Reinforcement Learning in Large State Action spaces

Anuj Mahajan

Hertford College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

To my family

Vandana, Mohan and Vaishali

Acknowledgements

This thesis would not have been possible without the support of a large number of amazing individuals and institutions. First and foremost, I would like to thank my advisor Shimon Whiteson for supporting me throughout my PhD, always providing useful feedback and perspectives. Working with Shimon has been a tremendous learning experience.

My sincere thanks and appreciation goes to Sergey Levine and Yarin Gal for their time and expertise to examine this thesis.

I would like to thank my collaborators during the PhD: Mikayel Samvelyan, Tarun Gupta, Mingfei Sun, Matthew Fellows, Kristian Hartikainen, Tabish Rashid and Ben Ellis. Also thanks to the other colleagues at WhiRL: Bei Peng, Wendelin Böhmer, Shangtong Zhang, Luisa Zintgraf, Jelena Luketina, Vitaly Kurin, Matt Smith, Jacob Beck, Risto Vuorio, Maximilian Igl and Supratik Paul, who made coming to the lab a fun experience.

A special thanks to Theja Tulabandhula who has been a fantastic mentor since my time at Xerox Research. Also thanks to Amy Zhang with whom, collaborating on state abstraction has been so much fun.

I did three wonderful internships during my PhD: at DeepMind London, J.P. Morgan London and Nvidia Santa Clara. Thanks to Max Jaderberg, Wojciech Czarnecki, Jakob Bauer, Jakub Sygnowski and other colleagues at Open Ended Learning team

at DeepMind, it was fun working on AI generalization with them. Thanks to Sammy Assefa, Prashant Reddy, Sumitra Ganesh and other collegues at J.P. Morgan for an awesome time despite the pandemic. Thanks to the colleagues at Nvidia: Yuke Zhu, Viktor Makoviychuk and other members of the ML algorithms team who made my first visit to US truly memorable.

Finally, I would like to thank my family whose love and affection has always been a driving force: My parents, Vandana and Mohan, for their continual support and motivation to work hard. My sister Vaishali for always being there to guide me and encouraging me to follow my passions.

Abstract

Reinforcement learning (RL) is a promising framework for training intelligent agents which learn to optimize long term utility by directly interacting with the environment. Creating RL methods which scale to large state-action spaces is a critical problem towards ensuring real world deployment of RL systems. However, several challenges limit the applicability of RL to large scale settings. These include difficulties with exploration, low sample efficiency, computational intractability, task constraints like decentralization and lack of guarantees about important properties like performance, generalization and robustness in potentially unseen scenarios.

This thesis is motivated towards bridging the aforementioned gap. We propose several principled algorithms and frameworks for studying and addressing the above challenges RL. The proposed methods cover a wide range of RL settings (single and multi-agent systems (MAS) with all the variations in the latter, prediction and control, model-based and model-free methods, value-based and policy-based methods). In this work we propose the first results on several different problems: e.g. tensorization of the Bellman equation which allows exponential sample efficiency gains (Chapter 4), provable suboptimality arising from structural constraints in MAS(Chapter 3), combinatorial generalization results in cooperative MAS(Chapter 5), generalization results on observation shifts(Chapter 7), learning deterministic policies in a probabilistic RL framework(Chapter 6). Our algorithms exhibit provably enhanced performance and sample efficiency along with better scal-

ability. Additionally, we also shed light on generalization aspects of the agents under different frameworks. These properties have been been driven by the use of several advanced tools (e.g. statistical machine learning, state abstraction, variational inference, tensor theory).

In summary, the contributions in this thesis significantly advance progress towards making RL agents ready for large scale, real world applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Artificial Intelligence (AI) holds tremendous promise in terms of being able to provide solutions for some of the biggest challenges we face today. These challenges come from a wide array of fields like agriculture, personalized medicine, energy production, sustainable development, better recommenders in the age of choices etc. Essentially, AI holds the potential to be applied to any task requiring human ingenuity and intellect and much beyond. This was also the founding vision for officially creating the field around 70 years ago.

However, historically, AI research has been focused on mimicking human-like reasoning abilities through creation of knowledge systems and using principles of formal reasoning towards creating expert systems that execute elaborate rules. Such systems typically required human experts towards carefully designing domain specific rules for automation and used elaborate search techniques for finding the solutions. This approach has driven lots of industrial development and enhanced our capabilities to create super-human systems for the fist time on narrow focus tasks: e.g. beating a human chess grand master like IBM's DeepBlue did.

While the aforementioned approach was effective in creating autonomous agents for controlled environment, such rule based systems often broke in the presence of uncer-

tainty and environment noise. Thus AI system remained far from delivering on their promise. This led to researchers thinking about ways to create AI systems which were robust to noise and came with guarantees about performance and learning. This led to the rapid development of Machine learning (ML), a field at the confluence of computer science, optimization and statistics. ML systems were extremely broad in terms of applicability and tolerance to noisy scenarios. Further, they have been extensively analysed and studied for leveraging structural properties in the data and learning in extreme situations involving limited amounts of data and computational intractability. Several innovative frameworks like statistical machine learning [108], support vector machines [229] and probabilistic graphical models [115] were created under this field. Over the past years this approach has transformed many areas in industry and otherwise. For example a lot of the tech companies like Amazon, Google and Netflix have benefited massively from automated advertising and recommendation systems driven by such innovations. Similarly, even the finance industry has several application for such systems. In fact, most of modern AI research still borrows heavily from concepts discovered during the rapid developmental phase of ML which started around 30 years ago.

Classical machine learning systems guaranteed principled development of complex solutions. They became good at modelling tasks like finding high level patterns in user behaviour, however, some problems which are otherwise easy for most humans and other living organisms, like reliably identifying a cat in a photograph remained unscathed for these classical systems. This was in part because these systems still relied heavily on expert knowledge of the underlying structure in the data for designing effective features for the task. Further, the lack of scalable compute methods limited the complexity and size of the models to consider while solving the problem (e.g. the curse of dimensionality is one such effect).

Things began to change around the previous decade when developments in other technological fields like acceleration hardware and software practices enabled Ma-

chine learning to enter a new phase of Deep Learning. This has brought a huge impact on automation for various research and industrial applications. Deep learning has been driving research in modern AI systems chiefly due to two factors: (1) Use of large neural networks for automated feature learning (2) Efficient training on very large amounts of data. Notable applications using deep learning include image recognition [124], speech-recognition [96], language translation [199], artificial data synthesis [230] amongst many others. These applications all entail training on a dataset consisting of inputs and their desired outputs and fall under the the supervised learning regime. Supervised learning is catered to a fixed underlying distribution of problem instances and the future data distribution is independent of current agent decisions/predictions.

However the advantages in automated learning of task relevant features has opened the doors to automation for extended decision making geared towards maximising long term utility. Such type of problems can be formalized under the Reinforcement Learning (RL) framework. In such tasks the autonomous agent* has to actively interact with the environment and learn from feedbacks, instead of being told the correct answer. These type of problems have tremendous potential for industrial applications like robotics, swarm intelligence, autonomous driving, financial markets etc. Deep reinforcement learning (DRL) which combines the deep learning technologies with reinforcement learning (aka learning from feedbacks) has proved to be very promising candidate for learning good agent policies for such applications.

1.1 Reinforcement Learning in Large State-Action Spaces

As mentioned above, Reinforcement Learning in general and Deep Reinforcement Learning specifically has immense potential in creating truly intelligent agents

*Hereon we use agents to imply autonomous agents unless specified otherwise

capable of long term decision making. Most real world RL applications can be characterized as those having large state action spaces. This makes learning in such spaces difficult due to a variety of reasons:

(1) Firstly, a large state action space makes the exploration difficult. This also makes it difficult to reduce uncertainties about the system. Thus innovative methods need to created so that the agents explore sufficiently and efficiently.

(2) A large state-action space also makes learning the underlying model parameters and objects used for decision making (like policy, value functions) difficult from a statistical perspective as the number of samples required for robust learning increases accordingly. In general, DRL agents have very poor sample efficiency. Even with rapid concurrent developments in scaling compute power and availability of big data, allowing for training at very large scales, it is still difficult to learn reasonable policies for large RL problems. Further, for decision tasks where obtaining the data is costly, DRL is not very helpful.

(3) For computational tractability, one has to often resort to approximate solutions for learning in large spaces which include strategies like factorization. The effects of such approximations are difficult to analyse due to iterated nature of the problem but they can lead to problems like severe sub-optimality.

(4) Several real world problems pose challenging constraints like learning decentralized control(see Section 2.5) for which any possible solution must have adverse computational complexity (e.g. Dec-POMDPs are NEXPTIME complete). Thus even large scale compute is often insufficient for such problems.

(5) Several instances show that DRL agents generalize very poorly. Small changes in deployment settings can often completely break learnt policies. This comes as a surprise given the amount of training data and compute required for DRL in addition to how good humans and other organisms are at generalization in comparison.

(6) Finally, as is typical of many deep learning approaches, DRL agents seldom come with guarantees about important properties like performance, generalization and

robustness in potentially unseen scenarios. These properties are especially difficult to study in large state-action spaces.

We now discuss instances of large state-action spaces relevant to this work. Many real world applications involve these instances alone or in combinations.

1.1.1 Multi-Agent systems

Many real world applications involve environments that contain a large number of learning agents, and are thus multi-agent in nature. Not all of the participating agents in such scenarios need to be machines. While we will primarily be interested in cooperative multi agent settings (MAS) most of the discussion also applies to general sum scenarios. In these settings a large number of agents need to coordinate towards maximising joint rewards, taking into account the presence of other agents in the environment. The state-action space in this setting grows exponentially in the number of agents, this makes it particularly susceptible to the needle in the haystack phenomenon as finding rewarding team actions and coordinating with the team members for exploration and adaptation becomes necessary [†]. Additionally, the number of varied interactions possible between the agents also grows exponentially in these systems which makes modelling and representing the underlying decision making objects like joint policy and value function computationally intractable. We will cover the multi-agent exploration and representation problem along with its implications on learning in Chapter 3, additionally, we will look at a completely new perspective towards tackling the representation problem in a statistically sound manner using the theory of tensors in Chapter 4. Often, agents only get a common reward, which means the agents need to learn to reason about their contribution to the rewards obtained and how they can improve. These systems also necessitate agents to demonstrate combinatorial generalization in addition to usual single agent generalization towards robust real world deployment, we will cover this in great

[†]Cooperative MAS have no mini-max performance guarantees unlike competitive MAS

detail in Chapter 5. Examples of cooperative MAS include autonomous vehicle fleet, swarm robotics, recommender systems etc.

