Reinforcement learning in partially observed and multi-agent systems

Jayakumar Subramanian

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

April 2020

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Doctor of Philosophy.

Abstract

In this thesis we investigate the problem of reinforcement learning in partially observed and multi-agent systems. The belief state, which is the most common information state used in planning for partially observed systems, is not directly applicable in reinforcement learning (RL). This is because one does not know the observation likelihood in order to recursively compute the belief state from the observations in RL. We define an approximate information state (AIS), which can be learned from observational data alone and hence is useful for RL. We present two variants of AIS—one that is sufficient for prediction of the per-step reward and next AIS distribution, and one that evolves in a state-like manner and is sufficient for predicting the per-step reward and the next observation distribution. For all variants, we derive error bounds in using AIS and also some existing approximation results in literature using the AIS concept. We present a (deep) RL algorithm that uses AIS, prove its convergence under standard technical assumptions and demonstrate its performance versus one of the state of the art approaches in four toy examples.

We then extend this concept of AIS to develop an RL algorithm for a class of large population cooperative multi-agent systems called mean-field teams. A contemporaneous result in literature is that the performance bound for such mean-field teams when using an infinite population assumption is $O(\frac{1}{\sqrt{n}})$. We derive this bound in terms of the AIS concept. Depending on the nature of the simulator available for RL, i.e, a complete system-level simulator or an agent-level simulator, we present two (deep) RL algorithms and demonstrate their performance using two examples.

Next, we consider (deep) RL in another class of large population systems called stationary mean-field games. The planning solution for such systems has been presented in literature. We present (deep) RL algorithms for these systems and also define two generalizations of the original equilibrium solution concepts for reinforcement learning. We prove the convergence of the algorithms and also demonstrate their performance using two examples.

Finally, we present an RL algorithm based on using concepts from renewal theory called Renewal Monte Carlo. We theoretically prove the convergence of this algorithm and numerically demonstrate its efficiency in problems with renewals. Though we have presented results for a single agent case, these can be extended to policy evaluation (as a low bias and low variance alternative to Monte Carlo and temporal difference (TD(λ) methods) in any dynamic programming decomposition that has this renewal structure.

Résumé

Dans cette thèse, nous étudions le problème d'apprentissage par renforcement dans les systèmes partiellement observés et multi-agents. L'état de croyance, qui est l'état d'information le plus couramment utilisé dans la planification de systèmes partiellement observés, n'est pas directement applicable dans l'apprentissage par renforcement (RL). En effet, on ne connaît pas la probabilité d'observation afin de calculer récursivement l'état de croyance à partir des observations dans RL. Nous définissons un état d'information approximatif (AIS), qui peut être appris à partir de données d'observation uniquement et est donc utile pour RL. Nous présentons deux variantes d'AIS — une qui est suffisante pour prédire la récompense par étape et la prochaine distribution AIS, et une qui évolue de manière semblable à un état et est suffisante pour prédire la récompense par étape et la distribution de la prochaine observation. Pour toutes les variantes, nous dérivons des limites d'erreur en utilisant AIS et également certains résultats d'approximation existants dans la littérature utilisant le concept de AIS. Nous présentons un algorithme RL (profond) qui utilise AIS, prouvons sa convergence sous des présomptions classiques et démontrons ses performances par rapport à l'une des approches de pointe sur quatre tâches.

Nous étendons ensuite ce concept d'AIS pour développer un algorithme RL pour une classe de systèmes multi-agents coopératifs à grande population appelés équipes de champ moyen. Un résultat contemporain dans la littérature est que la performance liée à de telles équipes de champ moyen lors de l'utilisation d'une hypothèse de population infinie est $O(\frac{1}{\sqrt{n}})$. Nous dérivons cette limite en termes de concept AIS. Selon la nature du simulateur disponible pour RL, c'est-à-dire un simulateur complet au niveau système ou un simulateur au niveau agent, nous présentons deux algorithmes RL (profonds) et démontrons leurs performances à l'aide de deux exemples.

Ensuite, nous considérons le RL (profond) dans une autre classe de systèmes à grande population appelés jeux stationnaires à champ moyen. La solution de planification pour de tels systèmes a été présentée dans la littérature. Nous présentons des algorithmes RL (profonds) pour ces systèmes et définissons également deux généralisations des concepts de solution d'équilibre originaux pour l'apprentissage par renforcement. Nous prouvons la convergence des algorithmes et démontrons également leurs performances à l'aide de deux exemples.

Finalement, nous présentons un algorithme RL basé sur l'utilisation de concepts de la

théorie du renouvellement appelés Renewal Monte Carlo. Nous prouvons théoriquement la convergence de cet algorithme et démontrons numériquement son efficacité dans les problèmes de renouvellement. Bien que nous ayons présenté des résultats pour un cas d'agent unique, ceux-ci peuvent être étendus à l'évaluation des politiques (comme alternative à faible biais et faible variance à Monte Carlo et à la différence temporelle (méthodes TD(λ)) dans n'importe quelle décomposition de programmation dynamique qui a cette structure de renouvellement.

Acknowledgments

I first wish to thank my supervisor, Prof. Aditya Mahajan, for all the guidance, support, inspiration and encouragement that he has given me during my stint at McGill University. I re-entered academia after around 10 years of completion of my post graduate studies, and this has been a wonderful journey of four years thanks primarily to Prof. Mahajan. Both as an instructor for the course on stochastic control that I attended and as my supervisor, his enthusiasm for the subject, desire for precision and clear communication have made a huge impact on my thinking. I am extremely grateful to him for his belief in me and his patience with me throughout the course of my study. It has been my great honor and privilege to have worked under his guidance.

I wish to thank my friend Dr. Aditya Paranjape for encouraging me to pursue doctoral studies and also introducing me to Prof. Aditya Mahajan and McGill. His advice and research collaboration have been very valuable to me. I wish to thank my professors at IIT Bombay, especially Prof. Mandal and Prof. Shimpi, for their continued support and guidance.

I would like to take this opportunity to thank McGill University for the excellent research atmosphere and also the financial support extended to me through the McGill Engineering Doctoral Award. I am also grateful for the excellent computational facilities available at McGill and those provided by Calcul Quebec, Compute Canada etc. Montreal's attractive and multi-cultural environment, while being one of the world's leading centers of machine learning, definitely contributed a lot to my experience.

At McGill, it has been my good fortune to learn from Prof. Peter Caines, who taught me a course on filtering and prediction for stochastic systems and was also part of my supervisory committee. The various formal and informal discussions I have had with him will remain forever invaluable to me. Prof. Joelle Pineau has also been a great source of research support and encouragement. As part of my supervisory committee, I have gained a lot from her advice and suggestions in improving my work, especially in the work on use of renewal theory in reinforcement learning.

I wish to thank all the students in the Systems and Control lab at McGill for being great friends throughout the course of my studies. Specifically, I wish to acknowledge the research discussions and support I received from Jhelum Chakravorty and Mohammad Afshari. In addition, they formed a key part of my social life in Montreal along with Ali

Pakniyat, Shuang Gao and Hamed Layeghi. I wish to thank Raihan Seraj for being an excellent collaborator and friend. I remember fondly our many discussions and time spent in coding examples and revising articles. Debarshi Ghoshal (Gogol) has been a great friend and confidant and the long discussions with him on various topics have enriched my stay in Montreal. I am also very grateful to my apartment mate Gandharv Patil for varied research discussions and collaborations and also for teaching me some handy cooking skills! I wish to thank Amit Sinha for discussions on approximate information state and am grateful to him for providing me with some of the numerical results for my thesis. I also wish to thank all my other friends from my lab and from the RL lab at McGill—Pierre-Luc Bacon, Riashat Islam, Sameen Yeasar Arnob, Nima Akbarzadeh, Borna Sayedana, Kushal Arora, Khimya Khetarpal, Shagun Sodhani, Thang Long Doan, Bogdan Mazoure, Fatih Gurturk for providing a very friendly and intellectually stimulating work environment. I am grateful to Bogdan for helping me with the French translation of my abstract. I am greatly indebted to Shagun for both the research and general conversations that we have had. Interacting with him has been a great pleasure and my only regret is that we met fairly late in the course of my study. Harm van Seijen and Mehdi Fatemi from Microsoft Research Montreal also taught me several aspects of research both from an academic and industrial research perspective and I am very thankful to them. I also with to thank Prof. Doina Precup, Prof. Mike Rabbat and Prof. Akshat Kumar for sparing the time for several research discussions and for exploring potential collaboration topics.

I have tried to recollect and acknowledge all the people who have helped me in my doctoral studies. I am sure I may have inadvertently missed a few names and for that I apologize in advance. Though I may have forgotten to mention their names, I have not forgotten their contribution to my study and will be grateful for that.

Finally I would like to thank my family—my parents, my sister Kaveri, my brother-in-law Venkat and my nieces Sindhuja and Shailaja for being there for me always. Without their support, my doctoral study would not have been possible. Their belief in me and the belief and support of my friends and former colleagues Sridhar Vaidyanath, Supratim Banerjee and Shalini Banerjee have helped me persevere through several difficult situations in my study. Though they probably know this, I wish to thank them again from the bottom of my heart.

This page is intentionally left blank.

Contents

List of Figures

List of Acronyms

MAS Multi-agent systems

RL Reinforcement learning

TD Temporal difference

MARL Multi-agent reinforcement learning

MDP Markov decision processes

MFT Mean-field teams

POMDP Partially observable Markov decision processes

Dec-POMDP Decentralized partially observable Markov decision processes

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

GRU Gated Recurrent Unit

AIS Approximate information state

RMC Renewal Monte Carlo

DP Dynamic programming

ADP Approximate dynamic programming

IPM Integral probability metric

This page is intentionally left blank.

Chapter 1

Introduction

1.1 Motivation

With the recent advances in machine learning and automation, we increasingly find ourselves immersed in a world with multiple agents that learn continuously and adapt autonomously to the environment to achieve their objectives. Examples of such multi-agent systems (MAS) include Internet of Things, self-driving cars, swarm robotics, teams of personalized recommenders, smart grids, among others. The objective of each agent in such systems is to choose a policy to make sequential decisions to maximize some objective over time. A fundamental question in such systems is the following: how should agents be designed so that they achieve optimal results? It is known in the literature that learning algorithms for single agent systems do not work in multi-agent settings. The reason is that in a multi-agent system, each agent is operating in an environment which depends on the behavior of other agents. The presence of multiple agents with potentially conflicting objectives renders finding an optimal policy for each agent in a MAS a difficult problem.

Examples of works in literature that address the problem of an agent learning to act optimally in a multi-agent system include [87, 89] among others. These works assume that the agents have complete knowledge of the system, i.e., each agent knows the dynamical evolution of the system and the reward function/utilities of all the agents in the system. We refer to these problems as planning in multi-agent systems. As opposed to this, in this thesis, we consider the problem of agents learning to act optimally in a multi-agent system, where the agents do not know the system dynamics nor the reward functions/utilities of themselves or the other agents. We refer to this problem as reinforcement learning (RL) in

multi-agent systems, often referred to in literature as multi-agent reinforcement learning (MARL). An overview of several approaches for MARL can be found in [17, 21, 46, 141]

In several modern multi-agent systems, especially the ones that involve human interaction, one does not know the system dynamics or the reward functions. Hence, finding an optimal policy in such systems is a problem of reinforcement learning in multi-agent systems rather than planning in multi-agent systems. In this thesis, we study reinforcement learning for specific classes of multi-agent systems, identify the key problems and present reinforcement learning algorithms for these classes of systems. In the process, we also address the related issues of reinforcement learning in single agent systems with partial observability and in fully observed systems which exhibit renewal/regenerative behavior.

The structure of the rest of this chapter is as follows. In the next section, we provide an overview of MARL. We start by describing MAS along with their salient features. We then discuss MARL and two specific problems in MARL—partial observability and agent coupling in the following two sub-sections. Subsequently we describe three classes of systems (single and multi-agent) that we study in this thesis. In the following two sections we provide the scope and organization of the remainder of this thesis. We finally discuss the claims of originality, publications and contribution of co-authors in the work described in this thesis.

1.2 An overview of multi-agent reinforcement learning (MARL)

1.2.1 Multi-agent systems MAS

As the name suggests, multi-agent systems are systems that have more than one agent or decision maker. Since, in general, each agent has a personal objective, the resulting optimization problem, from a centralized perspective, is one with a vector objective. A key characteristic of MAS is that all agents affect each other directly or indirectly. Hence, different elements of this vector objective cannot be optimized independently, thus making MAS different from multi single-agent systems.

There are two broad approaches (also called solution concepts) to optimizing this vector objective:

1. Find an equilibrium (i.e. a set of policies for all agents) where all agents' policies are such that no agent can achieve a better objective by unilateral deviation from the equilibrium.

2. Find an optimum (i.e. a set of policies for all agents) using a scalar objective formed by a (weighted) sum of the various elements of the vector objective.

The above two approaches leading to two different solution concepts also define two classes of MAS – games (where all agents are strategic, hence correspond to the first solution concept) and teams (where all agents are altruistic and hence correspond to the second solution concept). In general, in the case of teams, all agents receive an identical reward and this reward need not be the summation of any individual reward functions. In a case where all agents receive an identical reward, the scalar objective would just be this global reward. A special sub-class of teams and games can be considered where all agents behave in an identical manner, called swarms. This additional symmetry between agents can lead to further simplifications in finding an optimal policy for all agents. In this work, we restrict attention to games and teams.

Another important concept in MAS, in addition to the solution concept, is the concept of information structure. An information structure represents which agent knows what and when. Different information structures lead to different solutions for the MAS. In the sequel we describe information structures in more detail and also provide different examples.

1.2.2 Multi-agent reinforcement learning (MARL)

As described earlier, in MARL, the agents have to learn an optimal policy without knowing the system dynamics and the reward functions of any agent. The temporal sequence of interaction of each agent with the environment can be described as follows. At each point in time, each agent receives some observation from the environment, it takes an action, the environment evolves as a result of the actions of all agents and each agent then receives a scalar reward and the next observation. In general, the observations and the scalar rewards received by each agent are random variables. Each agent has to learn to act optimally using this history of observations and rewards obtained by interacting with the environment which includes other agents as well. In this thesis we assume that all agents act simultaneously. Extension of the algorithms to environments where agents act in a sequential manner can be done in a straightforward manner by carefully identifying the information structure of the problem.

We deconstruct this problem by comparing it with standard single agent RL problems. In a fully observed single agent RL problem, the agent's interaction with its environment

is as follows: the agent receives an observation (which is the state of the system) from the environment, it then takes an action, the environment evolves, the agent receives a reward from the environment and an observation, which is the next state of the system. This problem is modeled using the framework of Markov Decision Processes (Markov decision processes (MDP)s). There are several RL algorithms that have been developed and successfully used for different kinds of MDPs.

We can relax some of the requirements of MDPs to yield formalisms that can capture the features of MAS. The first requirement to be relaxed is full observability as in most MAS, each agent receives only a partial observation of the state of the system. Hence, we study RL in single agent partially observable Markov decision processes (POMDPs) to develop MARL algorithms. These can be extended to the multi-agent cooperative setting using the framework of Decentralized partially observable Markov decision processes (Dec-POMDP)s. The common paradigm used in RL for MAS is centralized learning with decentralized execution. In this setting, from the perspective of the centralized (meta-) agent, the Dec-POMDP becomes an RL problem in an associated centralized single agent POMDP. In this single agent POMDP, the centralized agent receives observations from all the agents, but only learns admissible policies for each agent, which are mappings from each agent's local observations to its local actions. Hence, RL in single agent POMDPs can provide a basis to design (decentralized) MARL algorithms for Dec-POMDPs. In this thesis, we present a novel single agent RL algorithm for POMDPs and extend the same to an MARL algorithm for a class of Dec-POMDPs called mean-field teams.

The second requirement that needs to be relaxed when moving from single agent systems to MAS is the assumption of stationarity (or time-homogeneity) of the environment. In almost all single agent RL problems, the environment is assumed to be stationary. Even in situations where the environment is considered to be non-stationary, it is assumed to change at a much larger timescale (i.e., at a much slower rate) than the timescale at which the agent operates in the environment. In MAS, for each agent, the other learning agents form a part of the environment and hence for each agent the environment is non-stationary. This non-stationarity is due to the dynamical and reward coupling between all agents. Hence, for each agent to learn, we must find some way to decouple the agent from the rest of the agents. In other words, each agent should be able to determine sensitivity of its performance objective, despite the unknown influence of the other agents. Various approaches to address this are presented in the literature. All these can be thought of as using what may be

viewed as a decoupling object, that conditionally decouples the performance of each agent from the rest of the agents in the system. For instance, some of the decoupling objects used in literature are opponent models, centralized critics, etc. In this thesis, we present two additional decoupling objects—one for teams and one for games.

Furthermore, MARL can also be broadly divided into two types – centralized and decentralized. A popular paradigm adopted in literature for developing provably convergent MARL algorithms is that of centralized learning and decentralized execution, where as the name suggests, RL is done in a centralized manner even though the actual execution in the real world has to be done in a decentralized manner. Thus, the conditions assumed during learning and execution are different in this paradigm. In contrast to this, we present fully decentralized learning algorithms with provable convergence guarantees in this thesis. Furthermore, the design of learning algorithms is also influenced by the type of MAS being considered—hence, we study games and teams separately for developing MARL algorithms.

1.3 Previous work

1.3.1 Previous work on RL in POMDPs

Exact and approximate planning in POMDPs is done in literature using the concept of information states. Informal definition of information state was used by [69] for adaptive control systems. Formal definitions for linear control systems were given by [18] for discrete time systems and by [31] for continuous time systems. An information state is a compression of the history of observations and actions that is sufficient for dynamic programming. A trivial information state is the entire history. One of the most popular information states used in planning in POMDPs is the belief state, which can be found in work such as [23, 108, 110] among others. Algorithms have been developed for planning using exact belief state evolution and also their particle filter like variants called point based methods. These methods are surveyed in [103]. Dynamical updates of belief states require knowledge of the transition kernel of the system as well as as the observation likelihood kernel. In RL problems, since the system dynamics are unknown, we cannot evaluate the belief state evolution. Hence, belief states are not a useful information state for RL.

There are two broad approaches that have been used in literature for RL in POMDPs:

1. Use of the entire history of observations and actions as an information state, i.e.,

the entire history is used as an input for the policy function and/or the (action-)value function. This is typically done using Recurrent Neural Network (RNN)s or their variants such as Long Short-Term Memory (LSTM)s or Gated Recurrent Unit (GRU)s as function approximators. An example of this approach is recurrent policy gradient [130]

2. Use of predictive state representations [78], i.e., compressions of history that are sufficient to predict future observations given future actions. These are sufficient statistics for uncontrolled processes and generally not sufficient for Dynamic programming (DP). They also involve use of RNNs with LSTMs or GRUs as function approximators. An example of this approach can be found in [45] and a variant that generalizes predictive state representations to causal state representations can be found in [140].

In this thesis, we present an alternative notion of an approximate information state and show how it can be constructed using the history of observation and action data. We then demonstrate the use of this approximate information state for RL in POMDPs and provide bounds for the resultant approximation error.

1.3.2 Previous work on MARL

A thorough overview of algorithms for MARL can be found in [21]. Some of the earliest works in MARL involved defining a framework for such problems [75] and algorithms for relatively simple MAS such as two player games, zero-sum games etc. [76, 77]. Some of the more recent approaches to MARL involve the paradigm of centralized learning and decentralized execution. Some popular algorithms that fall in this class include BICNET [90], MADDPG [79], and COMA [34]. A critique of the generic approaches in MARL can be found in [104, 105]. More recent reviews of MARL that complement [21] are [17, 46, 141].

Mean-field games (MFG) and mean-field teams (MFT) are classes of large population systems where the dynamics and rewards of each agent are decoupled from all other agents given the empirical distribution of the agents' states and/or actions [3, 5, 54–56, 70, 125, 126]. Some of the works that cover RL in such MAS include a model based adaptive control algorithm for MFG [66], a Q-learning based algorithm for MFG control of coupled oscillators [139], model-free Q-learning and actor critic algorithms for MFG [85, 138], a mean-field based solution for inverse RL [137].

1.4 Scope of this thesis

In this thesis we present RL algorithms for different single and multi-agent systems. This thesis is structured as follows:

1.4.1 POMDPs

In Chapter 2, we consider partially observable single agent systems and formally define the concept of an information state in such systems. This chapter is an extended version of the publications (C1) and (UJ1), given in Section 1.5.2. In this chapter, we provide two alternate characterizations leading to two different information states, both of which we prove are sufficient for dynamic programming. The first characterization involves predicting the expected reward and the next distribution of information state given the current information state and current action. The second characterization involves predicting the distribution of the next observation instead of the next information state, while ensuring that the information state evolves in a state-like manner. We show that the second characterization is a stricter requirement and it is a finer partition of the history compared to the first. We verify that entire history and belief states used in planning satisfy the conditions of an information state. But the dimension of the history increases with time, thus rendering RL using this difficult. The belief state cannot be used in RL as the the system dynamics and observation models are needed to determine its evolution. As an alternative to these, we present the concept of an Approximate information state (AIS) as a compression of history that is useful in approximately predicting the next expected reward and approximately predicting the next distribution of either the AIS or the observation while ensuring state-like evolution of the AIS similar to the two alternatives in the exact case. We bound the error in the value function obtained using AIS in terms of the approximation errors of the AIS in predicting the next expected reward and next distribution of AIS or observation as the case may be. We then demonstrate a method to learn an AIS from data obtained by sequential interaction with an environment. Finally we present a provably convergent RL algorithm that uses this learned AIS. We demonstrate the performance of this algorithm on a few numerical examples and show that it performs favorably when compared with one of the state-of-the-art POMDP RL algorithms.

1.4.2 Mean-field teams

In Chapter 3, we extend the above concept of AIS for a class of coooperative large population systems called mean-field teams. This is an extended version of the publications (UJ2), (W3) and (W4), given in 1.5.2, In this chapter, we present an AIS based interpretation of the mean-field limit and finite-agent approximations used in literature for mean-field teams and also bound the approximation error in terms of the AIS error. We verify that this is consistent with the results in literature [4]. We then develop an RL algorithm for mean-field teams and prove its convergence to a locally optimal solution. We demonstrate the empirical performance of our MARL algorithm for mean-field teams using two numerical examples.

1.4.3 Stationary mean-field games

In Chapter 4, we consider a different variant of large population MAS called stationary mean-field games. This chapter covers work presented in publication (SCJ1), given in Section 1.5.2. In this chapter, we define two new local solution concepts for stationary mean-field games and develop two RL algorithms. We prove the convergence of these two RL algorithms to the two newly defined solution concepts. We illustrate the performance of both these algorithms using two numerical examples.

1.4.4 MDPs with renewal

Chapter 5 covers work presented in publication (SCJ2), given in Section 1.5.2. In this chapter, we present a novel policy gradient based algorithm, called Renewal Monte Carlo (RMC), for single agent fully observed systems with a renewal/regenerative property. We extend this algorithm to systems with renewal in terms of post-decision states, i.e., we split the transition into two parts – a controlled one and an uncontrolled one and exploit the renewal property only in terms of the controlled transitions. Finally we present an approximate version of RMC, where the approximation error in the resultant value function is bounded (under some technical conditions) in terms of the approximation error in the definition of the renewal event. We empirically demonstrate the performance of RMC for each of these three cases.

In Chapter 6 we present our discussions, conclusions and scope for future work.

1.5 Claims of originality and publications

1.5.1 Claims of originality

The following are the original contributions presented in this thesis:

1. Definition of an alternative concept of information states sufficient for dynamic programming and that of approximate information states (AIS) for partially observed systems.

2. Derivation of approximation error bound in performance due to the use of AIS (under appropriate technical conditions) in terms of the approximation errors in the AIS definition.

3. Design of an algorithm for learning AIS from data obtained by interacting with an environment.

4. Development of an RL algorithm for POMDPs in terms of AIS with proof of convergence and demonstration of its empirical performance on some numerical examples.

5. Analysis of the approximation error of standard mean-field limit and finite-population approximations for mean-field teams using the AIS framework and verification of these results with literature.

6. Development of RL algorithms for mean-field teams using the AIS concept.

7. Definition of two new solution concepts for for stationary mean-field games.

8. Development of two RL algorithms for stationary mean-field games with proof of convergence to the above defined solution concepts.

9. Development of a novel policy gradient based algorithm called RMC for single agent fully observed systems with a renewal/regenerative property.

10. Extension of RMC algorithm to systems with renewal in terms of post-decision states, i.e., we split the transition into two parts – a controlled one and an uncontrolled one and exploit the renewal property only in terms of the controlled transitions.

11. Development of an approximate version of RMC, where the approximation error in the resultant value function is bounded (under some technical conditions) in terms of the approximation error in the definition of the renewal event.

1.5.2 List of publications

The following are the publications resulting from the work presented in this thesis:

Peer-reviewed publications in selective conferences and journals

(SCJ1) Subramanian J., and Mahajan A., "Reinforcement learning in stationary mean-field games", International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Montreal, Canada, 13-17 May, 2019.

(SCJ2) Subramanian J., and Mahajan, A., "Renewal Monte Carlo: Renewal theory based reinforcement learning", IEEE Transactions on Automatic Control, Aug 2020 (in print).

Peer reviewed conference publications

(C1) Subramanian J., and Mahajan, A., "Approximate information state for partially observed systems", IEEE Conference on Decision and Control (CDC), Nice, France, Dec 11-13, 2019.

(C2) Subramanian J., and Mahajan, A., "Renewal Monte Carlo: Renewal theory based reinforcement learning", IEEE Conference on Decision and Control (CDC), Miami, Florida, Dec 17-19, 2018.

(C3) Chakravorty, J., Subramanian, J. and Mahajan, A., "Stochastic approximation based methods for computing the optimal thresholds in remote-state estimation with packet drops", American Control Conference (ACC), Seattle, WA, May 24-26, 2017.

Under preparation journal publications

(UJ1) Subramanian J., and Mahajan, A., "Approximate Information State for Partially Observed Systems", to be submitted to Journal of Machine Learning Research, 2020.

(UJ2) Subramanian J., Seraj, R., and Mahajan, A., "Reinforcement learning using approximate information states for mean-field teams", under preparation.

Workshop and non-archival publications

(W1) Subramanian J., and Mahajan A., "Approximate information state for partially observed systems", NeurIPS 2019 Workshop on Optimization Foundation for Reinforcement Learning, Vancouver, British Columbia, Dec 8-14, 2019.

(W2) Subramanian J., and Mahajan A., "Approximate information state for partially observed systems", The Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM), Montreal, Canada, Jul 7-10, 2019.

(W3) Subramanian J., Seraj, R., and Mahajan, A., "Reinforcement learning for mean-field teams", The Multi-disciplinary Conference on Reinforcement Learning and Decision Making (RLDM), Montreal, Canada, Jul 7-10, 2019.

(W4) Subramanian J., Seraj, R., and Mahajan, A., "Reinforcement learning for mean-field teams", AAMAS Workshop on Adaptive and Learning Agents, Montreal Canada, 13-17 May, 2019.

(W5) Subramanian J., and Mahajan A., "A policy gradient algorithm to compute boundedly rational stationary mean field equilibria", Proceedings of the ICML/IJCAI/AAMAS Workshop on Planning and Learning (PAL-18), Stockholm, Sweden, July 15, 2018.

Other scientific outreach

(O1) Subramanian J., and Mahajan A., "Approximate dynamic programming and reinforcement learning for partially observed systems", poster presented at Montreal AI Symposium, Montreal, Quebec, Sep 6, 2019.

(O2) Subramanian J., and Mahajan A., "Approximate information state for partially observed systems", poster presented at Colloque REPARTI Workshop, Montreal, Quebec, Jun 13, 2019.

(O3) Subramanian J., "Multi-agent reinforcement learning: Stationary Mean Field Games", Graduation Day talk at Information Theory and Applications Workshop, San Diego, California, Feb 12, 2019.

(O4) Subramanian J., and Mahajan, A., "A new policy based RL algorithm with reduced bias and variance", poster presented at Montreal AI Symposium, Montreal, Quebec, Sep 26, 2017.

1.5.3 Contributions of co-authors

In all papers except (C3), (UJ2), (W3) and (W4), J. Subramanian and A. Mahajan contributed equally to model formulation and the analysis, while J. Subramanian was primarily responsible for derivations, programming the algorithms for producing numerical results and writing of the papers. In (C3), all three authors contributed equally while J. Chakravorty was primarily responsible for writing the paper. In (UJ2), (W3) and (W4), all three authors contributed equally to model formulation and the analysis, while J. Subramanian and R. Seraj were primarily responsible for derivations, programming the algorithms for producing numerical results and writing of the papers. We are grateful to Raihan Seraj for providing planning and RPG solutions for the examples in Chapter 2. We are grateful to Amit Sinha for providing numerical examples for the examples using AIS in Chapter 2. We are grateful to Joelle Pineau for useful feedback and for suggesting the idea of approximate RMC in Chapter 5. We are also grateful to the anonymous reviewers for suggestions that led to an improved exposition and more detailed numerical experiments for the work presented in Chapter 5.

Chapter 2

Approximate dynamic programming and RL for POMDPs

The theory of Markov decision processes focuses primarily on systems with full state observation. The focus of this thesis is on single and multi-agent systems with partial observations. Therefore, we begin by considering single-agent systems with partial state observations in this chapter. Though the chapter title only specifies POMDPs, we consider the more general case of partially observed input-output systems for the most of this chapter. For most of this chapter we restrict attention to partially observable single agent systems and we provide extension to a class of multi-agent cooperative systems, called mean-field teams, in the next chapter.

Generally, in planning problems, when systems with partial state observations are considered, they are converted to systems with full state observations by considering the belief state (which is the posterior belief on the state of the system given the history of observations and actions). Although this leads to an explosion in the size of the state space, the resulting value function has a nice property—it is piecewise linear and convex in the belief state [108]—which is exploited to develop efficient algorithms to compute the optimal policy [60, 103]. Thus, for planning, there is little value in studying alternative characterizations of partially observed models.

However, the belief state formulation is not as nice a fit for online reinforcement learning. Part of the difficulty is that the construction of the belief state depends on the system model. So, when the system model is unknown, the belief state cannot be constructed using

the observations. Therefore, critic based methods are not directly applicable. There are some results that circumvent this difficulty [10, 45, 78]. However, many of the recent results suggest that using RNNs (Recurrent Neural Networks [99]) or LSTMs (Long Short-Term Memories [49]) for modeling the policy function (actor) and/or the action-value function (critic) works for reinforcement learning in partially observed systems [8, 9, 43, 44, 130, 131]. In this chapter, we present a rigorous theory for planning and learning in partially observed models using the notions of information state and approximate information state. We then present numerical experiments that show that the approximate information state based approach works well on benchmark models.

This chapter is organized as follows. We first begin by defining states sufficient for optimal control (dynamic programming) for an input-output system with per-step rewards. This system has controlled as well as stochastic inputs, where the stochastic inputs are not observed. This forms the base model for planning as well as reinforcement learning in single agent POMDPs. For such systems, we present two versions of states sufficient for optimal control, which are called information states. We then present the notion of an Approximate information state (AIS), define two versions of it and derive performance (error) bounds when using AIS instead of the exact information state. Following this, we compare our approximation approach with other approaches in literature. We then present an RL algorithm that uses AIS, prove its convergence and demonstrate its performance by comparison with results from one of the state of the art methods for POMDPs.

2.1 Information state

Consider an input-output system with a reward process where the objective is to maximize the expected cumulative sum of the rewards obtained during the operation of the system. Informally, in such a system, a state sufficient for optimal control is a state sufficient for dynamic programming in this system. A state sufficient for dynamic programming is one such that using this state instead of the entire history is without loss of optimality. Such a state is called an information state for the system. We show that an information state only needs to be sufficient to predict the current reward and next state or observation distribution, rather than multi-step future outputs.

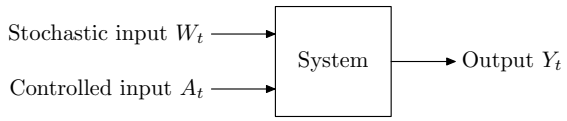


Fig. 2.1: A stochastic input-output system

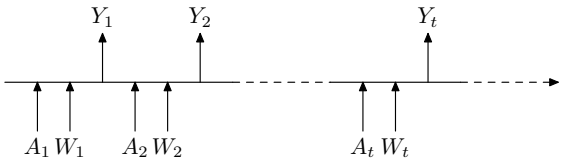


Fig. 2.2: The timing diagram of the input-output system.

2.1.1 Model

Consider a stochastic input-output system as shown in Fig. 2.1. The system operates in discrete time for a horizon T. At time t, the system has two inputs: a control input $A_t \in \mathcal{A}$ and a stochastic input $W_t \in \mathcal{W}$; and the system generates an output $Y_t \in \mathcal{Y}$ and a reward $R_t \in \mathbb{R}$. The order of inputs and outputs is shown in Fig. 2.2.

In general, the output and reward at time t are functions of all the (controlled and stochastic) inputs until 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 $\{f_t \colon \mathcal{A}^t \times \mathcal{W}^t \to \mathcal{Y}\}_{t=1}^T$ are called the system output functions and $\{r_t \colon \mathcal{A}^t \times \mathcal{W}^t \to \mathbb{R}\}_{t=1}^T$ are called the system reward functions.

For the ease of exposition, we assume that $\{\mathcal{A}\}_{t\geq 1}$, $\{\mathcal{W}\}_{t\geq 1}$, and $\{\mathcal{Y}\}_t \geq 1$ are finite sets. The results extend to general spaces under appropriate technical conditions.

We assume that $W_{1:T}$ are independent random variables defined on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Thus, if the control inputs $A_{1:T}$ are specified, then the output $Y_{1:T}$ are random variables on $(\Omega, \mathcal{F}, \mathbb{P})$.

In the rest of the exposition we interchangeably use the term actions to refer to control inputs.

An agent observes the history $H_t = (A_{1:t-1}, Y_{1:t-1})$ of control inputs and observations until time t, with $H_t \in \mathcal{H}_t$, the space of all histories at time t, and chooses the control input

$$A_t = \pi_t(H_t)$$

according to some history dependent policy $\pi \coloneqq \{\pi_t\}_{t=1}^T$. The performance of policy π is

given by

$$J(\pi) = \mathbb{E}^{\pi}\left[\sum_{t=1}^{T} R_t\right]. \tag{2.1}$$

The objective of the agent is to choose a policy π to maximize the expected total reward $J(\pi)$.

2.1.2 A dynamic programming decomposition

In this section, we present a dynamic program for (2.1) which uses the history of observations and actions as state. Such a dynamic program is not efficient for computing the optimal policy but it will serve as a reference for the rest of the analysis.

First consider the dynamic program for computing the value of any policy π. In particular, define the *reward-to-go* function as

$$V_t^{\pi}(h_t) \coloneqq \mathbb{E}^{\pi}\left[\sum_{s=t}^{T} R_s \;\middle|\; H_t = h_t\right]. \tag{2.2}$$

From definitions in (2.1) and (2.2), we have

$$J_1(h_1; \pi) = \mathbb{E}[V_1^{\pi}(h_1)].$$

Let $J_{T+1}(h_{T+1}; \pi) \coloneqq 0$. Then, the reward to go functions can be computed recursively as follows:

$$\begin{aligned} J_t(h_t; \pi) &\overset{(a)}{=} \mathbb{E}^{\pi}\left[R_t + \mathbb{E}\left[\sum_{s=t+1}^{T} R_s \;\middle|\; H_{t+1}\right] \;\middle|\; H_t = h_t\right] \\ &= \mathbb{E}^{\pi}\big[R_t + J_{t+1}(H_{t+1}; \pi) \;\big|\; H_t = h_t\big], \end{aligned} \tag{2.3}$$

where (a) follows from the tower property of conditional expectation and the fact that $H_t \subseteq H_{t+1}$. Note that $J_t(h_t; \pi)$ only depends on the future policy $(\pi_t, \dots, \pi_T)$ and not on the past policy $(\pi_1, \dots, \pi_{t-1})$. Thus, the dynamic program (2.3) gives a recursive method to compute $J(\pi)$.

Now, recursively define the following *value functions*. $V_{T+1}(h_{T+1}) \coloneqq 0$ and for $t \in$

$\{T, \dots, 1\}$:

$$Q_t(h_t, a_t) = \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t] \tag{2.4}$$

and

$$V_t(h_t) = \max_{a_t \in \mathcal{A}} Q_t(h_t, a_t). \tag{2.5}$$

Theorem 2.1.1 *A policy $\pi = (\pi_1, \dots, \pi_T)$ is optimal if and only if it satisfies*

$$\pi_t(h_t) \in \arg\max_{a_t \in \mathcal{A}} Q_t(h_t, a_t). \tag{2.6}$$

PROOF To prove this, we need to show the following:

(C) At any time t, $J_t(h_t, \pi) \leq V_t(h_t)$, with equality if and only if $(\pi_t, \pi_{t+1}, \dots, \pi_T)$ satisfy (2.6).

We prove this using backward induction. At $t = T + 1$, (C) is satisfied by definition and this forms the basis of induction. We assume that (C) holds for time $t + 1$, which is the induction hypothesis. Then, for time t, we have from (2.3),

$$\begin{aligned} J_t(h_t; \pi) &= \mathbb{E}^{\pi}\left[R_t + J_{t+1}(H_{t+1}; \pi) \;\middle|\; H_t = h_t\right] \\ &\overset{(a)}{\leq} \mathbb{E}^{\pi}\left[R_t + V_{t+1}(H_{t+1}) \;\middle|\; H_t = h_t\right] \\ &\overset{(b)}{\leq} V_t(h_t), \end{aligned}$$

where (a) follows from the induction hypothesis and (b) follows from the definition of the value function (2.5) and (2.4). From the induction hypothesis, the equality in (a) is achieved if and only if $\{\pi_s\}_{s>t}$ satisfy (2.6). From (2.5), we see that the equality in (b) is achieved if and only if $\pi_t(h_t) \in \arg\max_{a \in \mathcal{A}} Q_t(h_t, a)$, i.e., π_t satisfies (2.6). Hence, (C) holds at time t. ■

Remark 2.1.1 Traditionally the value function is defined as:

$$V_t(h_t) = \sup_{\pi} J_t(h_t; \pi),$$

and it can be shown that this value function satisfies the dynamic programmming recursion

of (2.4) and (2.5). Our definition of the value function is different, but it is easy to see that the two definitions are equivalent. □

2.1.3 Information state and a simplified dynamic program

Let $\mathcal{F}_t = \sigma(H_t)$ denote the filtration generated by the history of observations and control actions.

Definition 2.1.1 An information state $\{Z_t\}_{t\geq 1}$, $Z_t \in \mathcal{Z}$, is an $\mathcal{F}_t$ adapted process (therefore, there exist functions $\{\vartheta_t\}_{t=1}^T$ such that $Z_t = \vartheta_t(H_t)$) that satisfies the following properties:

(P1) Sufficient for performance evaluation, i.e.,

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

(P2) Sufficient to predict itself, i.e., for any Borel measurable subset B of $\mathcal{Z}$,

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

There is no restriction on the space $\mathcal{Z}$, although an information state is useful only when the space $\mathcal{Z}$ is "small" in an appropriate sense. We have assumed that the space $\mathcal{Z}$ is time-homogeneous for convenience. In some situations, it may be more convenient to construct an information state which takes values in spaces that are changing with time.

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

$$Z_{t+1} = \varphi_t(Z_t, Y_t, A_t).$$

(P2b) Is sufficient for predicting future observations, i.e., for any subset B of $\mathcal{Y}$,

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

Proposition 2.1.1 (P2a) *and* (P2b) *imply* (P2).

PROOF For any Borel measurable subset B of $\mathcal{Z}$, we have

$$\begin{aligned}
\mathbb{P}(Z_{t+1} \in B \,|H_t = h_t, A_t = a_t) &\overset{(a)}{=} \sum_{y_{t+1} \in \mathcal{Y}} \mathbb{P}(Y_t = y_t, Z_{t+1} \in B \mid H_t = h_t, A_t = a_t) \\
&\overset{(b)}{=} \sum_{y_t \in \mathcal{Y}} \mathbb{1}\{\varphi_t(\vartheta_t(h_t), y_t, a_t) \in B\} \times \mathbb{P}(Y_t = y_t \mid H_t = h_t, A_t = a_t) \\
&\overset{(c)}{=} \sum_{y_t \in \mathcal{Y}} \mathbb{1}\{\varphi_t(\vartheta_t(h_t), y_t, a_t) \in B\} \times \mathbb{P}(Y_t = y_t \mid Z_t = \vartheta_t(h_t), A_t = a_t) \\
&\overset{(d)}{=} \mathbb{P}(Z_{t+1} \in B \mid Z_t = \vartheta_t(h_t), A_t = a_t)
\end{aligned}$$

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. ■

Note that $Z_t = H_t$ is always an information state, so an information state always exists. It is straight-forward to show that if we construct a state space model for the above input-output model, then the belief on the state given the history of observations and control inputs is an information state. Below we present an example of a non-trivial information state that is much simpler than the belief state.

Example 1 (Machine Maintenance) Consider a machine which can be in one of n ordered states where the first state is the best and the last state is the worst. The production cost increases with the state of the machine. The state evolves in a Markovian manner. At each time, an agent has the option to either run the machine or stop and inspect it for a cost. After inspection, the agent may either repair it (at a cost that depends on the state) or replace it (at a fixed cost). The objective is to identify a maintenance policy to minimize the cost of production, inspection, repair, and replacement.

Let τ denote the time of last inspection and S_τ denote the state of the machine after inspection, repair, or replacement. Then, it can be shown that $(S_\tau, t - \tau)$ is an information state for the system. □

The main feature of an information state is that one can always write a dynamic program based on an information state.

