] B. Bakker, “Reinforcement learning with long short-term memory,” in *Neural Information Processing Systems (NIPS)*, 2002.
 - [69] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber, “Solving deep memory POMDPs with recurrent policy gradients,” in *International Conference on Artificial Neural Networks (ICANN)*, 2007.
 - [70] M. Hausknecht and P. Stone, “Deep recurrent Q-learning for partially observable MDPs,” in *AAAI Fall Symposium Series*, 2015.
 - [71] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, “Memory-based control with recurrent neural networks,” 2015. arXiv:1512.04455.
 - [72] P. Zhu, X. Li, P. Poupart, and G. Miao, “On improving deep reinforcement learning for POMDPs,” 2017. arXiv:1704.07978.
 - [73] A. Baisero and C. Amato, “Learning internal state models in partially observable environments;,” *Reinforcement Learning under Partial Observability, NeurIPS Workshop*, 2018.
 - [74] M. Igl, L. Zintgraf, T. A. Le, F. Wood, and S. Whiteson, “Deep variational reinforcement learning for POMDPs,” in *International Conference on Machine Learning (ICML)*, pp. 2117–2126, 2018.
 - [75] J. Subramanian, A. Sinha, R. Seraj, and A. Mahajan, “Approximate information state for reinforcement learning in partially observed systems.” <https://github.com/info-structures/ais>, 2020.
 - [76] J. D. Williams and S. Young, “Partially observable Markov decision processes for spoken dialog systems,” *Computer Speech & Language*, vol. 21, no. 2, pp. 393–422, 2007.

-
- [77] T. Smith and R. Simmons, “Heuristic search value iteration for POMDPs,” in *UA*, pp. 520–527, 2004.
 - [78] M. Svoreňová, M. Chmelík, K. Leahy, H. F. Eniser, K. Chatterjee, I. Černá, and C. Belta, “Temporal logic motion planning using POMDPs with parity objectives: Case study paper,” in *International Conference on Hybrid Systems: Computation and Control*, p. 233–238, 2015.
 - [79] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5026–5033, IEEE, 2012.
 - [80] R. Seraj, *Learning in the presence of partial observability and concept drifts*. PhD thesis, McGill University Libraries, 2019.
 - [81] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, “Acting optimally in partially observable stochastic domains,” in *AAAI*, vol. 94, pp. 1023–1028, 1994.
 - [82] R. J. Williams and J. Peng, “Function optimization using connectionist reinforcement learning algorithms,” *Connection Science*, vol. 3, no. 3, pp. 241–268, 1991.
 - [83] V. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.