1.1.2 Continuous state, action and context spaces

Several real world problems involve scenarios where either the state or the action or both are continuous. This characterization can be extended to systems having continuous observations. Moreover, several real world problems also contain inherent structure: like an underlying context which affects the agent rewards and transitions, such context can come from a continuous space e.g. the observation view angle of an autonomous car. Learning in these systems is difficult as policy search needs to be done on a continuous function space. Further, as it is impossible to try out each state actions combination due to its uncountable nature. Thus special focus is required to ensure adequate generalization both within the state-action space and across different such RL tasks using strategies like abstractions and metric learning. We will explore some of these problems in Chapter 5 and Chapter 7. Finally, due to the continuous nature of the state-action space, it becomes impossible to apply tabular approaches for uncertainty reduction and statistical robustness, hence once again problems like exploration, choosing the right policy class and designing sample efficient RL algorithms for inference and decision making need special attention. We will explore how some of these problems can be tackled by bringing the statistical methodology to bear under the framework of RL as inference in Chapter 6. Examples of continuous state-action problems include torque outputs for robotic joints, car steering angles in autonomous vehicles, visual inputs for robots, temperature of a chemical plant.

1.2 Contributions and Structure

The aforementioned challenges prevent DRL methods from being applied in large-scale practical scenarios. Thus, towards bridging the above gap, we propose several principled algorithms and frameworks for studying and addressing the challenges. We hope that this helps drive forward Deep-RL research for large scale systems and in the long term, increases their applicability for solving real world problems.

1.2.1 Core approach

The core strategy we use for tackling the problem of reinforcement learning in large state-action spaces is that of approximation. This helps us understand in a principled manner, the underlying similarities between the seemingly different problems. It also allows for finding common solution techniques which can used towards solving these hard problems and analyse the effects of approximation in terms of solution quality. Thus, it acts as the glue connecting various problems and solution methods developed in this thesis. The approximation approaches we use in this work can be broadly classified into those related to discrete optimization problems and those related to continuous optimization. The discrete problems we study in this thesis are particularly characterized by their combinatorial nature and typically admit factorization based methods: for instance, the problem of learning a monotonic decomposition of multi-agent action value function (Chapter 3) under the context of developing value based decentralized algorithms. On the other end of the spectrum, for instance in Chapter 7, we deal with the problem of generalizing across a large observation space that varies continuously with an underlying context, this allows for using elegant methods from state abstraction and metric learning to bear given the continuous structure. Quite uniquely, many of the problems covered here need a combination of approximation methods of both kinds to solve the problem. For instance, Chapter 3 also utilizes variational approximation from

the continuous domain to ensure the recovery of diverse monotonic projections. Similarly, in Chapter 5, where we study combinatorial generalization, we utilize both the discrete nature of team composition and the continuous dependence on the underlying agent capabilities for developing generalization bounds. In this thesis we also elucidate via the approximation strategy, what are the actual underlying challenges of solving a large (and hence difficult) RL problem. This often manifests as results obtained under the limit of approximation tending towards the original hard problem. For instance in Chapter 6, where we study the RL as inference problem, it becomes clear that the quality of the policy obtained via approximation is directly dependent on the complexity of the variational class used, and in the limit of using arbitrary non-parametric distributions, one can solve the problem exactly, albeit making the problem computationally intractable in the process. Similarly in Chapter 4, where we utilize tensor decompositions to approximate the joint action value function, we observe that as the approximation rank gets higher, the sample efficiency of the approach decreases making the problem difficult. We next discuss the structure of the thesis.

1.2.2 Thesis Structure

This thesis is divided into a background section followed by three main parts and a conclusion/discussion in the end. Each of the parts addresses several of the challenges involved in doing RL in large state-action spaces as outlined above. We next provide an overview of the different sections.

Background

In Chapter 2, we formally introduce the Reinforcement Learning problem and various settings used in this thesis. Additionally, we also discuss the necessary algorithmic and conceptual tools in RL along with methods in deep reinforcement learning which are common to the rest of the thesis. For the ease of exposition, background

concepts required only for a specifics chapter are introduced within the corresponding chapter.

Part 1: Multi Agent Systems

In the first part we address various challenges arising in learning in the cooperative multi agent setting.

In Chapter 3 we study the relation between representation and learning in multi agent systems. Centralised training with decentralised execution is an important setting for cooperative deep multi-agent reinforcement learning due to communication constraints during execution and computational tractability in training. In this work, we analyse value-based methods that are known to have superior performance in complex environments. We are the first to show that the representational constraints on the joint action-values introduced by the value based methods like VDN [198], QMIX [172] and other similar methods lead to provably poor exploration and sub-optimality. Furthermore, we propose a novel approach called MAVEN [140] that hybridises value and policy-based methods by introducing a latent space for hierarchical control. The value-based agents condition their behaviour on the shared latent variable controlled by a hierarchical policy. This allows MAVEN to achieve committed, temporally extended exploration, which is key to solving complex multi-agent tasks. Our experimental results show that MAVEN achieves significant performance improvements on the challenging SMAC domain [181].

In Chapter 4, we focus on the problem of sample efficient policy evaluation and critic learning for Cooperative multi-agent reinforcement learning (MARL). While RL in large action spaces is a challenging problem, MARL exacerbates matters by imposing various constraints on communication and observability. In this work, we consider the fundamental hurdle affecting both model based and model free(value-based and policy-gradient) approaches: an exponential blowup of the action space with the

number of agents. For model based methods, it makes sample efficient learning of the underlying parameters difficult. For policy gradient methods, it makes training the critic difficult and exacerbates the problem of the *lagging* critic similarly, for value-based methods, it poses challenges in accurately representing the optimal value function similarly. We show that from a learning theory perspective, both problems can be addressed by accurately representing the associated action-value function with a low-complexity hypothesis class. This requires accurately modelling the agent interactions in a sample efficient way. To this end, we propose a novel tensorised formulation of the Bellman equation. This gives rise to our method TESSERACT [142], which views the Q-function as a tensor whose modes correspond to the action spaces of different agents. Algorithms derived from TESSERACT decompose the Q-tensor across agents and utilise low-rank tensor approximations to model agent interactions relevant to the task. We provide probably approximately correct learning(PAC) analysis for TESSERACT-based algorithms and highlight their relevance to the class of rich observation MDPs. Empirical results in different domains confirm TESSERACT's gains in sample efficiency predicted by the theory. We are the first to apply tensor theory towards efficient learning in factored RL problems like MARL.

In Chapter 5 we study generalization in cooperative multi agent systems, which is a important property exhibited by several species of living organisms. As is commonly observed, such natural systems are very flexible to changes in their structure. Specifically, they exhibit a high degree of generalization when the abilities or the total number of agents changes within a system. We term this phenomenon as *Combinatorial Generalization* (CG). CG is particularly difficult for MAS as it leads to a combinatorial blow-up in the number of possible teams (w.r.t. agent capabilities) given a team size. Further the capabilities need to be grounded w.r.t. the dynamics of the environment which becomes increasingly hard with team size and the non-stattionarity introduced by other agents. CG is a highly desirable trait

for autonomous systems as it can increase their utility and deployability across a wide range of applications. While recent works addressing specific aspects of CG have shown impressive results on narrow domains, they provide no performance guarantees when generalizing towards novel situations. In this work [141], we shed light on the theoretical underpinnings of CG for cooperative multi-agent systems (MAS). Specifically, we study generalization bounds under a linear dependence of the underlying dynamics on the agent capabilities, which can be seen as a generalization of Successor Features to MAS. We then extend the results first for Lipschitz and then arbitrary dependence of rewards on team capabilities. Finally, empirical analysis on various domains using the framework of multi-agent reinforcement learning highlights important desiderata for multi-agent algorithms towards ensuring CG. This is the first work which defines a principled framework for studying combinatorial generalization in MAS.

Part 2: Learning in continuous state-action spaces

In the second part, we focus on learning in high dimensional continuous state-action spaces with emphasis on creating methods which can efficient and can principally reason about uncertainties in these spaces.

Chapter 6 discusses this problem in greater detail. An important approach towards achieving the above goal is applying probabilistic models to reinforcement learning, which enables the use of powerful optimisation tools such as variational inference in RL. However, existing inference frameworks and their algorithms pose significant challenges for learning optimal policies, e.g., the lack of mode capturing behaviour in pseudo-likelihood methods, difficulties learning deterministic policies in maximum entropy RL based approaches, and a lack of analysis when function approximators are used. We propose VIREL [58], a theoretically grounded inference framework for RL that utilises a parametrised action-value function to summarise future dynamics of the underlying MDP, generalising existing approaches. VIREL also benefits from

a mode-seeking form of KL divergence, the ability to learn deterministic optimal polices naturally from inference, and the ability to optimise value functions and policies in separate, iterative steps. Applying variational expectation-maximisation to VIREL, we show that the actor-critic algorithm can be reduced to expectation-maximisation, with policy improvement equivalent to an E-step and policy evaluation to an M-step. We derive a family of actor-critic methods from VIREL, including a scheme for adaptive exploration and demonstrate that our algorithms outperform state-of-the-art methods based on soft value functions in several domains.

Part 3: Learning to generalize across observation shifts

In the third part, we turn our attention to learning agent policies which are robust to changes in the environment and give good generalization across environment shifts. In Chapter 7 we focus on bisimulation metrics, which provide a powerful means for abstracting task relevant components of the observation and learning a succinct representation space for training the agent using reinforcement learning. In this work, we extend the bisimulation framework to also account for context dependent observation shifts. Specifically, we focus on the simulator based learning setting and use alternate observations to learn a representation space which is invariant to observation shifts using a novel bisimulation based objective. This allows us to deploy the agent to varying observation settings during test time and generalize to unseen scenarios. We further provide theoretical bounds for simulator fidelity and performance transfer guarantees for using a learnt policy to unseen shifts. Empirical analysis on the high-dimensional image based control domain [212] demonstrates the efficacy of our method.

1.2.3 Published works

Parts of this thesis are based on the following published works which I have authored:

- MAVEN: Multi-Agent Variational Exploration
  Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, Shimon Whiteson.
  NeurIPS, 2019

- VIREL: A Variational Inference Framework for Reinforcement Learning
  Matthew Fellows*, Anuj Mahajan*, Tim Rudner, Shimon Whiteson.
  NeurIPS, (Spotlight) 2019.

- Tessaract: Tensorised Actors for Multi-Agent Reinforcement Learning
  Anuj Mahajan, Mikayel Samvelyan, Lei Mao, Viktor Makoviychuk, Animesh Garg, Jean Kossaifi, Shimon Whiteson, Yuke Zhu, Animashree Anandkumar.
  ICML, 2021.

- Reinforcement Learning in Factored Action Spaces using Tensor Decompositions
  Anuj Mahajan, Mikayel Samvelyan, Lei Mao, Viktor Makoviychuk, Animesh Garg, Jean Kossaifi, Shimon Whiteson, Yuke Zhu, Animashree Anandkumar.
  QTNML, NeurIPS, 2021.