Theorem 2.1.2 *Let $\{Z_t\}_{t=1}^T$ be an information state. Recursively define value functions*

$\{\tilde{V}_t\}_{t=1}^{T+1}$, *where* $\tilde{V}_t \colon Z_t \mapsto \mathbb{R}$ *as follows:* $\tilde{V}_{T+1}(z_{T+1}) = 0$ *and for* $t \in \{T, \dots, 1\}$*:*

$$\begin{aligned} \tilde{Q}_t(z_t, a_t) &= \mathbb{E}[R_t + \tilde{V}_{t+1}(Z_{t+1}) \mid Z_t = z_t, A_t = a_t] \\ \tilde{V}_t(z_t) &= \max_{a_t \in \mathcal{A}} \tilde{Q}_t(z_t, a_t). \end{aligned} \tag{2.7}$$

Then, we have the following:

$$Q_t(h_t, a_t) = \tilde{Q}_t(\vartheta_t(h_t), a_t) \text{ and } V_t(h_t) = \tilde{V}_t(\vartheta_t(h_t)). \tag{2.8}$$

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,

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

where (a) follows from the induction hypothesis, (b) follows from the properties of information state, and (c) follows from the definition of $\tilde{Q}$. This shows that the action-value functions are equal. By maximizing over the actions, we get that the value functions are also equal. ■

Remark 2.1.2 In light of Theorem 2.1.2, an information state may be viewed as a generalization of the traditional notion of state [88, 134]. Traditionally, the state of an input-output system is sufficient for input-output mapping. In contrast, the information state is sufficient for dynamic programming.

The notion of information state is also related to sufficient statistics for optimal control [112]. However, in contrast to [112], we do not assume a state space model for the underlying system so it is easier to develop reinforcement learning algorithms using our notion of an information state. □

Informal definition of information state was used by [69] for adaptive control systems. Formal definitions for linear control systems were given by [18] for discrete time systems

and by [31] for continuous time systems.

Coming back to Example 1, Theorem 2.1.2 shows that we can write a dynamic program for that model using the information state $(S_\tau, t-\tau)$, which takes values in a countable set. This countable state dynamic program is considerably simpler than the standard belief state dynamic program typically used for that model. Another feature of the information state formulation is that the information state $(S_\tau, t-\tau)$ does not depend on the transition probability of the state of the machine or the cost of inspection or repair. Thus, if these model parameters were unknown, we can use a standard reinforcement learning algorithm to find an optimal policy which maps $(S_\tau, t-\tau)$ to current action.

Given these 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. In the next section, we present a notion of approximate information state and bound the approximation error.

2.2 Approximate information state (AIS)

Roughly speaking, a compression of the history is an approximate information state if it approximately satisfies (P1) and (P2). This intuition can be made precise as follows. Prior to defining an AIS, we state the definition of an Integral probability metric (IPM) [86]:

Definition 2.2.1 Let $\mathcal{P}$ denote the set of probability measures on a measurable space $(\mathcal{X}, \mathfrak{G})$. Given a class $\mathfrak{F}$ of real-valued uniformly bounded measurable functions on $(\mathcal{X}, \mathfrak{G})$, the Integral probability metric (IPM) between two probability distributions $\mu, \nu \in \mathcal{P}$ is given by:

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

□

Examples of IPM are:

- If $\mathfrak{F} = \{f : \|f\|_\infty \leq 1\}$, then $d_{\mathfrak{F}}$ is the total variation distance.
- If $\mathfrak{F} = \{f : |f|_L \leq 1\}$, i.e, the function $f \in \mathfrak{F}$ is Lipschitz with Lipschitz constant ≤ 1, then $d_{\mathfrak{F}}$ is the Wasserstein distance. This follows from the Kantorovich-Rubinstein duality [124].

- If $\mathfrak{F} = \{f : \|f\|_\infty + |f|_L \leq 1\}$, then $d_{\mathfrak{F}}$ is the Dudley metric.

We say a function f has a $\mathfrak{F}$-constant K if $f/K \in \mathfrak{F}$. We now define an AIS:

Definition 2.2.2 Given a function class $\mathfrak{F}$, positive numbers $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$, Polish spaces $\{\widehat{\mathcal{Z}}_t\}_{t=1}^T$ and compression functions $\{\widehat{\vartheta}_t : \mathcal{H}_t \to \widehat{\mathcal{Z}}_t\}_{t=1}^T$, the process $\{\widehat{Z}_t\}_{t=1}^T$, $\widehat{Z}_t = \widehat{\vartheta}_t(H_t)$, is called an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-approximate information state if there exist a transition approximation function $\widehat{p}_t : \widehat{\mathcal{Z}}_t \times \mathcal{A}_t \to \Delta(\widehat{\mathcal{Z}}_{t+1})$ and a reward approximation function $\widehat{r}_t : \widehat{\mathcal{Z}}_t \times \mathcal{A}_t \to \mathbb{R}$ that satisfy:

(AP1) Sufficient for approximate performance evaluation, i.e.,

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

(AP2) Sufficient to predict itself approximately. For any Borel subset B of $\widehat{\mathcal{Z}}$ define,

$$\mu_t(B) = \mathbb{P}(\widehat{Z}_{t+1} \in B \mid H_t = h_t, A_t = a_t) \text{ and}$$
$$\nu_t(B) = \widehat{p}_t(B \mid \widehat{\vartheta}_t(h_t), a_t).$$

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

We call the tuple $\{\widehat{\vartheta}_t, \widehat{r}_t, \widehat{p}_t\}_{t=1}^T$ as an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-AIS generator. □

Our main result of this section is to identify an approximate dynamic program based on an approximate information state.

Theorem 2.2.1 *Given a function class $\mathfrak{F}$, let $\{\widehat{Z}_t\}_{t=1}^T$, $\widehat{Z}_t \in \widehat{\mathcal{Z}}_t$ be an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-approximate information state with generator $\{(\widehat{\vartheta}_t, \widehat{r}_t, \widehat{p}_t)\}_{t=1}^T$. Recursively define value functions $\{\widehat{V}_t : \widehat{\mathcal{Z}}_t \to \mathbb{R}\}_{t=1}^{T+1}$ as follows: $\widehat{V}_{T+1}(\hat{z}_{T+1}) = 0$ and for $t \in \{T, \dots, 1\}$:*

$$\widehat{Q}_t(\hat{z}_t, a_t) = \mathbb{E}[R_t + \widehat{V}_{t+1}(\widehat{Z}_{t+1}) \mid \widehat{Z}_t = \hat{z}_t, A_t = a_t]$$
$$\widehat{V}_t(\hat{z}_t) = \max_{a_t \in \mathcal{A}} \widehat{Q}_t(\hat{z}_t, a_t). \tag{2.9}$$

Suppose $\widehat{V}_t$ has an $\mathfrak{F}$-constant K_t. Then for any time t and any history h_t, we have the following:

$$|Q_t(h_t, a_t) - \widehat{Q}_t(\widehat{\vartheta}_t(h_t), a_t)| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$$

$$|V_t(h_t) - \widehat{V}_t(\widehat{\vartheta}_t(h_t))| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \tag{2.10}$$

Furthermore, let policy $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_T)$ *be any policy such that:*

$$\hat{\pi}_t(\hat{z}_t) \in \arg\max_{a \in \mathcal{A}} (\widehat{Q}_t(\hat{z}_t, a)), \tag{2.11}$$

and, define policy $\pi = (\pi_1, \dots, \pi_T)$ *as* $\pi_t = \hat{\pi}_t \circ \widehat{\vartheta}_t$*. Then, we have the following:*

$$\begin{aligned} |Q_t(h_t, a_t) - Q_t^{\pi}(h_t, a_t)| &\leq 2[\varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})] \\ |V_t(h_t) - V_t^{\pi}(h_t)| &\leq 2[\varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})]. \end{aligned} \tag{2.12}$$

Proof We prove (2.10) by backward induction. By construction, (2.10) is true at time $T+1$. This forms the basis of induction. Assume that (2.10) is true at time $t+1$ and consider the system at time t. Let $C = \varepsilon_{t+1} + \sum_{s=t+2}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$. Then,

$$\begin{aligned} Q_t(h_t, a_t) &= \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t] \\ &\overset{(a)}{\leq} \mathbb{E}[R_t + \widehat{V}_{t+1}(\widehat{\vartheta}_{t+1}(H_{t+1})) \mid H_t = h_t, A_t = a_t] + C \\ &\overset{(b)}{\leq} \left(\widehat{r}_t(\widehat{\vartheta}_t(h_t), a_t) + \varepsilon_t\right) \\ &\quad + \left(\mathbb{E}[\widehat{V}_{t+1}(\widehat{Z}_{t+1}) \mid \widehat{Z}_t = \widehat{\vartheta}_t(h_t), A_t = a_t] + K_t\delta_t\right) + C \\ &= \widehat{Q}_t(\widehat{\vartheta}_t(h_t), a_t) + \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \end{aligned}$$

where (a) follows from the induction hypothesis and (b) follows from (AP1) and the definition of an IPM (See Def. 2.2.1). The reverse inequality in (2.10) can be proven using a similar argument. By maximizing over actions, we get the relationship between the value functions in (2.10). This completes the induction step and hence (2.10) holds by induction.

To prove (2.12), we note that $\widehat{V}_t^{\hat{\pi}}$ is the value of the performance of policy $\hat{\pi}$. By definition, since $\hat{\pi}_t$ is an optimal policy in the system using the approximate information

state, we have that:

$$\widehat{Q}_t^{\hat{\pi}}(\hat{z}_t, a_t) = \widehat{Q}_t(\hat{z}_t, a_t) \tag{2.13}$$

$$\widehat{V}_t^{\hat{\pi}}(\hat{z}_t) = \widehat{V}_t(\hat{z}_t) \tag{2.14}$$

Then, by using the triangle inequality we get:

$$|Q_t(h_t, a_t) - Q_t^{\pi}(h_t, a_t)| \leq |Q_t(h_t, a_t) - \widehat{Q}_t^{\hat{\pi}}(\widehat{\vartheta}_t(h_t), a_t)| + |\widehat{Q}_t^{\hat{\pi}}(\widehat{\vartheta}_t(h_t), a_t) - Q_t^{\pi}(h_t, a_t)|. \tag{2.15}$$

The first term on the RHS of (2.15) can be bounded using (2.10) as:

$$|Q_t(h_t, a_t) - \widehat{Q}_t^{\hat{\pi}}(\widehat{\vartheta}_t(h_t), a_t)| = |Q_t(h_t, a_t) - \widehat{Q}_t(\widehat{\vartheta}_t(h_t), a_t)| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \tag{2.16}$$

Similarly, the second term on the RHS of (2.15) can be bounded using an almost identical argument as (2.10)by backward induction as follows. We first note that:

$$V_t^{\pi}(h_t) = Q_t^{\pi}(h_t, \pi_t(h_t)). \tag{2.17}$$

We now need to prove:

$$|\widehat{Q}_t^{\hat{\pi}}(\widehat{\vartheta}_t(h_t), a_t) - Q_t^{\pi}(h_t, a_t)| \leq \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}), \tag{2.18}$$

$$|\widehat{V}_t^{\hat{\pi}}(\widehat{\vartheta}_t(h_t)) - V_t^{\pi}(h_t)| \leq \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \tag{2.19}$$

By construction, (2.19) is true at time $T+1$. This forms the basis of induction. Assume that (2.19) is true at time $t+1$ and consider the system at time t. Let $C = \varepsilon_{t+1} + \sum_{s=t+2}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$. Then,

$$\begin{aligned} Q_t^{\pi}(h_t, a_t) &= \mathbb{E}[R_t + V_{t+1}^{\pi}(H_{t+1}) \mid H_t = h_t, A_t = a_t] \\ &\overset{(a)}{\leq} \mathbb{E}[R_t + \widehat{V}_{t+1}^{\hat{\pi}}(\widehat{\vartheta}_{t+1}(H_{t+1})) \mid H_t = h_t, A_t = a_t] + C \\ &\overset{(b)}{\leq} \left(\widehat{r}_t(\widehat{\vartheta}_t(h_t), a_t) + \varepsilon_t\right) \end{aligned}$$

$$+ \left(\mathbb{E}[\widehat{V}_{t+1}^{\hat{\pi}}(\widehat{Z}_{t+1}) \mid \widehat{Z}_t = \widehat{\vartheta}_t(h_t), A_t = a_t] + K_t\delta_t\right) + C$$
$$= \widehat{Q}_t^{\hat{\pi}}(\widehat{\vartheta}_t(h_t), a_t) + \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}).$$

where (a) follows from the induction hypothesis and (b) follows from (AP1), the fact that $\widehat{V}_t^{\hat{\pi}} = \widehat{V}_t$ and so has $\mathfrak{F}$-constant K_t, and the definition of an IPM. The reverse inequality in (2.18) can be proven using a similar argument. By (2.17), we get the relationship between the value functions in (2.19).

Combining (2.15), (2.16) and (2.18), we get (2.12). ■

Corollary 2.2.1 *Given a function class $\mathfrak{F}$, suppose $\{Z_t\}_{t=1}^T$ is an information state and $\{\widehat{Z}_t\}_{t=1}^T$, $\widehat{Z}_t \in \widehat{\mathcal{Z}}_t$ is an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-approximate information state with generator $\{(\widehat{\vartheta}_t, \widehat{r}_t, \widehat{p}_t)\}_{t=1}^T$. Then for any realization h_t of H_t, we have the following:*

$$|Q_t(\vartheta_t(h_t), a_t) - \widehat{Q}_t(\widehat{\vartheta}_t(h_t), a_t)| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$$
$$|V_t(\vartheta_t(h_t)) - \widehat{V}_t(\widehat{\vartheta}_t(h_t))| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \tag{2.20}$$

□

Proof The result follows from Theorems 2.1.2 and 2.2.1. ■

Based on Prop. 2.1.1, we provide an alternative characterization of an approximate information state. We can replace (AP2) with the following stronger conditions:

(AP2a) Evolves in a state-like manner, i.e., there exist measurable functions $\{\hat{\varphi}_t\}_{t=1}^T$ such that $\widehat{Z}_{t+1} = \hat{\varphi}_t(\widehat{Z}_t, Y_t, A_t)$. Moreover, these functions have an $\mathfrak{F}$-constant M_t with respect to Y_t.

(AP2b) Is sufficient for predicting future observations approximately. For any Borel subset B of $\mathcal{Y}$ define, $\mu_t(B) = \mathbb{P}(Y_t \in B \mid H_t = h_t, A_t = a_t)$ and $\nu_t(B) = \mathbb{P}(Y_t \in B \mid \widehat{Z}_t = \widehat{\vartheta}_t(h_t), A_t = a_t) = \widehat{p}_t(B|\widehat{\vartheta}_t(h_t), a_t)$, where $\widehat{p}_t : \widehat{Z}_t \times A_t \mapsto \Delta(Y_{t+1})$. Then, $d_{\mathfrak{F}}(\mu_t, \nu_t) \leq \delta$,

Proposition 2.2.1 *If* (AP2) *is replaced by* (AP2a) *and* (AP2b)*, the result of Theorem 2.2.1 holds with K_t replaced by M_tK_t.*

PROOF If (AP2) is replaced by (AP2a) and (AP2b), the IPM definition holds with K_t replaced by $M_t K_t$. Using this in Theorem 2.2.1 gives the desired result. ■

2.3 Stochastic AIS

A compression of the history for an approximate information state need not be deterministic. A stochastic variant of AIS can be written as follows.

Definition 2.3.1 Given a function class $\mathfrak{F}$, positive numbers $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$, Polish spaces $\{\widehat{\mathcal{Z}}_t\}_{t=1}^T$ and compression functions $\{\widehat{\vartheta}_t^s : \mathcal{H}_t \to \Delta(\widehat{\mathcal{Z}}_t)\}_{t=1}^T$, the process $\{\widehat{Z}_t^s\}_{t=1}^T$, $\widehat{Z}_t^s \sim \widehat{\vartheta}_t(H_t)$, is called an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-stochastic approximate information state if there exist a transition approximation function $\widehat{p}_t : \widehat{\mathcal{Z}}_t \times \mathcal{A}_t \to \Delta(\widehat{\mathcal{Z}}_{t+1})$ and a reward approximation function $\widehat{r}_t : \widehat{\mathcal{Z}}_t \times \mathcal{A}_t \to \mathbb{R}$ that satisfy:

(APS1) Sufficient for approximate performance evaluation, i.e.,

$$\left|\mathbb{E}[R_t \mid H_t = h_t, A_t = a_t] - \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[\widehat{r}_t(\widehat{Z}_t^s, a_t)]\right| \leq \varepsilon_t.$$

(APS2) Sufficient to predict itself approximately. For any Borel subset B of $\widehat{\mathcal{Z}}^s$ define,

$$\mu_t(B) = \mathbb{P}(\widehat{Z}_{t+1}^s \in B \mid H_t = h_t, A_t = a_t) \text{ and}$$
$$\nu_t(B) = \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[\widehat{p}_t(B|\widehat{Z}_t^s, a_t)].$$

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

We call the tuple $\{\widehat{\vartheta}_t^s, \widehat{r}_t, \widehat{p}_t\}_{t=1}^T$ as an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-stochastic AIS generator.

We then have the following result.

Theorem 2.3.1 *Given a function class $\mathfrak{F}$, let $\{\widehat{Z}_t\}_{t=1}^T$, $\widehat{Z}_t^s \in \widehat{\mathcal{Z}}_t$, be an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-approximate information state with generator $\{(\widehat{\vartheta}_t^s, \widehat{r}_t, \widehat{p}_t)\}_{t=1}^T$. Recursively define value functions $\{\widehat{V}_t : \widehat{\mathcal{Z}}_t \to \mathbb{R}\}_{t=1}^{T+1}$ as follows: $\widehat{V}_{T+1}(\hat{z}_{T+1}^s) = 0$ and for $t \in \{T, \dots, 1\}$:*

$$\widehat{Q}_t(\hat{z}_t^s, a_t) = \mathbb{E}[R_t + \widehat{V}_{t+1}(\widehat{Z}_{t+1}^s) \mid \widehat{Z}_t^s = \hat{z}_t^s, A_t = a_t]$$
$$\widehat{V}_t(\hat{z}_t^s) = \max_{a_t \in \mathcal{A}} \widehat{Q}_t(\hat{z}_t^s, a_t). \tag{2.21}$$

Suppose $\widehat{V}_t$ has an $\mathfrak{F}$-constant K_t. Then for any time t and any history h_t, we have the following:

$$|Q_t(h_t, a_t) - \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t(h_t)}[\widehat{Q}_t(\widehat{Z}_t^s, a_t)]| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$$
$$|V_t(h_t) - \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t(h_t)}[\widehat{V}_t(\widehat{Z}_t^s)]| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \tag{2.22}$$

Furthermore, let policy $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_T)$ be any policy such that:

$$\hat{\pi}_t(\hat{z}_t^s) \in \arg\max_{a \in \mathcal{A}} (\widehat{Q}_t(\hat{z}_t^s, a)), \tag{2.23}$$

and, define policy $\pi = (\pi_1, \dots, \pi_T)$ as $\pi_t = \hat{\pi}_t \circ \widehat{\vartheta}_t^s$. Then, we have the following:

$$|Q_t(h_t, a_t) - Q_t^{\pi}(h_t, a_t)| \leq 2[\varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})]$$
$$|V_t(h_t) - V_t^{\pi}(h_t)| \leq 2[\varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})]. \tag{2.24}$$

Proof The proof is very similar to the proof for Thm. 2.2.1, with the difference being that for the value and action-value functions of the stochastic approximation state, we take an additional expectation over the stochastic AIS. Again we prove the result by backward induction. By construction, (2.22) is true at time $T+1$. This forms the basis of induction. Assume that (2.22) is true at time $t+1$ and consider the system at time t. Let $C = \varepsilon_{t+1} + \sum_{s=t+2}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$. Then,

$$\begin{aligned}
Q_t(h_t, a_t) &= \mathbb{E}[R_t + V_{t+1}(H_{t+1}) \mid H_t = h_t, A_t = a_t] \\
&\overset{(a)}{\leq} \mathbb{E}[R_t + \mathbb{E}_{\widehat{Z}_{t+1}^s \sim \widehat{\vartheta}_{t+1}^s(h_{t+1})}[\widehat{V}_{t+1}(\widehat{Z}_{t+1}^s)] \mid H_t = h_t, A_t = a_t] + C \\
&\overset{(b)}{\leq} \left(\mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[\widehat{r}_t(\widehat{Z}_t^s, a_t)] + \varepsilon_t\right) \\
&+ \left(\mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[\mathbb{E}_{\widehat{Z}_{t+1}^s \sim \widehat{p}(\widehat{Z}_t^s, a_t)}[\widehat{V}_{t+1}(\widehat{Z}_{t+1}^s)] + K_t\delta_t\right) + C
\end{aligned}$$

$$= \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[\widehat{Q}_t(\widehat{Z}_t^s, a_t)] + \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}).$$

where (a) follows from the induction hypothesis and (b) follows from (AP1) and the definition of an IPM. The reverse inequality can be proven using a similar argument. By maximizing over actions, we get the relationship between the value functions.

To prove (2.24), we note that $\widehat{V}_t^{\hat{\pi}}$ is the value of the performance of policy $\hat{\pi}$. By definition, since $\hat{\pi}_t$ is an optimal policy in the system using the stochastic approximate information state, we have that:

$$\widehat{Q}_t^{\hat{\pi}}(\hat{z}_t^s, a_t) = \widehat{Q}_t(\hat{z}_t^s, a_t) \tag{2.25}$$

$$\widehat{V}_t^{\hat{\pi}}(\hat{z}_t^s) = \widehat{V}_t(\hat{z}_t^s) \tag{2.26}$$

Furthermore, we have:

$$V_t^{\hat{\pi}}(h_t) = \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[Q_t^{\hat{\pi}}(h_t, \hat{\pi}_t(\widehat{Z}_t^s))].$$

Then, by using the triangle inequality we get:

$$\begin{aligned} |Q_t(h_t, a_t) - Q_t^{\pi}(h_t, a_t)| \leq |Q_t(h_t, a_t) - \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[Q_t^{\hat{\pi}}(\widehat{Z}_t^s, a_t)]| \\ + |\mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[Q_t^{\hat{\pi}}(\widehat{Z}_t^s, a_t)] - Q_t^{\pi}(h_t, a_t)|. \end{aligned} \tag{2.27}$$

The first term on the RHS of (2.27) can be bounded using (2.22) as:

$$\begin{aligned} |Q_t(h_t, a_t) - \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[Q_t^{\hat{\pi}}(\widehat{Z}_t^s, a_t)]| &= |Q_t(h_t, a_t) - \mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[\widehat{Q}_t(\widehat{Z}_t^s, a_t)]| \\ &\leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \end{aligned} \tag{2.28}$$

Similarly, the second term on the RHS of (2.15) can be bounded using an almost identical argument as (2.22) as:

$$|\mathbb{E}_{\widehat{Z}_t^s \sim \widehat{\vartheta}_t^s(h_t)}[Q_t^{\hat{\pi}}(\widehat{Z}_t^s, a_t)] - Q_t^{\pi}(h_t, a_t)| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \tag{2.29}$$

Combining (2.27), (2.28) and (2.29), we get (2.24). ■

It is important to note that the error term δ_t defined in (APS2), is an expectation over all possible $\widehat{Z}_t^s$ and hence, the above bounds for performance may seem tighter than they actually are.

2.4 Extension to infinite horizon

In this section, we extend the definition of AIS for the infinite horizon case. To do this, we first look at the definition of an information state for infinite horizon and then define a corresponding AIS.

2.4.1 Information state for infinite horizon

Definition 2.4.1 An $\mathcal{F}_t$-adapted process $\{Z_t\}_{t\geq 1}$, where $Z_t \in \mathcal{Z}$ (which is time-homogeneous), is an information state for infinite horizon if, in addition to satisfying (P1) and (P2), it satisfies the following:

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

We refer to such a process as time-homogeneous information state. □

In time-homogeneous infinite horizon POMDPs, the belief state is an information state because it satisfies (P1) and (P2) and also satisfies (S).

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

$$[\mathcal{B}V](z) = \max_{a\in\mathcal{A}} \mathbb{E}[R_t + \gamma V(Z_{t+1}) \mid Z_t = z, A_t = a], \tag{2.30}$$

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, if the rewards are uniformly bounded, the following fixed point equation has a unique bounded solution:

$$V = \mathcal{B}V. \tag{2.31}$$

Let V^* be the fixed point and π^* be any policy such that $\pi^*(z)$ achieves the arg max in the right hand side of (2.30) for $[\mathcal{B}V^*](z)$. Is is easy to see that V^* is the performance of the time homogeneous policy $(\pi^*, \pi^*, \dots)$. However, it is not obvious that V^* equals to the optimal performance J^{OPT} (defined below), because the proof of Theorem 2.1.2 relies on backward induction and is not applicable to infinite horizon models. So, we present an alternative proof below.

Theorem 2.4.1 *Let $\{Z_t\}_{t\geq 1}$ be a time-homogeneous information state process. Suppose the rewards are uniformly bounded and lie in the interval $[0, M]$. Let V^* be the unique bounded fixed point of the Bellman operator $\mathcal{B}$. Fix a starting time s and let J_s^{OPT} denote the optimal performance from time s onwards, i.e.,*

$$J_s^{\text{OPT}}(h_s) \coloneqq \max_{\pi} \mathbb{E}^{\pi}\Big[\sum_{t=s}^{\infty} \gamma^{t-s-1} R_t \;\Big|\; H_s = h_s\Big], \tag{2.32}$$

where the maximum is over all (possibly randomized) history dependent policies. Then, $J_s^{\text{OPT}}(h_s) = V^*(\vartheta_s(h_s))$.

Proof Fix a time $T > s$ and let

$$J_{s,T}^{\text{OPT}}(h_s) \coloneqq \max_{\pi} \mathbb{E}^{\pi}\Big[\sum_{t=s}^{T} \gamma^{t-s-1} R_t \;\Big|\; H_s = h_s\Big]$$

be the optimal performance for the time interval $[s, T]$. Note that $J_{s,\infty}^{\text{OPT}} = J_s^{\text{OPT}}$.

Let $V^{(0)} = 0$ and iteratively define $V^{(n+1)} = \mathcal{B}V^{(n)}$. From Theorem 2.1.1, we know that $J_{s,T}^{\text{OPT}}(h_s) = V^{(T-s)}(\vartheta_s(h_s))$. Now, we consider two directions:

- We first derive a lower bound on $J_{s,\infty}^{\text{OPT}}$. Note that

$$\begin{aligned} J_{s,\infty}^{\text{OPT}}(h_s) &= \max_{\pi} \mathbb{E}\Big[\sum_{t=s}^{\infty} \gamma^{t-s-1} R_t \;\Big|\; H_s = h_s\Big] \\ &\geq \max_{\pi} \mathbb{E}\Big[\sum_{t=s}^{T} \gamma^{t-s-1} R_t \;\Big|\; H_s = h_s\Big] \\ &= J_{s,T}^{\text{OPT}}(h_s) = V^{(T-s)}(\vartheta_s(h_s)). \end{aligned} \tag{2.33}$$

- Next, we derive an upper bound on $J^{\mathrm{OPT}}_{s,\infty}$. Note that

$$\begin{aligned} J^{\mathrm{OPT}}_{s,\infty}(h_s) &= \max_{\pi} \mathbb{E}\left[\sum_{t=s}^{\infty} \gamma^{t-s-1} R_t \;\middle|\; H_s = h_s\right] \\ &\le \max_{\pi} \mathbb{E}\left[\sum_{t=s}^{T} \gamma^{t-s-1} R_t \;\middle|\; H_s = h_s\right] + \sum_{t=T+1}^{\infty} \gamma^{t-s-1} M \\ &= J^{\mathrm{OPT}}_{s,T}(h_s) + \frac{\gamma^T}{1-\gamma} M \\ &= V^{(T-s)}(\vartheta_s(h_s)) + \frac{\gamma^T}{1-\gamma} M. \end{aligned} \tag{2.34}$$

Combining (2.33) and (2.34), we get

$$V^{(T-s)}(\vartheta_s(h_s)) \le J^{\mathrm{OPT}}_{s,T}(h_s) \le V^{(T-s)}(\vartheta_s(h_s)) + \frac{\gamma^T}{1-\gamma} M. \tag{2.35}$$

Recall that $\mathcal{B}$ is a contraction. Therefore, $\lim_{T\to\infty} V^{(T-s)} = V^*$. Hence, the result follows from (2.35) by taking the limit $T \to \infty$. ■

2.4.2 Approximate information state for infinite horizon

Definition 2.4.2 Given a function class $\mathfrak{F}$, positive numbers (ε, δ), a Polish space $\widehat{\mathcal{Z}}$ and compression functions $\{\widehat{\vartheta}_t : \mathcal{H}_t \to \widehat{\mathcal{Z}}\}_{t\ge 1}$, the process $\{\widehat{Z}_t\}_{t\ge 1}$, $\widehat{Z}_t = \widehat{\vartheta}_t(H_t)$, is called an (ε, δ)-time-homogeneous approximate information state for infinite horizon, if there exist a time-homogeneous transition approximation function $\widehat{p} : \widehat{\mathcal{Z}} \times \mathcal{A} \to \Delta(\widehat{\mathcal{Z}})$ and a time-homogeneous reward approximation function $\widehat{r} : \widehat{\mathcal{Z}} \times \mathcal{A} \to \mathbb{R}$ that satisfy (AP1) and (AP2) for all t. □

As before, define the Bellman operator $\widehat{\mathcal{B}}$: $[\widehat{\mathcal{Z}} \to \mathbb{R}] \to [\widehat{\mathcal{Z}} \to \mathbb{R}]$ as follows: for any uniformly bounded function $V : \widehat{\mathcal{Z}} \to \mathbb{R}$,

$$[\widehat{\mathcal{B}}V](\hat{z}) = \max_{a\in\mathcal{A}} \mathbb{E}_{\widehat{Z}_{t+1}\sim\widehat{p}(\hat{z},a)}[R_t + \gamma V(\widehat{Z}_{t+1}) | \widehat{Z}_t = \hat{z}, A_t = a]. \tag{2.36}$$

Because of (AS), the expectation on the right hand side does not depend on time. Then, similar to Theorem 2.4.1, we can establish the following.

Theorem 2.4.2 *Given a function class $\mathfrak{F}$, let $\{\widehat{Z}_t\}_{t=1}^{\infty}$ be a time-homogeneous (ε, δ)-approximate information state with generator $(\{\widehat{\vartheta}_t\}_{t>1}, \widehat{r}, \widehat{p})$, with the system having a time-homogeneous information state $Z_t = \vartheta_t(H_t)$, let $\widehat{Z}_t = \widehat{\vartheta}_t(H_t)$, and let the rewards be uniformly bounded. Let $\hat{V}^*$ be the unique bounded fixed point of $V = \widehat{\mathcal{B}}V$. Suppose $\widehat{V}^*$ has an $\mathfrak{F}$-constant K. Then,*

$$|J_s^{\mathrm{OPT}}(h_s) - \widehat{V}^*(\widehat{\vartheta}_t(h_s))| \leq \frac{\varepsilon + \gamma K \delta}{1-\gamma}. \tag{2.37}$$

Furthermore, let policy $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_T)$ be any policy such that:

$$\hat{\pi}(\hat{z}) \in \arg\max_{a \in \mathcal{A}} (\widehat{Q}(\hat{z}, a)), \tag{2.38}$$

where $\widehat{Q}(\hat{z}, a) = \mathbb{E}[R_t + \gamma V(\widehat{Z}_{t+1}) | \widehat{Z}_t = \hat{z}, A_t = a]$ and, define policy π as $\pi_t = \hat{\pi}_t \circ \widehat{\vartheta}_t$. Then, we have:

$$|J_s^{\mathrm{OPT}}(h_s) - J^{\pi}(\widehat{\vartheta}_s(h_s))| \leq \frac{2(\varepsilon + \gamma K \delta)}{1-\gamma}, \tag{2.39}$$

Proof The proof follows by combining ideas from Theorems 2.2.1 and 2.4.1. Let $\widehat{V}^{(0)} = 0$ and iteratively define $\widehat{V}^{(n+1)} = \widehat{\mathcal{B}}\widehat{V}^{(n)}$. We first prove

$$|V^{(T-t)}(\vartheta_t(h_t)) - \widehat{V}^{(T-t)}(\widehat{\vartheta}_t(h_t))| \leq \varepsilon_t + \sum_{s=t+1}^{T} \gamma^{s-t}(\varepsilon_s + K\delta_{s-1}). \tag{2.40}$$

We prove this result by backward induction. By construction, (2.40) is true at time $T+1$. This forms the basis of induction. Assume that (2.40) is true at time $t+1$ and consider the system at time t. Let $C = \varepsilon_{t+1} + \sum_{s=t+2}^{T} \gamma^{s-t-1}(\varepsilon_s + K\delta_{s-1})$. Then,

$$\begin{aligned}
Q^{(T-t)}(\vartheta_t(h_t), a_t) &= \mathbb{E}[R_t + \gamma V^{(T-t-1)}(\vartheta_{t+1}(H_{t+1})) \mid H_t = h_t, A_t = a_t] \\
&\overset{(a)}{\leq} \mathbb{E}[R_t + \gamma \widehat{V}^{(T-t-1)}(\widehat{\vartheta}_{t+1}(H_{t+1})) \mid H_t = h_t, A_t = a_t] + \gamma C \\
&\overset{(b)}{\leq} \Big(\mathbb{E}[R_t \mid \widehat{Z}_t = \widehat{\vartheta}_t(h_t), A_t = a_t] + \varepsilon_t\Big) \\
&\quad + \gamma\Big(\mathbb{E}[\hat{V}^{(T-t-1)}(\widehat{Z}_{t+1}) \mid \widehat{Z}_t = \widehat{\vartheta}_t(h_t), A_t = a_t] + K\delta_t\Big) + \gamma C \\
&= \widehat{Q}^{(T-t)}(\widehat{\vartheta}_t(h_t), a_t) + \varepsilon_t + \sum_{s=t+1}^{T} \gamma^{s-t}(\varepsilon_s + K\delta_{s-1}).
\end{aligned}$$

where (a) follows from the induction hypothesis and (b) follows from (AP1) and the definition of an IPM. The reverse inequality can be proven using a similar argument. By maximizing over actions, we get (2.40). Furthermore, using the fact that $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$ are time-homogeneous with values (ε, δ), and taking limits as $T \to \infty$ in (2.40), we get:

$$\begin{aligned} |V^*(\vartheta(h_t)) - \widehat{V}^*(\widehat{\vartheta}_t(h_t))| &\le \lim_{T\to\infty} \Big[\varepsilon + \sum_{s=1}^{T} \gamma^s(\varepsilon + K\delta)\Big] \\ &= \varepsilon + \frac{\gamma(\varepsilon + K\delta)}{1-\gamma} = \frac{\varepsilon + \gamma K\delta}{1-\gamma}. \end{aligned} \tag{2.41}$$

Then using (2.35) and (2.41), we get (2.37).

For proving (2.39), we consider the following triangle inequality:

$$|V^*(\vartheta_t(h_t)) - V^\pi(\vartheta_t(h_t))| \le |V^*(\vartheta_t(z_t)) - \widehat{V}^{\hat{\pi}}(\widehat{\vartheta}(h_t))| + |\widehat{V}^{\hat{\pi}}(\widehat{\vartheta}(h_t)) - V^\pi(\vartheta_t(h_t))|. \tag{2.42}$$

The first term on the RHS of (2.42) can be bounded using (2.37) as:

$$|V^*(\vartheta_t(h_t)) - \widehat{V}^{\hat{\pi}}(\widehat{\vartheta}(z_t))| = |V^*(\vartheta_t(h_t)) - \widehat{V}(\widehat{\vartheta}(h_t))| \le \frac{\varepsilon + \gamma K\delta}{1-\gamma}. \tag{2.43}$$

Similarly, the second term on the RHS of (2.42) can be bounded using an almost identical argument as (2.37) to get:

$$|\widehat{V}^{\hat{\pi}}(\widehat{\vartheta}(h_t)) - V^\pi(\vartheta_t(h_t))| \le \frac{\varepsilon + \gamma K\delta}{1-\gamma}. \tag{2.44}$$

Combining (2.42), (2.43) and (2.44), we get (2.39). ■

2.5 Comparison with existing results in literature

2.5.1 Relation with state compression

Suppose the approximate information state is a compression of an information state Z_t, rather than the history H_t. In particular, given a function class $\mathfrak{F}$, positive numbers $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$, Polish spaces $\{\widehat{\mathcal{Z}}_t\}_{t=1}^T$ (or these can also be discrete sets) and compression functions $\{\widehat{\vartheta}_t : \mathcal{Z}_t \to \widehat{\mathcal{Z}}_t\}_{t=1}^T$, the process $\{\widehat{Z}_t\}_{t=1}^T$, $\widehat{Z}_t = \widehat{\vartheta}_t(Z_t)$, is called an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-approximate information state if there exist a transition approximation function $\widehat{p}_t : \widehat{\mathcal{Z}}_t \times \mathcal{A}_t \to \Delta(\widehat{\mathcal{Z}}_{t+1})$

and a reward approximation function $\widehat{r}_t : \widehat{\mathcal{Z}}_t \times \mathcal{A}_t \to \mathbb{R}$ that satisfy:

1. $\left|\mathbb{E}[R_t \mid Z_t = z_t, A_t = a_t] - \widehat{r}(\widehat{\vartheta}(z_t, a_t))\right| \leq \varepsilon_t$.
2. Let $\mu_t(B) = \mathbb{P}(\widehat{Z}_{t+1} \in B \mid Z_t = z_t, A_t = a_t)$ and $\nu_t(B) = \widehat{p}(\widehat{\vartheta}_t(z_t), a_t)$. Then $d_{\mathfrak{F}}(\mu_t, \nu_t) \leq \delta_t$.

Then, similar to Theorem 2.2.1, we can show that:

$$|Q_t(\hat{z}_t, a_t) - \widehat{Q}_t(\tilde{\vartheta}_t(\hat{z}_t), a_t)| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$$
$$|V_t(\hat{z}_t) - \widehat{V}_t(\tilde{\vartheta}_t(\hat{z}_t))| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}).$$

These bounds are similar to bounds for aggregating Markov decision processes obtained in [13].

2.5.2 Relation with action compression

In this subsection, we consider the effect of transforming the action space. For instance, one might wish to transform a continuous action space problem to a quantized action space problem, such that the difference between the value functions in the two problems is bounded. An example of action space discretization can be found in [120]. Using an AIS like analysis, we can provide a theoretical error bound for such approaches as follows.

Definition 2.5.1 Consider an input-output process with history H_t, having an information state process Z_t. Let $\hat{\mathcal{A}}$ be any set and $\widehat{\vartheta}^a : \mathcal{A} \to \widehat{\mathcal{A}}$ be an action compression. Given a function class $\mathfrak{F}$, define positive numbers $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$, such that:

$$\left|\mathbb{E}[R_t \mid Z_t = z_t, A_t = a_t] - \mathbb{E}[R_t \mid Z_t = z_t, \widehat{A}_t = \widehat{\vartheta}^a(a_t)]\right| \leq \varepsilon_t,$$

and for any Borel subset B of $\mathcal{Z}$ define,

$$\mu_t(B) = \mathbb{P}(Z_{t+1} \in B \mid Z_t = z_t, A_t = a_t) \text{ and}$$
$$\nu_t(B) = \mathbb{P}(Z_{t+1} \in B \mid Z_t = z_t, \widehat{A}_t = \widehat{\vartheta}^a(a_t)).$$

Then, $d_{\mathfrak{F}}(\mu_t, \nu_t) \leq \delta_t$, where $d_{\mathfrak{F}}$ is an IPM. □

We then have:

Theorem 2.5.1 *Given a function class $\mathfrak{F}$, let $\widehat{A}_t = \widehat{\vartheta}^a(A_t) \in \widehat{\mathcal{A}}$ be an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-action compression, such that $\widehat{\mathcal{A}} \subset \mathcal{A}$. Recursively define value functions $\{\widehat{V}_t\}_{t=1}^{T+1}$, where $\widehat{V}_t \colon Z_t \mapsto \mathbb{R}$ as follows: $\widehat{V}_{T+1}(z_{T+1}) = 0$ and for $t \in \{T, \dots, 1\}$:*

$$\begin{aligned} \widehat{Q}_t(z_t, \hat{a}_t) &= \mathbb{E}[R_t + \widehat{V}_{t+1}(Z_{t+1}) \mid Z_t = z_t, \widehat{A}_t = \hat{a}_t] \\ \widehat{V}_t(z_t) &= \max_{\hat{a}_t \in \widehat{\mathcal{A}}} \widehat{Q}_t(z_t, \hat{a}_t). \end{aligned} \tag{2.45}$$

Suppose $\widehat{V}_t$ has $\mathfrak{F}$-constant K_t. Then, we have the following:

$$\begin{aligned} |Q_t(z_t, a_t) - \widehat{Q}_t(z_t, \widehat{\vartheta}^a(a_t))| &\leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}) \\ |V_t(z_t) - \widehat{V}_t(z_t)| &\leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \end{aligned} \tag{2.46}$$

Furthermore, let policy $\hat{\pi} = (\hat{\pi}_1, \dots, \hat{\pi}_T)$ be any policy such that:

$$\hat{\pi}_t(z_t) \in \arg\max_{a \in \widehat{\mathcal{A}}} (\widehat{Q}_t(\hat{z}_t, a)), \tag{2.47}$$

Then, we have the following:

$$\begin{aligned} |Q_t(z_t, a_t) - Q_t^{\hat{\pi}}(z_t, a_t)| &\leq 2[\varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})] \\ |V_t(z_t) - V_t^{\hat{\pi}}(z_t)| &\leq 2[\varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})]. \end{aligned} \tag{2.48}$$

Proof The proof is very similar to the proof for Thm. 2.2.1. We prove the result by backward induction. By construction, (2.46) is true at time $T+1$. This forms the basis of induction. Assume that (2.46) is true at time $t+1$ and consider the system at time t. Let $C = \varepsilon_{t+1} + \sum_{s=t+2}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$. Then,

$$Q_t(z_t, a_t) = \mathbb{E}[R_t + V_{t+1}(Z_{t+1}) \mid Z_t = z_t, A_t = a_t]$$

$$
\begin{aligned}
&\overset{(a)}{\leq} \mathbb{E}[R_t + \widehat{V}_{t+1}(Z_{t+1}) \mid Z_t = z_t, A_t = a_t] + C \\
&\overset{(b)}{\leq} \Big(\mathbb{E}[R_t \mid Z_t = z_t, \widehat{A}_t = \widehat{\vartheta}^a(a_t)] + \varepsilon_t\Big) \\
&+ \Big(\mathbb{E}[\hat{V}_{t+1}(Z_{t+1}) \mid Z_t = z_t, \widehat{A}_t = \widehat{\vartheta}^a(a_t)] + K_t\delta_t\Big) + C \\
&= \widehat{Q}_t(z_t, \hat{a}_t) + \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}).
\end{aligned}
$$

where (a) follows from the induction hypothesis and (b) follows from Def. 2.5.1 and the definition of an IPM. The reverse inequality can be proven using a similar argument. By maximizing over actions, we get the relationship between the value functions.