- Generalization in Cooperative Multi-Agent Systems
  Anuj Mahajan, Mikayel Samvelyan, Tarun Gupta, Benjamin Ellis, Mingfei Sun, Tim Rocktaschel, Shimon Whiteson.
  arxiv, 2022.

During the development of this thesis, I also worked on the following papers, which have not been included here:

- Open-Ended Learning Leads to Generally Capable Agents

*Equal contribution, author order decided by coin flip

Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, Wojciech Marian Czarnecki.
DeepMind Tech Report, 2021.

- Model based Multi-agent Reinforcement Learning with Tensor Decompositions
  Pascal Van Der Vaart, Anuj Mahajan, Shimon Whiteson.
  QTNML, NeurIPS, 2021.

- UneVEn: Universal Value Exploration for Multi-Agent Reinforcement Learning
  Tarun Gupta, Anuj Mahajan, Bei Peng, Wendelin Bohmer, Shimon Whiteson.
  ICML, 2021.

- RODE: Learning Roles to Decompose Multi-Agent Tasks
  Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, Chongjie Zhang.
  ICLR, 2021.

- Softdice for imitation learning: Rethinking off-policy distribution matching
  Mingfei Sun, Anuj Mahajan, Katja Hofmann, Shimon Whiteson.
  arxiv, 2021.

- SMACv2: A New Benchmark for Cooperative Multi-Agent Reinforcement Learning
  Benjamin Ellis, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, Shimon Whiteson.
  Openreview, 2022.

Chapter 2

Background

Contents

In this chapter we provide the necessary background information and formalisms used for the rest of the thesis. In particular we introduce Markov decision process (MDP) and Reinforcement Learning (RL), methods used for RL, partial observ-

ability and the multi-agent settings. Concepts which are required only for specific chapters are introduced as additional background in those chapters, see, e.g. tensor decompositions in Chapter 4. Furthermore, where appropriate, we revise some of the key concepts within the given chapters. Content in this chapter is based on all relevant papers and preprints mentioned in the introduction .

2.1 Markov Decision Process and Reinforcement learning

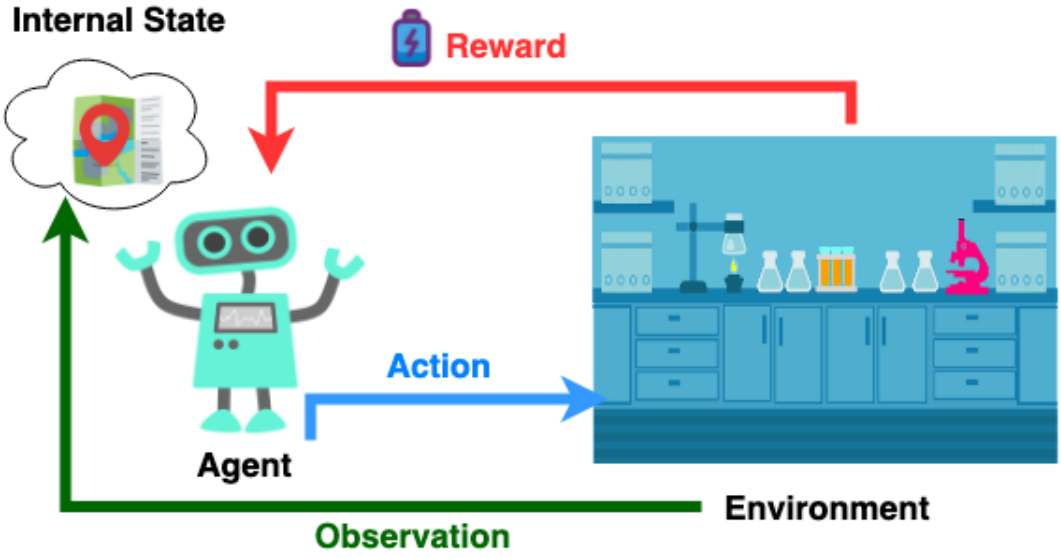


Figure 2.1: The reinforcement learning loop

We start with the simplest model for sequential decision making which can be captured using the formalism of a Markov Decision Process (MDP). This uses the Markov assumption ie. the rewards and the new environment states encountered by the agent while interacting with the environment are independent of the previous states and actions given the current state and agent action. An MDP is formally defined as a tuple $\langle S, U, P, r, \gamma, \rho \rangle$. Here S is the state space of the environment and ρ is the initial state distribution. At each time step t, an agent observes the

state $s \in S$ and chooses an action* $a \in U$ using its policy $\pi : S \to \mathcal{P}(U)$, where $\mathcal{P}(\cdot)$ represents the space of distributions on the argument set. This leads to a state transition governed by the distribution $P(s'|s,a) : S \times U \times S \to [0,1]$, and the agent receives reward† $r(s,a) : S \times U \to [0, R_{max}]$ which can be potentially stochastic. Fig. 2.1 illustrates the the reinforcement learning loop. We consider the discounted infinite horizon setting, where the discount factor is given by $\gamma \in [0,1)$. The episodic case which has a finite problem horizon can be viewed as a special case of the infinite horizon setting. The state-action trajectory of the agent is represented by $\tau \in T \equiv (S \times U)^*$, we overload the notation to also include rewards as necessary. We assume finite state and action sets although some of methods in this thesis are applicable to non-finite sets as well. The value of a policy is defined as:

$$J^\pi = \mathbb{E}_{\pi,\rho}\left[\sum_{t=0}^{\infty} \gamma^t r_\tau(s_t)\right]$$

The expectation on the RHS above is well defined given bounds on rewards and γ. We also define three other useful functions:

$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)|s_0 = s, a_0 = a\right]$$

$$V^\pi(s) = \mathbb{E}_{a\sim\pi} Q^\pi(s,a) \tag{2.1}$$

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s) \tag{2.2}$$

respectively called the action-value, value and advantage functions. The goal of the MDP problem is to find the optimal policy π^* corresponding to the optimal policy value J^*. It is well known that a deterministic optimal policy always exists for finite MDPs [201]. Further, the optimal value function V^* and optimal action value function Q^* also exhibit important properties like uniqueness and point-wise

*Following standard convention, for disambiguation, we use a to denote action in single agent and u for the same in multi-agent settings

†we use $R_{max} = 1$ unless specified

function dominance over the entire domain [201]. A standard assumption in RL is that both the rewards and transition kernels are not known to the agent. Thus to solve the RL problem, the agent has to estimate the underlying dynamics either explicitly (e.g. model based methods) or implicitly (e.g. value based methods).

2.1.1 Recurrence relations

The action value function Q^π of any policy π satisfies the recurrence relation called (scalar)-Bellman expectation equation [21]: $Q^\pi(s,a) = r(s,a) + \gamma \mathbb{E}_{s',a'}[Q^\pi(s',a')]$, which can equivalently be written in vectorized form as:

$$Q^\pi = R + \gamma P^\pi Q^\pi, \tag{2.3}$$

where R is the mean reward vector of size $|S \times U|$, P^π is the transition matrix $|S \times U| \times |S \times U|$ with $P^\pi((s,a),(s',a')) \triangleq \pi(a'|s')p(s'|s,a)$. The operation on RHS: $\mathcal{T}^\pi(\cdot) \triangleq R + \gamma P^\pi(\cdot)$ is the Bellman expectation operator for the policy π. In Chapter 4 we will study the novel tensorized form of this recurrence relation. We also have the Bellman optimality equation followed by any optimal policy π^*:

$$Q^{\pi^*}(s,a) = r(s,a) + \gamma \mathbb{E}_{s'}[\max_{a'} Q^{\pi^*}(s',a')] \tag{2.4}$$

whose Bellman optimality operator is given by $\mathcal{T}^*(\cdot) \triangleq r(s,a) + \gamma \mathbb{E}_{s'}[\max_{a'}(\cdot)]$. We can obtain similar recurrence relations for the value and advantage functions using Eq. (2.1), Eq. (2.2). It turns out that the above operators are contraction mappings and thus admit unique fixed point solutions due to Banach fixed point theorem. Hence, an interesting way to compute the action values Q^π for any given policy (known as the policy evaluation or prediction problem) and the optimal action values Q^* (known as the control problem) is to repeatedly apply these operators on arbitrary initial vector $q \in \mathbb{R}^{|S \times U|}$ until convergence.

2.2 Exploration-Exploitation trade-off

Reinforcement Learning deviates significantly from supervised learning methods because the data distribution on which the agents are trained for the control task is not stationary. This is because the observations (states, reward) of the agents are dependent on the agent policy in the first place. Further, there can be additional shifts in the agent's training data and internal state representation owing to design choices like use of function approximation for feature learning or due to events not in control of the agent like environment non-stationarity. Thus initially when the agent's policy is usually not performant or when the agent passes though non-stationarity, it must gather more information about the environment towards learning more reward optimizing behaviour. This leads to an interacting trade-off where the agent, while interacting with the environment, has to choose how much to explore by taking actions whose outcomes are uncertain versus how much to exploit by leveraging already found rewarding behaviour, given everything it has learned so far. Exploration involves taking information-seeking actions, that help the agent gather data from which it can learn about the environment and adjust to non-stationarity, while also learning the short and long-term consequences of the new actions. Thus exploratory actions can be potentially costly and sub-optimal but may pay off in the long term in comparison to exploitative actions. There has been extensive work in bandit theory and more recently in RL about principled ways to manage the Exploration Exploitation trade-off.

2.3 Contextual MDPs

An important class of MDP arises when we consider the presence of an underlying parametrized context θ, which governs the rewards and transitions in the MDP framework. We call this extension of the setting as the Contextual MDP setting (CMDP). Formally, we have $\mathcal{M} \triangleq \langle S, U, P_\theta, r_\theta, \gamma, \rho, \Theta, P_\Theta \rangle$, where Θ defines a

space of context parameters, P_Θ is a fixed distribution over the contexts. The important distinction here is that the transitions $P_\theta : S \times U \times S \times \Theta \to [0, 1]$, and the agent reward $r_\theta : S \times U \times \Theta \to [0, R_{max}]$ are now also function of the context parameter θ. Thus fixing a particular context θ gives us an instance of a regular MDP indexed by θ: $\mathcal{M}_\theta$. Fig. 2.2 illustrates the contextual MDP setting. It is important to note that this CMDP is equivalent to the regular MDP setting when we augment the state space S to also include the context space Θ (ie. new state space $\mathcal{S} \triangleq S \times \Theta$). Nevertheless, the explicit MDP treatment by fixing a given context is a very useful abstraction from an applications perspective and can be found in many real world settings. This view also helps design powerful algorithms which utilize the underlying contextual structure. We will extend the contextual MDP setting in Chapter 5 where we use contexts to define capabilities of agents in a multi agent setting towards studying combinatorial generalization. Similarly, in Chapter 7 we will build over the CMDP setting to include context based dependence of agent observations and study powerful methods which utilise MDP metrics for solving important practical problems.