To prove 2.48, we consider the following triangle inequality:

$$
|Q_t(z_t, a_t) - Q_t^{\hat{\pi}}(z_t, a_t)| \leq |Q_t(z_t, a_t) - \widehat{Q}_t^{\hat{\pi}}(z_t, \widehat{\vartheta}_t^s(a_t))| + |\widehat{Q}_t^{\hat{\pi}}(z_t, \widehat{\vartheta}_t^a(a_t)) - Q_t^{\hat{\pi}}(z_t, a_t)|. \quad (2.49)
$$

The first term in (2.49) can be bounded using (2.46) as:

$$
|Q_t(z_t, a_t) - \widehat{Q}_t^{\hat{\pi}}(z_t, \widehat{\vartheta}_t^s(a_t))| = |Q_t(z_t, a_t) - \widehat{Q}_t(z_t, \widehat{\vartheta}_t^s(a_t))| \leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \quad (2.50)
$$

Similarly, the second term on the RHS of (2.49) can be bounded using an almost identical argument as (2.46) by backward induction as follows. We first note that:

$$
V_t^{\hat{\pi}}(z_t) = Q_t^{\hat{\pi}}(z_t, \hat{\pi}_t(z_t)). \quad (2.51)
$$

We now need to prove:

$$
|\widehat{Q}_t^{\hat{\pi}}(z_t, \widehat{\vartheta}^a(a_t)) - Q_t^{\hat{\pi}}(z_t, a_t)| \leq \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}), \quad (2.52)
$$

$$
|\widehat{V}_t^{\hat{\pi}}(z_t) - V_t^{\hat{\pi}}(z_t)| \leq \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \quad (2.53)
$$

By construction, (2.53) is true at time $T+1$. This forms the basis of induction. Assume that (2.53) is true at time $t+1$ and consider the system at time t. Let $C =$

$\varepsilon_{t+1} + \sum_{s=t+2}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1})$. Then,

$$\begin{aligned} Q^{\hat{\pi}}{}_t(z_t, a_t) &= \mathbb{E}[R_t + V^{\hat{\pi}}_{t+1}(Z_{t+1}) \mid Z_t = z_t, A_t = a_t] \\ &\overset{(a)}{\leq} \mathbb{E}[R_t + \widehat{V}^{\hat{\pi}}_{t+1}(Z_{t+1}) \mid Z_t = z_t, A_t = a_t] + C \\ &\overset{(b)}{\leq} \Big(\mathbb{E}[R_t \mid Z_t = z_t, A_t = \widehat{\vartheta}^a_t(a_t)] + \varepsilon_t\Big) \\ &+ \Big(\mathbb{E}[\widehat{V}^{\hat{\pi}}_{t+1}(Z_{t+1}) \mid Z_t = z_t, A_t = \widehat{\vartheta}^a_t(a_t)] + K_t\delta_t\Big) + C \\ &= \widehat{Q}^{\hat{\pi}}_t(z_t, \widehat{\vartheta}^a_t(a_t)) + \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \end{aligned}$$

where (a) follows from the induction hypothesis and (b) follows from Def. 2.5.1, the fact that $\widehat{V}^{\hat{\pi}}_t = \widehat{V}_t$ and so has $\mathfrak{F}$-constant K_t, and the definition of an IPM. The reverse inequality in (2.52) can be proven using a similar argument. By (2.51), we get the relationship between the value functions in (2.53).

Combining (2.49), (2.50) and (2.52), we get (2.48). ■

2.5.3 Relation with observation compression (world models)

In this subsection, we consider the effect of spatial compression of the observations. This is relevant in vision based RL problems, where each observation is a 2D or 3D image. One way of doing this spatial compresssion is to use an autoencoder [39] to compress the observations. These compressed obervation representations could then be used to construct an AIS. Formally an AIS using this approach can be defined as:

Definition 2.5.2 Given a function class $\mathfrak{F}$, positive numbers $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$, Polish spaces $\{\widehat{\mathcal{Z}}_t\}_{t=1}^T$ and compression functions $\{\widehat{\vartheta}_t : \mathcal{H}_t \to \widehat{\mathcal{Z}}_t\}_{t=1}^T$, the process $\{\widehat{Z}_t\}_{t=1}^T$, $\widehat{Z}_t = \widehat{\vartheta}_t(H_t) = [\widehat{Y}_t, \bar{Y}_t]$, is an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-approximate information state with generator $\{(\widehat{\vartheta}_t, \widehat{r}_t, \widehat{p}_t)\}_{t=1}^T$ that satisfy:

(AP0) History compression:

$$\widehat{\vartheta}_t(H_t) = [\widehat{\vartheta}_{1,t}(Y_t), \widehat{\vartheta}_{2,t}(H_t))],$$

where $\widehat{\vartheta}_{1,t}$ is the spatial observation compression function (e.g. an autoencoder) and

$\widehat{\vartheta}_{2,t}$ is the temporal compression function.

(AP1) Sufficient for approximate performance evaluation, i.e.,

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

(AP2) Sufficient to predict itself approximately. For any Borel subset B of $\widehat{\mathcal{Z}}$ define,

$$\begin{aligned}\mu_t(B) &= \mathbb{P}(\widehat{Z}_{t+1} \in B \mid H_t = h_t, A_t = a_t) \text{ and} \\ \nu_t(B) &= \widehat{p}_t(B \mid \widehat{\vartheta}_t(h_t), a_t).\end{aligned}$$

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

Using this AIS, we get the following performance bound:

Corollary 2.5.1 *Given a function class $\mathfrak{F}$, suppose $\{\widehat{Z}_t = [\widehat{Y}_t, \bar{Y}_t]\}_{t=1}^T$, $\widehat{Z}_t \in \widehat{\mathcal{Z}}_t$ is an $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$-approximate information state with generator $\{(\widehat{\vartheta}_t, \widehat{r}_t, \widehat{p}_t)\}_{t=1}^T$ as defined in Defintion 2.5.2. Then for any realization h_t of H_t, we have the following:*

$$\begin{aligned}|Q_t(\vartheta_t(h_t), a_t) - \widehat{Q}_t(\widehat{\vartheta}_t(h_t), a_t)| &\leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}) \\ |V_t(\vartheta_t(h_t)) - \widehat{V}_t(\widehat{\vartheta}_t(h_t))| &\leq \varepsilon_t + \sum_{s=t+1}^{T} (\varepsilon_s + K_{s-1}\delta_{s-1}). \end{aligned} \tag{2.54}$$

□

Proof The proof follows from Theorem 2.2.1 with the AIS and generators defined as per Definition 2.5.2. ■

A similar model was proposed in world models paper [42] without any bounds. In this problem, the agent receives a 2D image as an observation Y_t at each time t, which is part of a video sequence. The agent has to choose an action A_t at time t and on taking this action, the agent receives a reward R_t and the next image observation Y_{t+1}. This problem is a POMDP with a very large observation space. Hence, it is not easy to directly use any of the standard POMDP approaches in this case. In [42], the authors present an approach whereby an agent learns a world model, i.e., a generative dynamical model that approximates the environment using spatial and temporal compressions of the 2D image observations. These

compressed representations then serve as an input to the policy function of an RL agent. This dual compression—compression in observation space as well as compression in time is described below.

Specifically, the world model approach consists of three components:

1. **Vision - Spatial compression**: The vision component outputs a compressed vector representation of each observation, i.e, each 2D image in a video sequence is compressed into a vector. This is done in [42] using a Variational Autoencoder. Let $\widehat{Y}_t$ denote this compressed observation, defined as a latent vector in [42]. This is given by:

$$\widehat{Y}_t = M_1(Y_t), \tag{2.55}$$

   where M_1 denotes the vision autoencoder.

2. **Memory - Temporal compression**: The memory component is used for temporal compression of the observation sequence. Specifically, at time t, the memory component outputs a distribution of the next latent vector given the entire history of latent vectors and actions. This is done in [42] using an RNN (denoted by M_2), where $\bar{Y}_t$ is the hidden state of this RNN. $\bar{Y}_t$ is the compressed representation of the latent vectors and actions at time t. Furthermore, let $\mu_t = \mathbb{P}(\widehat{Y}_{t+1}|H_t, A_t)$ and $\nu_t = \mathbb{P}(\widehat{Y}_{t+1}|\bar{Y}_t, \widehat{Y}_t, A_t) =: M_2(\widehat{Y}_{1:t}, A_{1:t})$. M_2 is trained in such a way so as to minimize the distance between the distributions μ_t, ν_t. In [42], M_2 outputs a mixture of Gaussian distributions to approximate the distribution of the next latent vector $\widehat{Y}_{t+1}$. This approach used in [42] is called the Mixture Density Network - RNN approach or the MDN-RNN approach.

3. **Controller**: The controller component is the policy function that uses the compressed outputs of the vision and memory components to generate an action A_t for time t. In [42], this component is a simple linear model that maps $(\widehat{Y}_t, \bar{Y}_t)$ to an action A_t as:

$$A_t = W_c[\widehat{Y}_t, \bar{Y}_t] + b_c, \tag{2.56}$$

   where W_c is matrix of appropriate dimensions and b_c is a vector of appropriate dimensions.

In [42], the authors only present empirical results and do not show any theoretical error bounds. If we define the AIS as $\widehat{Z}_t = [\widehat{Y}_t, \bar{Y}_t]$ and the functions $\widehat{\vartheta}_{1,t}(\widehat{Y}_t) = M_1(Y_t)$,

$\widehat{\vartheta}_{2,t} = \texttt{Hidden state}(M_2(\bar{Y}_t, \hat{Y}_t, A_t))$ and modify the definitions of μ_t and ν_t as follows:

$$\mu_t = \mathbb{P}(\widehat{Y}_{t+1}, \bar{Y}_{t+1} | H_t, A_t),$$
$$\nu_t = \mathbb{P}(\widehat{Y}_{t+1}, \bar{Y}_{t+1} | \bar{Y}_t, \widehat{Y}_t, A_t),$$

then we get the performance bounds for world models using Corollary 2.5.1.

2.5.4 Relation with predictive state representations (PSRs)

The notion of approximate information state is related to predictive state representation (PSR) [78], which predicts a distribution on the future observations given the current history and future actions. Thus, PSR is a state sufficient for input-output models. However, PSR does not predict future rewards, so it is not sufficient for performance evaluation, and therefore, for dynamic programming.

Predictive state representations (PSRs) predict a distribution of future observations conditioned on [45,78,97,107,115,135] history and future actions [45,78]. In other words, in these models, the predictive state is a sufficient statistic to predict future observations given history and future actions. Thus, it provides an alternative for a belief state in POMDPs. In RL using PSRs, one approach is to recursively predict a predictive state and use a reactive policy that depends on this predictive state. This is very similar to our approach for using AIS in RL as presented in Sec. 2.6. The key difference between our approach and some of the recent approaches for RL using PSR as in [45], is that the predictive state in [45] does not involve prediction of rewards. In contrast, our definition of approximate information state involves predicting both the next approximate information state (we could also use next observations (AP2a, AP2b)) and reward. However, we predict both these only for a single step, whereas the PSR models predict the conditional observations for the next k steps. Our comparison with PSRs also extends to methods using causal state representations, which are generalizations of PSRs using non-linear function approximations [140].

2.5.5 Relation with bisimulation

The notion of information state is also related to bisimulation based equivalence [24], which constructs an equivalence in the belief state that is sufficient for dynamic programming. In principle, the bisimulation equivalence may be relaxed using bisumulation metrics [33] to

obtain an approximate information state. The key difference in our definition of information state is that we do not assume a state space model. So, an approximate information state is a compression of the history and not just a compression of the beliefs. Therefore, it is easier to develop reinforcement learning algorithms based on approximate information state.

Bisimulation constructs an equivalence relationship on the state space that is sufficient to keep track of the states and the rewards. Bisimulation metric is a relaxation of bisimulation. Bisimulation metric based state aggregation procedures for MDPs are proposed in [1, 29, 30, 33]. These are used to propose a bisimulation based state aggregation in [24]. This aggregation is done in the belief space and while it does reduce the complexity of finding the planning solution, it does not help in learning. In contrast, our definition of approximate information state does not assume a state space model, but constructs a state space model from data (as explained in the next section). Therefore, it is applicable to learning models as well.

2.5.6 Relation with Deep MDPs

A similar state abstraction procedure for MDPs has been presented in [36] at almost the same time. Though the analysis and error bounds are similar, the motivation is very different and their results do not apply to POMDPs.

2.5.7 Relation with other approaches for POMDPs

A reinforcement learning algorithm based on properties very similar to our definition of approximate information state was presented in [8]. However, that paper did not include an approximation result similar to Theorem 2.2.1 and, therefore, did not provide any performance guarantees.

An alternate generalization of an information state called an Incrementally Expanding Representation (IER) has been presented in [3]. An IER is a countable sequence of finite sets that can be used as an information state. A restriction of this countable sequence to a finite sequence yields an approximation with bounded error. The IER approach requires choice of functions specifying the dynamical evolution of the representation. Our AIS approach learns a continuous, finite dimensional compression of history based on data. Furthermore, the analysis in the IER approach utilizes concepts of reachability and the contraction effect due to discounting, whereas, in the AIS approach, we use the error in

predicting the rewards and dynamics to derive an approximation error bound.

2.6 Reinforcement learning for POMDPs using AIS

In this section, we use an approximate information state to design reinforcement learning algorithms for infinite horizon POMDPs. We split our approach into two steps—a data-driven approach to construct an approximate information state and reinforcement learning using this approximate information state.

2.6.1 Constructing an approximate information state

The definition of approximate information state suggests two ways to construct an information state from data: either use $\hat{\vartheta}(h_t)$ to determine an approximate information state that satisfies conditions (AP1) and (AP2) or conditions (AP1), (AP2a), and (AP2b). The first approach is more efficient, but the second is easier to understand. So we first describe the latter and then the former.

Note that training a network requires the control inputs $\{A_t\}_{t\geq 1}$. In this section, we assume that the control and the observations have been generated according to a pure exploration policy. In the next section, we will consider the case when policy is being learned along with the approximate information state.

Construction based on (AP1), (AP2a) and (AP2b)

We use two function approximators:

- A recurrent neural network (RNN) or its refinements such as LSTM (Long Short-Term Memory) [49] or GRU (Gated Recurrent Unit) [28] with state $C_{t-1} = \widehat{Z}_{t-1}$, inputs (A_{t-1}, Y_{t-1}) and output $\widehat{Z}_t$. We denote this function approximator by ρ.
- A feed forward network with inputs $(\widehat{Z}_t, A_t)$ and output $(\tilde{R}_t, \tilde{\nu}_{t+1})$, where $\tilde{R}_t$ is a prediction of the expected reward and $\tilde{\nu}_{t+1}$ is the prediction of ν_{t+1}, the distribution of the next observation Y_t. We parameterize $\tilde{\nu}_{t+1}$ as multi-variate Gaussain. We denote this function approximator as ψ.

By construction ρ satisfies (AP2a). To minimize the ε in (AP1), we define the loss functions

$$\mathcal{L}_R = \frac{1}{B}\sum_{t=1}^{B} \texttt{smoothL1}(\tilde{R}_t - R_t),$$

where B is the batch size and

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

is the standard smooth approximation for L1 loss. To minimize the δ in (AP2), we define the loss function

$$\mathcal{L}_\nu = -\sum_{t=1}^{B-1} \log(\tilde{\nu}_{t+1}(Y_t)),$$

which is the negative log likelihood loss for $\tilde{\nu}_t$ and thus approximates the KL-divergence between μ_t and ν_t. We use the KL-divergence as a surrogate for the Wasserstein distance because: (i) Wasserstein distance is computationally expensive to compute; and (ii) KL-divergence upper bounds the total variation (due to Pinsker's inequality), which in turn upper bounds Wasserstein distance for metric spaces with bounded diameter. To train the networks ρ and ψ, we use a weighted combination of these losses to get a single scalar loss:

$$\mathcal{L}_{\rho,\psi} = \lambda\mathcal{L}_R + (1-\lambda)\mathcal{L}_\nu \tag{2.57}$$

where $\lambda \in [0, 1]$ is a hyperparameter.

Construction based on (AP1) and (AP2)

We use two function approximators:

- A recurrent neural network (RNN) or its refinements such as LSTM (Long Short-Term Memory) [49] or GRU (Gated Recurrent Unit) [28] with state C_{t-1}, inputs (A_{t-1}, Y_{t-1}) and output $\widehat{Z}_t$. We denote this function approximator by ρ.
- A feed forward network as in the previous case, except that $\tilde{\nu}_{t+1}$ is the prediction of ν_{t+1}, the distribution of the next approximate information state $\widehat{Z}_{t+1}$. We denote this function approximator as ψ.

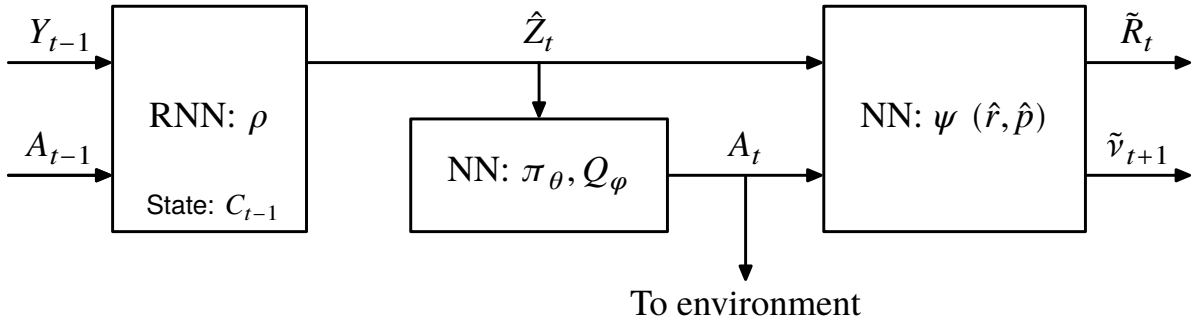


Fig. 2.3: Neural network based function approximators for RL using AIS.

Note that we do not require $C_{t-1} = \widehat{Z}_{t-1}$ in this case. This is because to satisfy (AP2), the approximate information state just needs to be a function of the history, and not necessarily evolve in a state-like manner. To minimize ε and δ, we train the networks ρ and ψ using the loss function $\mathcal{L}_{\rho,\psi}$ defined in (2.57), where $\mathcal{L}_R$ is as before and

$$\mathcal{L}_\nu = -\sum_{t=1}^{B-1} \log(\tilde{\nu}_{t+1}(\widehat{Z}_{t+1})). \tag{2.58}$$

2.6.2 Reinforcement learning

In this section, we present an approach to use the approximate information state for reinforcement learning. We propose the following RL architecture with three components. In this case, we learn the AIS along with the optimal policy, but the AIS loss is as defined before. For completeness, we restate the AIS component below.

1. **State approximator**, which consists of two parts: The first is an AIS encoder (ρ) which is an RNN/LSTM with with state C_{t-1}, inputs (A_{t-1}, Y_{t-1}) and output $\widehat{Z}_t$. The RNN/LSTM can be represented as:

$$\widehat{Z}_t, C_t = \rho(Y_{t-1}, A_{t-1}, C_{t-1}),$$

where C_t is the state of the RNN/LSTM at time t and this state is randomly initialized at $t = 1$. For the purpose of analysis, an RNN can be unrolled into a sequence of feedforward networks. Each of these networks has the same architecture as the RNN and have as an input the previous state of the RNN. Any such network can be expanded into multiple

layers with each layer acting on the output of the previous layer. For any layer l, if the vector x is its input, its output is given by $f_l(W_l x + b_l)$, where W_l is a matrix of appropriate dimensions and b_l is a vector of appropriate dimension and f_l is an activation function, which is typically a non-linear function. The output of the t^{th} network has the $\widehat{Z}_t$ as well as the RNN state C_t. Since this process of unrolling creates long chains of function applications, the process of backpropagation to compute gradients suffers from the problem of exploding or vanishing gradients. To overcome this problem, long short-term memory (LSTM) or gated recurrent unit (GRU) based architectures have been developed that have additional gates/state variables at each level of unrolling that determine what part of the history needs to be retained in future computations. Most current RNN implementations use either LSTMs or GRUs for this reason.

The second part is a feedforward NN (ψ) with inputs $(\widehat{Z}_t, A_t)$ and outputs $(\tilde{R}_t, \tilde{\nu}_{t+1})$. This can be represented as:

$$\tilde{R}_t, \tilde{\nu}_{t+1}) = \psi(\widehat{Z}_t, A_t),$$

where the function ψ network can be expanded into multiple layers with each layer acting on the output of the previous layer. For any layer l, if the vector x is its input, its output is given by $f_l(W_l x + b_l)$, where W_l is a matrix of appropriate dimensions and b_l is a vector of appropriate dimension and f_l is an activation function, which is typically a non-linear function. For an overview of neural network architectures used here, please see Appendix A.

We use the smooth L1 loss between the predicted $\tilde{R}_t$ and the observed reward R_t to minimize the ε parameter of (AP1) and use the negative log likelihood loss[1] of ν_{t+1} to minimize the parameter δ of (AP2). Thus, the overall loss function for the state approximator is $\mathcal{L}_{\text{AIS}} = \lambda\mathcal{L}_R + (1-\lambda)\mathcal{L}_\nu$, where $\lambda \in (0,1)$ is a hyper-parameter and, given a batch size of B,

$$\mathcal{L}_R = \frac{1}{B}\sum_{t=0}^{B-1} \texttt{smoothL1}(\tilde{R}_t - R_t),$$
$$\mathcal{L}_\nu = -\sum_{t=0}^{B-2} \log(\nu_{t+1}(\widehat{Z}_{t+1})).$$

Let ξ denote the combined parameters of the AIS encoder and predictor, which are updated

[1]The negative log likelihood loss approximates KL-divergence.

using stochastic gradient ascent

$$\xi_{k+1} = \xi_k + a_k \nabla_\xi \mathcal{L}_{\text{AIS}}(\xi_k), \tag{2.59}$$

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

2. **Critic**, which is a function approximator (Q_φ) (may be a linear function approximator, or a feedforward neural network) with the AIS $\widehat{Z}_t$ and action A_t as input and $Q(\widehat{Z}_t, A_t)$ as output. Let φ denote the parameters of this function approximator, which are updated using batch temporal-difference (TD):

$$\varphi_{k+1} = \varphi_k + b_k \nabla_\varphi \mathcal{L}_{\text{TD}}(\xi_k, \varphi_k, \theta_k), \tag{2.60}$$

where,

$$\mathcal{L}_{\text{TD}}(\xi_k, \varphi_k, \theta_k) \coloneqq \frac{1}{B} \sum_{t=0}^{B-1} \texttt{smoothL1}(Q_{\varphi_k}(\widehat{Z}_t, A_t), R_t + \gamma Q_{\varphi_k}(\widehat{Z}_{t+1}, A_{t+1})) \tag{2.61}$$

and θ_k are the parameters of the of the actor (explained below) and the learning rate $\{b_k\}_{k\geq 0}$ satisfies the standard conditions $\sum b_k = \infty$ and $\sum b_k^2 < \infty$. In addition, we require, $\lim_{k\to\infty} b_k/a_k = 0$ to ensure that the state approximator converges faster. Instead of the TD(1) update in (2.60), we can also use TD(λ) update with eligibility traces to reduce variance [117].

3. **Actor**, $\pi_\theta : \widehat{Z} \mapsto \Delta(A)$, which is a function approximator (again a linear function approximator or a feedforward neural network) with AIS $\widehat{Z}_t$ as input and a distribution on actions parameters $\Delta(A_t)$ as output. Let θ denote the parameters of the this policy function approximator. The parameter θ is updated using the policy gradient theorem [67, 118]:

$$\theta_{k+1} = \theta_k + c_k \nabla_\theta J(\xi_k, \varphi_k, \theta_k), \tag{2.62}$$

where the policy gradient is given by

$$\nabla_\theta J(\xi_k, \varphi_k, \theta_k) = \frac{1}{B} \sum_{t=0}^{B-1} Q_{\varphi_k}(\widehat{Z}_t, A_t) \nabla_\theta \log \pi_{\theta_k}(\widehat{Z}_t, A_t)$$

and the learning rate $\{c_k\}_{k\geq 0}$ satisfies $\sum c_k = \infty$ and $\sum c_k^2 < \infty$. In addition, we require $\lim_{k\to\infty} c_k/b_k = 0$ to ensure that the critic learns faster.

The choice of the learning rates implies that there is a separation of timescales between

the updates at the state approximator, the critic, and the actor. Thus, we can show convergence of the algorithm using ideas from [20]. We impose the following standard technical assumptions:

Assumption (A) *1. All network parameters $(\xi_k, \varphi_k, \theta_k)$ lie in convex and bounded subsets of Euclidean spaces.*

2. The gradient of the loss function $\nabla_\xi \mathcal{L}_{AIS}(\xi_k)$ of the state approximator is Lipschitz in ξ_k, the gradient of the TD loss $\nabla_\varphi \mathcal{L}_{TD}(\xi_k, \varphi_k, \theta_k)$ and the policy gradient $\nabla_{\theta_k} J(\xi_k, \varphi_k, \theta_k)$ is Lipschitz in $(\xi_k, \varphi_k, \theta_k)$ in terms of the sup norm.

3. All the gradients—$\nabla_\xi \mathcal{L}_{AIS}(\xi_k)$ at the state approximator; $\nabla_\varphi \mathcal{L}_{TD}(\xi_k, \varphi_k, \theta_k)$ at the critic; and $\nabla_{\theta_k} J(\xi_k, \varphi_k, \theta_k)$ at the actor—are unbiased with bounded variance. Furthermore, the critic and the actor function approximators are compatible as given in [118], i.e.,

$$\frac{\partial Q_{\varphi_k}(\widehat{Z}_t, A_t)}{\partial \varphi} = \frac{1}{\pi_{\theta_k}(\widehat{Z}_t, A_t)} \frac{\partial \pi_{\theta_k}(\widehat{Z}_t, A_t)}{\partial \theta}.$$

4. The learning rates are sequences of positive numbers $\{a_k\}_{k\geq 0}, \{b_k\}_{k\geq 0}, \{c_k\}_{k\geq 0}$ that satisfy:

$$\sum a_k = \infty, \text{ and } \sum a_k^2 < \infty,$$

$$\sum b_k = \infty, \sum b_k^2 < \infty, \text{ and } \lim_{k\to\infty} b_k/a_k = 0,$$

$$\sum c_k = \infty, \sum c_k^2 < \infty, \text{ and } \lim_{k\to\infty} c_k/b_k = 0.$$

The proposed RL framework has the following convergence guarantees.

Theorem 2.6.1 *Suppose in addition to Assumption 1 the following regularity conditions hold: the ODE corresponding to* (2.62) *is locally asymptotically stable and the ODEs corresponding to* (2.59) *and* (2.60) *are globally asymptotically stable with the ODE corresponding to* (2.60) *having a fixed point which is Lipschitz continuous in θ. Then, along any sample path, almost surely we have:* (a) *iteration* (2.59) *converges to a state estimator that minimizes the loss function $\mathcal{L}_{AIS}$;* (b) *iteration* (2.60) *converges to a critic that minimizes the error with respect to the true Q-function; and* (c) *iteration* (2.62) *converges to a local maximum of the performance $J(\xi^*, \varphi^*, \theta)$.*

PROOF The assumptions satisfy all the four conditions stated in [72, page 35], [20, Theorem 23]. The proof follows in a straightforward manner from combining this two-time scale algorithm proof with the fastest third time-scale of learning the state representation. Due to the specific choice of learning rates, the state representation algorithm sees a stationary actor and critic, while the actor and critic in turn see a converged state approximator ietration due to its faster learning rate. The convergence of the state approximator follows from [19, Theorem 2.2] and the fact that the model satisfies conditions (A1)–(A4) of [19, pg 10–11]. The Martinagle difference condition (A3) in this reference is satisfied by the unbiasedness assumption for the state approximator. The result then follows from by combining the theorem given in [72, page 35], [20, Theorem 23] along with [19, Theorem 2.2] and using a third fastest time scale for the state apparoximator. ■

The convergence guarantees of Theorem 5.2.1 provides the following approximation bounds.

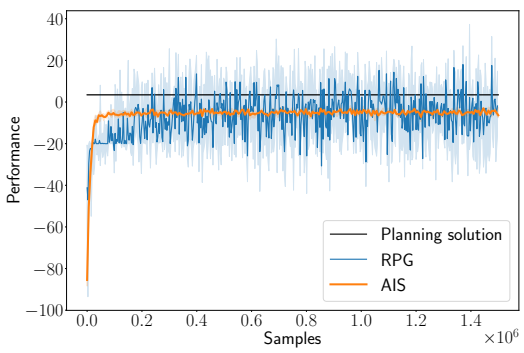
Theorem 2.6.2 *At convergence, let ε and δ be the error constants in* (AP1) *and* (AP2)*, and let $\kappa = \|V_\varphi - \hat{V}\|_\infty$ where V_φ is the converged value function at the critic and $\hat{V}$ is the solution of* (2.45)*. Then, by the triangle inequality, we have for any realization of history H_t,*

$$\left|V(H_t) - V_\varphi(\vartheta_t(H_t))\right| \leq \kappa + \frac{\varepsilon + \gamma L_V \delta}{1-\gamma}.$$

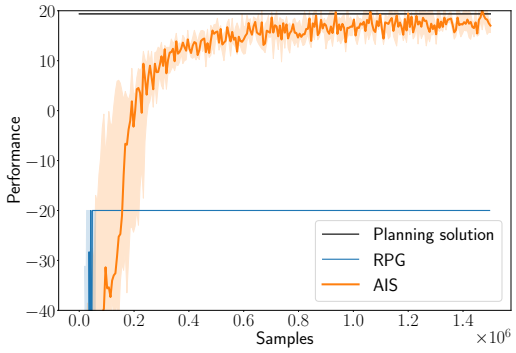
2.7 Numerical examples

In this section, we use the approximate information state based reinforcement learning for four small dimensional POMDP benchmarks: voicemail [132], tiger [60], 4×4 grid [23], and cheese maze [83]. See [102] for the details of the environments. The results for these numerical examples were provided by Amit Sinha. For these numerical experiments, we use Actor only methods, i.e, the Critic is replaced with a Monte Carlo return estimator. We use the approach described in Sec. 2.6.2, with the following choices for the networks:

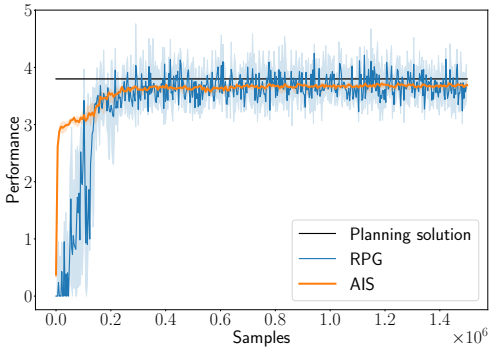
- The ρ network is a two layer recurrent neural network, where the input is one-hot



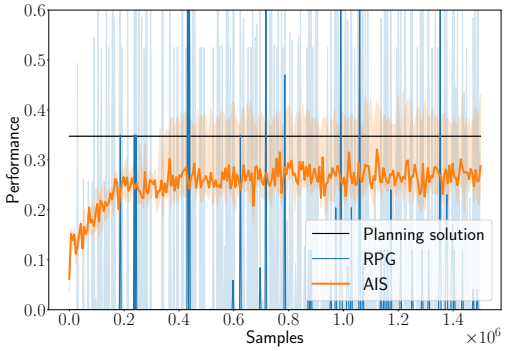
(a) Voicemail



(b) Tiger



(c) 4×4 grid



(d) Cheese maze

Fig. 2.4: Performance versus samples for all the problems. The solid line shows the median value and the shaded region shows the region between the first and third quartiles over 10 runs.

encoded[2], the first layer is a fully connected layer with 40 neurons and tanh activation and the second layer is an LSTM layer with 40 neurons. This network outputs $\widehat{Z}_t$ as the state of the LSTM cell.

- The ψ network has two parts—one for $\tilde{R}_t$ and one for $\tilde{\nu}_{t+1}$. Both these parts are two layer feedforward neural networks, where the input is an approximate information state and one-hot encoded action, the first layer for both these parts is a fully connected layer with 20 neurons and relu activation and the second layer is a fully connected layer with a single neuron for $\tilde{R}_t$ and neurons equal to the observation dimension ($|\mathcal{Y}|$) with softmax activation for $\tilde{\nu}_{t+1}$. This network outputs $\tilde{R}_t$ and a discrete probability distribution for $\tilde{\nu}_{t+1}$, which represents the probability distribution of the next observation.

- The policy network π_θ is a two layer feedforward neural network, where the input is an approximate information state, the first layer is a fully connected layer with 40 neurons and relu activation and the second layer is a fully connected layer with $|\mathcal{A}|$ neurons. The output of this network are the parameters of a $|\mathcal{A}|$ dimensional softmax distribution.

We train these networks for $\gamma = 0.95$ (except for cheese maze, where $\gamma = 0.7$), $\lambda = 0.1$, $B = 300$ and $\{a_k\}_{k\geq 1}$ and $\{b_k\}_{k\geq 1}$ chosen according to ADAM(0.005) and ADAM(0.003) [65] respectively. The performance gradients are estimated using REINFORCE [133]. The plots for 5000 iterations of the algorithm for all 4 problems are shown in Figure 2.4. These figures show the performance plots for our algorithm labelled AIS, the planning solution and one of the state-of-the-art Actor only algorithms for POMDPs called recurrent policy gradient (RPG) [131] for all four examples. The data for RPG is taken from [102]. The experiments with RPG used a two layer RNN for the policy function, where the first layer is a recurrent LSTM layer with 20 neurons and the second layer is a fully connected layer with 20 neurons. This outputs parameters for the softmax function to obtain distributions over actions. The RPG implementation also included a history dependent baseline for variance reduction. This was a two layer RNN where the first layer is a recurrent LSTM layer with 20 neurons

[2]In one-hot encoding an input belonging to a finite indexed set of cardinality N is converted into an N-dimensional vector representation. Suppose i is the index of the input. Then the output vector v has all elements as zero except for $v[i]$, which has the value 1. Hence, only one element of this vector has a non-zero value (i.e. $v[i] = 1$) and hence this is called a one-hot encoding.

and the second layer is a fully connected layer with a single neuron. In all three examples, our algorithm performed better than or as good as RPG.

2.8 Conclusion

In this chapter, we present two notions of information states for partially observed systems. We show that both these information states are sufficient for dynamic programming, with the second information state definition being a refinement of the more general first information state definition. We then relax the definition to describe an Approximate information state (AIS) that can be used to identify an approximately optimal policy. We also present an extension of the concept of AIS to stochastic compressions of history.

We compare the AIS concept with various approximation approaches in literature including approaches such as state compression, action compression, observation compression (world models), predictive state representations, bisimulation based approaches, deep MDPs.

The approximate information state is defined in terms of properties that can be estimated from data, so it can be used to develop sampling based reinforcement learning algorithms. We present an RL algorithm that learns an AIS from data and uses it for learning an optimal policy. We state and prove the convergence of this algorithm and also demonstrate its performance versus one of the state of the art RL algorithms for POMDPs called recurrent policy gradient (RPG) in four toy problems.

This page is intentionally left blank.

Chapter 3

RL using AIS for mean-field teams

3.1 Introduction

In this chapter, we develop reinforcement learning (RL) algorithms for a class of multi-agent systems called mean-field teams (MFT). Teams are multi-agent systems where agents have a common goal and receive a common reward at each time step. The team objective is to maximize the expected cumulative reward over a finite horizon or the expected cumulative discounted reward over an infinite horizon. Mean-field teams (MFT)s are teams with homogeneous, anonymous agents such that the agents are coupled in their dynamics and rewards only through the mean-field of the system (i.e., empirical distribution of the agents' state). In our work, we consider MFTs with a mean-field sharing information structure, i.e., each agent knows its local state and the empirical mean-field at each time step as described in [3, 5].

The planning solution for this problem was presented in [3, 5] in which a dynamic programming (DP) decomposition for MFTs is obtained using a decomposition approach from literature called the common information approach, which splits the decision making process into a centralized coordination rule that yields prescriptions to be followed by each agent based on their local information. Using the concept of approximate information state (AIS), we relate the solution of this finite agent problem to the solution of the corresponding mean-field limit problem, where we assume that the number of agents is infinite. Under certain regularity conditions, we show that the use of the mean-field limit solution in the finite population system results in bounded sub-optimality. Using this relation, we also bound the sub-optimality in using the solution obtained using an m agent system in a system

with n agents. We develop two RL approaches for MFTs using the AIS approach under the assumption of parametrized prescriptions based on the availability of either a system-level simulator or an agent-level simulator. In the former case, the AIS is the statistical mean-field, and in the latter it is the empirical mean-field of a system with possibly a different number of agents, which we refer to as an approxmate empirical mean-field. In addition to using the statistical or an approximate empirical mean-field as an AIS, we consider the parameters as actions and use conventional RL algorithms to solve the DP. We show that the performance sub-optimality of using the two proposed AIS processes are bounded and of the order of $O(\frac{1}{\sqrt{n}})$, where n is the number of agents. A similar result was recently presented in [4] for planning in MFTs.

Having presented an RL algorithm and the associated performance analysis, we illustrate their empirical performance through two examples based on stylized models of the demand response problem in smart grids and malware spread in networks. For the demand response example, we compute the planning solution as given in [3, 5] for comparison with the RL solutions. For variants of this problem with different numbers of agents ranging from 100 to 1000, we also numerically estimate the sub-optimality of using a mean-field limit RL solution and the sub-optimality of using an RL solution for the 100-agent system in the systems with varying number of agents.

3.1.1 Notation

Random variables are denoted by uppercase letters, their realization by the corresponding lowercase letter and their state spaces are denoted by corresponding calligraphic letters. Subscripts are used to index time while superscripts are used to index agents. $S_{1:t}$ is a shorthand for the vector $(S_1, \dots, S_t)$. $\mathbb{P}(\cdot)$ denotes probability of an event, $\mathbb{E}[\cdot]$ denotes expectation of a random variable, and $\mathbb{1}\{\cdot\}$ denotes the indicator function of a statement.

Given a finite set $\mathcal{S}$, $\Delta(\mathcal{S})$ denotes the set of all probability mass functions on $\mathcal{S}$. For a positive integer n, $\Delta^{(n)}(\mathcal{S})$ denotes the probability mass functions on $\mathcal{S}$ which can be normalized to a denominator of n, i.e.,

$$\Delta^{(n)}(\mathcal{S}) = \{z \in \Delta(\mathcal{S}) : \text{for all } s \in \mathcal{S}, nz_s \in \mathbb{Z}_{\geq 0}\}$$

Given a vector $s = (s^1 \dots s^n) \in \mathcal{S}^n$ of size n, $\xi(s) = \left(\sum_{i \in n} \delta_{s^i}\right)/n \in \Delta^{(n)}(\mathcal{S})$ denotes the empirical mean-field of s. Given a mean-field $z \in \Delta^{(n)}(\mathcal{S})$, $\Xi^{(n)}(z) = \{s \in \mathcal{S}^n : \xi(s) = z\}$.

denotes all vectors s of size n with empirical mean-field z.

3.2 System model and problem formulation

Consider a multi-agent team with n agents, indexed by the set $N = \{1, \dots, n\}$. The team operates in discrete time for a infinite horizon T. Let $S_t^i \in \mathcal{S}$ and $A_t^i \in \mathcal{A}$ denote the state and action of agent $i \in N$ at time t. Let $\mathbf{S}_t = (S_t^1, \dots, S_t^n) \in \mathcal{S}^n$ and $\mathbf{A}_t = (A_t^1, \dots, A_t^n) \in \mathcal{A}^n$ denote the state and action of all agents. Let $Z_t = \xi(\mathbf{S}_t)$ denote the empirical mean-field of the team at time t. The state space $\mathcal{S}$ and action space $\mathcal{A}$ are finite sets and are the same for all agents.

The initial states of all agents are independent, i.e., for any realization $\mathbf{s}_0 = (s_0^1, \dots, s_0^n) \in \mathcal{S}^n$, we have

$$\mathbb{P}(\mathbf{S}_0 = \mathbf{s}_0) = \prod_{i \in N} \mathbb{P}(S_0^i = s_0^i) =: \prod_{i \in N} P_0(s_0^i),$$

where P_0 denotes the initial state distribution of agents.

The global state evolves in a controlled Markov manner and the agents are exchangeable, so the evolution of a generic agent depends on the states and actions of other agents only through the empirical mean-field of the state, i.e., for any realizations $\{s_t\}_{t \geq 0}$ and $\{a_t\}_{t \geq 0}$ of $\{S_t\}_{t \geq 0}$ and $\{A_t\}_{t \geq 0}$, we have

$$\begin{aligned}
&\mathbb{P}(\mathbf{S}_{t+1} = \mathbf{s}_{t+1} \mid \mathbf{S}_{0:t} = \mathbf{s}_{1:t}, \mathbf{A}_{0:t} = \mathbf{a}_{0:t}) \\
&\quad = \mathbb{P}(\mathbf{S}_{t+1} = \mathbf{s}_{t+1} \mid \mathbf{S}_t = \mathbf{s}_t, \mathbf{A}_t = \mathbf{a}_t) \\
&\quad = \prod_{i \in N} \mathbb{P}(S_{t+1}^i = s_{t+1}^i \mid S_t^i = s_t^i, A_t^i = a_t^i, Z_t = \xi(\mathbf{s}_t)) \\
&\quad =: \prod_{i \in N} p(s_{t+1}^i \mid s_t^i, a_t^i, \xi(s_t)),
\end{aligned} \tag{3.1}$$

where $p \colon \mathcal{S} \times \mathcal{A} \times \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ denotes the controlled transition matrix.

The system has mean-field sharing information-structure, i.e., the information available to agent i is given by:

$$I_t^i = \{S_t^i, Z_t\}. \tag{3.2}$$

Agents use identical[1] (stochastic) control law $\pi_t \colon \mathcal{S} \times \Delta(\mathcal{S}) \to \Delta(\mathcal{A})$ to choose the control

[1] In general, restricting attention to identical policies may lead to a loss of optimality. See [5] for an

action at time t, i.e.,

$$A_t^i \sim \pi_t(S_t^i, Z_t). \tag{3.3}$$

We also use the notation $\pi_t(a_t^i|s_t^i, z_t)$ to denote the probability of choosing action a_t^i in state (s_t^i, z_t) when following policy π_t.

The team receives a per-step reward given by

$$R_{\texttt{team}}(\mathbf{S}_t, \mathbf{A}_t) = \frac{1}{n} \sum_{i \in N} r(S_t^i, A_t^i, \xi(\mathbf{S}_t)), \tag{3.4}$$

where $r \colon \mathcal{S} \times \mathcal{A} \times \Delta(\mathcal{S}) \to \mathbb{R}_{\geq 0}$ is the per-agent reward function.

Given a policy $\pi = (\pi_0, \pi_1, \dots, \pi_T)$ for the entire horizon, the expected total reward incurred by the team is given by

$$J(\pi) = \mathbb{E}^{\pi} \left[\sum_{t=0}^{T} R_{\texttt{team}}(\mathbf{S}_t, \mathbf{A}_t) \right], \tag{3.5}$$

We are interested in the following optimization problem.

Problem 1 *Given the sets $\mathcal{S}$ and $\mathcal{A}$, the number n of agents, the initial distribution P_0, the controlled transition matrix p and the reward function r, choose a policy $\pi^* = (\pi_0^*, \pi_1^*, \dots, \pi_T^*)$ to maximize the performance $J(\pi)$ given by* (3.5)*, i.e.,*

$$J^* = J(\pi^*) = \max_{\pi} J(\pi), \tag{3.6}$$

where the maximum is over all policies of the form 3.3.

Problem 1 is a decentralized control problem with a non-clssical information structure. A planning solution for this problem was proposed in [3, 5], which we describe in the next section. We then define the corresponding mean-field limit problem, where we assume that the number of agents $n \to \infty$ and also describe its planning solution. We then relate the solutions of these two problems by proposing that the transition and reward functions of the mean-field limit problem be used in the generator for an AIS process for the finite agent problem, Problem 1

example. Nonetheless, identical policies are attractive for reasons of fairness, simplicity, and robustness.

3.3 Planning solution for Problem 1

In this section we describe the planning solution for Problem 1 presented in [3, 5].

Given any policy $\pi = (\pi_0, \pi_1, \dots, \pi_T)$ and any realization, $z = (z_0, z_1, \dots, z_T)$ of the mean-field, define *prescriptions* $g_t : \mathcal{S} \to \Delta(\mathcal{A})$ given by

$$g_t(s) = \pi_t(s, z_t), \quad \forall s \in \mathcal{S}.$$

When the mean field trajectory is a random process, the prescription g_t is a random function which we denote G_t. The results of [3, 5] rely on the following two key properties. Let $(z_{0:t+1}, g_{1:t})$ denote any realization of $(Z_{0:t+1}, G_{0:t})$. Then, we have:

1. [5, Lemma 4] $\{Z_t\}_{t=0}^T$ is a controlled Markov process with control action G_t i.e.,

$$\begin{aligned}
&\mathbb{P}^\pi(Z_{t+1} = z_{t+1} \mid Z_{0:t} = z_{0:t}, G_{0:t} = g_{0:t}) \\
&\quad = \mathbb{P}(Z_{t+1} = z_{t+1} \mid Z_t = z_t, G_t = g_t) \\
&\quad =: P(z_{t+1} \mid z_t, g_t).
\end{aligned} \tag{3.7}$$

   Furthermore, we can show that for any $\mathbf{s}_t = (s_t^1, \dots, s_t^n) \in \Xi(z_t)$, we have

$$\begin{aligned}
&P(z_{t+1} \mid z_t, g_t) = \\
&\sum_{\mathbf{s}_{t+1} \in \Xi(z_{t+1})} \prod_{i \in N} \sum_{a \in \mathcal{A}} p(s_{t+1}^i | s_t^i, a, z_t) g_t(a | s_t^i),
\end{aligned} \tag{3.8}$$

   where $g_t(a|s_t^i) = \mathbb{P}(A_t = a | g_t, s_t^i)$ denotes the probability of taking action a given a prescription g_t and local state s_t^i, which is a slight abuse of notation.

2. [5, Lemma 3] The expected per-step reward simplifies as follows.

$$\begin{aligned}
&\mathbb{E}[R_{\texttt{team}}(\mathbf{S}_t, \mathbf{A}_t) \mid Z_{0:t} = z_{0:t}, G_{0:t} = g_{0:t}] \\
&\quad = \mathbb{E}[R_{\texttt{team}}(\mathbf{S}_t, \mathbf{A}_t) \mid Z_t = z_t, G_t = g_t] \\
&\quad = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} r(s, a, z) g(a|s) z(s) \\
&\quad =: R(z_t, g_t).
\end{aligned} \tag{3.9}$$

As shown in [5, Theorem 1], these two properties imply that the optimal policy π^* can be identified via the following dynamic program.

Theorem 3.3.1 *Define value function* $\{V_t : \mathcal{Z} \to \mathbb{R}\}_{t=0}^{T+1}$ *as follows:* $V_{T+1}(z) = 0$ *and for* $t \in \{T, T-1, \dots, 0\}$*, we have*

$$V_t(z) := \max_{g\colon \mathcal{S}\to\Delta(\mathcal{A})} \mathbb{E}\big[R(z,g) + V_{t+1}(Z_{t+1}) \mid Z_t = z, G_t = g\big]. \tag{3.10}$$

Let $\tilde{\psi}_t^*(z)$ *denote an* $\arg\max$ *of the right hand side of* (3.10)*. Then the policy,* $\pi^* = (\pi_1^*, \dots, \pi_T^*)$*, where*

$$\pi_t^*(s,z) = \tilde{\psi}_t^*(z)(s), \tag{3.11}$$

is optimal for Problem 1, i.e., it maximizes (3.6)*.*

3.4 Mean-field limits

Problem 1 can be computationally simplified by assuming an infinite population system, i.e., the mean-field limit system which has a deterministic evolution. This mean-field limit assumption is also useful when the agents do not know the exact number of agents in the team.

3.4.1 Model and problem formulation

We now consider the infinite-population mean-field limit of the controlled Markov process given by (3.7) and (3.9). In this case, the information available to agent i can be written as:

$$\bar{I}_t^i = \{S_t^i, \bar{z}_t\} \tag{3.12}$$

where $\bar{z}_t \in \bar{\mathcal{Z}}$ is the statistical mean-field at time t. In this problem as well, the agents use identical stochastic control laws $\bar{\pi}_t : \mathcal{S} \times \Delta(\mathcal{S}) \to \Delta(\mathcal{A})$ to choose the control action at time t, i.e.,

$$A_t^i \sim \bar{\pi}_t(S_t^i, \bar{Z}_t). \tag{3.13}$$

Let $\bar{\pi} = (\bar{\pi}_0, \bar{\pi}_1, \dots, \bar{\pi}_T)$ denote the team policy for all time. As before, we define prescriptions as:

$$\bar{g}_t(s) = \bar{\pi}_t(s, \bar{z}_t), \quad \forall s \in \mathcal{S}.$$

For any presciption $\bar{g}\colon \mathcal{S} \to \Delta(\mathcal{A})$, define an operator $\mathcal{P}_{\bar{g}}\colon \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ as follows: for any $\bar{z} \in \Delta(\mathcal{S})$ and any $s' \in S$,

$$[\mathcal{P}_{\bar{g}}\bar{z}](s') = \sum_{s \in \mathcal{S}} \bar{z}(s) p(s' \mid s, \bar{g}, \bar{z}).$$

The mean-field limit of the controlled Markov process given by (3.7) and (3.9) is a distribution valued controlled Markov process $\{\bar{z}_t\}_{t=0}^{T}$. The initial state $\bar{z}_0$ of the process is P_0. At any time $t \geq 0$ and a choice $\bar{g}_t\colon \mathcal{S} \to \Delta(\mathcal{A})$, the state evolves in a deterministic manner as

$$\bar{z}_{t+1} = \mathcal{P}_{\bar{g}_t}\bar{z}_t. \tag{3.14}$$

and the system receives a reward $R(\bar{z}_t, \bar{g})$. Given any policy $\bar{\psi} = (\bar{\psi}_0, \ldots, \bar{\psi}_T)$, where

$$\bar{g}_t = \bar{\psi}_t(\bar{z}_t), \tag{3.15}$$

the expected total reward obtained by the system is given by

$$\bar{J}(\bar{\pi}) = \sum_{t=0}^{T} R(\bar{z}_t, \bar{g}). \tag{3.16}$$

The mean-field limit optimization problem is the following.

Problem 2 *Given the sets $\mathcal{S}$ and $\mathcal{A}$, the initial distribution P_0, the controlled transition matrix p, and the reward function R, choose a policy $\bar{\pi}^* = (\bar{\pi}_0^*, \bar{\pi}_1^*, \ldots, \bar{\pi}_T^*)$ to maximize the performance $\bar{J}(\bar{\pi})$ given by* (3.16)*, i.e,*

$$\bar{J}^* = \bar{J}(\bar{\pi}^*) = \max_{\bar{\pi}} \bar{J}(\bar{\pi}), \tag{3.17}$$

where the maximum is over all policies of the form (3.15)*.*

Problem 2 is a deterministic control problem whose solution is given as follows.

Theorem 3.4.1 *Define value function $\{\bar{V}_t : \mathcal{Z} \to \mathbb{R}\}_{t=0}^{T+1}$ as follows: $\bar{V}_{T+1}(z) = 0$ and for $t \in \{T, T-1, \ldots, 0\}$, we have*

$$\bar{V}_t(z) := \max_{\bar{g}\colon \mathcal{S} \to \Delta(\mathcal{A})} \{\tilde{R}(z, \bar{g}) + \bar{V}_{t+1}(\mathcal{P}_{\bar{g}} z)\}. \tag{3.18}$$

Let $\bar{\psi}_t^*(z)$ *denote an* $\arg\max$ *of the right hand side of* (3.18)*. Then the policy,* $\bar{\psi}^* = (\bar{\psi}_1^*, \ldots, \bar{\psi}_T^*)$ *is optimal for Problem 2, i.e., it maximizes* (3.17)*.*

The relation between the value functions of these two problems is given in the next sub-section. This is followed by a theorem relating the performance of an optimal policy for Problem 2 in the system defined in Problem 1.

3.5 Approximation bounds

3.5.1 Preliminaries on Lipschitz continuity

Our results in the sequel will be based on Lipschitz continuity, Wasserstein distance and the Kantorovich-Rubinstein duality. Hence, in this section we give state some basic definitions and results.

Let $\bar{z}_1, \bar{z}_2 \in \Delta(\mathcal{S})$ be two distributions with a finite support $\mathcal{S}$. We assume that the space $\mathcal{S}$ is a metric space (and denote the metric by d_s) and equip the probability space $\Delta(\mathcal{S})$ with the Wasserstein metric, which we denote by $d_{\bar{z}}$. In particular, for any $\bar{z}_1, \bar{z}_2 \in \Delta(\mathcal{S})$,

$$d_{\bar{z}}(\bar{z}_1, \bar{z}_2) = \inf_{\lambda \in \Lambda(\bar{z}_1, \bar{z}_2)} \sum_{s_1, s_2 \in \mathcal{S}} \lambda(s_1, s_2) d_s(s_1, s_2),$$

where $\Lambda(\bar{z}_1, \bar{z}_2)$ denotes the collection of probability mass functions on $\mathcal{S} \times \mathcal{S}$ with marginals $\bar{z}_1$ and $\bar{z}_2$ on the first and the second factors, respectively. Then the Kantorovich-Rubinstein duality [124] can be stated as:

$$d_{\bar{z}}(\bar{z}_1, \bar{z}_2) = \sup_f \Big\{ \Big| \int_{s \in \mathcal{S}} f(s) d\bar{z}_1(s) - \int_{s \in \mathcal{S}} f(s) d\bar{z}_2(s) \Big| \Big\}, \tag{3.19}$$

where the supremum is taken over all Lipschitz continuous functions f with a Lipschitz constant $L_f \leq 1$. When the space $\mathcal{S}$ is discrete, the Kantorovich-Rubinstein duality reduces to:

$$d_{\bar{z}}(\bar{z}_1, \bar{z}_2) = \sup_f \Big\{ \Big| \sum_{s \in \mathcal{S}} f(s) \bar{z}_1(s) - \sum_{s \in \mathcal{S}} f(s) \bar{z}_2(s) \Big| \Big\}, \tag{3.20}$$

which can be written as:

$$\Big| \sum_{s \in \mathcal{S}} f(s) \bar{z}_1(s) - \sum_{s \in \mathcal{S}} f(s) \bar{z}_2(s) \Big| \leq d_{\bar{z}}(\bar{z}_1, \bar{z}_2). \tag{3.21}$$

Furthermore, we can extend the above inequality to Lipschitz continuous functions that have a Lipschitz constant L as:

$$\left|\sum_{s\in\mathcal{S}} f(s)\bar{z}_1(s) - \sum_{s\in\mathcal{S}} f(s)\bar{z}_2(s)\right| \leq L d_{\bar{z}}(\bar{z}_1, \bar{z}_2), \tag{3.22}$$

where f is a Lipschitz continuous function with Lipschitz constant L. Another extension of the above inequality to functions of more than one variable can be given by:

$$\left|\sum_{s\in\mathcal{S}} f(s,y)\bar{z}_1(s) - \sum_{s\in\mathcal{S}} f(s,y)\bar{z}_2(s)\right| \leq L d_{\bar{z}}(\bar{z}_1, \bar{z}_2), \tag{3.23}$$

where f is a Lipschitz continuous function in its first variable (s) for any given y, with Lipschitz constant L. A final extension of this inequality that we will use in our proofs is an extension to a function f of multiple arguments and conditional distributions $\bar{z}(s|y_1), \bar{z}(s|y_2)$ given by:

$$\left|\sum_{s\in\mathcal{S}} f(s,y)\bar{z}(s|y_1) - \sum_{s\in\mathcal{S}} f(s,y)\bar{z}(s|y_2)\right| \leq L d_{\bar{z}}(\bar{z}(s|y_1), \bar{z}(s|y_2)), \tag{3.24}$$

where f is a Lipschitz continuous function in its first variable (s) for any given y, with Lipschitz constant L.

3.5.2 Lipschitz continuity of the reward R, transition function $\mathcal{P}_{\bar{g}_t}$ and the value function $\bar{V}$

In order to derive approximation bounds, we impose certain technical assumptions on the model. We assume that the state space $\mathcal{S}$ is a metric space (and denote the metric by d_s) and equip the probability space $\Delta(\mathcal{S})$ with the Wasserstein metric, which we denote by $d_{\bar{z}}$.

We also assume that the action space $\mathcal{A}$ is a metric space (and denote the metric by d_a) and equip the probability space $\Delta(\mathcal{A})$ with the Wasserstein metric, which we denote by $d_{\bar{g}}$. We say that a prescription $\bar{g}\colon \mathcal{S} \to \Delta(\mathcal{A})$ is Lipschitz continuous with Lipschitz constant $L_{\bar{g}}$ if for any $s_1, s_2 \in \mathcal{S}$,

$$d_{\bar{g}}(\bar{g}(\cdot|s_1), \bar{g}(\cdot|s_2)) \leq L_{\bar{g}} d_s(s_1, s_2).$$

Since $\mathcal{S}$ is a finite set, all prescriptions are Lipschitz continuous with some Lipschitz constant.

We make the following assumptions on the model.

(A1) The per-step reward function r is uniformly bounded between 0 and $R_{\max}$.[2]

(A2) For any $\bar{z} \in \Delta(\mathcal{S})$ and $a \in \mathcal{A}$, the function $r(\cdot, a, \bar{z})\colon \mathcal{S} \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant L_{sr}.

(A3) For any $s \in \mathcal{S}$ and $\bar{z} \in \Delta(\mathcal{S})$, the function $r(s, \cdot, \bar{z})\colon \mathcal{A} \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant L_{ar}.

(A4) For any $s \in \mathcal{S}$ and $a \in \mathcal{A}$, the function $r(s, a, \cdot)\colon \Delta(\mathcal{S}) \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant $L_{\bar{z}r}$.

(A5) For any $\bar{z} \in \Delta(\mathcal{S})$ and $a \in \mathcal{A}$, the function $P(S^+|\cdot, a, \bar{z})\colon \mathcal{S} \to \Delta(\mathcal{S})$ is Lipschitz continuous with Lipschitz constant L_{sP} with respect to the Wasserstein metric, i.e., for any $s_1, s_2 \in \mathcal{S}$, we have:

$$\mathcal{K}(P(S^+|s_1, a, \bar{z}), P(S^+|s_2, a, \bar{z})) \leq L_{sP} d_s(s_1, s_2), \tag{3.25}$$

where $\mathcal{K}(\mu, \nu)$ denotes the Wasserstein distance between distributions μ and ν.

(A6) For any $s \in \mathcal{S}$ and $\bar{z} \in \Delta(\mathcal{S})$, the function $P(S^+|s, \cdot, \bar{z})\colon \mathcal{A} \to \Delta(\mathcal{S})$ is Lipschitz continuous with Lipschitz constant L_{aP} with respect to the Wasserstein metric, i.e., for any $a_1, a_2 \in \mathcal{A}$, we have:

$$\mathcal{K}(P(S^+|s, a_1, \bar{z}), P(S^+|s, a_2, \bar{z})) \leq L_{aP} d_a(a_1, a_2). \tag{3.26}$$

(A7) For any $s \in \mathcal{S}$ and $a \in \mathcal{A}$, the function $P(S^+|s, a, \cdot)\colon \Delta(\mathcal{S}) \to \Delta(\mathcal{S})$ is Lipschitz continuous with Lipschitz constant $L_{\bar{z}P}$ with respect to the Wasserstein metric, i.e., for any $\bar{z}_1, \bar{z}_2 \in \Delta(\mathcal{S})$, we have:

$$\mathcal{K}(P(S^+|s, a, \bar{z}_1), P(S^+|s, a, \bar{z}_2)) \leq L_{\bar{z}P} d_{\bar{z}}(\bar{z}_1, \bar{z}_2). \tag{3.27}$$

For ease of notation, we define the following: a transition kernel $\bar{p}\colon \mathcal{S} \times [\mathcal{S} \to \Delta(\mathcal{A})] \times$

[2]Note that any bounded reward function with negative rewards can be translated into this form without affecting the optimal policy.

$\Delta(\mathcal{S}) \to \Delta(\mathcal{S})$, which is given as follows: for any $s, s_+ \in \mathcal{S}$, $\bar{g}\colon \mathcal{S} \times \Delta(\mathcal{S}) \to \mathcal{A}$, and $z \in \Delta(\mathcal{S})$,

$$\bar{p}(s_+|s, \bar{g}, z) = \sum_{a \in \mathcal{A}} p(s_+|s, a, z)\bar{g}(a|s);$$

and a function $\bar{r}\colon \mathcal{S} \times [\mathcal{S} \to \Delta(\mathcal{A})] \times \Delta(\mathcal{S}) \to \mathbb{R}_{\geq 0}$, which is given as follows: for any $s \in \mathcal{S}$, $\bar{g}\colon \mathcal{S} \to \mathcal{A}$, and $z \in \Delta(\mathcal{S})$,

$$\bar{r}(s, \bar{g}, z) = \sum_{a \in \mathcal{A}} r(s, a, z)\bar{g}(a|s).$$

where we use $\bar{g}(a \mid s)$ to denote $\mathbb{P}(a|s; \bar{g})$, which is a slight abuse of notation.

Lemma 3.5.1 *For any Lipschitz continuous prescription $\bar{g}\colon \mathcal{S} \to \Delta(\mathcal{A})$ with Lipschitz constant $L_{\bar{g}}$ and any $\bar{z} \in \Delta(\mathcal{S})$, the function $\bar{r}(\cdot, \bar{g}, \bar{z})\colon \mathcal{S} \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant $L_{sr} + L_{ar}L_{\bar{g}}$.* □

PROOF Consider $s_1, s_2 \in \mathcal{S}$. From the triangle inequality, we have

$$\begin{aligned}
&\left|\bar{r}(s_1, \bar{g}, \bar{z}) - \bar{r}(s_2, \bar{g}, \bar{z})\right| \\
&\quad = \left|\sum_{a \in \mathcal{A}} r(s_1, a, \bar{z})\bar{g}(a \mid s_1) - \sum_{a \in \mathcal{A}} r(s_2, a, \bar{z})\bar{g}(a \mid s_2)\right| \\
&\quad \leq \left|\sum_{a \in \mathcal{A}} r(s_1, a, \bar{z})\bar{g}(a \mid s_1) - \sum_{a \in \mathcal{A}} r(s_2, a, \bar{z})\bar{g}(a \mid s_1)\right| \\
&\qquad + \left|\sum_{a \in \mathcal{A}} r(s_2, a, \bar{z})\bar{g}(a \mid s_1) - \sum_{a \in \mathcal{A}} r(s_2, a, \bar{z})\bar{g}(a \mid s_2)\right| \\
&\quad \overset{(a)}{\leq} L_{sr} d_s(s_1, s_2) + L_{ar} d_{\bar{g}}(\bar{g}(a \mid s_1), \bar{g}(a \mid s_2)) \\
&\quad \overset{(b)}{\leq} L_{sr} d_s(s_1, s_2) + L_{ar} L_{\bar{g}} d_s(s_1, s_2),
\end{aligned}$$

where the second term in (a) follows from Kantorovich-Rubinstein duality given by (3.24), as $r(s_2, a, \bar{z})$ is Lipschitz continuous with respect to a for any s_2 and $\bar{z}$ with Lipschitz constant L_{ar} as per assumption (A3) and the second term in (b) follows from the fact that the prescription is Lipschitz continuous. ■

Now, consider the function R defined in (3.9), which can be written as $R(\bar{z}, \bar{g}) = \sum_{s \in \mathcal{S}} \bar{z}(s)\bar{r}(s, \bar{g}, \bar{z})$.

Lemma 3.5.2 *For any Lipschitz continuous prescription $\bar{g}$ with Lipschitz constant $L_{\bar{g}}$, the function $R(\cdot, \bar{g})$ given by (3.9) is Lipschitz continuous with Lipschitz constant $L_R = L_{\bar{z}r} + L_{sr} + L_{ar}L_{\bar{g}}$.* □

PROOF Consider any $\bar{z}_1, \bar{z}_2 \in \Delta(\mathcal{S})$. From the triangle inequality, we have

$$
\begin{aligned}
&\big|R(\bar{z}_1, \bar{g}) - R(\bar{z}_2, \bar{g})\big| \\
&\quad = \big|\sum_{s\in\mathcal{S}} \bar{z}_1(s)\bar{r}(s, \bar{g}, \bar{z}_1) - \sum_{s\in\mathcal{S}} \bar{z}_2(s)\bar{r}(s, \bar{g}, \bar{z}_2)\big| \\
&\quad \le \big|\sum_{s\in\mathcal{S}} \bar{z}_1(s)\bar{r}(s, \bar{g}, \bar{z}_1) - \sum_{s\in\mathcal{S}} \bar{z}_1(s)\bar{r}(s, \bar{g}, \bar{z}_2)\big| + \big|\sum_{s\in\mathcal{S}} \bar{z}_1(s)\bar{r}(s, \bar{g}, \bar{z}_2) - \sum_{s\in\mathcal{S}} \bar{z}_2(s)\bar{r}(s, \bar{g}, \bar{z}_2)\big| \\
&\quad \le \sum_{s\in\mathcal{S}} \bar{z}_1(s)\big|\bar{r}(s, \bar{g}, \bar{z}_1) - \bar{r}(s, \bar{g}, \bar{z}_2)\big| + \big|\sum_{s\in\mathcal{S}} \bar{z}_1(s)\bar{r}(s, \bar{g}, \bar{z}_2) - \sum_{s\in\mathcal{S}} \bar{z}_2(s)\bar{r}(s, \bar{g}, \bar{z}_2)\big| \\
&\quad \overset{(a)}{\le} L_{\bar{z}r} d_{\bar{z}}(\bar{z}_1, \bar{z}_2) + (L_{sr} + L_{ar}L_{\bar{g}}) d_{\bar{z}}(\bar{z}_1, \bar{z}_2),
\end{aligned}
$$

where the first term in (a) follows from the Lipschitz continuity of r with respect to its third argument $\bar{z}$ (assumption A4) and the second term follows from the Kantorovich-Rubinstein duality given by (3.23), where we use the fact that $\bar{r}(s, \bar{g}, \bar{z}_2)$ is Lipschitz continuous in its first parameter with Lipschitz constant $L_{sr} + L_{ar}L_{\bar{g}}$ from Lemma 3.5.1. ■

Lemma 3.5.3 *For any Lipschitz continuous prescription $\bar{g}\colon \mathcal{S} \to \Delta(\mathcal{A})$ with Lipschitz constant $L_{\bar{g}}$, any $s^+ \in \mathcal{S}$ and any $\bar{z} \in \Delta(\mathcal{S})$, the function $\bar{p}(s^+|\cdot, \bar{g}, \bar{z})\colon \mathcal{S} \to [0, 1]$ is Lipschitz continuous with Lipschitz constant $L_{sP} + L_{aP}L_{\bar{g}}$.* □

PROOF Consider $s_1, s_2 \in \mathcal{S}$. From the triangle inequality, we have

$$
\begin{aligned}
&\big|\bar{p}(s^+|s_1, \bar{g}, \bar{z}) - \bar{p}(s^+|s_2, \bar{g}, \bar{z})\big| \\
&\quad = \Big|\sum_{a\in\mathcal{A}} p(s^+|s_1, a, \bar{z})\bar{g}(a \mid s_1) - \sum_{a\in\mathcal{A}} p(s^+|s_2, a, \bar{z})\bar{g}(a \mid s_2)\Big| \\
&\quad \le \Big|\sum_{a\in\mathcal{A}} p(s^+|s_1, a, \bar{z})\bar{g}(a \mid s_1) - \sum_{a\in\mathcal{A}} p(s^+|s_2, a, \bar{z})\bar{g}(a \mid s_1)\Big| \\
&\qquad + \Big|\sum_{a\in\mathcal{A}} p(s^+|s_2, a, \bar{z})\bar{g}(a \mid s_1) - \sum_{a\in\mathcal{A}} p(s^+|s_2, a, \bar{z})\bar{g}(a \mid s_2)\Big| \\
&\quad \overset{(a)}{\le} L_{sP} d_s(s_1, s_2) + L_{aP} d_s(\bar{g}(a \mid s_1), \bar{g}(a \mid s_2))
\end{aligned}
$$

$$\overset{(b)}{\leq} L_{sP} d_s(s_1, s_2) + L_{aP} L_{\bar{g}} d_s(s_1, s_2),$$

where the first term in (a) follows from the Lipschitz continuity of $p(s^+|s, a, \bar{z})$ with respect to its first argument as per assumption (A5) and and the second term follows from the Kantorovich-Rubinstein duality (3.24) and the fact that $p(s^+|s_2, a, \bar{z})$ is Lipschitz continuous in its second argument with Lipschitz constant L_{aP} from assumption (A6), and the second term in (b) follows from the fact that the prescription is Lipschitz continuous. ■

Lemma 3.5.4 *For any Lipschitz continuous prescription $\bar{g}$ with Lipschitz constant $L_{\bar{g}}$, the function $\mathcal{P}_{\bar{g}}\bar{z}$ given by* (3.14) *is Lipschitz continuous with Lipschitz constant $L_{\bar{P}}$, i.e.,*

$$d_{\bar{z}}(\mathcal{P}_{\bar{g}}\bar{z}_1, \mathcal{P}_{\bar{g}}\bar{z}_2) \leq L_{\bar{P}} d_{\bar{z}}(\bar{z}_1, \bar{z}_2),$$

where $L_{\bar{P}} = \frac{diam(\mathcal{S})|\mathcal{S}|}{2}(L_{\bar{z}P} + L_{sP} + L_{aP}L_{\bar{g}})$. □

Proof We first consider the total divergence distance between these distributions:

$$\begin{aligned}
d_{TV}(\mathcal{P}_{\bar{g}}\bar{z}_1, \mathcal{P}_{\bar{g}}\bar{z}_2) &\overset{(a)}{=} \frac{1}{2}\sum_{s^+\in\mathcal{S}}\left[\left|\sum_{s\in\mathcal{S}}[\bar{z}_1(s)\bar{p}(s^+|s,\bar{g},\bar{z}_1)] - \sum_{s\in\mathcal{S}}[\bar{z}_2(s)\bar{p}(s^+|s,\bar{g},\bar{z}_2)]\right|\right] \\
&\overset{(b)}{\leq} \frac{1}{2}\sum_{s^+\in\mathcal{S}}\left[\left|\sum_{s\in\mathcal{S}}[\bar{z}_1(s)\bar{p}(s^+|s,\bar{g},\bar{z}_1)] - \sum_{s\in\mathcal{S}}[\bar{z}_1(s)\bar{p}(s^+|s,\bar{g},\bar{z}_2)]\right|\right. \\
&\quad \left. + \left|\sum_{s\in\mathcal{S}}[\bar{z}_1(s)\bar{p}(s^+|s,\bar{g},\bar{z}_2)] - \sum_{s\in\mathcal{S}}[\bar{z}_2(s)\bar{p}(s^+|s,\bar{g},\bar{z}_2)]\right|\right] \\
&\overset{(c)}{\leq} \frac{1}{2}\sum_{s^+\in\mathcal{S}}\left[\left|L_{\bar{z}P} d_{\bar{z}}(\bar{z}_1, \bar{z}_2) + (L_{sP} + L_{aP}L_{\bar{g}}) d_{\bar{z}}(\bar{z}_1, \bar{z}_2)\right|\right] \\
&\overset{(d)}{=} \frac{|\mathcal{S}|}{2}(L_{\bar{z}P} + L_{sP} + L_{aP}L_{\bar{g}}) d_{\bar{z}}(\bar{z}_1, \bar{z}_2),
\end{aligned} \tag{3.28}$$

where (a) follows from the definition of $\bar{p}$, (b) from adding and subtracting the same term, the first term in (c) from the Lipschitz continuity of $\bar{p}$ with respect to $\bar{z}$ (since $\bar{p}$ is a convex combination of Lipschitz continuous functions of $\bar{z}$, with the constants being independent of $\bar{z}$) and the second term in (c) from the Kantorovich-Rubinstein duality given by (3.23), where we use the fact that $\bar{p}(s^+|s, \bar{g}, \bar{z}_2)$ is Lipschitz continuous in s with Lipschitz constant $L_{sP} + L_{aP}L_{\bar{g}}$ from Lemma 3.5.3, and (d) follows from the fact that the summands do not

depend on $s+$. Then, we have:

$$\begin{aligned} d_{\bar{z}}(\bar{z}_1^+, \bar{z}_2^+) &\leq \texttt{diam}(\mathcal{S}) d_{TV}(\bar{z}_1^+, \bar{z}_2^+) \\ &\leq \frac{\texttt{diam}(\mathcal{S})|\mathcal{S}|}{2}(L_{\bar{z}P} + L_{sP} + L_{aP}L_{\bar{g}}) d_{\bar{z}}(\bar{z}_1, \bar{z}_2). \end{aligned}$$

Thus, we get the desired Lipschitz continuity of the transition probabilities in terms of the Kantorovic metric. ■

We now consider the Lipschitz continuity of the value functions.

Lemma 3.5.5 *Recursively define value functions for Problem 2 as $\bar{V}_{T+1}(\bar{z}) = 0$ and for $t \in [0, T]$:*

$$\begin{aligned} \bar{V}_t(\bar{z}) &= \max_{\bar{g} \in \bar{\mathcal{G}}} \bar{Q}_t(\bar{z}, \bar{g}) \\ \bar{Q}_t(\bar{z}, \bar{g}) &= \bar{R}(\bar{z}, \bar{g}) + \bar{V}_{t+1}(\bar{z}^+) = R(\bar{z}, \bar{g}) + \bar{V}_{t+1}(\mathcal{P}_{\bar{g}}\bar{z}). \end{aligned} \tag{3.29}$$

For any Lipschitz continuous policy $\psi : \Delta(\mathcal{S}) \to [\mathcal{S} \to \Delta(\mathcal{A})]$ with Lipschitz constant L_ψ, the value function $\bar{V}_t : \Delta(\mathcal{S}) \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant $L_V = L_{\bar{R}} \sum_{s=0}^{T-t} L_{\bar{P}}^s$. □

Proof We follow the approach given in [94] modified for the finite horizon case. We first note that the MDP given by Problem 2 is a Lipschitz MDP as defined in [94] since both the reward and transition functions are Lipschitz (from Lemmas 3.5.2, 3.5.4).

We prove the claim by using backward induction.

Basis of induction For $t = T$, any $\bar{z}_1, \bar{z}_2 \in \Delta(\mathcal{S})$, and any $\bar{g} \in \bar{\mathcal{G}}$,

$$|Q_T(\bar{z}_1, \bar{g}) - Q_T(\bar{z}_2, \bar{g})| = |R(\bar{z}_1, \bar{g}) - R(\bar{z}_2, \bar{g})| \leq L_R d_{\bar{z}}(\bar{z}_1, \bar{z}_2) = L_R \sum_{s=0}^{0} L_{\bar{P}}^s d_{\bar{z}}(\bar{z}_1, \bar{z}_2),$$

which follows from Lemma 3.5.2. We see that this satisfies the induction hypothesis. Consequently,

$$\begin{aligned} |V_T(\bar{z}_1) - V_T(\bar{z}_1)| &= |\max_{\bar{g}} Q_T(\bar{z}_1, \bar{g}) - \max_{\bar{g}} Q_T(\bar{z}_2, \bar{g})| \leq \max_{\bar{g}} |(Q_T(\bar{z}_1, \bar{g}) - Q_T(\bar{z}_2, \bar{g}))| \\ &\leq L_R d_{\bar{z}}(\bar{z}_1, \bar{z}_2). \end{aligned}$$

Induction hypothesis For time $t+1$, we have:

$$|Q_{t+1}(\bar{z}_1, \bar{g}) - Q_{t+1}(\bar{z}_2, \bar{g})| \leq L_R \sum_{s=0}^{T-t-1} L_{\bar{P}}^s,$$

and

$$|V_{t+1}(\bar{z}_1) - V_{t+1}(\bar{z}_2)| \leq L_R \sum_{s=0}^{T-t-1} L_{\bar{P}}^s,$$

Induction step For time t:

$$\begin{aligned}
|Q_t(\bar{z}_1, \bar{g}) - Q_t(\bar{z}_1, \bar{g})| &\overset{(a)}{=} |R(\bar{z}_1, \bar{g}) + V_{t+1}(\bar{z}_1^+) - R(\bar{z}_2, \bar{g}) + V_{t+1}(\bar{z}_2^+)| \\
&\overset{(b)}{\leq} |R(\bar{z}_1, \bar{g}) - R(\bar{z}_2, \bar{g})| + |V_{t+1}(\bar{z}_1^+) - V_{t+1}(\bar{z}_2^+)| \\
&\overset{(c)}{\leq} L_R d_{\bar{z}}(\bar{z}_1, \bar{z}_2) + L_R \sum_{s=0}^{T-t-1} L_{\bar{P}}^s d_{\bar{z}}(\bar{z}_1^+, \bar{z}_2^+) \\
&\overset{(d)}{\leq} L_R d_{\bar{z}}(\bar{z}_1, \bar{z}_2) + L_{\bar{P}} L_R \sum_{s=0}^{T-t-1} L_{\bar{P}}^s d_{\bar{z}}(\bar{z}_1, \bar{z}_2) \\
&= L_R \sum_{s=0}^{T-t} L_{\bar{P}}^s d_{\bar{z}}(\bar{z}_1, \bar{z}_2),
\end{aligned}$$

where (a) follows from the fact that this model has deterministic transitions, (b) follows from the triangle inequality, the first term of (c) follows from Lemma 3.5.2 and the second term from the induction hypothesis, and (d) follows from Lemma 3.5.4. We now get:

$$\begin{aligned}
|V_t(\bar{z}_1) - V_t(\bar{z}_2)| = |\max_{\bar{g}} Q_t(\bar{z}_1, \bar{g}) - \max_{\bar{g}} Q_t(\bar{z}_2, \bar{g})| &\leq \max_{\bar{g}} |(Q_t(\bar{z}_1, \bar{g}) - Q_t(\bar{z}_2, \bar{g}))| \\
&\leq L_{\bar{R}} \sum_{s=0}^{T-t} L_{\bar{P}}^s d_{\bar{z}}(\bar{z}_1, \bar{z}_2).
\end{aligned}$$

■

3.5.3 Relation between the solutions of Problems 1 and 2

We define an AIS process for Problem 1 in terms of Problem 2 as follows.

Theorem 3.5.1 *Consider the following AIS generators* $\{(\widehat{\vartheta}_t, \widehat{r}_t, \widehat{p}_t)\}_{t=1}^T$ *given by:*

$$\bar{z}_t = \widehat{\vartheta}_t(z_{1:t}, g_{1:t}) = z_t \tag{3.30}$$

$$\widehat{r}_t(\bar{z}_t, g_t) = R(\bar{z}_t, g_t) = R(z_t, g_t) \tag{3.31}$$

$$\widehat{p}_t(\bar{z}_t, g_t) = \mathcal{P}_{g_t}\bar{z}_t = \mathcal{P}_{g_t} z_t. \tag{3.32}$$

Then, $\{\bar{Z}_t\}_{t=1}^T$, $\bar{Z}_t \in \Delta(\mathcal{S}_t)$ *is an* $\{(\varepsilon_t, \delta_t)\}_{t=1}^T$*-approximate information state for Problem 1 with the error bounds given by:*

$$\varepsilon_t = 0 \tag{3.33}$$

$$\delta_t \leq \frac{K}{\sqrt{n}}, \tag{3.34}$$

where K *is a constant that depends on the state space* $\mathcal{S}$ *and the metric* d_s*. Note that the transition dynamics and reward functions in the AIS generator are identical to the transition dynamics and reward functions of Problem 2. Hence, the system with these AIS generators has the same DP as Problem 2.*

Proof We have:

$$\Big|\mathbb{E}[R_t | Z_t = z_t, G_t = g_t] - \widehat{r}_t(\bar{z}_t, g_t)\Big| \overset{(a)}{=} \Big|R(z_t, g_t) - R(z_t, g_t)\Big| = 0 =: \varepsilon_t, \tag{3.35}$$

where (a) follows from (3.9) and (3.31). Furthermore, we have:

$$d_{\bar{z}}(\mathbb{P}(\bar{Z}_{t+1} | Z_t = z_t, G_t = g_t), \widehat{p}(\bar{z}_t, g_t)) \overset{(a)}{=} d_{\bar{z}}(P(z_{t+1} \mid z_t, g_t), \mathcal{P}_{g_t} z_t) \overset{(b)}{\leq} \frac{K}{\sqrt{n}} =: \delta_t, \tag{3.36}$$

where (a) follows from (3.7) and (3.32), and (b) from [109], where K is a constant that depends on the state space $\mathcal{S}$ and the metric d_s. ■

We now relate the performance of an optimal policy for Problem 2, in Problem 1. This relation can be stated as follows.

Corollary 3.5.1 *Let* $\bar{\pi}^*(\cdot, \bar{z}) = \bar{\psi}^*(\bar{z})$ *denote an optimal policy for Problem 2. Then, we have:*

$$|\bar{J}^* - J^*] \leq T L_V \frac{K}{\sqrt{n}} = \frac{\bar{K}}{\sqrt{n}}, \tag{3.37}$$

and

$$\tilde{J}^* - J(\bar{\pi}^*) \leq 2TL_V \frac{K}{\sqrt{n}} = 2\frac{\bar{K}}{\sqrt{n}}, \tag{3.38}$$

where $\bar{K} = TL_V K$. □

PROOF The results given by (3.37) and (3.38) follow from the definition of the performances as:

$$\tilde{J} = \mathbb{E}_{Z_0}[\tilde{V}_0(Z_0)], \text{ and } \bar{J} = \mathbb{E}_{\bar{Z}_0}[\bar{V}_0(\bar{Z}_0)],$$

Theorem 3.5.1, and Theorem 2.2.1 from Chapter 2, where L_V is the Lipschitz constant of the value function, denoted by K_s in Theorem 2.2.1. ■

3.5.4 Relation between the solutions of Problem 1 with different number of agents

In this section, we bound the sub-optimality due to the use of a solution of an $m-agent$ system as defined in Problem 1 and an $n-agent$ system as defined in Problem 1. In order to do this, we use the relation between both these systems and the corresponding mean-field limit system as given below.