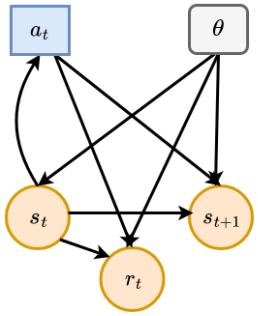


Figure 2.2: The contextual MDP setting

2.4 Partial Observability

Markovian transitions is often an unrealistic assumptions. This is because in most real-world scenarios, the complete relevant information about the system is hardly ever observable to the agent. For example, in Stratego, which is a game of incomplete information, the opponent setup and pieces are not known to the player. Thus, the player has to reason about the about the opponent state through the course

of the game progression, further they also have to account for aspects of battle psychology like concealment, bluffing and guessing. Similarly, in the deployment of an android robot in real world terrain, important information about physical state of the environment like coefficient of friction and ground plasticity are not directly observable to the agent, and these must be indirectly accounted for by the agent's policy. Partial observability can also arise in Deep Reinforcement Learning (DRL) when an agent encounters new situations and observation shifts that it still has to learn about from a feature extraction perspective. Finally, in cooperative multi-agent settings where constraints on communication prevent the agents from knowing the teammates state again requires maintaining beliefs and indirectly inferring the values of relevant variables towards reasonable execution.

Such situations can be modelled under the Partially Observable Markov Decision Process framework (POMDP) [106]. The POMDP can be formulated as $\langle S, U, P, r, \gamma, \rho, Z, O \rangle$. Here the MDP framework has been extended to allow for Z the observation set from which the agent observations come, and the observation function $O : S \to \mathcal{P}(Z)$ which gives the probability distribution over possible observations given a state. Agents get to see the obsevation in Z instead of the environment state. The optimal policy in POMDP conditions on either the agents observation trajectory τ or a sufficient statistic for the agent observation history. Note that the agent history itself follows the Markov property by definition. Belief based approaches for solving POMDPs, maintain a belief distribution over the state space given the observation history. They use posterior updates on the beliefs as the agent interacts with the environment [37]. Notice how this can potentially grow the joint belief-history distribution exponentially in the length of the problem horizon. Thus POMDP are computationally costly to solve. [189] show that the value function for the POMDPs are piece-wise linear in beliefs. In practice, partial observability under the DRL framework is accommodated using RNN based methods [89] for summarizing the agent trajectories. We use this approach for various algorithms

discussed in this thesis.

2.5 Multi-Agent Settings

In this thesis the multi-agent system (MAS) scenario which we will be primarily concerned with is the cooperative scenario. We next discuss the various formulation which we use along with their relation to the other formulations.

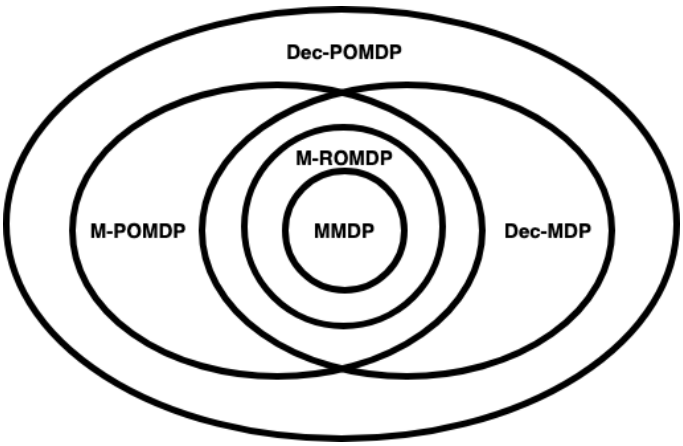


Figure 2.3: Different settings in MAS

Cooperative MARL settings: In the most general setting, a fully cooperative multi-agent task can be modelled as a decentralized partially observable MDP (Dec-POMDP) [22, 158]. A Dec-POMDP is formally defined as a tuple $\langle S, U, P, r, Z, O, n, \rho, \gamma \rangle$. Building over the POMDP framework, the most important addition is the presence of multiple agents (n in number) which add new algorithmic complexity to the problem. At each time step t, every agent $i \in \mathcal{A} \equiv \{1, ..., n\}$ observes its observation and chooses an action $u^i \in U$ which forms the joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$. The state transition function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$ now conditions on the joint action, and similarly the rewards $r(s, \mathbf{u}) : S \times \mathbf{U} \to [0, 1]$ which are shared by all agents. As before we have $\gamma \in [0, 1)$ being the discount factor. A Dec-POMDP is partially observable: each agent i does not have access to the full state and instead samples observations $z \in Z$ according to its own observation distribution $O(s, i) : S \times \mathcal{A} \to \mathcal{P}(Z)$. The action-observation history for an agent i is $\tau^i \in T \equiv (Z \times U)^*$. We use u^{-i} to denote the action of all the agents other than i

and similarly for the policies π^{-i}. Note that in general fo Dec-POMDPs the agents cannot exchange their action-observation histories with others and must condition their policy solely on local trajectories, $\pi^i(u^i|\tau^i) : T \times U \rightarrow [0, 1]$.

There are several interesting specializations of the Dec-POMDP which are of theoretical as well as practical interest. If the Dec-POMDP is such that the observations across the jointly identify a unique underlying state, the problem is called a Dec-MDP.

Similarly, when the observations are invertible for each agent, so that the observation space is partitioned w.r.t. S, i.e., $\forall i \in \mathcal{A}, \forall s_1, s_2 \in S, \forall z_i \in Z, P(z_i|s_1) > 0 \wedge s_1 \neq s_2 \implies P(z_i|s_2) = 0$, we classify the problem as a multi-agent richly observed MDP (M-ROMDP) [142] which extend ROMDPS[10] to the multi-agent setting. For M-ROMDP, we typically have $|Z| >> |S|$, thus for this work, we assume a setting with no information loss due to observation but instead, redundancy across different observation dimensions. Such is the case for many real world tasks like 2D robot navigation using observation data from different sensors.

When the observation distributions admit no special structure, but the observation distribution is independent of the agent index (ie. identical across agents), the problem is called an M-POMDP[142]. M-POMDPs can be thought of as POMDPs with factored action spaces, several algorithmic techniques applicable to M-POMDPs can be used to improve POMDP as well.

Finally, when the observation function is a unique bijective map $O : S \rightarrow Z$, we refer to the scenario as a multi-agent MDP (MMDP) [29], which can simply be denoted by the tuple : $\langle S, U, P, r, n, \gamma \rangle$

Fig. 2.3 gives the relation between different scenarios for the cooperative setting. For ease of exposition, we present our theoretical results for the MMDP case, though they can easily be extended to other cases by incurring additional sample

complexity.

Similar to the single agent setting, the value of a joint policy is defined as $J^{\pi} = \mathbb{E}_{\pi,\rho}\left[\sum_{t=0}^{\infty} \gamma^t r_{\tau}(s_t)\right]$. Similarly, the joint action-value function given a policy π is defined as: $Q^{\pi}(s_t, \mathbf{u}_t) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k})|s_t, \mathbf{u}_t\right]$. The goal is to find the optimal joint policy π^* corresponding to the optimal joint policy value J^*.

2.5.1 Centralised Training with Decentralised Execution

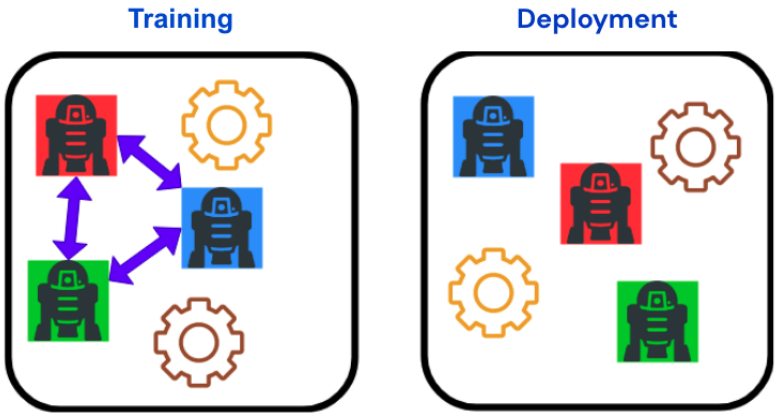


Figure 2.4: CTDE learning settting

The Dec-POMDP framework in general imposes no information exchange between the agents during execution. However, with the aim of potentially learning agent policies efficiently, we will consider a centralised training scenario, where the training algorithm may utilise extra information such as access to the underlying state or unbounded communication between the agents. As long as we enforce that the final agent policies do not rely on such privileged information, we can satisfy the no information exchange constraint required during execution for the Dec-POMDP. This training/execution setup is called Centralised Training with Decentralised Execution (CTDE) [137] and has become standard approach for policy search in Dec-POMDP. Fig. 2.4 illustrates the CTDE setup. We will be using the CTDE settings for the multi-agent reinforcement learning (MARL) problems discussed in this thesis. Note that CTDE is naturally applicable for many real world scenarios where we have access

to additional information about the environment during training phase: for example learning decentralized policy for a robot swarm in a controlled lab environment or simulator which will be subsequently used for deployment in real world.

2.6 Computational complexity of solving different settings

Our current understanding of computational complexity theory is that:

$$\text{Class P} \subseteq \text{Class NP} \subseteq \text{Class PSPACE} \subseteq \text{Class EXPTIME} \subseteq \text{Class NEXPTIME}$$

$$\text{Class P} \subset \text{Class EXPTIME}, \quad \text{Class NP} \subset \text{Class NEXPTIME}$$

where the bottom relations are strict inclusions. Finite horizon MDPs are P-complete under the dynamic programming framework. However, solving a POMDP becomes PSPACE-complete [164], this means that the worst instances of POMDP problems can potentially take exponential time, similarly solving a Dec-MDP for $n = 2$ agents is PSPACE-hard. Even more strikingly, solving a Dec-POMDP is NEXPTIME-complete [22] for $n \geq 2$ agents and similarly solving a Dec-MDP is NEXPTIME-complete [22] for $n \geq 3$ agents. NEXPTIME is the class of decision problems solvable by a nondeterministic Turing machine in exponential time. Since NEXPTIME is a strict superset of NP, it not possible to solve is the Dec-POMDP problems in polytime compute resources. This makes solving decentralized POMDPs computationally intractable. Similarly, the hierarchy of complexity also reflects in the sample requirements for robustly learning the underlying dynamics in the RL setting, with Dec-POMDPS being very sample inefficient. Thus creating sample efficient and computationally tractable solutions for the problem is a research intensive area.