Corollary 3.5.2 *The performance error due to using an* $m-$ *agent optimal policy, denoted by* $\hat{\pi}^*(\cdot)$*, where* $\hat{z}$ *denotes the empirical mean-field in the* $m-$ *agent system, in the original* $n-agent$ *system can be given as:*

$$|J^* - J(\hat{\pi}^*)| \leq 2\bar{K}(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{m}}). \tag{3.39}$$

□

PROOF The proof follows by using the following triangle inequality:

$$|J^* - J(\hat{\pi}^*)| \leq |J^* - J(\bar{\pi}^*)| + |J(\hat{\pi}^*) - J(\bar{\pi}^*)|, \tag{3.40}$$

where each of the two terms on the RHS can be bounded by the error given in (3.38). Thus, we get the desired result. ■

3.5.5 Extension to infinite horizon

The approximation results developed for mean-field teams in the previous sections can be extended to the infinite horizon case in a fairly straightforward manner. In order to do so, we first consider the Lipschitz continuity of the infinite horizon value function as follows. In this case, the transition functions and the reward functions are assumed to be time-homogeneous and we consider a disount factor of $\gamma \in (0,1)$. This is proven in [94, Theorem 1], which we state below:

Lemma 3.5.6 *[94, Theorem 1] Define the value function for the infinite horizon version of Problem 2 as $\bar{V}(\bar{z})$ that satisfies:*

$$\begin{aligned} \bar{V}(\bar{z}) &= \max_{\bar{g} \in \bar{\mathcal{G}}} \bar{Q}(\bar{z}, \bar{g}) \\ \bar{Q}(\bar{z}, \bar{g}) &= R(\bar{z}, \bar{g}) + \sum_{\bar{z}^+ \in \Delta(\mathcal{S})} \mathbb{P}(\bar{z}^+ | \bar{z}, \bar{g}) \bar{V}(\bar{z}^+). \end{aligned} \tag{3.41}$$

For any Lipschitz continuous stationary policy $\psi : \Delta(\mathcal{S}) \to [\mathcal{S} \to \Delta(\mathcal{A})]$ with Lipschitz constant L_ψ, if $\gamma L_{\bar{P}}(1 + L_\psi) < 1$, then the value function $\bar{V} : \Delta(\mathcal{S}) \to \mathbb{R}$ is Lipschitz continuous with Lipschitz constant $L_V^\infty = \frac{L_{\bar{R}}}{1 - \gamma L_{\bar{P}}(1 + L_\psi)}$. □

We now get the following approximation error in the infinite horizon case.

Lemma 3.5.7 *Consider the infinite horizon extensions of Problems 1 and 2 with time-homogeneous transition and reward functions and a discount factor of $\gamma \in (0,1)$. Let $V : \Delta(\mathcal{S}) \to \mathbb{R}$ and $\bar{V} : \Delta(\mathcal{S}) \to \mathbb{R}$ denote their respective value functions and $J, \bar{J}$ their respective performances. We then have:*

$$|J^* - J(\bar{\pi}^*)| \leq \frac{2\bar{K}^\infty}{\sqrt{n}}, \tag{3.42}$$

where $\gamma \in (0,1)$ is the discount factor, $\bar{K}^\infty = \frac{\gamma L_V^\infty K}{(1-\gamma)}$ and L_V^∞ is as per Lemma 3.5.6 □

Proof This result follows from Lemma 3.5.6 and Theorem 2.4.2 from Chapter 2. ■

3.6 Mean-field team reinforcement learning (MFT-RL)

In principle, the dynamic program of Theorem 3.3.1 provides a computational algorithm to identify the optimal policy, but it suffers from several computational challenges. The state space $\mathcal{Z}$ of the dynamic program is the set of all empirical distributions with denominator n. Although this set is finite and bounded by $(n+1)^{|\mathcal{S}|}$, it can grow quickly with the size of the state space $\mathcal{S}$. Furthermore, the prescriptions are functions from a finite set to a simplex. For example, if $|\mathcal{S}| = |\mathcal{A}| = 4$ and $n = 1000$, then z_t takes approximately 10^{12} values and g_t takes values in $(\Delta_4)^4$, where Δ_4 denotes the simplex in $\mathbb{R}^4$. Carrying out exact dynamic programming with such state and action spaces is computationally challenging. In this section, we propose a reinforcement learning framework for computing the optimal policy for the above model. In the sequel, we restrict attention to the infinite horizon case, as is common in the reinforcement learning literature.

For RL, we need access to a simulator (or environment) to get data. For mean-field teams, we consider two different types of simulators and propose two different RL algorithms. The two types of simulators are—a system level simulator that generates the next statistical mean-field given the current mean-field, $\bar{z}_t$, and prescription, $\bar{g}_t$, i.e., we get $\bar{z}_{t+1} = \mathcal{P}_{\bar{g}}\bar{z}_t$ or an agent-level simulator, where we get a sample of the next state of the agent, given the current state, action and mean-field, i.e, we get $S^i_{t+1} \sim \mathbb{P}(S^i_{t+1}|S^i_t = s^i_t, A^i_t \sim G_t(S^i_t), Z_t)$ for each agent $i \in N$. By using trajectories generated using this simulator for n agents in parallel, we get the evolution of the empirical mean-field Z_t. In both cases, we assume that the initial state $\bar{Z}_0$ or Z_0, as the case may be, is known.

In the first case, when we have access to a system level simulator, the problem reduces to a standard RL in MDP problem as we have access to the (approximate) information state $\bar{Z}_t$. Hence, we can use any standard MDP RL algorithm such as TRPO [100], PPO [101], NAFDQN [41] etc. to solve this problem. Under appropriate conditions these algorithms will converge to a policy which is optimal for the model [117] and Theorem 3.5.1 gives the error between this solution and the optimal solution.

In the second case, where we have access only to an agent-level simulator, we can construct an approximate system level simulator by simulating several agents in parallel. Then, we can use the same approach for RL as the first case. If the number of parallel simulations equals the number of agents in the system, then using RL, we can get a simulation based solution for Problem 1. Note that this simulation can, in principle, be done

by each agent independently. When either n is too large or each agent does not know the total number of agents in the system, we could consider simulating $m < n$ parallel agents. Under appropriate conditions any standard RL algorithm will converge to a policy which is optimal for the model [117] and Corollary 3.5.2 gives the error between this solution and the optimal solution.

We thus have two variants of RL based on having access to a system-level simulator or an agent level simulator. These RL solutions can be used in Problem 1 with bounded sub-optimality. In the sequel, we demonstrate the numerical performance of these algorithms. For the agent-level simulator, we demonstrate performance using a stylized version of the demand-response problem and a stylized version of the malware spread problem. In order to demonstrate the sub-optimality in using a mean-field limit policy or a policy learned in a system with different number of agents, we compute both the mean-field limit solution using a system-level simulator and a 100-agent solution using an agent-level simulator, and we evaluate the performance of these learned polciies in systems with number of agents varying from 100 to 1000. We also compare these values with the respective planning solutions from literature [3, 5].

The key step that enables us to use standard RL algorithms for both these types of simulators is given in the following section.

3.6.1 Restriction to parameterized policies

The action space $\mathcal{G}$ of the above dynamic program is all functions from $\mathcal{S}$ to $\Delta(\mathcal{A})$. We assume that $\mathcal{G}$ is approximated by some family of parametrized functions $\mathcal{G}_\Phi = \{g_\phi\}_{\phi\in\Phi}$ (where Φ is a compact and convex set) such as Gibbs/Boltzmann functions or neural networks. With such a parametrization, the dynamic program of (3.18) may be approximated as:

$$V(z) = \max_{\phi\in\Phi} \mathbb{E}[R(z, g_\phi) + \gamma V(Z_{t+1}) \mid Z_t = z, G_t = g_\phi]. \tag{3.43}$$

Let $\tilde{\psi}^*(z)$ be an arg max of the right hand side of (3.43). Then the policy,

$$\pi(s, z) = g_{\tilde{\psi}^*(z)}(s), \tag{3.44}$$

is the best policy for (3.6) when $g_t(\cdot, z_t)$ is restricted to belong to $\mathcal{G}_\Phi$.

3.7 Numerical experiments

In this section, we first describe the two benchmark problems—demand response in smart grids and malware spread in networks. We first illustrate that TRPO, PPO and NAFDQN converge to an approximate solution in the agent-level simulator based approach for these two problems. Subsequent to this we only consider the demand response problem. For this example, we then show the convergence to an approximate solution in the system-level simulator case (mean-field limit case) using TRPO, PPO and NAFDQN. Following this, we compare the performance of the solution obtained using the mean-field limit model in finite population models with different agent populations. We also compare the performance of the solution obtained using the agent level simulator with $n = 100$ agents in systems with different agent populations. For clarity of exposition, we only show solutions obtained using TRPO for these comparisons. For both these cases, as expected we see that the sub-optimality in performance is bounded. For the RL implementations in this section, we have used the Chainer RL code base [25, 122].

3.7.1 Benchmark domains

We consider the following domains to illustrate different decentralized reinforcement learning algorithms.

Demand response in smart grids

This is a stylized model for demand response in smart grids [5]. The system consists of n agents, where $\mathcal{S} = \{0, 1\}$, $\mathcal{A} = \{\emptyset, 0, 1\}$, the dynamics are given by:

$$P(\cdot \mid \cdot, \emptyset, z) = M \tag{3.45}$$

$$P(\cdot \mid \cdot, 0, z) = (1 - \varepsilon_1) \left[\begin{smallmatrix} 1 & 0 \\ 1 & 0 \end{smallmatrix}\right] + \varepsilon_1 M \tag{3.46}$$

$$P(\cdot \mid \cdot, 1, z) = (1 - \varepsilon_2) \left[\begin{smallmatrix} 0 & 1 \\ 0 & 1 \end{smallmatrix}\right] + \varepsilon_2 M, \tag{3.47}$$

where M denotes the "natural" dynamics of the systems and ε_1 and ε_2 are small positive constants.

The per-step reward is given by:

$$R_t = -\left(\frac{1}{n}\sum_{i\in N}\left(c_0\mathbb{1}_{\{A_t^i=0\}} + c_1\mathbb{1}_{\{A_t^i=1\}}\right) + KL(z_t\|\zeta)\right), \tag{3.48}$$

where c_0 and c_1 are costs for taking actions 0 and 1 respectively, ζ is a given target distribution and $KL(z_t\|\zeta)$ denotes the Kullback-Leibler divergence between z_t and ζ.

In our experiments, we consider we consider a system with $n = 100$ agents, initial state distribution $P_0 = [1/3, 2/3]$, $M = \left[\begin{smallmatrix} 0.25 & 0.75 \\ 0.375 & 0.625 \end{smallmatrix}\right]$, $c_0 = 0.1$, $c_1 = 0.2$, $\zeta = [0.7, 0.3]$, $\varepsilon_1 = \varepsilon_2 = 0.2$ and discount factor $\gamma = 0.9$.

Malware spread in networks

This is a stylized model for malware spread in networks [51–53]. The system consists of n agents where $\mathcal{S} = [0, 1]$, $\mathcal{A} = \{0, 1\}$. The dynamics are given by:

$$S_{t+1}^i = \begin{cases} S_t^i + (1 - S_t^i)\omega_t, & \text{for } A_t = 0, \\ 0 & \text{for } A_t = 1, \end{cases}$$

where $\omega_t \sim \texttt{Uniform}[0, 1]$. The per-step reward is given by:

$$R_t = -\Big(\frac{1}{n}\sum_{i\in N}(k + \langle z_t\rangle)S_t^i + \lambda A_t^i\Big),$$

where $\langle z_t\rangle$ denotes the average of Z_t, and λ is the cost of taking action 1.

In our experiments, we consider $k = 0.2$, initial state distribution $P_0 = \texttt{Uniform}(\mathcal{S})$, $\lambda = 0.5$ and discount factor $\gamma = 0.9$. For the simulation, we discretize the state space into 11 bins—$0, 0.1, \ldots, 1$.

3.7.2 Simulation results

We first present the results for the agent based simulator for both the benchmark problems. We consider three RL algorithms—TRPO, PPO and NAFDQN. Figure 3.1 shows the result for the demand response domain and Figure 3.2 shows the result for the malware spread domain. For each of these, the dark line shows the median performance and the shaded region shows the region between the first and third quartiles across multiple independent

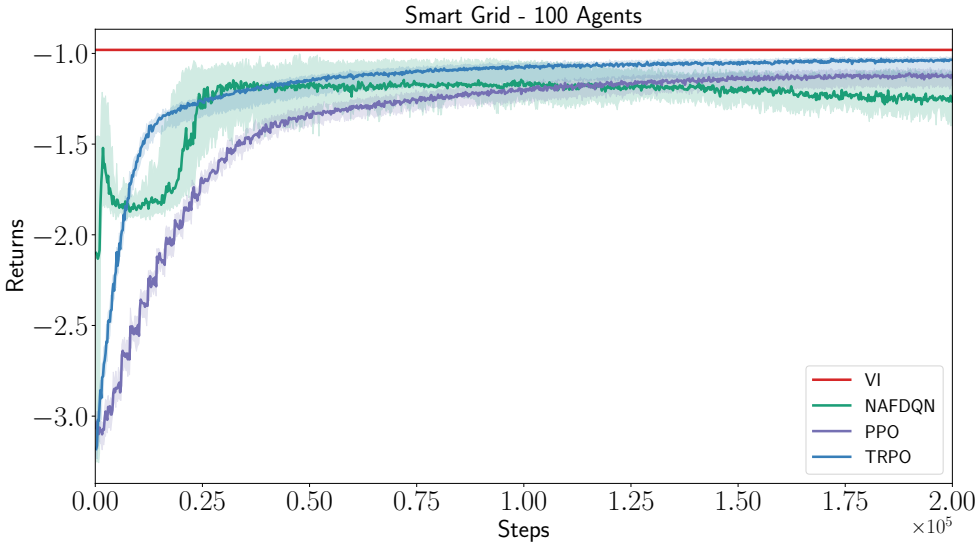


Fig. 3.1: Performance of different variants of MFT-RL for demand response domain (25 independent runs).

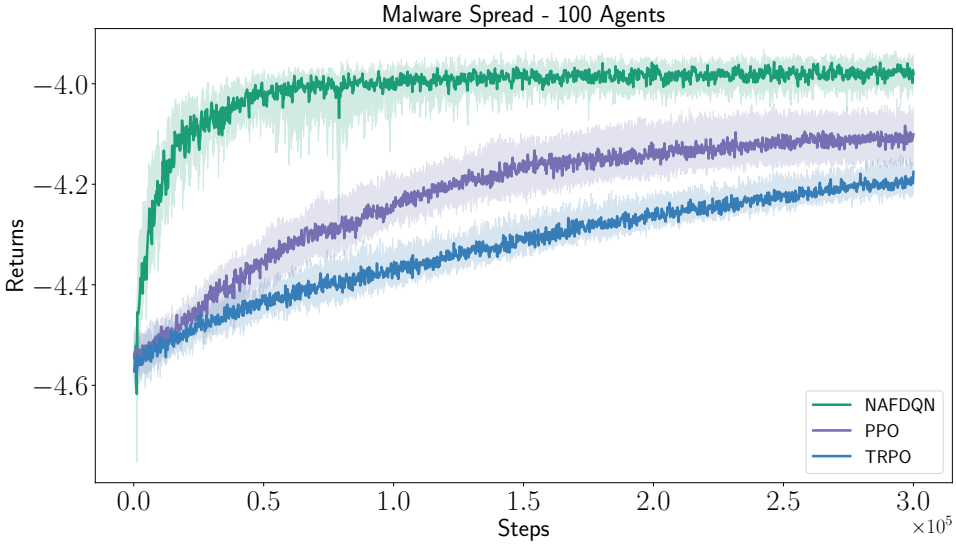


Fig. 3.2: Performance of different variants of MFT-RL for malware spread domain (15 independent runs).

runs. For the demand response domain we also show the optimal performance obtained using the value iteration algorithm presented in [3, 5].

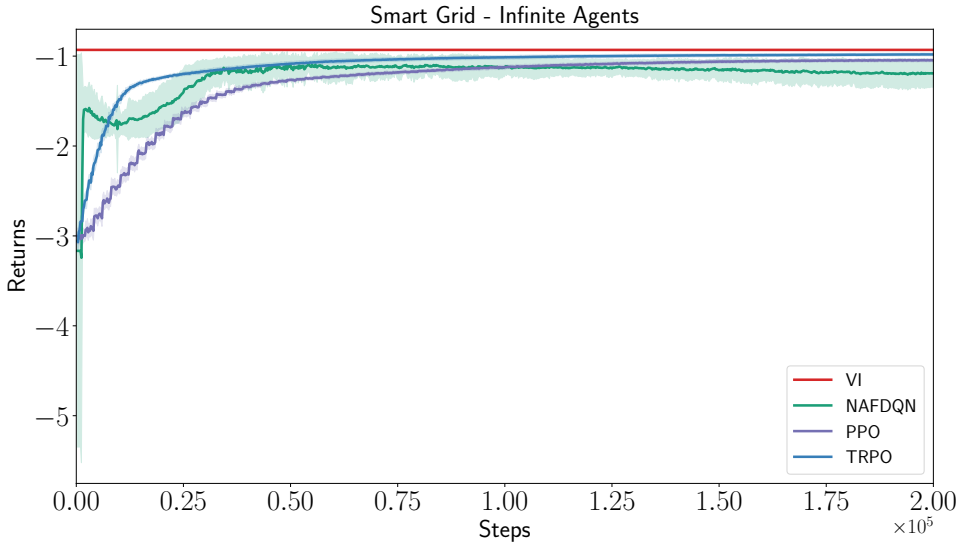


Fig. 3.3: Performance of different variants of MFT-RL for demand response domain (25 independent runs).

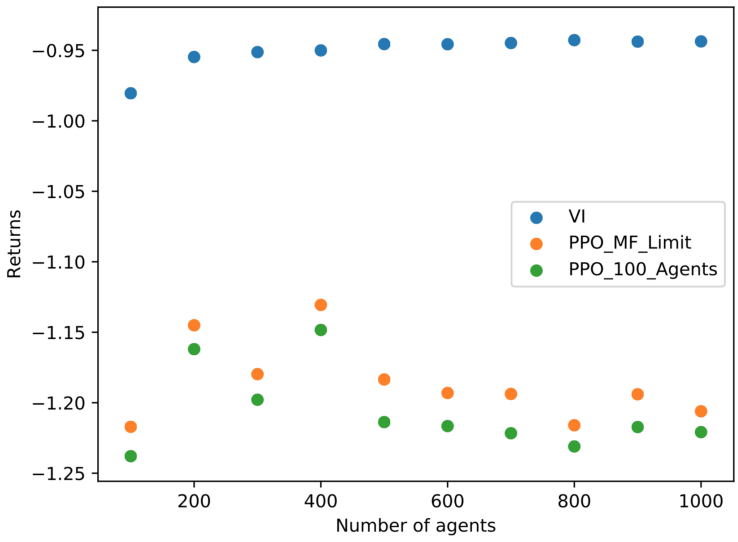


Fig. 3.4: Performance of policy obtained in mean-field limit system and 100-agent system in systems with larger number of agents.

For the system based simulator, we only consider the demand response problem. Figure 3.3 shows that TRPO, PPO and NAFDQN algorithms converge to solutions with performance close to the optimal performance.

Finally, in Figure 3.4, we compare the performance of the policies obtained using PPO in the agent-level simulator (labeled as 'PPO_100_Agents') and the system-level simulator (labeled as 'PPO_MF_Limit') in systems with different agent populations. We see that the sub-optimality in using either of these solutions is bounded by comparing with the planning solution from literature obtained using value iteration (labeled as 'VI').

3.8 Conclusion

In this chapter, we related the solutions of a mean-field team problem with a finite number of agents to the solution of the correspoding mean-field limit problem. We showed using the theory of approximate information state (AIS) that the sub-optimality of using the solution obtained in the mean field limit system in the original n-agent system is bounded and of the order of $O(\frac{1}{\sqrt{n}})$ as expected from results in literature [4]. Furthermore, using this relation, we bound the sub-optimality of using the solution obtained using an m-agent system in an n-agent system and show that this is of the order of $O(\frac{1}{\sqrt{n}} + \frac{1}{\sqrt{m}})$. We then present an approach for RL in such systems using parametrized prescriptions. Specifically we cover two case—where we have access to a system-level simulator (equivalent to the mean-field limit problem, Problem 2) and where we have access to an agent-level simulator (equivalent to the n-agent problem, Problem 1). We illustrate the performance of our algorithms using two exmaples which are stylized models of the demand response and malware spread problems. For the demand response problem, we also show the sub-optimality in performance otained in using a mean-field limit RL solution and an m-agent RL solution in systems with varying number of agents ranging from 100 to 1000. Thus, using our approach, we can obtain RL or ADP solutions to any arbitrary population mean-field team problem with mean-field sharing solutions using approaches with finite complexity (mean-field limit or a fixed population solution) and bounded sub-optimality.

This page is intentionally left blank.

Chapter 4

RL in Stationary Mean-field Games

4.1 Introduction

In this chapter, we present an MARL algorithm for a special class of mean-field systems, where the agents can be non-cooperative, play stationary strategies and the mean-field also becomes stationary. These systems are called stationary mean-field games. As discussed earlier, in such multi-agent systems, the presence of other agents makes MARL different from traditional single agent RL. When we view the MARL setup from the point of view of a particular agent, say agent i, all other agents are part of the environment. Since these agents are also learning and changing their policies, the environment faced by agent i changes with time. Due to this perception of non-stationary environment, traditional single agent RL algorithms cannot be used in MARL.

Another feature of MARL is that the agents may be strategic (i.e., selfish) and wish to maximize their individual reward or they might be cooperative and wish to maximize their team reward. Depending on the case, the learning process in MARL should converge to a variation of Nash equilibrium or of social-welfare optimal (or team optimal) solution.

There is a rich literature on MARL which models the multi-agent interaction using the framework of stochastic dynamic games starting with [75], where a Q-learning algorithm that converges to a minimax solution of a zero-sum game was proposed. This was extended to an algorithm that converges to the Nash equilibrium of a general sum game (under some conditions) in [50]. Several other variations have been proposed in the literature and we refer the reader to [17, 46, 141] for a detailed survey.

In recent years, there has been considerable interest in using deep neural networks in

MARL. Most papers adopt the paradigm of *centralized training with decentralized execution* in which a centralized critic estimates the Q-function and decentralized actors optimize the policy of the agents. Examples include BICNET [90], MADDPG [79], and COMA [34].

These approaches, in general, do not scale with the number of agents. In the literature on planning for multi-agent systems, various frameworks have been proposed which easily scale to thousands of homogeneous agents. These include swarm based models [57], mean-field games (MFG) [55, 70, 125], mean-field teams [3, 5, 6], and cooperative multi-agent systems [5, 56, 57, 70]. The central theme in all these approaches is the idea of mean-field (MF) approximation from statistical physics [128].

Motivated by the success of the planning frameworks, there have been several approaches which use mean-field approximation for reinforcement learning. The earliest of these is [66], which proposed a model based adaptive control algorithm for mean-field games. A Q-learning based algorithm for MFG control of coupled oscillators is proposed in [139]. Model-free Q-learning and actor critic algorithms for mean-field games have been proposed in [85, 138]. A detailed description of these papers is presented in Sec. 4.5.3. Another related work is [137], which proposed a mean-field based solution for inverse RL. Mean-field games are related to the notion of anonymous games, which considers static games with large number of anonymous agents [12, 59]. A learning framework for such games was presented in [63].

In the last decade, mean-field models have been successfully used in many planning problems in control engineering, network economics, and finance, but these results haven't been translated to the learning setup. A remarkable feature of mean-field models is that as the number of agents becomes large, the non-stationarity problem has negligible impact on the solution. In a mean-field model, agents are homogeneous and coupled only through the mean-field. Agents impact each other only through the mean-field distribution and once this is fixed, the agents are decoupled. Thus, MF models circumvent the non-stationarity problem by changing the solution concept. It has been shown that under appropriate conditions, the mean-field equilibrium is also a ε-Nash equilibrium, where ε is $O(1/\sqrt{n})$.

In this chapter, we present reinforcement learning algorithms for *stationary* mean-field games. In the game theory and stochastic control literature, there are two very closely related modeling frameworks that are referred to as mean-field games and *stationary* mean-field games. We highlight the difference between these two modeling frameworks in Sec. 4.5.2. The current literature on using mean-field ideas in MARL focuses on computing Nash equilibrium of mean-field games. We propose reinforcement learning algorithms that

compute *stationary mean-field equilibrium* and *social-welfare optimal solution* of stationary mean-field games. Both the modeling framework and the solution concepts are different from what has previously appeared in the MARL literature. Our main contribution is to obtain RL algorithms for stationary MF models. Most existing works for RL for MF assumes non-stationary solution concept.

4.2 Background

4.2.1 Mean-field games (MFG)

Consider a mean-field game with n homogeneous agents, indexed by the set $N = \{1, 2, \ldots, n\}$. Each agent has the same state and action spaces, which we denote by $\mathcal{S}$ and $\mathcal{A}$ respectively. Both $\mathcal{S}$ and $\mathcal{A}$ are finite sets. At any time t, $S_t^i \in \mathcal{S}$ and $A_t^i \in \mathcal{A}$ denote the state and action of agent $i \in N$. In a MFG, the dynamical evolution and the reward of each agent are decoupled from the rest of the agents given the mean-field, where the mean-field or the empirical distribution of the system is given by:

$$z_t(x) = \frac{1}{n} \sum_{i \in N} \mathbb{1}\{S_t^i = x\}, \quad \forall x \in \mathcal{S}. \tag{4.1}$$

Note that $z_t \in \Delta(\mathcal{S})$, the space of probability mass functions on $\mathcal{S}$. The state of agent i evolves according to:

$$S_{t+1}^i \sim P(S_t^i, A_t^i, z_t), \tag{4.2}$$

where $P(x, a, z) \in \Delta(\mathcal{S})$ is the transition probability distribution given the state x, action a and mean-field z. With a slight abuse of notation, we use $P(y|x, a, z)$ to denote the probability that the next state is y given that the current state, action and mean-field are x, a and z respectively. The per-step reward for each agent $i \in N$ is given by:

$$\mathsf{R}_t^i = r(S_t^i, A_t^i, z_t, S_{t+1}^i). \tag{4.3}$$

The utility or the expected total reward for agent $i \in N$ is given by:

$$U^i = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t \mathsf{R}_t^i\Big], \tag{4.4}$$

where $\gamma \in (0, 1)$ is the discount factor.

The main idea of mean-field games is to approximate the above finite population system by an infinite population system, where the empirical mean-field almost surely converges to the statistical mean-field due to the strong law of large numbers. Thus the agents assume that:

$$z_t(x) \approx \frac{1}{n} \sum_{i \in N} \mathbb{P}(S_t^i = x). \tag{4.5}$$

In addition, it is assumed that agents use an identical time varying policy $(\pi_1, \pi_2, \dots)$, where $\pi_t \colon \mathcal{S} \to \Delta(\mathcal{A})$ is the stochastic policy at time t and $A_t^i \sim \pi_t(S_t^i)$. When all agents follow policy $(\pi_1, \pi_2, \dots)$, the statistical mean-field evolves according to the discrete time McKean Vlasov equation:

$$z_{t+1}(y) = \sum_{x \in \mathcal{S}} \sum_{a \in \mathcal{A}} z_t(x) \pi_t(a|x) P(y|x, a, z_t), \quad \forall y \in \mathcal{S}, \tag{4.6}$$

which we denote as:

$$z_{t+1} = \Phi(z_t, \pi_t). \tag{4.7}$$

4.2.2 Stationary MFG

In *stationary* MFG, the following additional assumptions are made [2, 125, 126].

(A1) Time homogeneous policy: All agents follow a time-homogeneous, stochastic policy, $\pi_t = \pi \colon \mathcal{S} \to \Delta(\mathcal{A})$ for all t, i.e., each agent chooses an action given by $A_t^i \sim \pi(S_t^i)$. With a slight abuse of notation, we use $\pi(a|x)$ to denote the probability of choosing action a in state x under policy π. Let Π denote the space of all such policies.

(A2) Stationarity of mean-field: When all agents follow a policy $\pi \in \Pi$, the mean-field of states $\{Z_t\}_{t \geq 0}$ converges almost surely to a constant limit z, which we call the stationary mean-field. Note that the stationary mean-field satisfies:

$$z = \Phi(z, \pi). \tag{4.8}$$

(A3) Agent's performance evaluation: Agents evaluate their performance by assuming that the population is infinite and the corresponding mean-field takes its stationary

value at all times. In particular, given a policy $\pi \in \Pi$ and a candidate stationary mean-field distribution $z \in \Delta(\mathcal{S})$, agent i evaluates its performance starting from initial state $x \in \mathcal{S}$ as:

$$V_{\pi,z}(x) = \mathbb{E}_{\substack{A_t^i \sim \pi(X_t^i) \\ X_{t+1}^i \sim P(X_t^i, A_t^i, z)}} \left[\sum_{t=0}^{\infty} \gamma^t r(X_t^i, A_t^i, z, X_{t+1}^i) \middle| X_0^i = x \right].$$

Such a *mean-field approximation* may be written as the solution of the following Bellman fixed-point equation.

$$V_{\pi,z}(x) = \sum_{a \in \mathcal{A}} \pi(a|x) \Big[\sum_{y \in \mathcal{X}} P(y|x,a,z) \times \big[r(x,a,z,y) + \gamma V_{\pi,z}(y) \big] \Big].$$

4.2.3 Solution concepts

When agents are strategic (non-cooperative), the following refinement of Markov perfect equilibrium (MPE) is used as a solution concept [2].

Definition 4.2.1 (Stationary mean-field equilibrium (SMFE)) A stationary mean-field equilibrium (SMFE) is a pair of policy $\pi \in \Pi$ and mean-field $z \in \Delta(\mathcal{S})$ which satisfies the following two properties:

1. *Sequential rationality:* For any other policy π',

$$V_{\pi,z}(x) \geq V_{\pi',z}(x), \quad \forall x \in \mathcal{X}.$$

2. *Consistency:* The mean-field z is stationary under policy π, i.e.,

$$z = \Phi(z, \pi).$$

□

The sufficient conditions for the existence of an SMFE are given in [2].

When agents are cooperative, the following refinement of social welfare optimal solution is used as a solution concept.

Definition 4.2.2 (Stationary mean-field social-welfare optimal policy (SMF-SO)) A policy $\pi \in \Pi$ is stationary mean-field social welfare optimal (SMF-SO) if it satisfies the following property:

- *Optimality:* For any other policy $\pi' \in \Pi$,

$$V_{\pi,z}(x) \geq V_{\pi',z'}(x), \quad \forall x \in \mathcal{S},$$

where z and z' are the stationary mean-field distributions corresponding to π and π', respectively, i.e., satisfy

$$z = \Phi(z, \pi) \quad \textit{and} \quad z' = \Phi(z', \pi').$$

□

If the model satisfies sufficient conditions such that each policy has an associated stationary distribution, then SMF-SO is the policy which yields the best performance. In our case, since there are only a finite number of policies due to the finiteness of the state and action spaces, an SMF-SO always exists.

Comparison of the two solution concepts

The definitions of sequential rationality and optimality are different. In particular, sequential rationality is defined with respect to the mean-field z; while considering the performance of an alternative policy $\pi' \in \Pi$ it is assumed that the mean-field does not change. In contrast, optimality is a property of a policy; while considering the performance of an alternative policy $\pi' \in \Pi$, the mean-field approximation of the performance is with respect to the stationary mean-field corresponding to π'. Thus, in general, SMFE and SMF-SO are different.

4.2.4 Local solution concepts

Both the solution concepts described in Sec. 4.2.3 are global concepts, i.e., they are defined over all possible policies $\pi \in \Pi$. They are difficult to verify by agents with bounded rationality or limited computational resources. So, we define local variations of these solution concepts that are easier to verify. It is worth highlighting that when these local solution concepts are unique (as is the case in many examples), they coincide with the the global ones. To define these local solution concepts, we make two assumptions:

1. The initial states of all agents are independent and identically distributed according

to $\xi_0 \in \Delta(\mathcal{S})$. Thus, the performance of any policy $\pi \in \Pi$ is given by:

$$J_{\pi,z} = \mathbb{E}_{X\sim\xi_0}[V_{\pi,z}(X)] = \sum_{x\in\mathcal{S}} V_{\pi,z}(x)\xi_0(x).$$

2. The policy $\pi \in \Pi$ is parametrized by $\theta \in \Theta$, where Θ is a convex and closed subset of a Euclidean space. We denote the policy parametrized by $\theta \in \Theta$ as π_θ. Examples of such parametrizations include Gibbs/Boltzmann distribution and neural networks.

Both these assumptions are standard in the reinforcement learning literature on policy gradient methods [118]. Based on these assumptions, we define the following local variants of SMFE and SMF-SO.

Definition 4.2.3 (Local stationary mean-field equilibrium (LSMFE)) A local stationary mean-field equilibrium (LSMFE) is a pair of policy $\pi_\theta \in \Pi$ and mean-field $z \in \Delta(\mathcal{S})$ which satisfies the following two properties:

1. *Local sequential rationality:* $\partial J_{\pi_\theta,z}/\partial\theta = 0$.
2. *Consistency:* $z = \Phi(z, \pi_\theta)$. □

Definition 4.2.4 (Local stationary mean-field social welfare optimal policy (LSMF-SO)) A policy $\pi_\theta \in \Pi$ is local stationary mean-field social welfare optimal (LSMF-SO) if it satisfies the following property:

- *Local optimality:* $dJ_{\pi_\theta,z_\theta}/d\theta = 0$, where z_θ is the stationary mean-field distribution corresponding to π_θ, i.e., satisfies $z_\theta = \Phi(z_\theta, \pi_\theta)$. □

Comparison of the two local solution concepts

From the chain rule of derivatives, we have

$$\frac{dJ_{\pi,z}(x)}{d\theta} = \frac{\partial J_{\pi,z}(x)}{\partial\pi}\frac{\partial\pi}{\partial\theta} + \frac{\partial J_{\pi,z}(x)}{\partial z}\frac{\partial z}{\partial\theta}.$$

The first term is equal to $\partial J_{\pi_\theta,z}(x)/\partial\theta$. In general, $\partial J_{\pi,z}(x)/\partial z \neq 0$ and $\partial z/\partial\theta \neq 0$. Thus, local optimality is not the same as local sequential rationality. This is also illustrated by the numerical results presented in Sec. 4.4.

Comparison of global and local solution concepts

Local variants of Nash equilibrium have been studied in the literature [96]. An interesting feature for MFG is that uniqueness of SMFE does not imply that LSMFE is same as SMFE. This is because unlike standard Nash equilibrium, SMFE and LSMFE are a collection of a strategy profile and stationary distribution. Sufficient conditions for LSMFE to be unique (and agree with the SMFE) are:

1. SMFE is unique.

2. The value function is concave in the policy parameters for every value of mean-field.

If an SMFE exists for a problem and the policy parametrization is such that it covers the entire space of policies, then this parametrized SMFE satisfies the conditions of an LSMFE. Hence, we can conclude that there is at least one LSMFE solution. Furthermore, concavity of value function in terms of policy parameters guarantees a unique optimum and hence unique best-response for every value of the stationary mean-field. This property, along with the uniqueness of the SMFE, ensure uniqueness of the LSMFE and its equivalence with the SMFE. If an SMF-SO exists for a problem and the policy parametrization is such that it covers the entire space of policies, then this parametrized SMF-SO satisfies the conditions of an LSMF-SO. Hence, we can conclude that there is at least one LSMF-SO solution. Furthermore, if the SMF-SO is unique, and if the value function evaluated at the corresponding stationary mean-field is concave with respect to the policy parameters, then the LSMF-SO is unique and agrees with the SMF-SO. Conditions for unique local equilibrium are satisfied for the malware spread model presented in Sec. 4.4 [51–53].

4.3 RL for stationary MFG

In this section we propose two RL algorithms corresponding to each of the local solution concepts defined in Sec. 4.2.4. For both cases we assume that the agent has access to a simulator that yields the next state and the per-step reward for an agent, given the agent's current local state, current action and the current mean-field.

4.3.1 RL algorithm for learning LSMFE

The key idea behind the RL algorithm to learn an LSMFE is as follows. Suppose $G_{\theta,z}$ is an unbiased estimator of $\partial J_{\pi_\theta,z}/\partial\theta$. Then, we can start with an initial guess $\theta_0 \in \Theta$ and $z_0 \in \Delta(\mathcal{X})$ and at each step of the iteration, update the guess (θ_k, z_k) using two-timescale stochastic gradient ascent [20]:

$$z_{k+1} = z_k + \beta_k\big[\hat{\Phi}(z_k, \pi_{\theta_k}) - z_k\big], \tag{4.9a}$$

$$\theta_{k+1} = \big[\theta_k + \alpha_k G_{\theta_k,z_k}\big]_\Theta, \tag{4.9b}$$

where $[\,\cdot\,]_\Theta$ denotes projection on to the set Θ and $\hat{\Phi}(z,\pi)$ is an unbiased approximation of $\Phi(z,\pi)$ which is generated as follows: generate a mini-batch of m samples $(X^j, A^j, Y^j)_{j=1}^m$ where $X^j \sim z$, $A^j \sim \pi(\cdot|X^j)$, and $Y^j \sim P(X^j, A^j, z)$ and set

$$\hat{\Phi}(z,\pi)(y) = \frac{1}{m}\sum_{j=1}^{m} \mathbb{1}\{Y^j = y\}.$$

The learning rates $\{\alpha_k, \beta_k\}_{k\geq 0}$ are chosen according to the standard conditions for two-timescale algorithms: $\sum \alpha_k = \infty$, $\sum \beta_k = \infty$, $\sum(\alpha_k^2 + \beta_k^2) < \infty$, $\lim\limits_{k\to\infty} \alpha_k = 0$, $\lim\limits_{k\to\infty} \beta_k = 0$ and $\lim\limits_{k\to\infty} \alpha_k/\beta_k = 0$. Then, we have the following:

Proposition 4.3.1 *If the following conditions are satisfied:*

1. *$\Phi(z,\pi_\theta)$, $\partial J_{\pi_\theta,z}/\partial\theta$ are Lipschitz in θ, z.*
2. *$\hat{\Phi}(z,\pi)$ and $G_{\theta,z}$ are unbiased estimators of of $\Phi(z,\pi)$ and $\partial J_{\pi_\theta,z}/\partial\theta$. Moreover, the estimation error $G_{\theta,z} - \partial J_{\pi_\theta,z}/\partial\theta$ has bounded variance.*
3. *For all $\theta \in \Theta$, the ODE corresponding to* (4.9a)*, i.e.,*

$$\dot{z} = \Phi(z,\pi_\theta) - z$$

   has a unique globally asymptotically stable equilibrium point, which we denote by $f(\theta)$.

4. *$f(\theta)$ is Lipschitz in θ.*

Then, almost surely:

1. $\|z_n - f(\theta_n)\| \to 0$ *as* $n \to \infty$.

2. *Suitable continuous time interpolation of* $\{\theta_n\}$ *is an asymptotic pseudotrajectory of the semiflow induced by the ODE corresponding to* (4.9b) *for* θ*, i.e,*

$$\dot{\theta} = \partial J_{\pi_\theta, z}/\partial\theta.$$

3. *The iteration* (4.9) *converges to a LSMFE.*

PROOF Note that, because the image space of Φ is bounded, the estimation error $\hat{\Phi}(z,\pi) - \Phi(z,\pi)$ is uniformly bounded. Thus, the conditions stated in the proposition along with the learning rate conditions specified for iteration (4.9) satisfy all the conditions (B1) to (B4) stated in [72, page 35], [20, Theorem 23]. Specifically condition (B1) in [72, page 35] is satisfied as we have assumed that both θ and z line in compact metric spaces. (B2) is satisfied by assumption (1) in the proposition. (B3) and (B4) are satisfied by the learning rate conditions for iteration (4.9) and assumptions 2,3 and 4 in the proposition. Thus, all the 4 conditions are satisfied and the result then follows from the application of the theorem given in [72, page 35], [20, Theorem 23]. Consequently, iteration (4.9) almost surely converges to a limit (θ^*, z^*) such that:

$$\partial J_{\pi_{\theta^*}, z^*}/\partial\theta = 0 \quad \textit{and} \quad z^* = f(\theta^*),$$

which implies (π_{θ^*}, z^*) is a LSMFE. ■

In theory, two-timescale algorithms are nice because they are amenable to a proof of convergence. However, in practice, two-time scale algorithms converge slowly because there are no good methods to adapt the learning rates. So, rather than running a two-timescale algorithm, it is often better to run a large but fixed number of iterations of variable running at the faster timescale for every iteration of variable running at the slower timescale. For iteration (4.9) this is equivalent to running multiple iterations of (4.9a) (with a fixed learning rate β) for every iteration of (4.9b). In the sequel, we run B iterations of (4.9b) with $\beta_k = 1$, which is shown in Algorithm 1 and is equivalent to a particle based Monte Carlo computation of the generated mean-field of the system.

Algorithm 1: `Stationary_MF`

input : θ : Policy parameter, ξ_0 : Initial state distribution
B : Iteration count, m : Batch size
output : z : Final mean-field
for $j = 1 : m$ **do**
  for $i \in N$ **do**
    Sample $X_0^{i,j} \sim \xi_0$
  $z_0^j = \xi_0$
  for $t = 0 : B$ **do**
    for $i \in N$ **do**
      Sample $A_t^{i,j} \sim \pi(X_t^{i,j})$
    Sample $X_{t+1}^{i,j} \sim P(X_t^{i,j}, A_t^{i,j}, z_t^j)$
    for $x \in \mathcal{X}$ **do**
      $z_{t+1}^j(x) = \frac{1}{n} \sum_{i \in N} \mathbb{1}\{X_{t+1}^{i,j} = x\}$
$z = \frac{1}{m} \sum_{j=1}^{m} z_{B+1}^j$
return z

To convert iteration (4.9) to a complete algorithm, we need an algorithm that computes an unbiased estimator $G_{\theta,z}$ for $\partial J_{\pi_\theta,z}/\partial\theta$ for a given z. Since z is fixed, $\partial J_{\pi_\theta,z}/\partial\theta$ may be computed using any of the standard policy gradient based approaches for reinforcement learning: likelihood ratio based gradient estimators [67, 118] or simultaneous perturbation based gradient estimators [16, 64, 82, 111].