2.7 Methods and algorithms in RL

We now discuss the main algorithmic approaches for deep reinforcement learning used in this thesis.

2.7.1 Model based vs Model free

Reinforcement Learning Methods can be broadly categorized into model based and model free algorithms. The model based approaches explicitly learn the underlying dynamics of the environment (rewards, transitions, emission probabilities) which is then subsequently used for planning and control. An important aspect here is to obtain robust statistical estimates for the environment models while being sample efficient. Using the model, other auxillary objects like policy and value functions are learnt in combination with the environment experience. A general principal is: the more complex the model the used the better it performs on real world tasks while being less susceptible to biases, however, this exposes the model to being less sample efficient and overly sensitive to environment noise (thus requiring careful regularization). Model free algorithms only implicitly model the environment are thus more readily deployable. They are directly concerned with computing the policy/value functions and thus less susceptible to errors in comparison to model based methods. Model free methods however offer less interpretability and offer a coarser way to reason about environment uncertainty in comparison to their model based counterparts.

2.7.2 Value based and Policy based methods

Yet another dimension of classifying the algorithms is based on the components they use for computing and representing the agent policy.

Value based methods typically use the action value function(Q^π) estimate to derive a behaviour policy which is iteratively improved using the policy evaluation and

policy improvement loop. (see policy improvement theorem in [201]). Q-Learning offers a sample efficient approach which allows reusing experience from previous policies and can be easily combined with various exploratory strategies. Derived from the Bellman optimally Eq. (2.4), Q-learning uses the residual loss minimization:

$$\mathcal{L}_{QL} = \mathbb{E}_{\pi_{exp}}[(r(s,\mathbf{u}) + \gamma \max_{\mathbf{u}'} Q^{\phi^-}(s',\mathbf{u}') - Q^{\phi}(s,\mathbf{u}))^2].$$

where π_{exp} denote exploration policy samples (or a experience buffer) and ϕ are the parameters used for function approximation (ϕ^- represent older parameters used for bootstrapping). Neural network based function approximation along with other stabilising techniques like replay buffer and target networks [153] have shown promising performance on problems with large state action spaces, where tabular methods would be computationally intractable. The most common exploration strategy for Q-Learning is annealed ϵ greedy where the greedy action corresponding to $\arg\max_{\mathbf{u}} Q^{\phi}(s,\mathbf{u})$ is picked with probability (w.p.) $1-\epsilon$ and a random action is picked w.p. ϵ. As we shall see in coming chapters (e.g. Chapter 3) better exploration methods combined with novel representation classes can prove much more effective, specially in large problems like MARL.

Policy gradient methods directly optimize an agents policy (typically parameterized by θ) by performing gradient ascent on the policy value objective J^{π}. The simplest form of policy gradient is REINFORCE [241], in which the gradient is given by: $\nabla \mathcal{J}_{\theta} = \mathbb{E}_{\pi}[G_{\tau} \nabla \pi_{\theta}(\mathbf{u}|\mathbf{s})]$, where $G_{\tau} \triangleq \sum_{t=0}^{\infty} \gamma^t r_t$ is the discounted return. This gradient is unbiased but tends to be very noisy as it uses the Monte-Carlo return. Actor-critic algorithms offer a promising approach in terms of reducing the gradient estimate variance. Here, an estimator for the action-value function $Q^{\phi} \approx Q^{\pi}$ given the policy π is used for weighing the score function ($\nabla \log(\pi_{\theta}(\mathbf{u}|\mathbf{s}))$) as we do not

have access to the true Q^π-function. The overall gradient thus becomes:

$$\nabla \mathcal{J}_\theta = \int_S \rho^\pi(s) \int_{\mathbf{U}} \nabla \pi_\theta(\mathbf{u}|\mathbf{s}) Q^\phi(s, \mathbf{u}) d\mathbf{u} ds$$

where $\rho^\pi(s) \triangleq (1-\gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\rho, \pi)$ is the discounted future state distribution (similar gradient formula also exists for the average reward formulation). The bias induced in approximating the gradient using an estimate of Q^π can be removed using compatible function approximation [206]. The parametrised approximation Q^ϕ is usually trained using the bootstrapped target objective derived using the samples from π by minimising the mean squared temporal difference(TD) error: $\mathbb{E}_\pi[(r(s, \mathbf{u}) + \gamma Q^\phi(s', \mathbf{u}') - Q^\phi(s, \mathbf{u}))^2]$. Methods such as n-step TD and TD(λ) can be used to enable faster critic learning and ameliorate the lagging critic problem, these however become insufficient for large state-action spaces (Chapter 4). Various techniques aimed towards reducing the gradient variance by using a baseline reduction have been found. Typically Q^ϕ is replaced by $Q^\phi(s, \mathbf{u}) - b(s)$ where $b(s)$ is the state dependent baseline. A common choice for $b(s) = V^\pi(s)$, which effectively uses the advantage for weighing the scores. Another option is to use the temporal difference $r(s, \mathbf{u}) + \gamma V^\pi(s') - V^\pi(s)$, which is an unbiased estimate of the advantage $A(s, \mathbf{u})$. Hence, the bias and variance of the policy gradient estimate depends strongly on the particular choice of estimator used for weighing the score. We will also be using deep neural networks for function approximation for empirical analysis on large domains throughout this work.

2.8 Variational Inference and EM

We will use tools from variation inference in Chapter 3 (for maximizing mutual information between joint multi-agent trajectories and a latent behaviour space) and Chapter 6 (for maximizing the RL as inference objective). Here we give a brief overview of these techniques. Fig. 2.5 shows the representation of

a generative graphical model that produces observations x from a distribution $x \sim p_\omega(x|h)$, has hidden variables h, and is parameterised by a set of parameters, ω. In learning a model, we often seek the parameters that maximises the log-marginal-likelihood (LML), which can be found by marginalising the joint distribution $p_\omega(x, h)$ over hidden variables, this is given by:

$$\ell_\omega(x) \coloneqq \log p_\omega(x) = \log \left(\int p_\omega(x, h) dh \right). \tag{2.5}$$

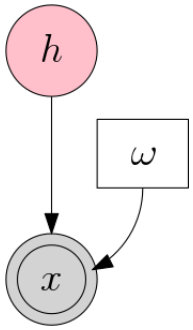


Figure 2.5: Graphical model of inference problem.

In many cases, we also need to infer the corresponding posterior,

$$p_\omega(h|x) = \frac{p_\omega(x, h)}{\int p_\omega(x, h) dh}.$$

Evaluating the marginal likelihood in Eq. (2.5) and obtain the corresponding posterior, however, is intractable for most distributions. To compute the marginal likelihood and ω^*, we can use the Expectation Maximization (EM) algorithm [50] and variational inference (VI). We review these two methods now.

For any valid probability distribution $q(h)$ with support over h we can rewrite the LML as a difference of two divergences [105],

$$\begin{aligned} \ell_\omega(x) =& \int q(h) \log \left(\frac{p_\omega(x, h)}{q(h)} \right) dh - \int q(h) \log \left(\frac{p_\omega(h|x)}{q(h)} \right) dh, \\ =& \mathcal{L}(\omega, q(h)) + \mathrm{KL}(q(h) \parallel p_\omega(h|x)), \end{aligned}$$

where $\mathcal{L}(\omega, q(h)) \coloneqq \int q(h) \log \left(\frac{p_\omega(x,h)}{q(h)} \right) dh$ is known as the evidence lower bound (ELBO). Intuitively, as $\mathrm{KL}(q(h) \parallel p_\omega(h|x)) \geq 0$, it follows that $\ell_\omega(x) \geq \mathrm{ELBO}\,(q(h); \omega)$, hence $\ell_\omega(x) \geq \mathrm{ELBO}\,(q(h); \omega)$ is a lower bound for the LML. The derivation of this bound can also be viewed as applying Jensen's inequality directly to Eq. (2.5) [28].

Note that when the ELBO and marginal likelihood are identical, the resulting KL divergence between the function $q(h)$ and the posterior $p(h|x)$ is zero, implying that $q(h) = p_\omega(h|x)$.

Maximising the LML now reduces to maximising the ELBO, which can be achieved iteratively using EM [50, 244]; an expectation step (E-step) finds the posterior for the current set of model parameters and then a maximisation step (M-step) maximises the ELBO with respect to ω while keeping $q(h)$ fixed as the posterior from the E-step.

As finding the exact posterior in the E-step is still typically intractable, we resort to variational inference (VI), a powerful tool for approximating the posterior using a parametrised variational distribution $q_\theta(h)$ [105, 20]. VI aims to reduce the KL divergence between the true posterior and the variational distribution, $\mathrm{KL}(q_\theta(h) \parallel p_\omega(h|x))$. Typically VI never brings this divergence to zero but nonetheless yields useful posterior approximations. As minimising $\mathrm{KL}(q_\theta(h) \parallel p_\omega(h|x))$ is equivalent to maximising the ELBO for the variational distribution (e.g. see Eq. (D.13) from Theorem D.3 for an RL as inference application), the variational E-step amounts to maximising the ELBO with respect to θ while keeping ω constant. The variational EM algorithm can be summarised as:

$$\text{Variational E-Step: } \theta_{k+1} \leftarrow \arg\max_\theta \mathcal{L}(\omega_k, \theta),$$

$$\text{Variational M-Step: } \omega_{k+1} \leftarrow \arg\max_\omega \mathcal{L}(\omega, \theta_{k+1}).$$

Part I

Multi Agent Systems

Chapter 3

Maven: Multi-agent variational exploration

Contents

3.1 Introduction

Cooperative *multi-agent reinforcement learning* (MARL) is a key tool for addressing many real-world problems such as coordination of robot swarms [99] and autonomous cars [36]. However, two key challenges stand between cooperative MARL and such real-world applications. First, scalability is limited by the fact that the size of the joint action space grows exponentially in the number of agents. Second, while the training process can typically be centralised, partial observability and communication constraints often mean that execution must be decentralised, i.e., each agent can condition its actions only on its local action-observation history, a setting known as *centralised training with decentralised execution* (CTDE).

While both policy-based [63] and value-based [172, 209, 197] methods have been developed for CTDE, value based thend to perform better than policy based methods, as measured on SMAC, a suite of StarCraft II micromanagement benchmark tasks [181]. We focus on the recent value based methods here. VDN [197] tries to address the challenges mentioned above by learning *factored* value functions. By decomposing the joint value function into factors that depend only on individual agents, VDN can cope with large joint action spaces. Furthermore, because such factors are combined in a way that respects a monotonicity constraint, each agent can select its action based only on its own factor, enabling decentralised execution. QMIX [172] similarly learns a more general monotonic factorization. However, this process of decentralisation comes with a price, as the monotonicity constraint restricts these algorithms to suboptimal value approximations as we shall see in this work.

QTRAN[188], another recent method, performs this trade-off differently by formulating multi-agent learning as an optimisation problem with linear constraints and relaxing it with $L2$ penalties for tractability.

In this work, we shed light on a problem unique to decentralised MARL that arises due to inefficient exploration. Inefficient exploration hurts decentralised

MARL, not only in the way it hurts single agent RL[152] (by increasing sample inefficiency[145, 143]), but also by interacting with the representational constraints necessary for decentralisation to push the algorithm towards suboptimal policies. Single agent RL can avoid convergence to suboptimal policies using various strategies like increasing the exploration rate (ϵ) or policy variance, ensuring optimality in the limit. However, we show, both theoretically and empirically, that the same is not possible in decentralised MARL.

Furthermore, we show that *committed* exploration can be used to solve the above problem. In committed exploration [162], exploratory actions are performed over extended time steps in a coordinated manner. Committed exploration is key even in single-agent exploration but is especially important in MARL, as many problems involve long-term coordination, requiring exploration to discover temporally extended joint strategies for maximising reward. Unfortunately, none of the existing methods for CTDE are equipped with *committed* exploration.

To address these limitations, we propose a novel approach called *multi-agent variational exploration* (MAVEN) that hybridises value and policy-based methods by introducing a latent space for hierarchical control. MAVEN's value-based agents condition their behaviour on the shared latent variable controlled by a hierarchical policy. Thus, fixing the latent variable, each joint action-value function can be thought of as a mode of joint exploratory behaviour that persists over an entire episode. Furthermore, MAVEN uses mutual information maximisation between the trajectories and latent variables to learn a diverse set of such behaviours. This allows MAVEN to achieve committed exploration while respecting the representational constraints. We demonstrate the efficacy of our approach by showing significant performance improvements on the challenging SMAC domain.

3.2 Background

We use the Dec-POMDP framework with CTDE learning setting. An important concept we would be using which pertains to the value based methods is *decentralisability* (see IGM in [188]) which asserts that local agent utilities q_i, satisfy $\forall s, \mathbf{u}$:

$$\arg\max_{\mathbf{u}} Q^*(s, \mathbf{u}) = \left(\arg\max_{u^1} q_1(\tau^1, u^1) \ldots \arg\max_{u^n} q_n(\tau^n, u^n) \right)', \tag{3.1}$$

Fig. 3.1 illustrates the value based CTDE learning process.

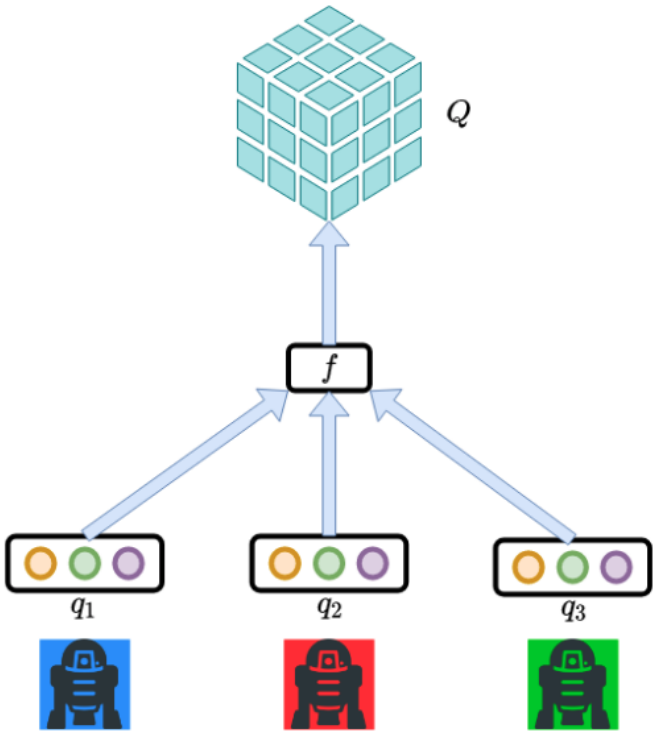


Figure 3.1: Value based CTDE learning. f combines local agent utilities for computing joint action values Q.

Monotonic decomposition: **QMIX** [172] is a value-based method that learns a monotonic approximation Q_{qmix} for the joint action-value function. Figure A.1 in Appendix A.2 illustrates its overall setup, reproduced for convenience. QMIX factors the joint-action Q_{qmix} into a monotonic nonlinear combination of individual

utilities q_i of each agent which are learnt via a *utility network*. A *mixer network* with nonnegative weights is responsible for combining the agent's utilities for their chosen actions u^i into $Q_{qmix}(s, \mathbf{u})$. This nonnegativity ensures that $\frac{\partial Q_{qmix}(s,\mathbf{u})}{\partial q_i(s,u^i)} \geq 0$, which in turn guarantees Eq. (3.1). During learning, the QMIX agents use ϵ-greedy exploration over their individual utilities to ensure sufficient exploration. For **VDN** [197] the factorization is further restrained to be just the sum of utilities: $Q_{vdn}(s, \mathbf{u}) = \sum_i q_i(s, u^i)$. Monotonic decomposition allows for an efficient, tractable maximisation as it can be performed in $\mathcal{O}(n|U|)$ time as opposed to $\mathcal{O}(|U|^n)$. Additionally, it allows for easy decentralisation as each agent can independently perform an argmax.

QTRAN [188] is another value-based method. Theorem 1 in the QTRAN paper guarantees optimal decentralisation by using linear constraints between agent utilities and joint action values, but it imposes $\mathcal{O}(|S||U|^n)$ constraints on the optimisation problem involved, where $|\cdot|$ gives set size. This is computationally intractable to solve in discrete state-action spaces and is impossible given continuous state-action spaces. The authors propose two algorithms (QTRAN-base and QTRAN-alt) which relax these constraints using two L2 penalties. While QTRAN tries avoid QMIX's limitations, we found that it performms poorly in practice on complex MARL domains (see Section 3.5) as it deviates from the exact solution due to these relaxations.

3.3 Analysis

In this section, we analyse the policy learnt by value based methods which use monotonic approximation in the case where they cannot represent the true optimal action-value function. We first start with QMIX and then discuss to similar algorithms like VDN [197]. Intuitively, monotonicity implies that the optimal action of agent i does not depend on the actions of the other agents. This motivates us to characterise the class of Q-functions that cannot be represented by QMIX, which

we call *nonmonotonic* Q functions.

Definition 3.1 (Nonmonotonicity)**.** *For any state $s \in S$ and agent $i \in \mathcal{A}$ given the actions of the other agents $u^{-i} \in U^{n-1}$, the Q-values $Q(s,(u^i,u^{-i}))$ form an ordering over the action space of agent i. Define $C(i,u^{-i}) := \{(u^i_1,...,u^i_{|U|}) | Q(s,(u^i_j,u^{-i})) \geq Q(s,(u^i_{j+1},u^{-i})), j \in \{1,\ldots,|U|\}, u^i_j \in U, j \neq j' \implies u^i_j \neq u^i_{j'}\}$, as the set of all possible such orderings over the action-values. The joint-action value function is* ***nonmonotonic*** *if $\exists i \in \mathcal{A}, u^{-i}_1 \neq u^{-i}_2$ s.t. $C(i,u^{-i}_1) \cap C(i,u^{-i}_2) = \varnothing$.*

A simple example of a nonmonotonic Q-function is given by the payoff matrix of the two-player three-action matrix game shown on Table 3.1(a). Table 3.1(b) shows the values learned by QMIX under *uniform visitation*, i.e., when all state-action pairs are explored equally.

	A	B	C
A	10.4	0	10
B	0	10	10
C	10	10	10

(a)

	A	B	C
A	6.08	6.08	8.95
B	6.00	5.99	8.87
C	8.99	8.99	11.87

(b)

	A	B	C
A	10.43	0.06	9.96
B	0.05	9.72	9.83
C	10.03	9.84	9.97

(c)

Table 3.1: (a) An example of a nonmonotonic payoff matrix, (b) QMIX values under uniform visitation. (c) MAVEN values under uniform exploration, $k_z = 4$

Of course, the fact that QMIX cannot represent the optimal value function does not imply that the policy it learns must be suboptimal. However, the following analysis establishes the suboptimality of such policies.

Theorem 3.1 (Uniform visitation)**.** *For n-player, $k \geq 3$-action matrix games ($|\mathcal{A}| = n, |U| = k$), under uniform visitation, Q_{qmix} learns a δ-suboptimal policy for any time horizon T, for any $0 < \delta \leq R\left[\sqrt{\frac{a(b+1)}{a+b}} - 1\right]$ for the payoff matrix (n-dimensional) given by the template below, where $b = \sum_{s=1}^{k-2} \binom{n+s-1}{s}$, $a = k^n - (b+1)$,*

$R > 0$*:*

$$\begin{bmatrix} R+\delta & 0 & \dots & R \\ 0 & & \iddots & \\ \vdots & \iddots & & \vdots \\ R & \dots & & R \end{bmatrix}$$

Proof. see Appendix A.1.1 □

We next consider ϵ-greedy visitation, in which each agent uses an ϵ-greedy policy and ϵ decreases over time. Below we provide a probabilistic bound on the maximum possible value of δ for QMIX to learn a suboptimal policy for any time horizon T.

Theorem 3.2 (ϵ-greedy visitation)**.** *For n-player, $k \geq 3$-action matrix games, under ϵ-greedy visitation $\epsilon(t)$, Q_{qmix} learns a δ-suboptimal policy for any time horizon T with probability $\geq 1 - \left(\exp(-\frac{T\upsilon^2}{2}) + (k^n - 1)\exp(-\frac{T\upsilon^2}{2(k^n-1)^2})\right)$, for any $0 < \delta \leq R\left[\sqrt{a\left(\frac{\upsilon b}{2(1-\upsilon/2)(a+b)} + 1\right)} - 1\right]$ for the payoff matrix given by the template above, where $b = \sum_{s=1}^{k-2} \binom{n+s-1}{s}$, $a = k^n - (b+1)$, $R > 0$ and $\upsilon = \epsilon(T)$.*

Proof. see Appendix A.1.2 □

We next cover similar suboptimality results for other value based methods.

Since the class of joint action values learnt by VDN is a subset of that of QMIX, it is intuitive that the suboptimality incurred by the policies learnt by it would be greater, this is in fact confirmed by the following theorem:

Theorem 3.3 (Uniform visitation VDN)**.** *For n player, $k \geq 3$ action matrix games ($|\mathcal{A}| = n, |U| = k$), under uniform visitation; Q_{vdn} learns a δ-suboptimal policy for any time horizon T, for any $0 < \delta \leq R\left[\binom{k+n-3}{n-1} - 1\right]$ for the payoff matrix (n dimensional) given by the template above, $R > 0$.*

Note that the above two upper bounds in Theorems 3.1 and A.3 for the uniform visitation case are tight further the latter bound is $\mathcal{O}(R\max\{k,n\}^{n-2})$ in comparison to the former which is of $\mathcal{O}(R\max\{k,n\}^{\frac{n}{2}})$. We similarly show results for the ϵ-greedy case and also for IQL[209], see Appendix A.1.3 for proofs and details.