Likelihood ratio based gradient estimation

One approach to estimate the performance gradient is to use likelihood ratio based estimates [38, 98, 133]. Suppose the policy $\pi_\theta(S)$ is differentiable with respect to θ. For any time t, define the likelihood function $\Lambda_\theta^t = \nabla_\theta \log[\pi_\theta(A_t \mid S_t)]$. Then from [10, 118, 133] we know that:

$$\frac{\partial V_{\theta,z}(x)}{\partial\theta} = \mathbb{E}_{A_t \sim \pi_\theta(S_t)} \Big[\sum_{t=0}^{\infty} \gamma^t \Lambda_\theta^t V_{\pi_\theta,z}(X_t) \;\Big|\; X_0 = x \Big].$$

Thus,

$$\frac{\partial J_{\theta,z}}{\partial\theta} = \mathbb{E}_{X \sim \xi_0} \left[\frac{\partial V_{\theta,z}(X)}{\partial\theta} \right].$$

An algorithm to compute LSMFE based on the likelihood ratio approach is given in

Algorithm 2: Likelihood ratio based algorithm to compute LSMFE

input : θ_0 : Initial policy, z_0 : Initial mean-field
ξ_0 : Initial state distribution
K : Iteration count
B : Iterations for mean-field update
m : Batch size for mean-field update

output : (θ^*, z^*) : Estimated LSMFE solution

for *iterations* $k = 0 : K$ **do**
  $z_{k+1} = \texttt{Stationary\_MF}(\theta_k, \xi_0, B, m)$
  $G_{\theta_k, z_{k+1}} = \texttt{PolicyGradient}(\theta_k, \xi_0, z_{k+1})$
  $\theta_{k+1} = [\theta_k + \alpha_k G_{\theta_k, z_{k+1}}]_\Theta$

return θ_{K+1}

Algorithm 2. The `PolicyGradient` function in Algorithm 2 can be obtained by an actor only method such as Monte Carlo [116] or Renewal Monte Carlo [113] or using an actor critic method such as SARSA [116]. Additionally, variance reduction techniques such as subtracting a baseline or using mini-batch averaging may also be used.

Simultaneous perturbation based gradient estimation

Another approach to estimate the performance gradient is to use simultaneous perturbation based methods [16,64,82,111]. This approach is useful when the policy π_θ is not differentiable with respect to its parameters θ. Now, given any distribution ξ_0, we can estimate $J_{\pi_\theta, z}$ using $V_{\pi_\theta, z}$ as:

$$J_{\pi_\theta, z} = \mathbb{E}_{X \sim \xi_0}[V_{\pi_\theta, z}(X)].$$

To generate the two-sided form of simultaneous perturbation based estimate, we generate two random parameters $\theta^+ = \theta + c\,\eta$ and $\theta^- = \theta - c\,\eta$, where η is a random variable with the same dimension as θ and c is a small constant. Let $\pi^+ = \pi_{\theta^+}$ and $\pi^- = \pi_{\theta^-}$. Then, the two-sided simultaneous perturbation estimate is given by

$$G_{\theta, z} = \frac{\eta}{2c}(J_{\pi^+, z} - J_{\pi^-, z}).$$

When $\eta_i \sim$ Rademacher(± 1), the above method is called simultaneous perturbation stochastic approximation (SPSA) [82, 111]; when $\eta_i \sim$ Normal$(0, I)$ it is called smoothed functional stochastic approximation (SFSA) [16, 64].

Algorithm 3: Simultaneous perturbation based algorithm to compute LSMFE

input : θ_0 : Initial policy, z_0 : Initial mean-field
ξ_0 : Initial state distribution
K : Iteration count, c : Perturbation size
B : Iterations for mean-field update
m : Batch size for mean-field update

output : (θ^*, z^*) : Estimated LSMFE solution

for *iterations* $k = 1 : K$ **do**

$z_{k+1} = \texttt{Stationary\_MF}(\theta_k, \xi_0, B, m)$
Let $\eta \sim \texttt{Rademacher}(\pm 1)$ or $\eta \sim \mathcal{N}(0,1)$
$\theta_k^+ = \theta_k + \eta\beta$ and $\theta_k^- = \theta_k - \eta\beta$.
$\widehat{J}_k^+ = \texttt{PolicyEvaluation}(\theta_k^+, \xi_0, z_{k+1})$
$\widehat{J}_k^- = \texttt{PolicyEvaluation}(\theta_k^-, \xi_0, z_{k+1})$
$G_{\theta_k, z_{k+1}} = \frac{\eta}{2c}(\widehat{J}_k^+ - \hat{J}_k^-)$
$\theta_{k+1} = [\theta_k + \alpha_k G_{\theta_k, z_{k+1}}]_\Theta$

return θ_{K+1}

An algorithm to compute LSMFE using the simultaneous perturbation approach is given in Algorithm 3. As in the case of the likelihood ratio based approach, the `PolicyEvaluation` function in Algorithm 3 may be obtained by an actor only method such as Monte Carlo [116] or Renewal Monte Carlo [113] or using an actor critic method such as SARSA [116].

4.3.2 RL algorithm for learning LSMF-SO

The key idea behind the RL algorithm to learn an LSMF-SO is as follows. Suppose T_θ is an unbiased estimator for $dJ_{\pi_\theta, z_\theta}/d\theta$, where z_θ is the fixed point of $z = \Phi(z, \pi_\theta)$. Then, we start with an initial guess $\theta_0 \in \Theta$, and at each step of the iteration, update the guess using stochastic gradient ascent:

$$\theta_{k+1} = \left[\theta_k + \alpha_k T_{\theta_k}\right]_\Theta, \tag{4.10}$$

where $\{\alpha_k\}_{k\geq 0}$ is a sequence of learning rates that satisfies the standard conditions: $\sum \alpha_k = \infty$ and $\sum \alpha_k^2 < \infty$. Then, we have the following:

Proposition 4.3.2 *If the following conditions are satisfied:*

1. $dJ_{\pi_\theta, z_\theta}/d\theta$ *is Lipschitz continuous in* θ.

2. *T_θ is an unbiased estimator of $dJ_{\pi_\theta,z_\theta}/d\theta$ and the error $T_\theta - dJ_{\pi_\theta,z_\theta}/d\theta$ has bounded variance.*

3. *The ODE for θ, i.e,*

$$\dot{\theta} = dJ_{\pi_\theta,z_\theta}/d\theta,$$

has isolated limit points that are locally asymptotically stable.

Then, almost surely:

1. *Suitable continuous time interpolation of $\{\theta_n\}$ is an asymptotic pseudotrajectory of the semiflow induced by the ODE for θ.*

2. *The iteration converges to a LSMF-SO.*

PROOF The conditions stated above and the learning rate conditions satisfy the standard stochastic approximation convergence conditions as given in [19, 68]. Specifically, the model satisfies all the conditions (A1)–(A4) of [19, pg 10–11] Condition (A1) is satisfied by the first condition of the proposition, (A2) is satisfied by the choice of learning rates in (4.10), (A3) is satisfied by the second condition in the proposition and (A4) is satisfied due to projection of the iterates in (4.10) to Θ. Hence, the iteration (4.10) converges almost surely to a limit θ^* such that:

$$dJ_{\pi_\theta,z_\theta}/d\theta = 0,$$

which implies (π_{θ^*}, z^*) is a LSMF-SO, where $z^* = \hat{\Phi}(z^*, \pi_{\theta^*})$. ■

To convert iteration (4.10) to a complete algorithm, we need an algorithm that computes an unbiased estimator $T_{\theta,z}$ of $dJ_{\pi_\theta,z_\theta}/d\theta$. Likelihood ratio based gradient estimators do not work in this case because, in order to compute $d\mathbb{E}[r(X_t^i, A_t^i, z_\theta)]/d\theta$, we need to compute $dz_\theta/d\theta$ and there are no good methods to do so. There are some results in the literature on the sensitivity of the stationary distribution of a Markov chain to its transition probability (e.g., [35] and references therein), but these results only provide loose bounds on $dz_\theta/d\theta$. However, it is possible to adapt simultaneous perturbation based methods to generate estimators of $dJ_{\pi_\theta,z_\theta}/d\theta$. We present one such estimator in the next section.

4.3.3 Simultaneous perturbation based gradient estimation

We first consider estimating z_θ for a given π_θ. Under (A2), when each agent follows policy π_θ, the mean-field converges to the stationary distribution z_θ. Then, we can estimate z_θ by

Algorithm 4: Simultaneous perturbation based algorithm to compute LSMF-SO

input : θ_0 : Initial policy
ξ_0 : Initial state distribution
K : Iteration count, c : Perturbation size
B : Iterations for mean-field update
m : Batch size for mean-field update

output : θ^* : Estimated LSMF-SO solution

for *iterations* $k = 1 : K$ **do**

Let $\eta \sim$ `Rademacher`(± 1) or $\eta \sim \mathcal{N}(0,1)$
$\theta_k^+ = \theta_k + \eta\beta$ and $\theta_k^- = \theta_k - \eta\beta$.
$z_k^+ =$ `Stationary_MF`(θ^+, ξ_0, B, m)
$z_k^- =$ `Stationary_MF`(θ^-, ξ_0, B, m)
$\widehat{J}_k^+ =$ `PolicyEvaluation`$(\theta_k^+, \xi_0, z_k^+)$
$\widehat{J}_k^- =$ `PolicyEvaluation`$(\theta_k^-, \xi_0, z_k^-)$
$T_{\theta_k} = \frac{\eta}{2c}(\widehat{J}_k^+ - \hat{J}_k^-)$
$\theta_{k+1} = [\theta_k + \alpha_k T_{\theta_k}]_\Theta$

return θ_{K+1}

simply running the system for a sufficiently long time. An algorithm based on this idea is shown in Algorithm 1.

Then, to generate the two-sided simultaneous perturbation based estimate of $dJ_{\pi_\theta, z_\theta}/d\theta$, we generate two random parameters $\theta^+ = \theta + c\,\eta$ and $\theta^- = \theta - c\,\eta$, where η and c are as in Sec. 4.3.1. Let $\pi^+ = \pi_{\theta^+}$ and $\pi^- = \pi_{\theta^-}$. Generate $z^+ = z_{\pi^+}$ and $z^- = z_{\pi^-}$ using Algorithm 1. Then, the two-sided simultaneous perturbation estimate is given by

$$T_\theta = \frac{\eta}{2c}\big(J_{\pi^+, z^+} - J_{\pi^-, z^-}\big).$$

An algorithm to compute LSMF-SO using simultaneous perturbation approach is given in Algorithm 4. As was the case for Algorithm 3, the `PolicyEvaluation` function in Algorithm 4 may be obtained by an actor only method such as Monte Carlo [116] or Renewal Monte Carlo [113] or using an actor critic method such as SARSA [116].

4.4 Numerical experiment

4.4.1 Example 1: Malware spread

Environment

We consider the malware spread model presented in [51–53, 58]. This model is representative of several problems with positive externalities. Examples of such models include flue vaccination, economic models involving entry and exit of firms, collusion among firms, mergers, advertising, investment, network effects, durable goods, consumer learning etc. Hence, we consider the malware spread problem as a representative problem where an analytical solution is available. In this model, let $X \in [0, 1]$ denote the state (level of infection) of agent i, where where $X_t^i = 0$ is the most healthy state and $X_t^i = 1$ is the least healthy state. The action space $\mathcal{A} = \{0, 1\}$, where $A_t^i = 0$ implies DO NOTHING and $A_t^i = 1$ implies REPAIR. The dynamics are given by

$$X_{t+1}^i = \begin{cases} X_t^i + (1 - X_t^i)\omega_t, & \text{for } A_t^i = 0, \\ 0, & \text{for } A_t^i = 1, \end{cases}$$

where $\{\omega_t\}_{t\geq 1}$ is a $[0, 1]$-valued i.i.d. process with probability density f. The above dynamics imply that if the agent takes the DO NOTHING action, then its state deteriorates to a worse condition in the interval $[1 - X_t^i, 1]$; if the agent takes the REPAIR action, then its state resets to the most healthy state.

The rewards are coupled through the mean $\langle Z_t \rangle$ of the mean field Z_t (i.e., $\langle Z_t \rangle >= \int_0^1 x Z_t(x) dx$). Each agent incurs a cost $(k + \langle Z_t \rangle) X_t^i$, which captures the risk of getting infected, and an additional cost of λ for taking the REPAIR action, i.e.,

$$r(X_t^i, A_t^i, Z_t) = -(k + \langle Z_t \rangle) X_t^i - \lambda A_t^i.$$

For this example, the existence of an SMFE is proven in [51–53]. Furthermore, it is shown in [51–53] that every policy has a stationary distribution. Hence, the SMF-SO exists.

Model and policy parameters

We consider $n = 1000$ agents, $f = \text{Uniform}[0, 1]$, $k = 0.2$, $\lambda = 0.5$ and $\gamma = 0.9$. The continuous state space $\mathcal{X} = [0, 1]$ is discretized into 101 uniformly sized cells $\{0, 0.01, \ldots, 1\}$. We consider two different policy parametrizations:

1. **Threshold based policy**: We consider threshold-based policies parametrized by $\theta \in [0, 1]$ such that[1]:

$$\pi_\theta(x) = \begin{cases} 0, & \text{if } x < \theta, \\ 1, & \text{if } x \geq \theta. \end{cases} \tag{4.11}$$

   We use this policy parametrization to estimate both LSMFE and LSMF-SO. The parameterized policies of the form (4.11) are not differentiable with respect to θ, so we estimate the gradient using simultaneous perturbation methods (Algorithms 3 and 4) with $c = 0.1$, $\eta \sim \text{Rademacher}(\pm 1)$, initial value of the threshold chosen uniformly at random, i.e., $\theta_0 \sim \text{Uniform}[0, 1]$. In both algorithms, policy evaluation is done using Monte Carlo with $m = 1000$ trajectories of length $H = 200$.

2. **Neural network (NN) based policy**: We consider a neural network policy with two hidden layers with 5 neurons and tanh activation. We estimate the gradient using the likelihood ratio method. We use REINFORCE [133] to compute the performance gradient and backpropagate this gradient over the NN to compute $G_{\theta,z}$. Policy gradient estimation is done using Monte Carlo (actor only) with $m = 10$ and $H = 200$. Since we have a likelihood ratio based gradient estimation approach only for the RL algorithm for LSMFE (Sec. 4.3.1), we use this policy parametrization only to estimate LSMFE (Algorithm 2).

For both the policy parametrizations, $z_0 = \xi_0 = \text{Uniform}(\mathcal{X})$, $B = 200$ and $K = 200$. We choose the learning rate using ADAM [65].[2] We repeat the experiment 100 times for both the policy parametrizations.

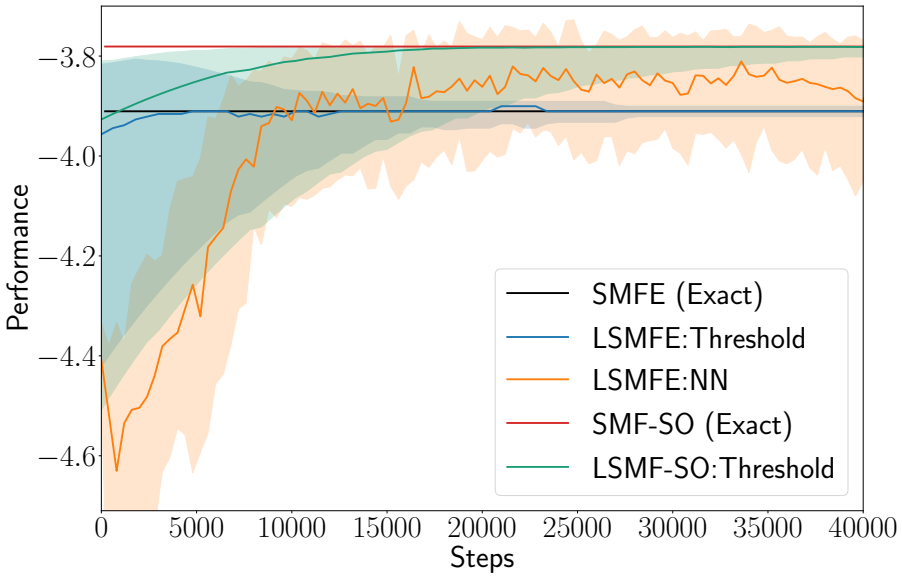


Fig. 4.1: Performance versus steps: RL algorithm converging to LSMFE or LSMF-SO for the malware spread example. The solid line shows the median value and the shaded region shows the region between the first and third quartiles over 100 runs.

Results

The performance J for LSMFE (using both parameterizations) and LSMF-SO (using threshold based policies) are shown in Fig. 4.1. For the threshold based policy, J and $\langle z^* \rangle$ were evaluated using exact policy evaluation. For the neural network policy, they were estimated using 10 Monte Carlo evaluation runs.

For comparison, the exact SMF-SO and SMFE solutions are also plotted. The SMF-SO solution is computed by a brute force search over all $\theta \in [0, 1]$. The SMFE solution is computed using the method described in [51]. These exact solutions are also shown in Fig. 4.1. The plots show that the convergence of the SPSA based RL algorithm is fairly fast and the variation across multiple runs is small. It is worth highlighting that LSMFE and LSMF-SO are different.

[1]It is shown in [51, 53] that such a parametrization is without loss of optimality.

[2]The α parameter of ADAM is set equal to 0.01 for the threshold based policy and 0.1 for the NN policy. All other ADAM parameters are equal to their default values.

4.4.2 Example 2: Investments in product quality

Environment

We consider the investment decisions of firms in a fragmented market with a large number of firms. This model is adapted from [127]. In this model, each firm produces n_p products. The state of each firm S_t^i is represented by a n_p vector with each element $S_t^{i,j} \in [0,1]$, $j \in \{1,\dots,n_p\}$ denoting the normalized product quality for product j for firm i. At each time step, each firm $i \in N$ has to choose whether or not to invest in improving the quality of each of its products $j \in \{1,\dots,n_p\}$. Investment decisions are binary for each product. Thus the action space for firm i is $\mathcal{A}^i = \{0,1\}^{n_p}$, with $|\mathcal{A}^i| = 2^{n_p}$. When agent i decides to invest in product j, the quality of product j manufactured by i increases uniformly at random from its current value to the maximum value of 1, if the average mean-field for that product is below a particular threshold q. If this average mean-field value is above q, then the agent gets only half of the product quality improvement as compared to the former case. This implies that when the average quality of product j in the economy is below q, it is easier for each agent to improve its quality for product j. When the agent does not invest any amount in product j, its product quality for product j remains unchanged. This is given as:

$$S_{t+1}^{i,j} = \begin{cases} \omega_t(1 - S_t^{i,j}), & \text{if } \langle z^j \rangle < q \text{ and } A_t^{i,j} = 1, \\ 0.5\omega_t(1 - S_t^{i,j}), & \text{if } \langle z^j \rangle \geq q \text{ and } A_t^{i,j} = 1, \\ S_t^{i,j}, & \text{if } A_t^{i,j} = 0, \end{cases} \tag{4.12}$$

where ω_t is a $[0,1]$-valued i.i.d. process with probability density f and $\langle Z_t^j \rangle$ is the mean of Z_t^j (i.e., equal to $\int_0^1 x Z_t^j(x)dx$, $j \in \{1,\dots,n_p\}$).

At each step, each agent i incurs a cost due to its investment and earns a positive reward due to its own product quality for each product $j \in \{1,\dots,n_p\}$ and a negative reward due to the average product quality for product i, i.e., $\langle z_t^j \rangle$. This per-step reward accumulated over all products is given as:

$$r(S_t^i, A_t^i, Z_t^i) = \sum_{j=1}^{n_p} \Big[d^j S_t^{i,j} - c^j \langle z_t^j \rangle - \lambda A_t^{i,j} \Big] \tag{4.13}$$

This example is adapted from a model in [127] for which they prove existence of SMFE.

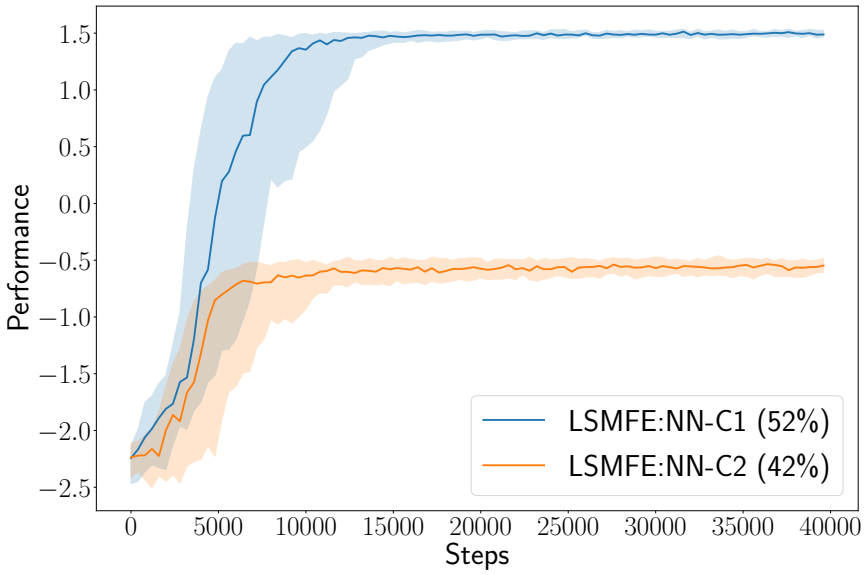


Fig. 4.2: Performance versus steps: RL algorithm converging to LSMFE for the product quality investments example. The solid line shows the median value and the shaded region shows the region between the first and third quartiles over 100 runs.

Model and policy parameters

We consider $n = 100$ agents, $f = \text{Uniform}[0, 1]$, $n_p = 3$, $q = 0.4$, $c = [0.21, 0.22, 0.23]$, $d = [0.31, 0.32, 0.33]$, $\lambda = [0.2, 0.21, 0.22]$, $B = 200$, $K = 200$, $X_0^{i,j} \sim \text{Uniform}[0, 1]$ and $\gamma = 0.9$. The policy is parametrizd using a two layer neural network with 8 and 16 neurons respectively with a tanh activation function for all hidden units. We use ADAM with learning rate of 0.1.

Results

In this example, we only demonstrate the computation of LSMFE using a neural network policy. We performed 100 independent runs for this example. We then clustered the tails (last 10 iterations) of these 100 trajectories and found that there are multiple LSMFE for this example. In Fig. 4.2, we plot the median, and the region between the first and third quantiles for the trajectories corresponding to the two most populated clusters. These two clusters, named C1 and C2 in Fig. 4.2, comprise of 52% and 42% of the total number of

trajectories respectively.

4.5 Discussion

4.5.1 Finite vs. infinite populations

In both MFG and stationary MFG, the finite population system is approximated by an infinite population system. The infinite population system has two features: (i) each agent has an infinitesimal impact on the evolution of the mean-field which can be ignored; and (ii) the empirical mean-field can be approximated by the statistical mean-field, which evolves in a deterministic manner for a given policy. Thus, the strategic interactions between agents in a general n-player game is replaced by two consistency requirements: the policy is a best-response to the mean-field and the mean-field is consistent with the policy. As a result, the n-agent learning problem is reduced to optimality and consistency between a single generic (or canonical) agent and the mean-field.

However, since we are approximating the finite population system by an infinite population system, the approximation is meaningful only if the corresponding approximation error is small. There are several results in the mean-field games literature that show that under various (generally mild) technical conditions, the infinite population result is a $O(1/\sqrt{n})$ or a $O(1/n)$ approximation of the corresponding finite population result [56, 125]. These conditions are often model specific, so we don't list them here. What is important to note from the point of view of learning is that under these conditions, the learning algorithms proposed in this chapter converge to $O(1/\sqrt{n})$ or $O(1/n)$ of the corresponding finite population solution.

4.5.2 Difference between MFG and stationary MFG models

MFG and stationary MFG are closely related but there is a fundamental difference between them. In MFG, assumptions (A1)–(A3) are not imposed. Thus the policy $\tilde{\boldsymbol{\pi}} = (\tilde{\pi}_1, \tilde{\pi}_2, \dots)$, $\tilde{\pi}_t : \mathcal{S} \to \Delta(\mathcal{A})$, is, in general, a time-varying policy (we denote the space of all such policies as $\tilde{\boldsymbol{\Pi}}$) and it is not assumed that the mean-field trajectory $\boldsymbol{z} = (z_1, z_2, \dots)$ converges to a limit. Thus, given a mean-field trajectory $\boldsymbol{z}$, the performance of a policy $\tilde{\boldsymbol{\pi}} \in \tilde{\boldsymbol{\Pi}}$ is given by:

$$\tilde{V}_{\tilde{\boldsymbol{\pi}},\boldsymbol{z}}(x) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(X_t^i, A_t^i, z_t, X_{t+1}^i) \;\Big|\; X_0^i = x\Big].$$

Note that even though the mean-field trajectory is fixed, the environment and rewards perceived by a generic agent are time-varying. Therefore, one cannot write a fixed-point Bellman equation for $\tilde{V}_{\tilde{\boldsymbol{\pi}},\boldsymbol{z}}$. Nonetheless, a time-varying Bellman equation can be written and, it is for this reason that, most of the literature on MFG apart from the special case of linear dynamics and quadratic cost considers finite horizon systems.

The commonly used solution concept for MFG is the following:

Definition 4.5.1 (NE-MFG) A Nash equilibrium for MFG is a pair of time-varying policy $\tilde{\boldsymbol{\pi}} = (\tilde{\pi}_1, \tilde{\pi}_2, \dots) \in \tilde{\boldsymbol{\Pi}}$ and a trajectory of mean-fields $\boldsymbol{z} = (z_1, z_2, \dots)$ which satisfies the following two conditions:

1. *Sequential rationality:* For any other policy $\tilde{\boldsymbol{\pi}}' = (\tilde{\pi}'_1, \tilde{\pi}'_2, \dots)$, we have:

$$V_{\tilde{\boldsymbol{\pi}},\boldsymbol{z}}(x) \geq V_{\tilde{\boldsymbol{\pi}}',\boldsymbol{z}}(x), \quad \forall x \in \mathcal{X}.$$

2. *Consistency:* The mean-field z evoles as:

$$z_{t+1} = \Phi(z_t, \tilde{\pi}_t), \quad \forall t.$$

□

It is worth highlighting that NE-MFG is a pair of a trajectory of time-varying policy and time-varying mean-field. In contrast, SMFE is a pair of single policy and a single mean-field. Thus, SMFE is considerably easier to compute and implement as compared to NE-MFG. This simplicity comes at the cost of generality. The conditions for existence of SMFE are generally stricter than those for NE-MFG.

4.5.3 Related work

In view of the above discussion, we revisit the related work on mean-field approximation in MARL.

In [66], a model based adaptive control algorithm for computing NE-MFG of linear quadratic systems is considered. It is assumed that the parameter set of the system is a specified compact subset of Euclidean space. Agents use maximum likelihood estimation to estimate the most likely dynamics and use certainty equivalent control laws corresponding to the estimated model. The results of [66] are difficult to generalize beyond the linear quadratic model.

In [139], a Q-learning algorithm for computing NE-MFG for a family of coupled oscillators is considered. The mean-field approximation is used to develop an approximate dynamic program (ADP) for the best-response equation and the ADP is solved using Q-learning. The approximation used for the ADP is specific for the model considered in [139] and does not apply to general models.

In [138], a Q-learning algorithm for computing NE-MFG for a stochastic game is presented. It is assumed that all agents observe the global state ($(x^1, x^2, \dots, x^n)$ in our model) and choose policies that map global state to local actions. The mean-field approximation is used to simplify the Q-function of the best-response and the simplified Q function is solved using Q-learning or DPG. When each agent has a local state (as is the case in the models presented in this chapter), the global state is n-dimensional and it is impractical to assume that all agents know the global state. For example, in the malware example presented earlier, it will mean that all agents know the state of health of all agents in the system. Even if the global state were known, searching over policies $\pi : (x^1, x^2, \dots, x^n) \mapsto \Delta(A^i)$ will suffer from the curse of dimensionality.

In [85], a fictitious play based learning algorithm for computing NE-MFG of finite horizon common interest MFG is presented.[3] In this algorithm, one starts with a guess for the mean-field trajectory and the policy and improves them using actor critic functions. The proof of convergence relies on a technical property for NE-MFG for finite horizon MFGs proved in [22] and it is not immediately clear how that technical property can be extended to infinite horizon stationary MFG.

It is worth highlighting that all the previous work on mean-field based learning algorithms for MARL compute NE-MFG. As far as we are aware, this is the first work to propose mean-field based learning algorithms to compute SMFE and SMF-SO for stationary MFGs.

4.5.4 Remarks on the generality of the model

For simplicity of exposition, we presented our results for the simplest model of stationary MFG. The results presented for this model continue to hold for the following generalizations.

- The coupling in the dynamics and reward is through the mean-field of states and actions rather than mean-field of just the states. In this case, the argument presented in this chapter continues to work with minor changes because mean-field of states

[3] In [85], it is claimed that all MFGs are common interest games, but that is, in general, not the case.

and actions is a function of the policy and the mean-field of states. In particular, if $\bar{z}$ denotes the mean-field of states and actions, then, in the infinite population limit

$$\bar{z}_t(x, a) = \bar{z}_t(x)\pi_t(a|x), \quad \forall x \in \mathcal{S}, a \in \mathcal{A}. \tag{4.14}$$

- The states and/or actions are continuous rather than discrete. In this case, the arguments hold under the standard conditions on measurability of dynamics, upper semi-continuity of the rewards, compactness of the action space, and the growth conditions on the rewards to ensure that value functions are well defined. An appropriate parametrization of the policy using a sufficiently rich family of function approximators such as radial basis functions or neural networks is also needed.

- There is a heterogeneous population consisting of multiple sub-populations of homogeneous agents. Such a model can be converted to a homogeneous population model by considering the type of the agent as a component of the state.

Chapter 5

Renewal Monte Carlo: Renewal theory based RL

5.1 Introduction

In recent years, reinforcement learning [14,61,116,119] has emerged as an effective framework for learning how to act optimally in unknown environments. Policy gradient methods [62, 67, 100, 101, 106, 118] have played a prominent role in the success of reinforcement learning. Such methods have two critical components: policy evaluation and policy improvement. In policy evaluation, the performance of a parameterized policy is evaluated while in policy improvement, the policy parameters are updated using stochastic gradient ascent.

Policy gradient methods may be broadly classified as Monte Carlo methods and temporal difference methods. In Monte Carlo methods, performance of a policy is estimated using the discounted return of one or more sample paths; in temporal difference methods, an initial estimate for the (action-) value function is chosen arbitrarily and then improved iteratively using temporal differences. Monte Carlo methods are attractive because they have zero bias, are simple and easy to implement, and work for both discounted and average reward setups as well as for models with continuous state and action spaces. However, they suffer from various drawbacks. First, they have a high variance because a single sample path is used to estimate performance. Second, in Monte Carlo methods it is implicitly assumed that the model is episodic (i.e., there is an end state and the system stops when it reaches the end state). To use these methods for infinite horizon models, the trajectory

is arbitrarily truncated to treat the model as an episodic model. For that reason, the resultant policy is not asymptotically optimal. Third, the policy improvement step cannot be carried out in tandem with policy evaluation. One must wait until the end of the episode to estimate the performance and only then can the policy parameters be updated. For these reasons the literature on policy gradient methods largely ignores Monte Carlo methods and almost exclusively focuses on temporal difference methods such as actor-critic with eligibility traces [116].

In this chapter an online reinforcement learning algorithm called Renewal Monte Carlo (RMC) is presented. RMC works for infinite horizon Markov decision processes with a designated start state. RMC is a Monte Carlo algorithm that retains the key advantages of Monte Carlo—viz., simplicity, ease of implementation, and low bias—while circumventing the main drawbacks of Monte Carlo—viz., high variance and delayed updates. The key intuition behind RMC is that, under any reasonable policy, the reward process is ergodic. Therefore, using ideas from renewal theory, it can be shown that the performance of any parameterized policy π_θ is proportional to R_θ/T_θ, where R_θ and T_θ are the expected discounted reward and the expected discounted time of the reward process over a regenerative cycle. Hence, the performance gradient is proportional to $H_\theta = \nabla R_\theta T_\theta - R_\theta \nabla T_\theta$. Hence, any policy for which H_θ is zero is locally optimal.

In RMC, R_θ and T_θ are estimated from Monte Carlo evaluations over multiple regenerative cycles; ∇R_θ and ∇T_θ are estimated using either likelihood ratio or simultaneous perturbation based estimators; and the root of H_θ is obtained using stochastic approximation. We show that under mild technical conditions, RMC converges to a locally optimal policy.

The RMC algorithm is generalized to post-decision state models, where regenerative cycle is defined as successive visits to an initial post-decision state.

An approximate RMC algorithm is proposed where successive visits to a pre-specified "renewal set" is viewed as a regenerative cycle. We show that if the value function for the system is locally Lipschitz continuous on the renewal set, then RMC converges to approximate locally optimal policy.

The effectiveness of RMC is illustrated on three examples: randomly generated Markov decision processes, event-driven communication, and inventory control. The last two examples have continuous state space and show that RMC works well for continuous state models as well.

Although renewal theory is commonly used to estimate performance of stochastic

systems [37, 38], those methods assume that the probability law of the primitive random variables and its weak derivative are known, which is not the case in reinforcement learning. Renewal theory is also commonly used in queuing theory and Markov decision processes (MDPs) with average reward criteria and a known system model. There is some prior work on using renewal theory for reinforcement learning [80, 81], where renewal theory based estimators for the average return and differential value function for average reward MDPs are developed. In RMC, renewal theory is used in a different manner for discounted reward MDPs (and the results generalize to average cost MDPs).

5.2 RMC Algorithm

Consider a Markov decision process (MDP) with state $S_t \in \mathcal{S}$ and action $A_t \in \mathcal{A}$. The system starts in an initial state $s_0 \in \mathcal{S}$ and at each time t there is a controlled transition from S_t to S_{t+1} according to a transition kernel $P(A_t)$. At each time t, a per-step reward $R_t = r(S_t, A_t, S_{t+1})$ is received.

A (time-homogeneous and Markov) policy π maps the current state to a distribution on actions, i.e., $A_t \sim \pi(S_t)$. We use $\pi(a|s)$ to denote $\mathbb{P}(A_t = a|S_t = s)$. The performance of a policy π is given by

$$J_\pi = \mathbb{E}_{A_t \sim \pi(S_t)} \left[\sum_{t=0}^{\infty} \gamma^t R_t \;\middle|\; S_0 = s_0 \right], \tag{5.1}$$

where $\gamma \in (0, 1)$ is the discount factor. We are interested in identifying an optimal policy, i.e., a policy that maximizes the performance. When $\mathcal{S}$ and $\mathcal{A}$ are Borel spaces, we assume that the model satisfies the standard regularity conditions under which time-homogeneous Markov policies are optimal [47].

Suppose policies are parameterized by a closed and convex subset Θ of the Euclidean space.[1] Given $\theta \in \Theta$, we use π_θ to denote the policy parameterized by θ and J_θ to denote J_{π_θ}. We assume that for all policies π_θ, $\theta \in \Theta$, the designated start state s_0 is positive recurrent.

The typical approach for policy gradient based reinforcement learning is to start with an initial choice $\theta_0 \in \Theta$ and iteratively update it using stochastic gradient ascent. In particular,

[1] Examples of such parametized policies include the weights of a Gibbs soft-max policy, the weights of a deep neural network, or the thresholds in a control limit policy, and so on.

let $\widehat{\nabla} J_{\theta_m}$ be an unbiased estimator of $\nabla_\theta J_\theta\big|_{\theta=\theta_m}$, then update

$$\theta_{m+1} = \left[\theta_m + \alpha_m \widehat{\nabla} J_{\theta_m}\right]_\Theta \tag{5.2}$$

where $[\theta]_\Theta$ denotes the projection of θ onto Θ, and $\{\alpha_m\}_{m\geq 1}$ are learning rates that satisfy the standard assumptions:

$$\sum_{m=1}^{\infty} \alpha_m = \infty \quad \text{and} \quad \sum_{m=1}^{\infty} \alpha_m^2 < \infty. \tag{5.3}$$

Under standard technical conditions [19], which include the assumption that ∇J_θ is Lipschitz conitnuous with respect to θ the above iteration converges to a θ^* that is locally optimal, i.e., $\nabla_\theta J_\theta\big|_{\theta=\theta^*} = 0$. In RMC, we approximate $\nabla_\theta J_\theta$ by a renewal theory based estimator as explained below.

Let $\tau^{(n)}$ denote the stopping time when the system returns to the start state s_0 for the n-th time. In particular, let $\tau^{(0)} = 0$ and for $n \geq 1$ define $\tau^{(n)} = \min\{t > \tau^{(n-1)} : s_t = s_0\}$. We call the sequence of (S_t, A_t, R_t) from $\tau^{(n-1)}$ to $\tau^{(n)} - 1$ as the *n-th regenerative cycle*. Let $\mathsf{R}^{(n)}$ and $\mathsf{T}^{(n)}$ denote the total discounted reward and total discounted time of the n-th regenerative cycle, i.e.,

$$\mathsf{R}^{(n)} = \Gamma^{(n)} \sum_{t=\tau^{(n-1)}}^{\tau^{(n)}-1} \gamma^t R_t \quad \text{and} \quad \mathsf{T}^{(n)} = \Gamma^{(n)} \sum_{t=\tau^{(n-1)}}^{\tau^{(n)}-1} \gamma^t, \tag{5.4}$$

where $\Gamma^{(n)} = \gamma^{-\tau^{(n-1)}}$. By the strong Markov property [84], $\{\mathsf{R}^{(n)}\}_{n\geq 1}$ and $\{\mathsf{T}^{(n)}\}_{n\geq 1}$ are i.i.d. sequences. Let R_θ and T_θ denote $\mathbb{E}[\mathsf{R}^{(n)}]$ and $\mathbb{E}[\mathsf{T}^{(n)}]$, respectively. Define

$$\widehat{\mathsf{R}} = \frac{1}{N}\sum_{n=1}^{N} \mathsf{R}^{(n)} \quad \text{and} \quad \widehat{\mathsf{T}} = \frac{1}{N}\sum_{n=1}^{N} \mathsf{T}^{(n)}, \tag{5.5}$$

where N is an arbitrarily chosen number of cycles. Then, $\widehat{\mathsf{R}}$ and $\widehat{\mathsf{T}}$ are unbiased and asymptotically consistent estimators of R_θ and T_θ.

From ideas of renewal theory [32], we have the following.

Proposition 5.2.1 (Renewal Relationship) *The performance of policy π_θ is given by:*

$$J_\theta = \frac{\mathsf{R}_\theta}{(1-\gamma)\mathsf{T}_\theta}. \tag{5.6}$$

PROOF Consider the performance:

$$\begin{aligned} J_\theta &= \mathbb{E}_{A_t\sim\pi_\theta(S_t)}\left[\sum_{t=0}^{\tau^{(1)}-1}\gamma^t R_t + \gamma^{\tau^{(1)}}\sum_{t=\tau^{(1)}}^{\infty}\gamma^{t-\tau^{(1)}}R_t \;\middle|\; S_0 = s_0\right] \\ &\overset{(a)}{=} \mathsf{R}_\theta + \mathbb{E}_{A_t\sim\pi_\theta(S_t)}[\gamma^{\tau^{(1)}}]\, J_\theta \end{aligned} \tag{5.7}$$

where the second expression in (a) uses the independence of random variables from $(0, \tau^{(1)}-1)$ to those from $\tau^{(1)}$ onwards due to the strong Markov property [84].

Now, by definition, $\mathsf{T}_\theta = (1 - \mathbb{E}_{A_t\sim\pi_\theta(S_t)}[\gamma^{\tau^{(1)}}])/(1-\gamma)$. Rearranging terms, we get $\mathbb{E}_{A_t\sim\pi_\theta(S_t)}[\gamma^{\tau^{(1)}}] = 1 - (1-\gamma)\mathsf{T}_\theta$. Substituting this in (5.7), we get the result of the proposition. ■

Differentiating both sides of (5.6) with respect to θ, we get

$$\nabla_\theta J_\theta = \frac{H_\theta}{\mathsf{T}_\theta^2(1-\gamma)}, \text{ where } H_\theta = \mathsf{T}_\theta\nabla_\theta\mathsf{R}_\theta - \mathsf{R}_\theta\nabla_\theta\mathsf{T}_\theta. \tag{5.8}$$

Therefore, instead of using stochastic gradient ascent to find a local maximum of J_θ, we can use stochastic approximation to find a root of H_θ.

Theorem 5.2.1 *Consider the sequence $\{\theta_m\}_{m\geq 1}$ where the initial $\theta_0 \in \Theta$ is chosen arbitrarily, and for $m > 0$,*

$$\theta_{m+1} = \left[\theta_m + \alpha_m \widehat{H}_m\right]_\Theta, \tag{5.9}$$

where $\{\alpha_m\}_{m\geq 1}$ satisfies (5.3) and $\widehat{H}_m$ is an unbiased estimator of H_{θ_m}. Then, the sequence $\{\theta_m\}_{m\geq 1}$ converges almost surely and

$$\lim_{m\to\infty} \nabla_\theta J_\theta\big|_{\theta_m} = 0.$$

Algorithm 5: RMC Algorithm with likelihood ratio based gradient estimates.

input : Intial policy θ_0, discount factor γ, initial state s_0, number of regenerative cycles N

for *iteration* $m = 0, 1, \ldots$ **do**

  for *regenerative cycle* $n_1 = 1$ *to* N **do**

    Generate n_1-th regenerative cycle using policy π_{θ_m}.

    Compute $\mathsf{R}^{(n_1)}$ and $\mathsf{T}^{(n_1)}$ using (5.4).