The reliance of QMIX on ϵ-greedy action selection prevents it from engaging in committed exploration [162], in which a precise sequence of actions must be chosen in order to reach novel, interesting parts of the state space. Moreover, Theorems 3.1 and 3.2 imply that the agents can latch onto suboptimal behaviour early on, due to the monotonicity constraint. Theorem 3.2 in particular provides a surprising result: For a fixed time budget T, increasing QMIX's exploration rate lowers its probability of learning the optimal action due to its representational limitations. Intuitively this is because the monotonicity constraint can prevent the Q-network from correctly remembering the true value of the optimal action (currently perceived as suboptimal). We hypothesise that the lack of a principled exploration strategy coupled with these representational limitations can often lead to catastrophically poor exploration, which we confirm empirically.

3.4 Methodology

In this section, we propose *multi-agent variational exploration* (MAVEN), a new method that overcomes the detrimental effects of QMIX's monotonicity constraint on exploration. MAVEN does so by learning a diverse ensemble of monotonic approximations with the help of a latent space. Its architecture consists of value-based agents that condition their behaviour on the shared latent variable z controlled by a hierarchical policy that off-loads ϵ-greedy with committed exploration. Thus, fixing z, each joint action-value function is a monotonic approximation to the optimal action-value function that is learnt with Q-learning. Furthermore, each such approximation can be seen as a mode of committed joint exploratory behaviour.

The latent policy over z can then be seen as exploring the space of *joint behaviours* and can be trained using any policy learning method. Intuitively, the z space should map to diverse modes of behaviour.

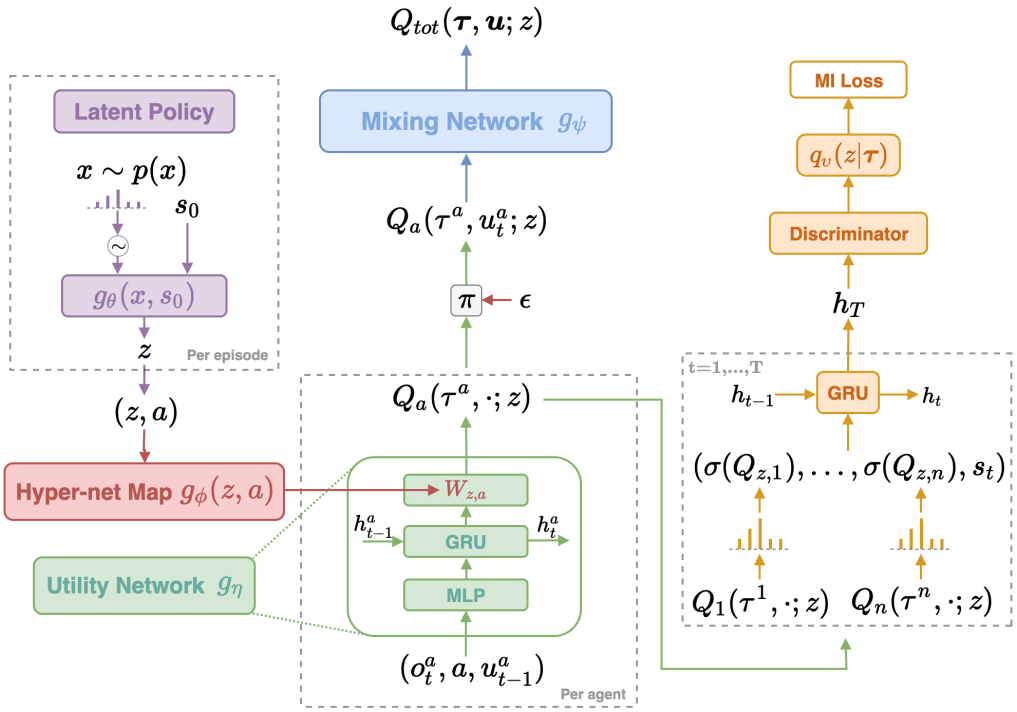


Figure 3.2: Architecture for MAVEN.

Fig. 3.2 illustrates the complete setup for MAVEN. We first focus on the lefthand side of the diagram, which describes the learning framework for the latent space policy and the joint action values. We parametrise the hierarchical policy by θ, the agent utility network with η, the hypernet map from latent variable z used to condition utilities by ϕ, and the mixer net with ψ. η can be associated with a feature extraction module per agent and ϕ can be associated with the task of modifying the utilities for a particular mode of exploration. We model the hierarchical policy $\pi_z(\cdot|s_0; \theta)$ as a transformation of a simple random variable $x \sim p(x)$ through a neural network parameterised by θ; thus $z \sim g_\theta(x, s_0)$, where s_0 is initial state. Natural choices for $p(x)$ are uniform for discrete z and uniform or normal for continuous z.

We next provide a coordinate ascent scheme for optimising the parameters. Fixing z gives a joint action-value function $Q(\mathbf{u}, s; z, \phi, \eta, \psi)$ which implicitly defines a greedy deterministic policy $\pi_{\mathcal{A}}(\mathbf{u}|s; z, \phi, \eta, \psi)$ (we drop the parameter dependence wherever its inferable for clarity of presentation). This gives the corresponding Q-learning loss:

$$\mathcal{L}_{QL}(\phi, \eta, \psi) = \mathbb{E}_{\pi_{\mathcal{A}}}[(Q(\mathbf{u}_t, s_t; z) - [r(\mathbf{u}_t, s_t) + \gamma \max_{\mathbf{u}_{t+1}} Q(\mathbf{u}_{t+1}, s_{t+1}; z)])^2],$$

where t is the time step. Next, fixing ϕ, η, ψ, the hierarchical policy over $\pi_z(\cdot|s_0; \theta)$ is trained on the cumulative trajectory reward $\mathcal{R}(\boldsymbol{\tau}, z|\phi, \eta, \psi) = \sum_t r_t$ where $\boldsymbol{\tau}$ is the joint trajectory.

Algorithm 1 MAVEN

Initialize parameter vectors $\upsilon, \phi, \eta, \psi, \theta$
Learning rate $\leftarrow \alpha$, $\mathcal{D} \leftarrow \{\}$
for each episodic iteration **do**
  $s_0 \sim \rho(s_0)$, $x \sim p(x)$, $z \sim g_\theta(x; s_0)$
  for each environment step t **do**
    $\mathbf{u}_t \sim \pi_{\mathcal{A}}(u|s_t;; z, \phi, \eta, \psi)$
    $s_{t+1} \sim p(s_{t+1}|s_t, \mathbf{u}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, \mathbf{u}_t, r(s_t, \mathbf{u}_t), r^z_{aux}(\mathbf{u}_t, s_t), s_{t+1})\}$
  end for
  for each gradient step **do**
    $\phi \leftarrow \phi + \alpha \hat{\nabla}_\phi(\lambda_{MI}\mathcal{J}_V - \lambda_{QL}\mathcal{L}_{QL})$ (Hypernet update)
    $\eta \leftarrow \eta + \alpha \hat{\nabla}_\eta(\lambda_{MI}\mathcal{J}_V - \lambda_{QL}\mathcal{L}_{QL})$ (Feature update)
    $\psi \leftarrow \psi + \alpha \hat{\nabla}_\psi(\lambda_{MI}\mathcal{J}_V - \lambda_{QL}\mathcal{L}_{QL})$ (Mixer update)
    $\upsilon \leftarrow \upsilon + \alpha \hat{\nabla}_\upsilon \lambda_{MI}\mathcal{J}_V$ (Variational update)
    $\theta \leftarrow \theta + \alpha \hat{\nabla}_\theta \mathcal{J}_{RL}$ (Latent space update)
  end for
end for

Thus, the hierarchical policy objective for z, freezing the parameters ψ, η, ϕ is given by:

$$\mathcal{J}_{RL}(\theta) = \int \mathcal{R}(\tau_{\mathcal{A}}|z) p_\theta(z|s_0) \rho(s_0) dz ds_0.$$

However, the formulation so far does not encourage diverse behaviour corresponding

to different values of z and all the values of z could collapse to the same joint behaviour. To prevent this, we introduce a *mutual information* (MI) objective between the observed trajectories $\boldsymbol{\tau} \triangleq \{(\mathbf{u}_t, s_t)\}$, which are representative of the joint behaviour and the latent variable z. The actions $\mathbf{u}_t$ in the trajectory are represented as a stack of agent utilities and σ is an operator that returns a per-agent Boltzmann policy w.r.t. the utilities at each time step t, ensuring the MI objective is differentiable and helping train the network parameters (ψ, η, ϕ). We use an RNN [97] to encode the entire trajectory and then maximise $MI(\sigma(\boldsymbol{\tau}), z)$. Intuitively, the MI objective encourages visitation of diverse trajectories $\boldsymbol{\tau}$ while at the same time making them identifiable given z, thus elegantly separating the z space into different exploration modes. The MI objective is:

$$\mathcal{J}_{MI} = \mathcal{H}(\sigma(\boldsymbol{\tau})) - \mathcal{H}(\sigma(\boldsymbol{\tau})|z) = \mathcal{H}(z) - \mathcal{H}(z|\sigma(\boldsymbol{\tau})),$$

where $\mathcal{H}$ is the entropy. However, neither the entropy of $\sigma(\boldsymbol{\tau})$ nor the conditional of z given the former is tractable for nontrivial mappings, which makes directly using MI infeasible. Therefore, we introduce a variational distribution $q_\upsilon(z|\sigma(\boldsymbol{\tau}))$ [235, 26] parameterised by υ as a proxy for the posterior over z, which provides a lower bound on $\mathcal{J}_{MI}$ (see Appendix A.1.4).

$$\mathcal{J}_{MI} \geq \mathcal{H}(z) + \mathbb{E}_{\sigma(\boldsymbol{\tau}),z}[\log(q_\upsilon(z|\sigma(\boldsymbol{\tau})))].$$