  Set $\widehat{\mathsf{R}}_m = \texttt{mean}(\mathsf{R}^{(n_1)} : n_1 \in \{1, \ldots, N\})$.

  Set $\widehat{\mathsf{T}}_m = \texttt{mean}(\mathsf{T}^{(n_1)} : n_1 \in \{1, \ldots, N\})$.

  for *regenerative cycle* $n_2 = N + 1$ *to* $2N$ **do**

    Generate n_2-th regenerative cycle using policy π_{θ_m}.

    Compute $\mathsf{R}_\sigma^{(n_2)}$, $\mathsf{T}_\sigma^{(n_2)}$ and Λ_σ for all σ using (5.12).

    Set $\widehat{\mathsf{R}}^{(n_2)} = \sum_{\sigma=\tau^{n_2-1}}^{\tau^{n_2}-1} \mathsf{R}_\sigma^{(n_2)} \Lambda_\sigma$.

    Set $\widehat{\mathsf{T}}^{(n_2)} = \sum_{\sigma=\tau^{n_2-1}}^{\tau^{n_2}-1} \mathsf{T}_\sigma^{(n_2)} \Lambda_\sigma$.

  Set $\widehat{\nabla}\mathsf{R}_m = \texttt{mean}(\widehat{\mathsf{R}}^{(n_2)} : n_2 \in \{N + 1, \ldots, 2N\})$

  Set $\widehat{\nabla}\mathsf{T}_m = \texttt{mean}(\widehat{\mathsf{T}}^{(n_2)} : n_2 \in \{N + 1, \ldots, 2N\})$

  Set $\widehat{H}_m = \widehat{\mathsf{T}}_m \widehat{\nabla}\mathsf{R}_m - \widehat{\mathsf{R}}_m \widehat{\nabla}\mathsf{T}_m$.

  Update $\theta_{m+1} = \left[\theta_m + \alpha_m \widehat{H}_m\right]_\Theta$.

PROOF The convergence of the $\{\theta_m\}_{m \geq 1}$ follows from [19, Theorem 2.2] and the fact that the model satisfies conditions (A1)–(A4) of [19, pg 10–11] Condition (A1) is satisfied as we assume ∇J_θ is Lipschitz continuous with respect to θ and hence H_θ is Lipschitz continuous with respect to θ, (A2) is satisfied by the choice of learning rates (5.3), (A3) is satisfied as we assume $\widehat{H}_m$ is an unbiased estimator of H_{θ_m} and (A4) is satisfied due to projection of the iterates to Θ . ■

Proposition 5.2.2 *Let* $\widehat{\mathsf{R}}_m$, $\widehat{\mathsf{T}}_m$, $\widehat{\nabla}\mathsf{R}_m$ *and* $\widehat{\nabla}\mathsf{T}_m$ *be unbiased estimators of* R_{θ_m}, T_{θ_m}, $\nabla_\theta \mathsf{R}_{\theta_m}$, *and* $\nabla_\theta \mathsf{T}_{\theta_m}$, *respectively such that* $\widehat{\mathsf{T}}_m \perp \widehat{\nabla}\mathsf{R}_m$ *and* $\widehat{\mathsf{R}}_m \perp \widehat{\nabla}\mathsf{T}_m$.[2] *Then,*

$$\widehat{H}_m = \widehat{\mathsf{T}}_m \widehat{\nabla}\mathsf{R}_m - \widehat{\mathsf{R}}_m \widehat{\nabla}\mathsf{T}_m \tag{5.10}$$

is an unbiased estimator of H_{θ_m}. *Furthermore, assume that*

1. H_θ *is continuous,*

[2] $X \perp Y$ denotes that random variables X and Y are independent.

2. *the estimate $\widehat{H}_m$ has bounded variance,*

3. *The differential equation $d\theta/dt = H_\theta$ has isolated limit points that are locally asymptotically stable.*

Then, the sequence $\{\theta_m\}_{m\geq 1}$ generated by (5.9) converges almost surely and

$$\lim_{m\to\infty} \nabla_\theta J_\theta\big|_{\theta_m} = 0.$$

PROOF The independence assumption implies that $\widehat{H}_m$ is unbiased. The model satisfies conditions (A2.1)–(A2.6) of [68, pg. 126]. (A2.1) is satisfied as $\widehat{H}_m$ is assumed to gave a bounded variance, (A2.2) is satisfied as $\widehat{H}_m$ is an unbiased estimator of H_θ, (A2,3) is satisfied as we assume H_θ is continuous, (A2.4) and (A2.5) are satisfied as we assume that $\widehat{H}_m$ is an unbiased estimator of H_θ and hence the terms in these conditions do not arise in our iterations and finally since H_θ is derived from a gradient (A2.6) is satisfied. Hence [68, Thm 2.1] implies that $\{\theta_m\}_{m\geq 1}$ converges. The convergence to a local maximum follows from the discussion in [68, Sec. 5.8]. ■

We can estimate R_θ and T_θ using (5.5). We present two methods to estimate the gradients of R_θ and T_θ: (i) a likelihood ratio based gradient estimator which works when the policy is differentiable with respect to the policy parameters; and (ii) is a simultaneous perturbation based gradient estimator that uses finite differences, which is useful when the policy is not differentiable with respect to the policy parameters.

5.2.1 Likelihood ratio based gradient estimator

One approach to estimate the performance gradient is to use likelihood radio based estimates [38, 98, 133]. Suppose the policy $\pi_\theta(a|s)$ is differentiable with respect to θ. For any time t, define the likelihood function

$$\Lambda_t = \nabla_\theta \log[\pi_\theta(A_t \mid S_t)], \tag{5.11}$$

and for $\sigma \in \{\tau^{(n-1)}, \ldots, \tau^{(n)} - 1\}$, define

$$\mathsf{R}_\sigma^{(n)} = \Gamma^{(n)} \sum_{t=\sigma}^{\tau^{(n)}-1} \gamma^t R_t \quad \text{and} \quad \mathsf{T}_\sigma^{(n)} = \Gamma^{(n)} \sum_{t=\sigma}^{\tau^{(n)}-1} \gamma^t. \tag{5.12}$$

In this notation $\mathsf{R}^{(n)} = \mathsf{R}^{(n)}_{\tau^{(n-1)}}$ and $\mathsf{T}^{(n)} = \mathsf{T}^{(n)}_{\tau^{(n-1)}}$. Then, define the following estimators for $\nabla_\theta \mathsf{R}_\theta$ and $\nabla_\theta \mathsf{T}_\theta$:

$$\widehat{\nabla}\mathsf{R} = \frac{1}{N} \sum_{n=1}^{N} \sum_{\sigma=\tau^{(n-1)}}^{\tau^{(n)}-1} \mathsf{R}_\sigma^{(n)} \Lambda_\sigma, \tag{5.13}$$

$$\widehat{\nabla}\mathsf{T} = \frac{1}{N} \sum_{n=1}^{N} \sum_{\sigma=\tau^{(n-1)}}^{\tau^{(n)}-1} \mathsf{T}_\sigma^{(n)} \Lambda_\sigma, \tag{5.14}$$

where N is an arbitrarily chosen number.

Proposition 5.2.3 *$\widehat{\nabla}\mathsf{R}$ and $\widehat{\nabla}\mathsf{T}$ defined above are unbiased and asymptotically consistent estimators of $\nabla_\theta \mathsf{R}_\theta$ and $\nabla_\theta \mathsf{T}_\theta$.*

PROOF Let P_θ denote the probability induced on the sample paths when the system is following policy π_θ. For $t \in \{\tau^{(n-1)}, \ldots, \tau^{(n)} - 1\}$, let $D_t^{(n)}$ denote the sample path $(S_s, A_s, S_{s+1})_{s=\tau^{(n-1)}}^{t}$ for the n-th regenerative cycle until time t. Then,

$$P_\theta(D_t^{(n)}) = \prod_{s=\tau^{(n-1)}}^{t} \pi_\theta(A_s|S_s) \mathbb{P}(S_{s+1}|S_s, A_s)$$

Therefore,

$$\nabla_\theta \log P_\theta(D_t^{(n)}) = \sum_{s=\tau^{(n-1)}}^{t} \nabla_\theta \log \pi_\theta(A_s|S_s) = \sum_{s=\tau^{(n-1)}}^{t} \Lambda_s. \tag{5.15}$$

Note that R_θ can be written as:

$$\mathsf{R}_\theta = \Gamma^{(n)} \sum_{t=\tau^{(n-1)}}^{\tau^{(n)}-1} \gamma^t \mathbb{E}_{A_t \sim \pi_\theta(S_t)}[R_t].$$

Using the log derivative trick,[3] we get

$$\begin{aligned}
\nabla_\theta \mathsf{R}_\theta &= \Gamma^{(n)} \sum_{t=\tau^{(n-1)}}^{\tau^{(n)}-1} \gamma^t \, \mathbb{E}_{A_t \sim \pi_\theta(S_t)}[R_t \nabla_\theta \log P_\theta(D_t^{(n)})] \\
&\overset{(a)}{=} \Gamma^{(n)} \mathbb{E}_{A_t \sim \pi_\theta(S_t)} \Bigg[\sum_{t=\tau^{(n-1)}}^{\tau^{(n)}-1} \Bigg[\gamma^t R_t \sum_{\sigma=\tau^{(n-1)}}^{t} \Lambda_\sigma \Bigg] \Bigg] \\
&\overset{(b)}{=} \mathbb{E}_{A_t \sim \pi_\theta(S_t)} \Bigg[\sum_{\sigma=\tau^{(n-1)}}^{\tau^{(n)}-1} \Lambda_\sigma \Bigg[\Gamma^{(n)} \sum_{t=\sigma}^{\tau^{(n)}-1} \gamma^t R_t \Bigg] \Bigg] \\
&\overset{(c)}{=} \mathbb{E}_{A_t \sim \pi_\theta(S_t)} \Bigg[\sum_{\sigma=\tau^{(n-1)}}^{\tau^{(n)}-1} \mathsf{R}_\sigma^{(n)} \Lambda_\sigma \Bigg]
\end{aligned} \tag{5.16}$$

where (a) follows from (5.15), (b) follows from changing the order of summations, and (c) follows from the definition of $\mathsf{R}_\sigma^{(n)}$ in (5.12). $\widehat{\nabla}\mathsf{R}$ is an unbiased and asymptotically consistent estimator of the right hand side of the last equation in (5.16). The result for $\widehat{\nabla}\mathsf{T}$ follows from a similar argument. ■

Algorithm 5 combines the above estimates with the stochastic gradient ascent iteration of Theorem 5.2.1. An immediate consequence of Proposition 5.2.2 and Theorem 5.2.1 is the following.

Corollary 5.2.1 *The sequence $\{\theta_m\}_{m\geq 1}$ generated by Algorithm 5 converges to a local maximum.* □

Remark 5.2.1 Algorithm 5 is presented in its simplest form. It is possible to use standard variance reduction techniques such as subtracting a baseline [40,91,133] to reduce variance. □

Remark 5.2.2 In Algorithm 5, we use two separate runs to compute $(\widehat{\mathsf{R}}_m, \widehat{\mathsf{T}}_m)$ and $(\nabla\widehat{\mathsf{R}}_m, \nabla\widehat{\mathsf{T}}_m)$ to ensure that the independence condition of Proposition 5.2.2 is satisfied. In practice, we found that using a single run to compute both $(\widehat{\mathsf{R}}_m, \widehat{\mathsf{T}}_m)$ and $(\nabla\widehat{\mathsf{R}}_m, \nabla\widehat{\mathsf{T}}_m)$ has negligible effect on the accuracy of convergence (but speeds up convergence by a factor of two). □

[3] Log-derivative trick: For any distribution $p(x|\theta)$ and any function f,

$$\nabla_\theta \mathbb{E}_{X\sim p(X|\theta)}[f(X)] = \mathbb{E}_{X\sim p(X|\theta)}[f(X)\nabla_\theta \log p(X|\theta)].$$

Remark 5.2.3 It has been reported in the literature [121] that using a biased estimate of the gradient given by:

$$\mathsf{R}_\sigma^{(n)} = \Gamma^{(n)} \sum_{t=\sigma}^{\tau^{(n)}-1} \gamma^{t-\sigma} R_t, \tag{5.17}$$

(and a similar expression for $T_\sigma^{(n)}$) leads to faster convergence. We call this variant *RMC with biased gradients* and, in our experiments, found that it does converge faster than RMC. □

5.2.2 Simultaneous perturbation based gradient estimator

Another approach to estimate performance gradient is to use simultaneous perturbation based estimates [16, 64, 82, 111]. The general one-sided form of such estimates is

$$\widehat{\nabla}\mathsf{R}_\theta = \delta(\widehat{\mathsf{R}}_{\theta+c\delta} - \widehat{\mathsf{R}}_\theta)/c$$

where δ is a random variable with the same dimension as θ and c is a small constant. The expression for $\widehat{\nabla}\mathsf{T}_\theta$ is similar. When $\delta_i \sim \text{Rademacher}(\pm 1)$, the above method corresponds to simultaneous perturbation stochastic approximation (SPSA) [82, 111]; when $\delta \sim \text{Normal}(0, I)$, it corresponds to smoothed function stochastic approximation (SFSA) [16, 64].

Substituting these estimates in (5.10) and simplifying, we get

$$\widehat{H}_\theta = \delta(\widehat{\mathsf{T}}_\theta \widehat{\mathsf{R}}_{\theta+c\delta} - \widehat{\mathsf{R}}_\theta \widehat{\mathsf{T}}_{\theta+c\delta})/c.$$

The complete algorithm in shown in Algorithm 6. Since $(\widehat{\mathsf{R}}_\theta, \widehat{\mathsf{T}}_\theta)$ and $(\widehat{\mathsf{R}}_{\theta+c\delta}, \widehat{\mathsf{T}}_{\theta+c\delta})$ are estimated from separate sample paths, $\widehat{H}_\theta$ defined above is an unbiased estimator of H_θ. Then, an immediate consequence of Proposition 5.2.2 and Theorem 5.2.1 is the following.

Corollary 5.2.2 *The sequence $\{\theta_m\}_{m\geq 1}$ generated by Algorithm 6 converges to a local maximum.* □

Algorithm 6: RMC Algorithm with simultaneous perturbation based gradient estimates.

input : Intial policy θ_0, discount factor γ, initial state s_0, number of regenerative cycles N, constant c, perturbation distribution Δ

for *iteration* $m = 0, 1, \ldots$ **do**

  for *regenerative cycle* $n_1 = 1$ *to* N **do**

    Generate n_1-th regenerative cycle using policy π_{θ_m}.

    Compute $\mathsf{R}^{(n_1)}$ and $\mathsf{T}^{(n_1)}$ using (5.4).

  Set $\widehat{\mathsf{R}}_m = \texttt{mean}(\mathsf{R}^{(n_1)} : n_1 \in \{1, \ldots, N\})$.

  Set $\widehat{\mathsf{T}}_m = \texttt{mean}(\mathsf{T}^{(n_1)} : n_1 \in \{1, \ldots, N\})$.

  Sample $\delta \sim \Delta$.

  Set $\theta'_m = \theta_m + c\delta$.

  for *regenerative cycle* $n_2 = N + 1$ *to* $2N$ **do**

    Generate n_2-th regenerative cycle using policy π_{θ_m}.

    Compute $\mathsf{R}^{(n_2)}$ and $\mathsf{T}^{(n_2)}$ using (5.4).

  Set $\widehat{\mathsf{R}}'_m = \texttt{mean}(\mathsf{R}^{(n_2)} : n_2 \in \{N + 1, \ldots, 2N\})$.

  Set $\widehat{\mathsf{T}}'_m = \texttt{mean}(\mathsf{T}^{(n_2)} : n_2 \in \{N + 1, \ldots, 2N\})$.

  Set $\widehat{H}_m = \delta(\widehat{\mathsf{T}}_m \widehat{\mathsf{R}}'_m - \widehat{\mathsf{R}}_m \widehat{\mathsf{T}}'_m)/c$.

  Update $\theta_{m+1} = \left[\theta_m + \alpha_m \widehat{H}_m\right]_\Theta$.

5.2.3 Remark on average reward setup

The results presented above also apply to average reward models where the objective is to maximize

$$J_\pi = \lim_{t_h \to \infty} \frac{1}{t_h} \mathbb{E}_{A_t \sim \pi(S_t)} \left[\sum_{t=0}^{t_h - 1} R_t \;\middle|\; S_0 = s_0 \right]. \tag{5.18}$$

Let the stopping times $\tau^{(n)}$ be defined as before. Define the total reward $\mathsf{R}^{(n)}$ and duration $\mathsf{T}^{(n)}$ of the n-th regenerative cycle as

$$\mathsf{R}^{(n)} = \sum_{t=\tau^{(n-1)}}^{\tau^{(n)} - 1} R_t \quad \text{and} \quad \mathsf{T}^{(n)} = \tau^{(n)} - \tau^{(n-1)}.$$

Let R_θ and T_θ denote the expected values of $\mathsf{R}^{(n)}$ and $\mathsf{T}^{(n)}$ under policy π_θ. Then from standard renewal theory we have that the performance J_θ is equal to $\mathsf{R}_\theta / \mathsf{T}_\theta$ and, therefore $\nabla_\theta J_\theta = H_\theta / T_\theta^2$, where H_θ is defined as in (5.8). We can use both variants of RMC presented

above to obtain estimates of H_θ and use these to update the policy parameters using (5.9).

5.3 RMC for Post-Decision State Model

In many models, the state dynamics can be split into two parts: a controlled evolution followed by an uncontrolled evolution. For example, many continuous state models have dynamics of the form $S_{t+1} = f(S_t, A_t) + N_t$, where $\{N_t\}_{t\geq 0}$ is an independent noise process. For other examples, see the inventory control and event-triggered communication models in Sec. 5.5. Such models can be written in terms of a post-decision state model described below.

Consider a post-decision state MDP with pre-decision state $S_t^- \in \mathcal{S}^-$, post-decision state $S_t^+ \in \mathcal{S}^+$, action $A_t \in \mathcal{A}$. The system starts at an initial state $s_0^+ \in \mathcal{S}^+$ and at time t:

1. there is a controlled transition from S_t^- to S_t^+ according to a transition kernel $P^-(A_t)$;
2. there is an uncontrolled transition from S_t^+ to S_{t+1}^- according to a transition kernel P^+;
3. a per-step reward $R_t = r(S_t^-, A_t, S_t^+)$ is received.

Remark 5.3.1 When $\mathcal{S}^+ = \mathcal{S}^-$ and P^+ is identity, then the above model reduces to the standard MDP model, considered in Sec. 5.2. When P^+ is a deterministic transition, the model reduces to a standard MDP model with post decision states [93, 123]. □

As in Sec. 5.2, we choose a (time-homogeneous and Markov) policy π that maps the current pre-decision state $\mathcal{S}^-$ to a distribution on actions, i.e., $A_t \sim \pi(S_t^-)$. We use $\pi(a|s^-)$ to denote $\mathbb{P}(A_t = a|S_t^- = s^-)$.

The performance when the system starts in post-decision state $s_0^+ \in \mathcal{S}^+$ and follows policy π is given by

$$J_\pi = \mathbb{E}_{A_t\sim\pi(S_t)}\left[\sum_{t=0}^{\infty} \gamma^t R_t \;\middle|\; S_0^+ = s_0^+\right], \tag{5.19}$$

where $\gamma \in (0, 1)$ is the discount factor. As before, we are interested in identifying an optimal policy, i.e., a policy that maximizes the performance. When $\mathcal{S}$ and $\mathcal{A}$ are Borel spaces, we assume that the model satisfies the standard conditions under which time-homogeneous

Markov policies are optimal [47]. Let $\tau^{(n)}$ denote the stopping times such that $\tau^{(0)} = 0$ and for $n \geq 1$,

$$\tau^{(n)} = \min\{t > \tau^{(n-1)} : s^+_{t-1} = s^+_0\}.$$

The slightly unusual definition (using $s^+_{t-1} = s^+_0$ rather than the more natural $s^+_t = s^+_0$) is to ensure that the formulas for $\mathsf{R}^{(n)}$ and $\mathsf{T}^{(n)}$ used in Sec. 5.2 remain valid for the post-decision state model as well. Thus, using arguments similar to Sec. 5.2, we can show that both variants of RMC presented in Sec. 5.2 converge to a locally optimal parameter θ for the post-decision state model as well.

5.4 Approximate RMC

In this section, we present a variant of RMC that trades off accuracy with the speed of convergence. One potential limitation of RMC is that the system may take a long time to revisit the initial state. We can circumvent this limitation by considering a "renewal set" B around the start state and pretending that a renewal takes place whenever the state enters B. Doing so, results in a loss in accuracy. Since each regenerative cycles does not start in the same state, the renewal relationship of Proposition 5.2.1 is no longer valid. Nonetheless, in this section, we show that if the model has sufficient regularity so that the value function is locally Lipschitz in the renewal set, the error due to this approximation is bounded.

Suppose that the state and action spaces $\mathcal{S}$ and $\mathcal{A}$ are separable metric spaces (with metrics d_S and d_A). Given a "renewal set" B containing the start state s_0 and let $\rho^B = \sup_{s \in B} d_S(s, s_0)$ denote the radius of B with respect to s_0. Given a policy π, let $\tau^{(n)}$ denote the stopping times for successive visits to B, i.e., $\tau^{(0)} = 0$ and for $n \geq 1$,

$$\tau^{(n)} = \min\{t > \tau^{(n-1)} : s_t \in B\}.$$

Define $\mathsf{R}^{(n)}$ and $\mathsf{T}^{(n)}$ as in (5.4) and let R^B_θ and T^B_θ denote the expected values of $\mathsf{R}^{(n)}$ and $\mathsf{T}^{(n)}$, respectively. Define

$$J^B_\theta = \frac{\mathsf{R}^B_\theta}{(1-\gamma)\mathsf{T}^B_\theta}.$$

Theorem 5.4.1 *Given a policy π_θ, let V_θ denote the value function and $\overline{\mathsf{T}}^B_\theta = \mathbb{E}_{A_t \sim \pi_\theta(S_t)}[\gamma^{\tau^{(1)}} | S_0 = s_0]$ (which is always less than γ). Suppose the following condition is satisfied:*

(C) *The value function V_θ is locally Lipschitz in B, i.e., there exists a L_θ such that for any $s, s' \in B$,*

$$|V_\theta(s) - V_\theta(s')| \leq L_\theta d_S(s, s').$$

Then

$$\left|J_\theta - J_\theta^B\right| \leq \frac{L_\theta \overline{\mathsf{T}}_\theta^B}{(1-\gamma)\mathsf{T}_\theta^B}\rho^B \leq \frac{\gamma}{(1-\gamma)} L_\theta \rho^B. \tag{5.20}$$

PROOF We follow an argument similar to Proposition 5.2.1.

$$\begin{aligned} J_\theta = V_\theta(s_0) &= \mathbb{E}_{A_t \sim \pi_\theta(S_t)}\Bigg[\sum_{t=0}^{\tau^{(1)}-1} \gamma^t R_t \\ &\qquad + \gamma^{\tau^{(1)}} \sum_{t=\tau^{(1)}}^{\infty} \gamma^{t-\tau^{(1)}} R_t \;\Bigg|\; S_0 = s_{\tau^{(1)}}\Bigg] \\ &\overset{(a)}{=} \mathsf{R}_\theta^B + \mathbb{E}_{A_t \sim \pi_\theta(S_t)}[\gamma^{\tau^{(1)}} | S_0 = s_0]\, V_\theta(s_{\tau^{(1)}}) \end{aligned} \tag{5.21}$$

where (a) uses the strong Markov property [84]. Since V_θ is locally Lipschitz with constant L_θ and $s_{\tau^{(1)}} \in B$, we have that

$$|J_\theta - V_\theta(s_{\tau^{(1)}})| = |V_\theta(s_0) - V_\theta(s_{\tau^{(1)}})| \leq L_\theta \rho^B.$$

Substituting the above in (5.21) gives

$$J_\theta \leq \mathsf{R}_\theta^B + \overline{\mathsf{T}}_\theta^B (J_\theta + L_\theta \rho^B).$$

Substituting $\mathsf{T}_\theta^B = (1 - \overline{\mathsf{T}}_\theta^B)/(1-\gamma)$ and rearranging the terms, we get

$$J_\theta \leq J_\theta^B + \frac{L_\theta \overline{\mathsf{T}}_\theta^B}{(1-\gamma)\mathsf{T}_\theta^B}\rho^B.$$

The proof for the other direction is similar. The second inequality in (5.20) follows from $\overline{\mathsf{T}}_\theta^B \leq \gamma$ and $\mathsf{T}_\theta^B \geq 1$. ■

Based on Theorem 5.4.1, a policy that minimizes J_θ^B is approximately optimal. Such a policy can be identified by modifying both variants of RMC to declare a renewal whenever the state lies in B.

Local Lipschitz continuity of value functions can be verified for specific models (e.g., the model presented in Sec. 5.5.3). Sufficient conditions for *global* Lipschitz continuity have been identified in [48, Theorem 4.1], [95, Lemma 1, Theorem 1], and [92, Lemma 1]). We state these conditions below.

Proposition 5.4.1 *Let V_θ denote the value function for any policy π_θ. Suppose the model satisfies the following conditions:*

1. *The transition kernel P is Lipschitz, i.e., there exists a constant L_P such that for all $s, s' \in \mathcal{S}$ and $a, a' \in \mathcal{A}$,*

$$\mathcal{K}(P(\cdot|s,a), P(\cdot|s',a')) \leq L_P\big[d_S(s,s') + d_A(a,a')\big],$$

   where $\mathcal{K}$ is the Kantorovich metric (also called the Wasserstein distance) between probability measures.

2. *The per-step reward r is Lipschitz, i.e., there exists a constant L_r such that for all $s, s', s_+ \in \mathcal{S}$ and $a, a' \in \mathcal{A}$,*

$$|r(s,a,s_+) - r(s',a',s_+)| \leq L_r\big[d_S(s,s') + d_A(a,a')\big].$$

In addition, suppose the policy satisfies the following:

3. *The policy π_θ is Lipschitz, i.e., there exists a constant L_{π_θ} such that for any $s, s' \in \mathcal{S}$,*

$$\mathcal{K}(\pi_\theta(\cdot|s), \pi_\theta(\cdot|s')) \leq L_{\pi_\theta}\, d_S(s,s').$$

4. *$\gamma L_P(1 + L_{\pi_\theta}) < 1$.*

5. *The value function V_θ exists and is finite.*

Then, V_θ is globally Lipschitz. In particular, for any $s, s' \in \mathcal{S}$,

$$|V_\theta(s) - V_\theta(s')| \leq L_\theta d_S(s,s'),$$

where $L_\theta = L_r(1 + L_{\pi_\theta})/\big(1 - \gamma L_P(1 + L_{\pi_\theta})\big)$.

5.5 Numerical Experiments

We present three experiments to evaluate the performance of RMC: a randomly generated MDP, event-triggered communication, and inventory management. The code for all the experiments is available at [114].

5.5.1 Randomized MDP (GARNET)

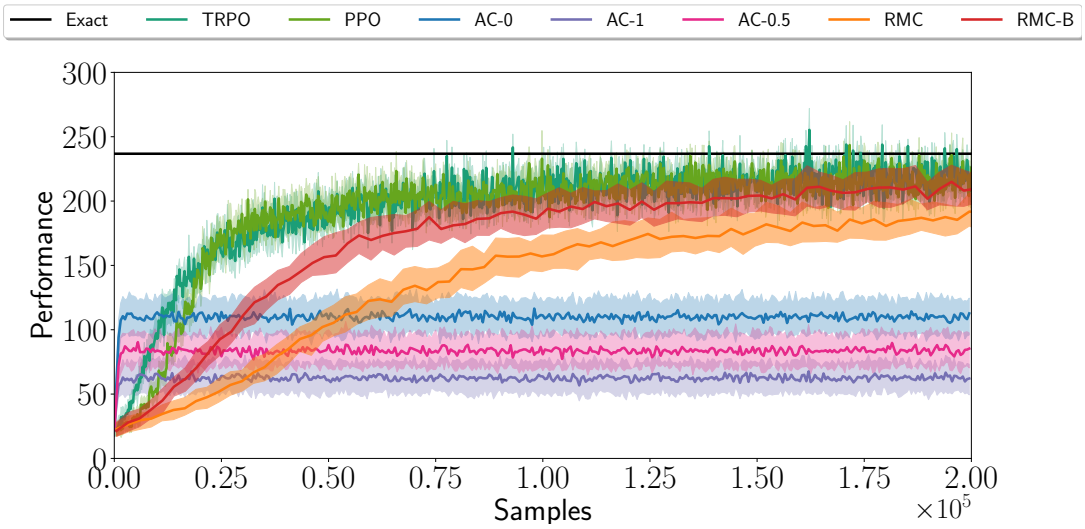


Fig. 5.1: GARNET: Comparison of RMC with other state of the art algorithms. The solid lines show the median values and the shaded area shows the region between the first and third quartiles.

In this experiment, we study a randomly generated GARNET$(100, 10, 50)$ model [15], which is an MDP with 100 states, 10 actions, and a branching factor of 50 (which means that each row of all transition matrices has 50 non-zero elements, chosen Unif$[0, 1]$ and normalized to add to 1). For each state-action pair, with probability $p = 0.05$, the reward is chosen Unif$[10, 100]$, and with probability $1-p$, the reward is 0. The discount factor $\gamma = 0.9$. The first state is chosen as start state. The policy is parameterized by a Gibbs soft-max distribution (which has states $\times$ actions $= 100 \times 10$ parameters) where each parameter belongs to the interval $[-10, 10]$ and the temperature is kept constant and equal to 1.

We compare the performance of the following algorithms:

1. RMC with likelihood ratio based gradient estimator (see Sec. 5.2.1) where the gradient is estimated using a single run (see Remark 5.2.2 in Sec. 5.2). The policy parameters are updated after $N = 4$ renewals and the learning is adapted using ADAM$(0.05)^4$ [65].

2. RMC with biased gradient denoted by RMC-B (see Remark 5.2.2) where all parameters are same as in RMC.

3. Actor critic with eligibility traces for the critic [116], which we refer to as AC-λ with $\lambda \in \{0, 0.5, 1\}$, where the learning rate for the actor is adapted using ADAM(0.1) [65].

4. TPRO [100] and PPO [101], which are two state of the art policy gradient based RL algorithms for models with discrete action spaces, where we use the default architecture and parameters from ChainerRL [25].

We run each algorithm for 2×10^5 samples and repeat this experiment 100 times. To compare the performance of these algorithms, we periodically evaluate the performance of π_{θ_m} for each trajectory using Monte Carlo evaluation (over 200 samples averaged over 10 independent runs). The median, first quartile, and third quartile across 100 runs are shown in Fig. 5.1. The optimal performance (which is computed using value iteration and the knowledge of the model) is also shown.

We observe that AC-λ, TRPO, and PPO learn faster (which is expected because the critic is keeping track of the entire value function) and have variance comparable to RMC and RMC-B. AC-λ gets stuck in a local minimum while RMC, RMC-B, TRPO, and PPO do not. Policy gradient algorithms only guarantee convergence to a local optimum. We are not sure why AC-λ converges to a different local maximum from RMC, RMC-B, TRPO and PPO. We also observe that RMC-B (which is RMC with biased evaluation of the gradient) learns faster than RMC.

It is worth highlighting that although TRPO/PPO converge in fewer number of samples compared to RMC/RMC-B, they require significantly more computational resources. In our experiments, each run of TRPO took ≈ 10 minutes (wall clock time), PPO took ≈ 16 minutes, AC-λ took ≈ 1 minute, whereas RMC/RMC-B took ≈ 40 seconds.

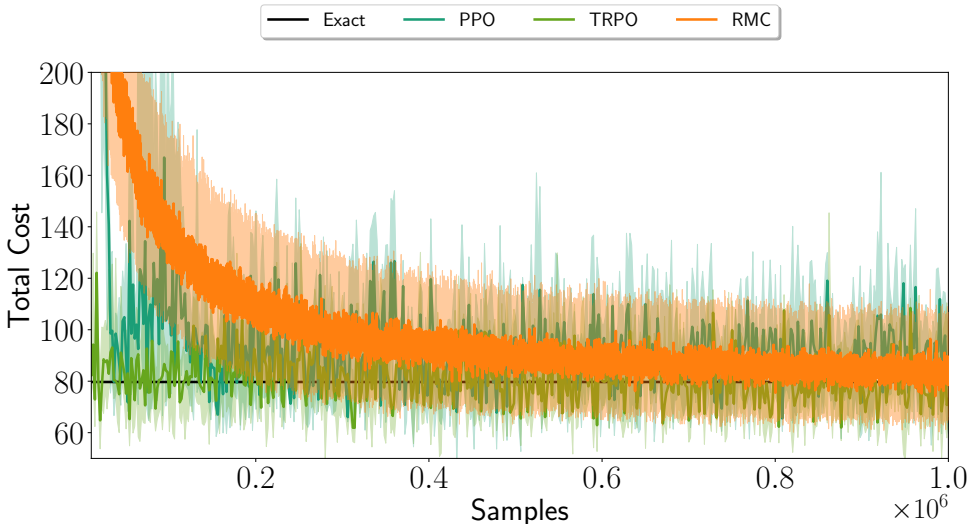


Fig. 5.2: Event-Triggered communication: Comparison of RMC with other state of the art algorithms. The solid lines show the median values and the shaded area shows the region between the first and third quartiles.

5.5.2 Event-Triggered Communication

In this experiment, we study an event-triggered communication problem that arises in networked control systems [27, 74]. A transmitter observes a first-order autoregressive process $\{X_t\}_{t\geq 1}$, i.e., $X_{t+1} = \alpha X_t + W_t$, where $\alpha, X_t, W_t \in \mathbb{R}$, and $\{W_t\}_{t\geq 1}$ is an i.i.d. process. At each time, the transmitter uses an event-triggered policy (explained below) to determine whether to transmit or not (denoted by $A_t = 1$ and $A_t = 0$, respectively). Transmission takes place over an i.i.d. erasure channel with erasure probability p_d. Let S_t^- and S_t^+ denote the "error" between the source realization and its reconstruction at a receiver. It can be shown that S_t^- and S_t^+ evolve as follows [27, 74]: when $A_t = 0$, $S_t^+ = S_t^-$; when $A_t = 1$, $S_t^+ = 0$ if the transmission is successful (w.p. $(1 - p_d)$) and $S_t^+ = S_t^-$ if the transmission is not successful (w.p. p_d); and $S_{t+1}^- = \alpha S_t^+ + W_t$. Note that this is a post-decision state model, where the post-decision state resets to zero after every successful transmission.[5]

[4]We use ADAM(α) to denote the choice of the α parameter of ADAM. All other parameters have their default value.

[5]Had we used the standard MDP model instead of the post-decision state model, this restart would not have always resulted in a renewal.

The per-step cost has two components: a communication cost of λA_t, where $\lambda \in \mathbb{R}_{>0}$ and an estimation error $(S_t^+)^2$. The objective is to minimize the expected discounted cost.

An event-triggered policy is a threshold policy that chooses $A_t = 1$ whenever $|S_t^-| \geq \theta$, where θ is a design choice. Under certain conditions, such an event-triggered policy is known to be optimal [27,74]. When the system model is known, algorithms to compute the optimal θ are presented in [26,136]. In this section, we use RMC to identify the optimal policy when the model parameters are not known.

In our experiment we consider an event-triggered model with $\alpha = 1$, $\lambda = 500$, $p_d = 0.0$, $W_t \sim \mathcal{N}(0,1)$, $\gamma = 0.9$.

We compare the performance for the following algorithms:

1. RMC with simultaneous perturbation based gradient estimate (see Sec. 5.2.2)[6], where the policy is parameterized by the threshold θ. We choose $c = 0.3$, $N = 1$ and $\Delta = \mathcal{N}(0,1)$ in Algorithm 6. The learning rate is adapted using ADAM(0.01) [65].

2. TPRO [100] and PPO [101], which are two state of the art policy gradient based RL algorithms for models with discrete action spaces, where we use the default architecture and parameters from ChainerRL [25].

We run each algorithm for 2×10^6 samples and repeat this experiment 100 times for RMC and 10 times for TRPO and PPO. To compare the performance of these algorithms, we periodically evaluate the performance of π_{θ_m} for each trajectory using Monte Carlo evaluation (over 200 samples averaged over 10 independent runs). The median, first quartile, and third quartile across the runs are shown in Fig. 5.2. The optimal total cost computed using [26] and the knowledge of the model is also shown in Fig. 5.2.

We observe that all three algorithms converge to the optimal values. TRPO and PPO converge in fewer number of samples (which is expected because the critic is keeping track of the entire value function), but require significantly more computational resources. In our experiments, each run of TRPO took ≈ 1.4 hours (wall clock time), PPO took ≈ 2.7 hours whereas RMC took ≈ 0.5 seconds.

[6] An event-triggered policy is a parametric policy but $\pi_\theta(a|s^-)$ is not differentiable in θ. Therefore, the likelihood ratio method cannot be used to estimate performance gradient.

5.5.3 Inventory Control

In this experiment, we study an inventory management problem that arises in operations research [7, 11]. Let $S_t \in \mathbb{R}$ denote the volume of goods stored in a warehouse, $A_t \in \mathbb{R}_{\geq 0}$ denote the amount of goods ordered, and D_t denotes the demand. The state evolves according to $S_{t+1} = S_t + A_t - D_{t+1}$.

We work with the normalized cost function:

$$C(s) = a_p s(1-\gamma)/\gamma + a_h s \mathbb{1}_{\{s \geq 0\}} - a_b s \mathbb{1}_{\{s<0\}},$$

where a_p is the procurement cost, a_h is the holding cost, and a_b is the backlog cost (see [129, Chapter 13] for details).

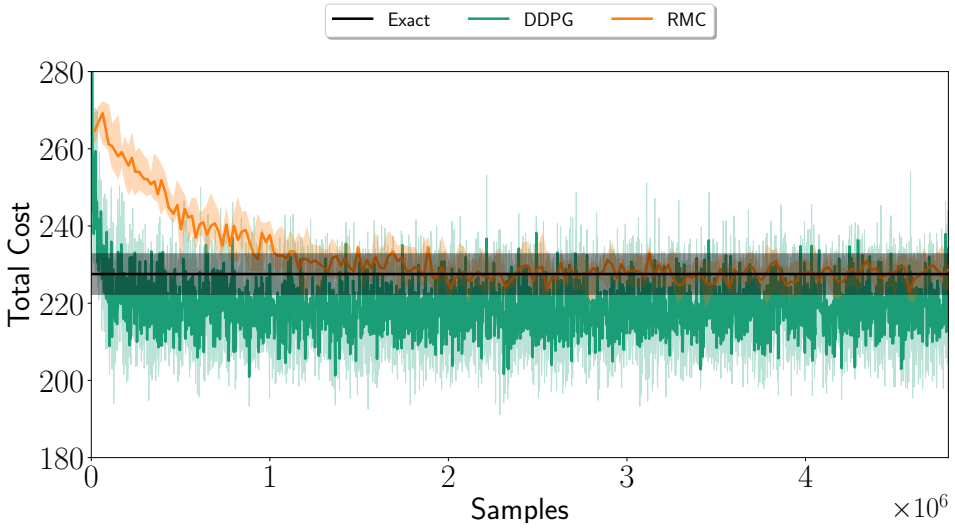


Fig. 5.3: Inventory control: Comparison of RMC with other state of the art algorithms. The solid lines show the median values and the shaded area shows the region between the first and third quartiles.

It is known that there exists a threshold θ such that the optimal policy is a base stock policy with threshold θ (i.e., whenever the current stock level falls below θ, one orders up to θ). Furthermore, for $s \leq \theta$, we have that [129, Sec. 13.2]

$$V_\theta(s) = C(s) + \frac{\gamma}{(1-\gamma)} \mathbb{E}[C(\theta - D)]. \tag{5.22}$$

So for $B \subset (0, \theta)$, the value function is locally Lipschitz in B with

$$L_\theta = \left(a_h + \frac{1-\gamma}{\gamma} a_p \right).$$

So, we can use approximate RMC to learn the optimal policy.

In our experiments, we consider an inventory management model with $a_h = 1$, $a_b = 1$, $a_p = 1.5$, $D_t \sim \text{Exp}(\lambda)$ with $\lambda = 0.025$, start state $s_0 = 1$, discount factor $\gamma = 0.9$.

We compare the performance for the following algorithms:

1. RMC with simultaneous perturbation based gradient (see Sec. 5.2.2), where the policy is parameterized by the threshold θ. We choose $c = 3.0$, $N = 100$, and $\Delta = \mathcal{N}(0, 1)$ in Algorithm 6 and choose $B = (0, 1)$ for approximate RMC. The learning rate is adapted using ADAM(0.25) [65].

2. DDPG [73], which is of one of state of the art RL algorithms for models with continuous action spaces, where we use the default architecture and implementation from ChainerRL [25].

We run each algorithm for $\approx 5 \times 10^6$ samples and repeat this experiment 100 times for RMC and 10 times for DDPG. To compare the performance of these algorithms, we use Monte Carlo evaluation (over 200 samples averaged over 100 independent runs for RMC and 10 independent runs for DDPG) periodically to evaluate the performance of π_{θ_m} for each trajectory. The median, first quartile and third quartile across the runs are shown in Fig. 5.3. The optimal performance computed using [129, Sec. 13.2][7] is also shown.

We observe that DDPG learns in fewer number of samples but it takes more time. In our experiments each run of DDPG took ≈ 10 hours (wall clock time) whereas RMC took ≈ 30 seconds. In addition, RMC converges smoothly to an approximately optimal parameter value with total cost within the bound predicted in Theorem 5.4.1. The grey rectangular region in Fig. 5.3 shows this bound.

[7] For $\text{Exp}(\lambda)$ demand, the optimal threshold is (see [129, Sec. 13.2])

$$\theta^* = \frac{1}{\lambda} \log \left(\frac{a_h + a_b}{a_h + a_p(1-\gamma)/\gamma} \right).$$

5.6 Conclusions

We present a renewal theory based reinforcement learning algorithm called Renewal Monte Carlo (RMC). RMC retains the key advantages of Monte Carlo methods and has low bias, is simple and easy to implement, and works for models with continuous state and action spaces. In addition, due to the averaging over multiple renewals, RMC has low variance. We generalize the RMC algorithm to post-decision state models and present a variant that converges faster to an approximately optimal policy, where the renewal state is replaced by a renewal set. The error in using such an approximation is bounded by the size of the renewal set.

In certain models, one is interested in the performance at a reference state that is not the start state. In such models, we can start with an arbitrary policy and ignore the trajectory until the reference state is visited for the first time and use RMC from that time onwards (assuming that the reference state is the new start state).

Chapter 6

Conclusion

In this thesis we studied reinforcement learning in certain classes of multi-agent systems. In order to do so, we first relaxed two requirements of MDPs, which are the most common mathematical fraework for single agent reinforcement learning. The first requirement that we relaxed is that of full observabililty and the second is that of stationarity (time homogeneity) of the environment.

Relaxation of the requirement of full observability led us to the problem of reinforcement learning in POMDPs, for which we developed a novel RL approach. We extended this approach to present an RL algorithm for a class of large population cooperative systems called mean-field teams.

Relaxation of the stationarity requirement, led us to the problem of decoupling the learning of each agent from the other agents. Several approaches for such decoupling between agents have been proposed in literature for various classes of MAS. In this thesis, we presented an RL algorithm for a class of large population systems called stationary mean-field games, where the decoupling is achieved by the mean-field (empirical distribution of the states and/or actions) of the system.

We also studied single agent RL in problems where the system had a renewal/regenerative property. For such systems, we developed a policy gradient based algorithm which is much simpler than conventional RL algorithms as it exploits the renewal property.

6.1 Summary

6.1.1 AIS for POMDPs

In the first technical chapter of this thesis, we studied RL in POMDPs, motivated by the fact that partial observability is an essential characteristic in most multi-agent systems. We provided two different definitions of information state, i.e., state sufficient for dynamic programming in partially observable Markov decision processes. We then relaxed the conditions to be satisfied by an information state process to yield two different definitions for approximate information states (AIS) along with the associated generators for the AIS. We showed that planning using AIS instead of an information state has bounded sub-optimality, where this bound is derived in terms of the approximation quantities that are used in the definition of the AIS. We also bounded the loss in performance when a policy learned using an AIS is used in the original system.

We showed that AIS can be learned from data and thus is useful for RL in systems where the transition and observation likelihood models are not known. We proposed an RL algorithm that learns an AIS along with learning an optimal policy for the AIS based system. We proved the convergence of this algorithm under standard technical conditions using the multi-time scale stochastic approximation theory. We also presented an recurrent neural network based function approximation architecture for implementation of this RL algorithm and demonstrated its numerical performance on four toy problems. In all these four cases, our proposed algorithm performed favorably when compared with one of the state of the art algorithms for RL in POMDPs.

6.1.2 RL in mean-field teams

The first multi-agent model in which we studied RL in this thesis is the mean-field team model. Mean-field teams are cooperative multi-agent systems with a large number of agents that have negligible individual impact on the system, and where the agents are coupled with each other only through the empirical distribution of their states and/or actions. The solution concept in such systems is defined as team optimality, where all agents act to optimize the cumulative team reward. It has been shown in literature that such systems can be modeled as decentralized POMDPs. Furthermore, using a decomposition approach from literature, called the common information approach, such systems can be converted into

POMDPs. Having used such a reduction to a POMDP, we first showed that considering a mean-field limit model, i.e., a model where the number of agents is assumed to be infinite, can be seen as using an AIS with a generator comprised of an identity mapping for the AIS and the infinite population deterministic transition and reward functions for the generator's transition and reward functions. Using this, we bounded the sub-optimality of using this mean-field limit system to derive an optimal policy for the finite population system. Then, using the mean-field limit system based AIS, we related the relation between the performance of an m-population system's optimal policy in an n-population system.

We defined two RL approaches based on access to either a system-level simulator or an agent-level simulator. We demonstrated the performance of RL using an agent-level simulator for two problems—which are stylized models of the demand response problem and the malware spread problem. Furthermore, for the demand response problem, we showed that the optimal performance of our RL algorithm matches very closely with the optimal planning solution from literature. For this problem, we also numerically illustrate the sub-optimality of using a mean-field limit solution (obtained using a system-level simulator) and a 100-agent system solution (obtained using an agent-level simulator) in systems with populations ranging from 100 to 1000.

6.1.3 RL in stationary mean-field games

The second multi-agent model in which we studied RL in this thesis is the stationary mean-field game model. Mean-field games are non-cooperative multi-agent systems with a large number of agents that have negligible individual impact on the system, and where the agents are coupled with each other only through the empirical distribution of their states and/or actions. Stationary mean-field games are mean-field games where all agents play identical stationary policies and the resultant mean-field becomes stationary. Typically two solution concepts—stationary mean-field equilibrium, sand stationary mean-field social welfare optimal policy. We defined localized extensions of these two solution concepts and presented two RL algorithms that provably converge to these two solution concepts under standard technical conditions. We numerically demonstrated the performance of these algorithms on two stylized models and for one, where a planning based solution was available from literature, we showed convergence to a policy with close to optimal performance.

6.1.4 Renewal Monte Carlo

In the last technical chapter of this thesis, we studied single-agent systems which have a renewal or regenerative structure. We showed that in such systems, we can use renewal theory to estimate the infinite horizon performance of a policy using a finite length trajectory that covers a single renewal cycle. Using the gradients of the renewal cycle discounted cumulative reward and renewal cycle discounted cumulative time with respect to the policy parameters, we defined a policy gradient theorem. We proved the convergence of the RL algorithm that uses this gradient formulation. We extended this policy gradient theorem to systems where renewals are defined in terms of post-decision states. We also defined the concept of an approximate renewal and extended our renewal theory based policy gradient theorem to such systems. We theoretically bounded the error due to this approximation in terms of the approximation radius used in defining this approximate renewal. For all three variants of our algorithm, called Renewal Monte Carlo (RMC), we demonstrated the performance on three problems respectively and also showed that RMC performs better or as good as some of the more complex RL algorithms in problems with renewal/regenerative property.

6.2 Future work

We believe that our proposed frameworks of AIS and RL in mean-field systems can be extended to other models as well. For instance, for single agent POMDPs, an AIS and its associated generator can serve as a model for model-based RL. This approach can be combined with the RL approach presented in Chapter 2 to develop a Dyna Q [117] like algorithm for POMDPs. This combined model-free and model-based RL approach can be extended to mean-field teams in a straightforward manner. Furthermore, using the common information approach for dynamic programming decomposition for teams, we can extend the AIS based RL algorithm for a class of cooperative multi-agent systems. The differentiating characteristic here is that this approach permits using a decentralized critic in addition to a decentralized actor and thus will be a fully decentralized learning algorithm. This is in contrast to the most popular learning framework in multi-agent systems—centralized learning and decentralized execution, or the variant where decentralized learning is accomplished using inter-agent communication that is only available during learning and not execution.

Another extension of our mean-field RL algorithm could be in models representing games between mean-field teams.

Our RMC approach can also be used as an alternative algorithm for MARL in systems with renewal. Furthermore, it can also be combined with other conventional TD algorithms as an added estimate in states where approximate renewals occur, albeit at a loss of the simplicity of the RMC algorithm. RMC in MARL

6.3 Final thoughts

In this thesis, we attempted to address the problem of RL in multi-agent systems. Multi-agent systems is a vast field with various sub-areas that have very different characteristics. While are nowhere close to arriving at a generic RL algorithm that can address this entire gamut of sub-areas, we believe we have developed some tools—AIS and simulation based RL, that will prove useful in several of these sub-areas.

Out of the two key problems in MARL—partial observability and non-stationarity of environment, we have been able to address the first one to a considerable extent using the AIS approach. This approach not only provides a theoretical framework to analyze RL in POMDPs, but also provides a practical algorithm for the same. Though we have only illustrated this algorithm using recurrent neural networks, newer approaches to handle temporal sequences such as transformers show significant promise. This is based on their successes in problems in natural language processing, which also have a significant temporally extended behavior.

For the non-stationarity problem, we chose to follow a model-specific approach— we restricted attention to mean-field systems where the mean-field decouples the agents from each other and conditioned on the mean-field the environment becomes stationary. Almost all other theoretical approaches to MARL in literature can be considered as using some kind of decoupling object such as knowledge of the opponent's reward in zero-sum games, centralized critic in MADDPG, COMA etc. It would be interesting to see if a more general decoupling scheme that extends beyond specific models or centralized training frameworks can be derived for MARL.

This page is intentionally left blank.

Appendix A

Background of neural network architectures

In this appendix, we present a brief overview of the various (artificial) neural network architectures and terminologies that have been used in this thesis. We begin by describing a layer in neural networks, activation functions and then describe architectures built using these layers and activation functions—primarily feed-forward neural networks and recurrent neural networks. There have been several challenges in training recurrent neural networks and two ways to overcome these have been presented in literature. These are long short-term memory (LSTM) and gated recurrent units (GRU), which are then detailed in two following subsections. Most of the explanations in this appendix are based on the text book [39]. The original references for the various concepts mentioned here and the history can be found in [39].

The process of computing an output using a neural network involves sequential transformations of the input using the layers and activation functions and this step is called forward-propagation. The various parameters used in these layers and activation functions are called parameters or weights of the neural network. Neural networks are trained using a gradient based update rule to update their parameters or weights, that minimizes some pre-defined loss function of the output of the neural network. The gradient of this loss function with respect to the weights is computed using chain rule of differentiation. This process of gradient computation is called back-propagation or backprop. The following

equation represents a generic backprop operation:

$$\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in Pa(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}},$$

where $u^{(n)}$ is the n^{th} neuron (i.e. parameters or weights corresponding to that neuron) or could be the final loss function, $u^{(j)}$ is one of the parents of $u^{(n)}$, i.e., one of the nodes that feeds into $u^{(n)}$. The set of parents of any node $u^{(i)}$ is denoted by $Pa(u^{(i)})$. A detailed algorithm to for backprop is given in [39, Algorithm 6.2]. Therefore, if we can compute the gradient of one neuron or layer with respect to its immediately preceding (input) neuron or layer, we can use the above chain rule to compute the gradient of the loss with respect to all the weights of the neural network. The gradients of different layers with respect to their inputs and the gradients of the various activation functions are given below.

A.1 Layer and activation

A layer in a neural network is a set of neurons that process input in parallel. A layer can be thought of as a transformation function that transforms an input x into an output y. There are various types of layers used in neural networks, thought the only one used in this thesis is the fully connected layer. Some common layer types are:

1. **Fully connected/Dense layer**: Here all the elements of the input vector (x) affect all the elements of the output vector (y). Let m be the dimension of the input vector and n be the number of neurons in this layer. Then a fully connected transformation function is represented by an $m \times n$ matrix W and an n-dimensional vector called bias and the transformation function is given as:

   $$y = Wx + b.$$

   This is one of the most common transformation functions and this is the only one used in this thesis. The gradient of the output of this layer with respect to its parameter W is x and with respect to b is 1 (of appropriate dimension).

2. **Convolutional layer**: Typically, convolutional layers are used to process image inputs. Here the input x is 2 or 3 dimensional. For illustration, we consider a 2D

image of size $m \times n$. A convolutional layer has a kernel K, usually a square kernel of size $k \times k$ and stride s, which is usually 1. In this case, the output is also 2-dimensional and its $(i, j)^{th}$ term is given by :

$$y_{i,j} = (x * K)_{i,j} = \sum_a \sum_b x_{a,b} K_{i-a,j-b}.$$

For further details on choosing different strides and different edge and corner paddings and multiple convolution channels, please see [39]. The convolutional layer has parameters equal to the kernel size and the gradients with respect to these parameters can be computed using the above linear equation. Each parameter in the kernel would receive gradient input from more than one output neuron based on the kernel size and stride length and the final gradient with respect to that parameter would be the addition of all these gradients.

3. **Pooling layer**: A pooling layer, also typically used for multi-dimensional inputs, compresses the input elements in a region into a single value which depends on the pooling function. Common pooling functions are max, min, mean etc. For instance, if the input is a 2-dimensional one and the pooling layer has a $k \times k$ square region with a max pooling function, then it compresses each $k \times k$ region of the input to the maximum of these $k \times k$ values. For a pooling layer, the gradient is only propagated from the output to all the neurons that are used in computing the output of the pooling layer. For example, if a max pooling layer is used, the the gradient is only passed to the input neuron with the maximum value in the pooling region and all other neurons in this region get a zero gradient.

An activation function is typically a non-linear function that acts on the output of the layer to yield an output of the same dimension. Often the transformation function is non-linear and squashes the layer output onto some predefined range. Some common activation functions are:

1. **Sigmoid**: $y = \sigma(x) := \frac{1}{1+e^{-x}}$, gradient: $\sigma(x)(1 - \sigma(x))$.
2. **Tanh**: $y = \tanh(x)$, gradient: $1 - \tanh^2(x)$.
3. **ReLU**: This is called a rectified linear unit: $y = \max(0, x)$, (sub-)gradient: 1 if $x > 0$, else 0.

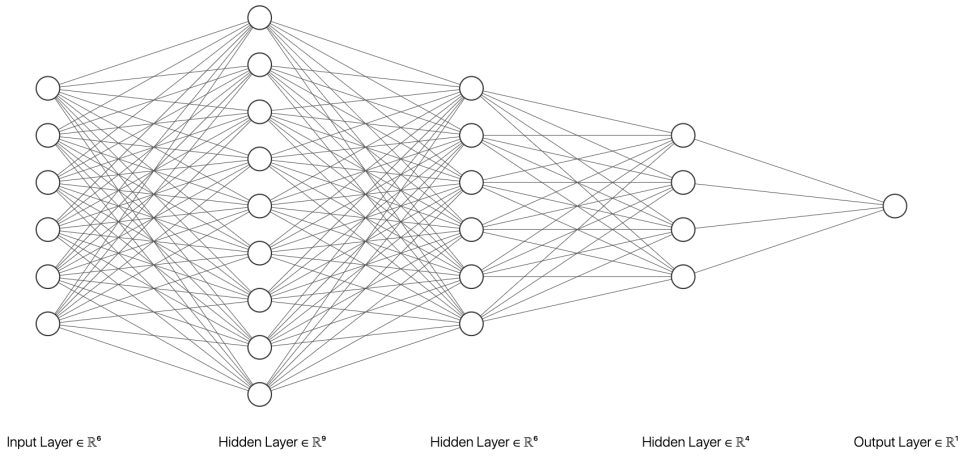


Fig. A.1: A fully connected neural network with 3 hidden layers.

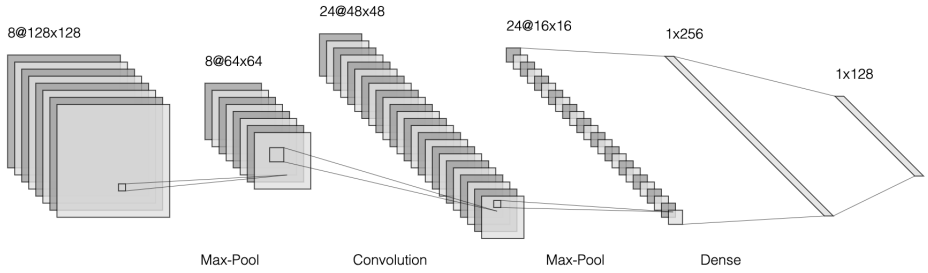


Fig. A.2: A convolutional neural network with 4 hidden layers.

A.2 Feed-forward neural network

A feed-forward neural network is constructed using a sequence of layers and activation functions. The first layer is the input itself and the last layer is called the output layer. All layers between the input and the output are called hidden layers. If there are more than 2 hidden layers, the neural network is considered deep. The following two figures given an example of a 3 hidden layer fully connected neural network (Fig. A.1), where each node represents a neuron and a 4 hidden layer convolutional neural network (Fig. A.2). Both these figures are generated using the tool given in [71]. The backprop for a feed-forward neural network is done using chain rule as mentioned earlier, where the gradient for each layer and activation function is as given in Sec. A.1.

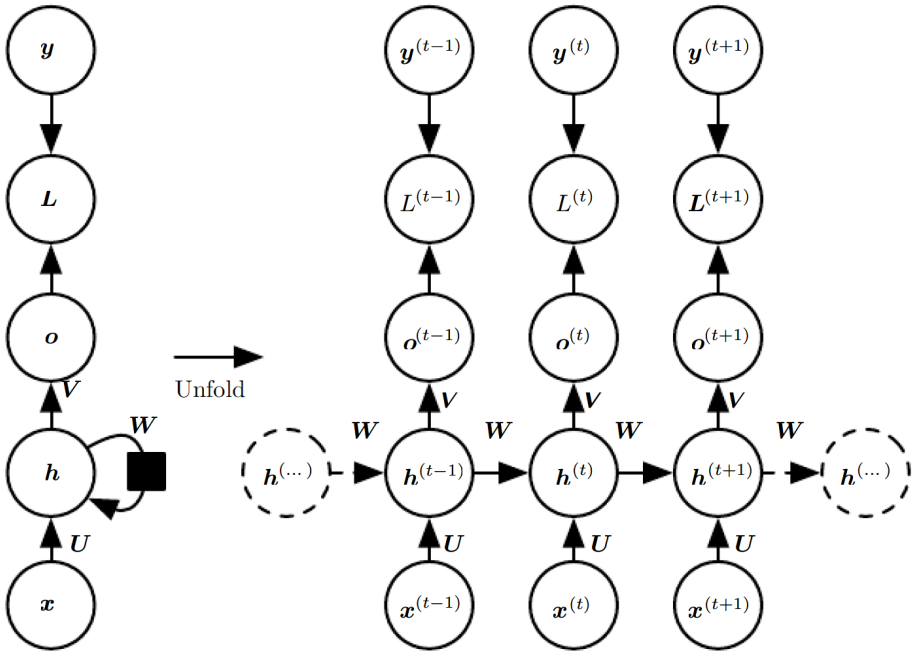


Fig. A.3: Diagram showing unrolling of a recurrent neural network (RNN).

A.3 Recurrent neural network (RNN)

A feed-forward network, though powerful, cannot capture temporal dependencies in data. For this purpose, recurrent neural networks are used. They have an architecture similar to a feed-forward network. However, they also have a hidden state that represents the sequence length processed thus far. Conceptually a recurrent neural network can be viewed as a time unrolled feed-forward neural network, where at each time t, the current unrolled feed-forward network uses a (hidden) state $h^{(t)}$ and the current element of the input sequence $x^{(t)}$ to yield an output $o^{(t)}$ and the next state $h^{(t+1)}$. This can be written as:

$$o^{(t)}, h_{t+1} = f(x^{(t)}, h^{(t)}),$$

where f represents the unrolled feed-forward neural network. This is given in Fig. A.3, which is taken from [39].

The gradient for a recurrent neural network is computed using back-propagation through time, which is similar to the backprop for feed-forward neural network now applied to the unrolled network.

These recurrent neural networks can process sequences of arbitrary length. In deep learning, networks are trained using backpropagation, which involves using chain rule to compute gradients of a loss function with respect to the parameters of the network. In the case of recurrent neural networks the sequence of chain rule applications and thus the number of product terms in the gradient computation increases with the input sequence length. Hence, when processing large sequences of inputs, we often face the problem of vanishing or exploding gradients. In order to address this issue, two approaches have been proposed in literature, which in effect break these long sequences by defining additional functions called gates to compute and update the RNN state. These are described in the following subsections.

A.3.1 Long short-term memory (LSTM)

The block diagram for an LSTM is given in Fig. A.4, which is taken from [39]. In an LSTM, there are three additional gates associated with the hidden state. These gates are functions given below. This entire network of gates and hidden state is called an LSTM cell. It is important to note that the LSTM cell has an internal recurrence in addition to the external recurrence of the RNN. This internal recurrence introduces a self-loop that generates paths where gradients can persist for a longer time. This is the key idea behind an LSTM. A more detailed explanation can be found in [39].

1. **Forget gate**: The forget gate function is given by:

$$f_i^{(t)} = \sigma\Big(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\Big),$$

   where σ is the sigmoid function, b^f, U^f, W^f are the biases, input weights and recurrent weights for the forget gate.

2. **(External) Input gate**: The input gate function is given by:

$$g_i^{(t)} = \sigma\Big(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\Big),$$

   where σ is the sigmoid function, b^g, U^g, W^g are the biases, input weights and recurrent weights for the input gate.

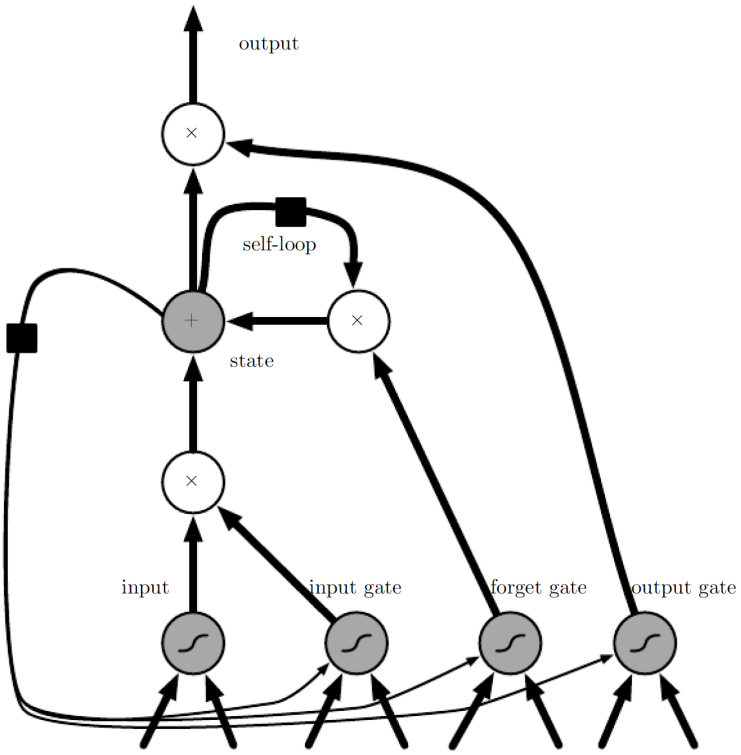


Fig. A.4: Block diagram of a long short-term (LSTM) cell.

3. **Output gate**: The output gate function is given by:

$$q_i^{(t)} = \sigma\Big(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\Big),$$

where σ is the sigmoid function, b^o, U^o, W^o are the biases, input weights and recurrent weights for the output gate.

4. **LSTM cell state update**: The update equation for the hidden state of the LSTM cell, denoted by $\mathbf{s^{(t)}}$ is given by:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\Big(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\Big),$$

where b, U, W denote the biases, input weights and recurrent weights for the LSTM cell.

The output of the LSTM cell is then computed as:

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)}$$

The gradient with respect to all the above gates and output can be computed using chain rule in a straightforward manner as they only involve a sigmoid or tanh function and a linear transformation (dense layer). The gradients of these are given in Sec. A.1.

A.3.2 Gated recurrent unit (GRU)

Another gated RNN architecture is the GRU. The main difference between a GRU and an LSTM is that in a GRU there is a single gate that controls the forget and LSTM cell state update part. Thus a GRU has two gates given by:

1. **Update gate**:
$$u_i^{(t)} = \sigma\Big(b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t-1)}\Big),$$
where σ is the sigmoid function, b^u, U^u, W^u are the biases, input weights and recurrent weights for the update gate.

2. **Reset gate**:
$$r_i^{(t)} = \sigma\Big(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t-1)}\Big),$$
where σ is the sigmoid function, b^r, U^r, W^r are the biases, input weights and recurrent weights for the reset gate.

The output of the GRU is computed as:

$$h_i^{(t)} = u_i^{(t-1)} h_i^{(t-1)} + (1 - u_i^{(t-1)}) \sigma\Big(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\Big),$$

where σ is the sigmoid function, b, U, W are the biases, input weights and recurrent weights for the GRU output.

The gradient with respect to all the above gates and output can be computed using chain rule in a straightforward manner as they only involve a sigmoid function and a linear transformation (dense layer). The gradients of these are given in Sec. A.1.

A.4 Conclusion

The purpose of this appendix is to give the reader a brief overview of neural network architectures. More detail regarding architectures, training algorithms for neural networks, original references, as well as history of development can be found in [39].

This page is intentionally left blank.

References

[1] David Abel, D. Ellis Hershkowitz, and Michael L. Littman. Near optimal behavior via approximate state abstraction. *arXiv:1701.04113*, 2017.

[2] Sachin Adlakha, Ramesh Johari, and Gabriel Y Weintraub. Equilibria of dynamic games with many players: Existence, approximation, and market structure. *Journal of Economic Theory*, 156:269–316, 2015.

[3] Jalal Arabneydi. *New Concepts in Team Theory: Mean Field Teams and Reinforcement Learning*. PhD thesis, McGill University, 2016.

[4] Jalal Arabneydi and Amir G. Aghdam. A certainty equivalence result in team-optimal control of mean-field coupled markov chains. In *56th IEEE Annual Conference on Decision and Control, CDC 2017, Melbourne, Australia, December 12-15, 2017*, pages 3125–3130, 2017.

[5] Jalal Arabneydi and Aditya Mahajan. Team optimal control of coupled subsystems with mean-field sharing. In *IEEE Conference on Decision and Control*, pages 1669–1674. IEEE, 2014.

[6] Jalal Arabneydi and Aditya Mahajan. Linear Quadratic Mean Field Teams: Optimal and Approximately Optimal Decentralized Solutions. *ArXiv e-prints*, August 2016.

[7] Kenneth J Arrow, Theodore Harris, and Jacob Marschak. Optimal inventory policy. *Econometrica: Journal of the Econometric Society*, pages 250–272, 1951.

[8] Andrea Baisero and Christopher Amato. Learning internal state models in partially observable environments;. *Reinforcement Learning under Partial Observability, NeurIPS Workshop*, 2018.

[9] Bram Bakker. Reinforcement learning with long short-term memory. In *NIPS*, 2002.

[10] Jonathan Baxter and Peter L Bartlett. Infinite-horizon policy-gradient estimation. *JAIR*, 15:319–350, 2001.

[11] Richard Bellman, Irving Glicksberg, and Oliver Gross. On the optimal inventory equation. *Management Science*, 2(1):83–104, 1955.

[12] James Bergin and Dan Bernhardt. Anonymous sequential games: existence and characterization of equilibria. *Economic Theory*, 5(3):461–489, 1995.

[13] D Bertsekas. Convergence of discretization procedures in dynamic programming. *IEEE Trans. Autom. Control*, 20(3):415–419, 1975.

[14] D.P. Bertsekas and J.N. Tsitsiklis. *Neuro-dynamic Programming*. Anthropological Field Studies. Athena Scientific, 1996.

[15] S Bhatnagar, R.S. Sutton, M. Ghavamzadeh, and M. Lee. Natural actor-critic algorithms. Technical report, Dept. of Computing Science, University of Alberta, Canada, 2009.

[16] Shalabh Bhatnagar, HL Prasad, and LA Prashanth. *Stochastic Recursive Algorithms for Optimization: Simultaneous Perturbation Methods*, volume 434. Springer, 2013.

[17] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, 53:659–697, 2015.

[18] T. Bohlin. Information pattern for linear discrete-time models with stochastic coefficients. *IEEE Transactions on Automatic Control*, 15(1):104–106, Feb 1970.

[19] Vivek Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge University Press, 2008.

[20] Vivek S Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291–294, 1997.

[21] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*, pages 183–221. Springer, 2010.

[22] Pierre Cardaliaguet and Saeed Hadikhanloo. Learning in mean field games: the fictitious play. *ESAIM: Control, Optimisation and Calculus of Variations*, 23(2):569–591, 2017.

[23] Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, 1994.

[24] Pablo Samuel Castro, Prakash Panangaden, and Doina Precup. Equivalence relations in fully and partially observable markov decision processes. In *IJCAI*, 2009.

[25] ChainerRLRepository. https://github.com/chainer/chainerrl.

[26] Jhelum Chakravorty and Aditya Mahajan. Fundamental limits of remote estimation of Markov processes under communication constraints. *IEEE Trans. Autom. Control*, 62(3):1109–1124, March 2017.

[27] Jhelum Chakravorty, Jayakumar Subramanian, and Aditya Mahajan. Stochastic approximation based methods for computing the optimal thresholds in remote-state estimation with packet drops. In *Proc. American Control Conference*, pages 462–467, Seattle, WA, May 2017.

[28] Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[29] Gheorghe Comanici and Doina Precup. Basis function discovery using spectral clustering and bisimulation metrics. In *AAAI*, 2011.

[30] Gheorghe Comanici, Doina Precup, and Prakash Panangaden. Basis refinement strategies for linear value function approximation in MDPs. In *NIPS*, 2015.

[31] M.H.A Davis and P.P Varaiya. Information states for linear stochastic systems. *Journal of Mathematical Analysis and Applications*, 37(2):384–402, feb 1972.

[32] William Feller. *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley and Sons, 1966.

[33] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite markov decision processes. In *UAI*, 2004.

[34] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 2974–2982, 2018.

[35] R.E. Funderlic and C.D. Meyer. Sensitivity of the stationary distribution vector for an ergodic markov chain. *Linear Algebra and its Applications*, 76:1 – 17, 1986.

[36] Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. Deepmdp: Learning continuous latent space models for representation learning. *CoRR*, 2019.

[37] Peter Glynn. Optimization of stochastic systems. In *Proc. Winter Simulation Conference*, pages 52–59, Dec. 1986.

[38] Peter Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33:75–84, 1990.

[39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[40] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[41] Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, 2016.

[42] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[43] Matthew Hausknecht and Peter Stone. Deep recurrent Q-learning for partially observable MDPs. In *2015 AAAI Fall Symposium Series*, 2015.

[44] Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-based control with recurrent neural networks. *arXiv:1512.04455*, 2015.

[45] Ahmed Hefny, Zita Marinho, Wen Sun, Siddhartha Srinivasa, and Geoffrey Gordon. Recurrent predictive state policy networks. *arXiv:1803.01489*, 2018.

[46] P. Hernandez-Leal, B. Kartal, and M. E. Taylor. Is multiagent deep reinforcement learning the answer or the question? A brief survey. *ArXiv e-prints*, October 2018.

[47] Onésimo Hernández-Lerma and Jean Bernard Lasserre. *Discrete-time Markov Control Processes: Basic Optimality Criteria*, volume 30. Springer, 1996.

[48] K. Hinderer. Lipschitz continuity of value functions in Markovian decision processes. *Mathematical Methods of Operations Research*, 62(1):3–22, Sep 2005.

[49] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[50] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

[51] M. Huang and Y. Ma. Mean field stochastic games: Monotone costs and threshold policies. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 7105–7110, Dec 2016.

[52] M. Huang and Y. Ma. Mean field stochastic games with binary action spaces and monotone costs. *ArXiv e-prints*, January 2017.

[53] M. Huang and Y. Ma. Mean field stochastic games with binary actions: Stationary threshold policies. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 27–32, Dec 2017.

[54] Minyi Huang, Peter E Caines, and Roland P Malhamé. Large-population cost-coupled LQG problems with nonuniform agents: individual-mass behavior and decentralized epsilon-Nash equilibria. *IEEE Transactions on Automatic Control*, 52(9):1560–1571, 2007.

[55] Minyi Huang, Peter E. Caines, and Roland P. Malhamé. The Nash certainty equivalence principle and Mckean-Vlasov systems: An invariance principle and entry adaptation. In *IEEE Conference on Decision and Control*, 2007.

[56] Minyi Huang, Roland P Malhamé, and Peter E Caines. Large population stochastic dynamic games: closed-loop Mckean-Vlasov systems and the Nash certainty equivalence principle. *Communications in Information & Systems*, 6(3):221–252, 2006.

[57] M. Hüttenrauch, A. Šošić, and G. Neumann. Deep Reinforcement Learning for Swarm Systems. *ArXiv e-prints*, July 2018.

[58] Libin Jiang, Venkat Anantharam, and Jean Walrand. How bad are selfish investments in network security? *IEEE/ACM Transactions on Networking (TON)*, 19(2):549–560, 2011.

[59] Boyan Jovanovic and Robert W. Rosenthal. Anonymous sequential games. *Journal of Mathematical Economics*, 17(1):77 – 87, 1988.

[60] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[61] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[62] Sham M Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems*, pages 1531–1538, Dec. 2002.

[63] Ian A Kash, Eric J Friedman, and Joseph Y Halpern. Multiagent learning in large anonymous games. *Journal of Artificial Intelligence Research*, 40:571–598, 2011.

[64] V. Katkovnik and Y. Kulchitsky. Convergence of a class of random search algorithms. *Automation and Remote Control*, 33(8):1321–1326, 1972.

[65] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[66] Arman C Kizilkale and Peter E Caines. Mean field stochastic adaptive control. *IEEE Transactions on Automatic Control*, 58(4):905–920, 2013.

[67] Vijay R Konda and John N Tsitsiklis. On actor-critic algorithms. *SIAM Journal on Control and Optimization*, 42(4):1143–1166, 2003.

[68] Harold Kushner and G George Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

[69] H. Kwakernaak. *Theory of Self-Adaptive Control Systems*, chapter Admissible Adaptive Control, pages 14–18. Springer, 1965.

[70] Jean-Michel Lasry and Pierre-Louis Lions. Mean field games. *Japanese Journal of Mathematics*, 2(1):229–260, 2007.

[71] Alexander Lenail. http://alexlenail.me/NN-SVG/index.html.

[72] D. S. Leslie. *Reinforcement learning in games*. PhD thesis, The University of Bristol, 2004.

[73] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations, San Juan, Puerto Rico, May 2-4*, 2016.

[74] G. M. Lipsa and Nuno Martins. Remote state estimation with communication costs for first-order LTI systems. *IEEE Trans. Autom. Control*, 56(9):2013–2025, September 2011.

[75] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning.*, 1994.

[76] Michael L. Littman. Friend-or-foe q-learning in general-sum games. In *International Conference on Machine Learning.*, 2001.

[77] Michael L Littman. Value-function reinforcement learning in markov games. *Cognitive Systems Research*, 2(1):55–66, 2001.

[78] Michael L Littman, Richard S Sutton, and Satinder P Singh. Predictive representations of state. In *NIPS*, 2002.

[79] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

[80] P. Marbach and J. N. Tsitsiklis. Simulation-based optimization of Markov reward processes. *IEEE Trans. Autom. Control*, 46(2):191–209, Feb 2001.

[81] P. Marbach and J. N. Tsitsiklis. Approximate gradient methods in policy-space optimization of Markov reward processes,. *Discrete Event Dynamical Systems*, 13(2):111–148, 2003.

[82] John L Maryak and Daniel C Chin. Global random optimization by simultaneous perturbation stochastic approximation. *IEEE Trans. Autom. Control*, 53(3):780–783, April 2008.

[83] R. Andrew McCallum. Overcoming incomplete perception with utile distinction memory. In *ICML*, 1993.

[84] Sean P Meyn and Richard L Tweedie. *Markov chains and stochastic stability*. Springer, 2012.

[85] David Mguni, Joel Jennings, and Enrique Munoz de Cote. Decentralised learning in systems with many, many strategic agents. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 4686–4693, 2018.

[86] Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997.

[87] Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Transactions on Automatic Control*, 58(7):1644–1658, 2013.

[88] A. Nerode. Linear automaton transformations. *Proceedings of American Mathematical Society*, 9:541–544, 1958.

[89] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2015.

[90] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[91] Jan Peters and Stefan Schaal. Policy gradient methods for robotics. In *International Conference on Intelligent Robots and Systems, 2006 IEEE/RSJ*, pages 2219–2225. IEEE, Oct. 2006.

[92] Matteo Pirotta, Marcello Restelli, and Luca Bascetta. Policy gradient in Lipschitz Markov decision processes. *Machine Learning*, 100(2):255–283, Sep 2015.

[93] Warren B Powell. *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons, second edition, 2011.

[94] Emmanuel Rachelson and Michail G. Lagoudakis. On the locality of action domination in sequential decision making. In *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2010, Fort Lauderdale, Florida, USA, January 6-8, 2010*, 2010.

[95] Emmanuel Rachelson and Michail G. Lagoudakis. On the locality of action domination in sequential decision making. In *International Symposium on Artificial Intelligence and Mathematics*, Fort Lauderdale, US, Jan. 2010.

[96] Lillian J Ratliff, Samuel A Burden, and S Shankar Sastry. On the characterization of local nash equilibria in continuous games. *IEEE Transactions on Automatic Control*, 61(8):2301–2307, 2016.

[97] Matthew Rosencrantz, Geoff Gordon, and Sebastian Thrun. Learning low dimensional predictive representations. In *ICML*, 2004.

[98] Reuven Y Rubinstein. Sensitivity analysis and performance extrapolation for computer simulation models. *Operations Research*, 37(1):72–81, 1989.

[99] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[100] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, June 2015.

[101] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[102] Raihan Seraj. Learning in the presence of partial observability and concept drifts. Master's thesis, McGill University, 2019.

[103] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *AAMAS*, 2013.

[104] Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent reinforcement learning: a critical survey. Technical report, Stanford University, 2003.

[105] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007.

[106] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian others Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[107] Satinder P Singh, Michael L Littman, Nicholas K Jong, David Pardoe, and Peter Stone. Learning predictive state representations. In *ICML*, 2003.

[108] Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

[109] Max Sommerfeld, Jörn Schrieber, Yoav Zemel, and Axel Munk. Optimal transport: Fast probabilistic approximation with exact solvers. *Journal of Machine Learning Research*, 20(105):1–23, 2019.

[110] Edward J Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations research*, 26(2):282–304, 1978.

[111] James C Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992.

[112] Charlotte Striebel. Sufficient statistics in the optimal control of stochastic systems. *Journal of Mathematical Analysis and Applications*, 12:576–592, 1965.

[113] Jayakumar Subramanian and Aditya Mahajan. Renewal monte carlo: Renewal theory based reinforcement learning. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 5759–5764. IEEE, 2018.

[114] Jayakumar Subramanian and Aditya Mahajan. Renewal Monte Carlo. https://codeocean.com/capsule/027c3bab-27cf-4f47-8153-6533c2bfc1e5, Aug. 2019.

[115] Wen Sun, Arun Venkatraman, Byron Boots, and J. Andrew Bagnell. Learning to filter with predictive state inference machines. In *ICML*, 2016.

[116] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 1998.

[117] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT Press, 2018.

[118] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, pages 1057–1063, Nov. 2000.

[119] Csaba Szepesvári. *Algorithms for reinforcement learning*. Morgan & Claypool Publishers, 2010.

[120] Yunhao Tang and Shipra Agrawal. Discretizing continuous action space for on-policy optimization. *CoRR*, abs/1901.10500, 2019.

[121] Philip Thomas. Bias in natural actor-critic algorithms. In *International Conference on Machine Learning*, pages 441–448, June 2014.

[122] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in Conference on Neural Information Processing Systems*, 2015.

[123] Benjamin Van Roy, Dimitri P Bertsekas, Yuchun Lee, and John N Tsitsiklis. A neuro-dynamic programming approach to retailer inventory management. In *36th IEEE Conference on Decision and Control, 1997*, volume 4, pages 4052–4057, Dec. 1997.

[124] Cédric Villani. *Optimal transport: Old and New*. Springer, 2008.

[125] Gabriel Y. Weintraub, C. Lanier Benkard, and Benjamin Van Roy. Oblivious Equilibrium: A Mean Field Approximation for Large-Scale Dynamic Games. In *Advances in Neural Information Processing Systems*, pages 1489–1496, December 2005.

[126] Gabriel Y Weintraub, C Lanier Benkard, and Benjamin Van Roy. Markov perfect industry dynamics with many firms. *Econometrica*, 76(6):1375–1411, 2008.

[127] Gabriel Y Weintraub, C Lanier Benkard, and Benjamin Van Roy. Computational methods for oblivious equilibrium. *Operations research*, 58(4-part-2):1247–1265, 2010.

[128] Pierre Weiss. L'hypothèse du champ moléculaire et la propriété ferromagnétique. *J. Phys. Theor. Appl.*, 6(1):661–690, 1907.

[129] Peter Whittle. *Optimization Over Time: Dynamic Programming and Optimal Control*. John Wiley and Sons, Ltd., 1982.

[130] Daan Wierstra, Alexander Foerster, Jan Peters, and Juergen Schmidhuber. Solving deep memory POMDPs with recurrent policy gradients. In *International Conference on Artificial Neural Networks*, 2007.

[131] Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Recurrent policy gradients. *Logic Journal of the IGPL*, 18(5):620–634, 2010.

[132] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.

[133] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[134] Hans S. Witsenhausen. Some remarks on the concept of state. In Y. C. Ho and S. K. Mitter, editors, *Directions in Large-Scale Systems*, pages 69–75. Plenum, 1976.

[135] Britton Wolfe, Michael R James, and Satinder Singh. Learning predictive state representations in dynamical systems without reset. In *ICML*, 2005.

[136] Y. Xu and J. P. Hespanha. Optimal communication logics in networked control systems. In *43rd IEEE Conference on Decision and Control*, pages 3527–3532, Dec. 2004.

[137] Jiachen Yang, Xiaojing Ye, Rakshit Trivedi, Huan Xu, and Hongyuan Zha. Deep mean field games for learning optimal behavior policy of large populations. In *International Conference on Learning Representations*, 2018.

[138] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2018.

[139] H. Yin, P. G. Mehta, S. P. Meyn, and U. V. Shanbhag. Learning in mean-field games. *IEEE Transactions on Automatic Control*, 59(3):629–644, March 2014.

[140] Amy Zhang, Zachary C. Lipton, Luis Pineda, Kamyar Azizzadenesheli, Anima Anandkumar, Laurent Itti, Joelle Pineau, and Tommaso Furlanello. Learning causal state representations of partially observable environments. *CoRR*, abs/1906.10437, 2019.

[141] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*, 2019.