We refer to the righthand side of the above inequality as the variational MI objective $\mathcal{J}_V(\upsilon, \phi, \eta, \psi)$. The lower bound matches the exact MI when the variational distribution equals $p(z|\sigma(\boldsymbol{\tau}))$, the true posterior of z. The righthand side of Fig. 3.2 gives the network architectures corresponding to the variational MI loss. Since

$$\mathbb{E}_{\boldsymbol{\tau},z}[\log(q_\upsilon(z|\sigma(\cdot)))] = \mathbb{E}_{\boldsymbol{\tau}}[-KL(p(z|\sigma(\cdot))||q_\upsilon(z|\sigma(\cdot))] - \mathcal{H}(z|\sigma(\cdot)),$$

where the nonnegativity of the KL divergence on the righthand side implies that a bad variational approximation can hurt performance as it induces a gap between the true objective and the lower bound [150, 12]. This problem is especially important if z is chosen to be continuous as for discrete distributions the posterior can be represented exactly as long as the dimensionality of υ is greater than the number of categories k_z for the random variable z. The problem can be addressed by various state-of-the-art developments in amortised variational inference [178, 177]. The variational approximation can also be seen as a discriminator/critic that induces an auxiliary reward field $r^z_{aux}(\boldsymbol{\tau}) = \log(q_\upsilon(z|\sigma(\boldsymbol{\tau}))) - \log(p(z))$ on the trajectory space. Thus the overall objective becomes:

$$\max_{\upsilon,\phi,\eta,\psi,\theta} \mathcal{J}_{RL}(\theta) + \lambda_{MI}\mathcal{J}_V(\upsilon,\phi,\eta,\psi) - \lambda_{QL}\mathcal{L}_{QL}(\phi,\eta,\psi),$$

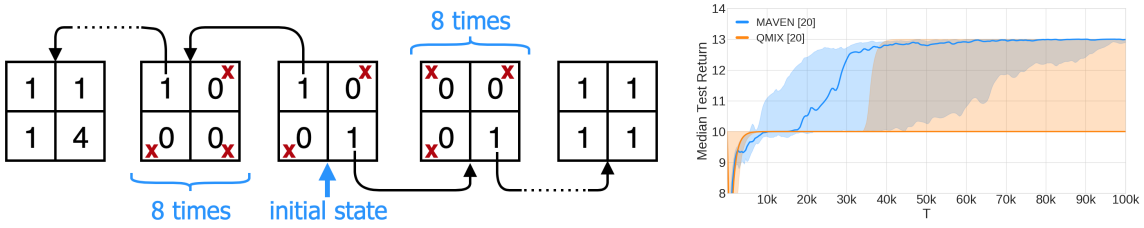
where $\lambda_{MI}, \lambda_{QL}$ are positive multipliers. For training (see Algorithm 1), at the beginning of each episode we sample an x and obtain z and then unroll the policy until termination and train $\psi, \eta, \phi, \upsilon$ on the Q-learning loss corresponding to greedy policy for the current exploration mode and the variational MI reward. The hierarchical policy parameters θ can be trained on the true task return using any policy optimisation algorithm. At test time, we sample z at the start of an episode and then perform a decentralised argmax on the corresponding Q-function to select actions. Thus, MAVEN achieves committed exploration while respecting QMIX's representational constraints.

3.5 Experimental Results

We now empirically evaluate MAVEN on various new and existing domains.

3.5.1 m-step matrix games

To test the how nonmonotonicity and exploration interact, we introduce a simple m-step matrix game. The initial state is nonmonotonic, zero rewards lead to termination, and the differentiating states are located at the terminal ends; there are $m-2$ intermediate states. Fig. 3.3(a) illustrates the m-step matrix game for $m = 10$. The optimal policy is to take the top left joint action and finally take the bottom right action, giving an optimal total payoff of $m+3$. As m increases, it becomes increasingly difficult to discover the optimal policy using ϵ-dithering and a *committed* approach becomes necessary. Additionally, the initial state's nonmonotonicity provides inertia against switching the policy to the other direction. Fig. 3.3(b) plots median returns for $m = 10$. QMIX gets stuck in a suboptimal policy with payoff 10, while MAVEN successfully learns the true optimal policy with payoff 13. This example shows how representational constraints can hurt performance if they are left unmoderated.



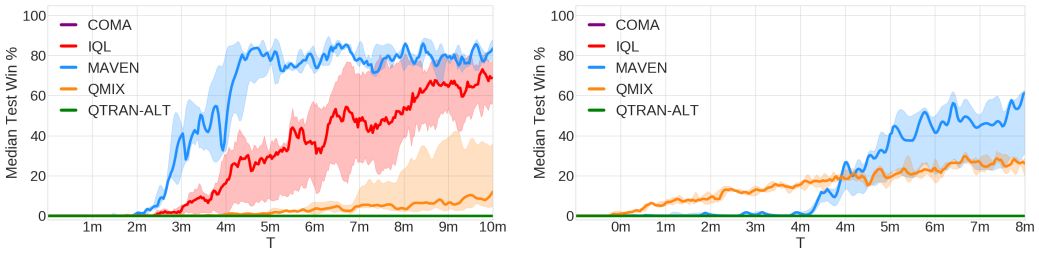
(a) (b)

Figure 3.3: (a) m-step matrix game for $m = 10$ case (b) median return of MAVEN and QMIX method on 10-step matrix game for 100k training steps, averaged over 20 random initializations (2nd and 3rd quartile is shaded).

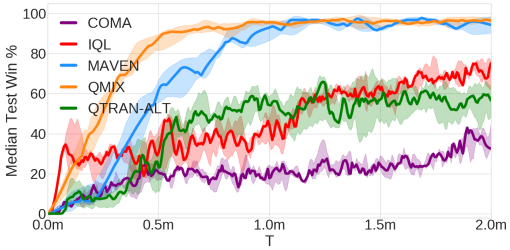
3.5.2 StarCraft II

StarCraft Multi-Agent Challenge We consider a challenging set of cooperative StarCraft II maps from the SMAC benchmark [181] which Samvelyan et al. have classified as **Easy, Hard** and **Super Hard**. Our evaluation procedure is similar to [172, 181]. We pause training every 100000 time steps and run 32 evaluation episodes with decentralised greedy action selection. After training, we report the median *test*

win rate (percentage of episodes won) along with 2nd and 3rd quartiles (shaded in plots). We use grid search to tune hyperparameters. Appendix A.3.1 contains additional experimental details. We compare MAVEN, QTRAN, QMIX, COMA [63]



(a) `corridor` **Super Hard** (b) `6h_vs_8z` **Super Hard**



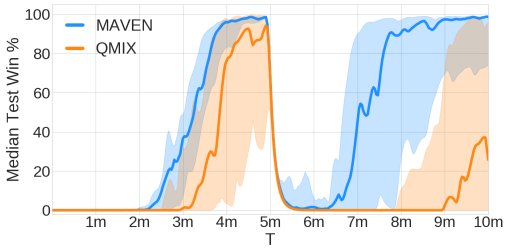
(c) `2s3z` **Easy**

Figure 3.4: The performance of various algorithms on three SMAC maps.

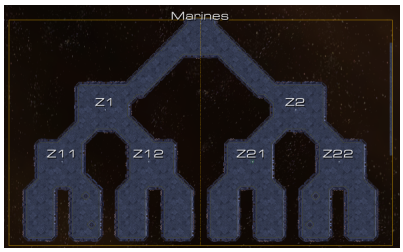
and IQL [209] on several SMAC maps. Here we present the results for two **Super Hard** maps `corridor` & `6h_vs_8z` and an **Easy** map `2s3z`. The `corridor` map, in which 6 Zealots face 24 enemy Zerglings, requires agents to make effective use of the terrain features and block enemy attacks from different directions. A properly *coordinated* exploration scheme applied to this map would help the agents discover a suitable unit positioning quickly and improve performance. `6h_vs_8z` requires fine grained 'focus fire' by the allied Hydralisks. `2s3z` requires agents to learn "focus fire" and interception. Figs. 3.4(a) to 3.4(c) show the median win rates for the different algorithms on the maps; additional plots can be found in Appendix A.3.2. The plots show that MAVEN performs substantially better than all alternate approaches on the **Super Hard** maps with performance similar to QMIX on **Hard** and **Easy** maps.Thus MAVEN performs better as difficulty increases. Furthermore, QTRAN does not yield satisfactory performance on most SMAC maps (0% win rate). The

map on which it performs best is 2s3z (Fig. 3.4(c)), an **Easy** map, where it is still worse than QMIX and MAVEN. We believe this is because QTRAN enforces decentralisation using only relaxed L2 penalties that are insufficient for challenging domains.

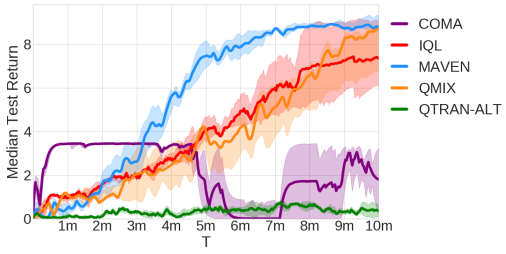
(a) `2_corridors`



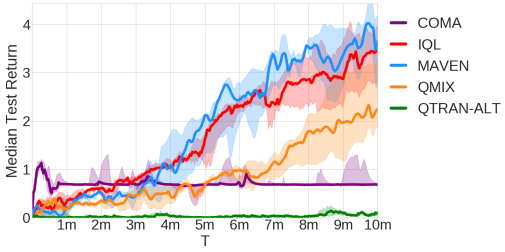
(b) `Shorter corridor closed at 5mil steps`



(c) `zealot_cave`



(d) `zealot_cave depth 3`



(e) `zealot_cave depth 4`

Figure 3.5: State exploration and policy robustness

Exploration and Robustness Although SMAC domains are challenging, they are not specially designed to test state-action space exploration, as the units involved start engaging immediately after spawning. We thus introduce a new SMAC map designed specifically to assess the effectiveness of multi-agent exploration techniques and their ability to adapt to changes in the environment. The `2-corridors` map features two Marines facing an enemy Zealot. In the beginning of training, the agents

can make use of two corridors to attack the enemy (see Fig. 3.5(a)). Halfway through training, the short corridor is blocked. This requires the agents to adapt accordingly and use the long corridor in a coordinated way to attack the enemy. Fig. 3.5(b) presents the win rate for MAVEN and QMIX for `2-corridors` when the gate to short corridor is closed after 5 million steps. While QMIX fails to recover after the closure, MAVEN swiftly adapts to the change in the environment and starts using the long corridor. MAVEN's latent space allows it to explore in a *committed* manner and associate use of the long corridor with a value of z. Furthermore, it facilitates recall of the behaviour once the short corridor becomes unavailable, which QMIX struggles with due to its representational constraints. We also introduce another new map called `zealot_cave` to test state exploration, featuring a tree-structured cave with a Zealot at all but the leaf nodes (see Fig. 3.5(c)). The agents consist of 2 marines who need to learn 'kiting' to reach all the way to the leaf nodes and get extra reward only if they always take the right branch except at the final intersection. The depth of the cave offers control over the task difficulty. Figs. 3.5(d) and 3.5(e) give the average reward received by the different algorithms for cave depths of 3 and 4. MAVEN outperforms all algorithms compared.

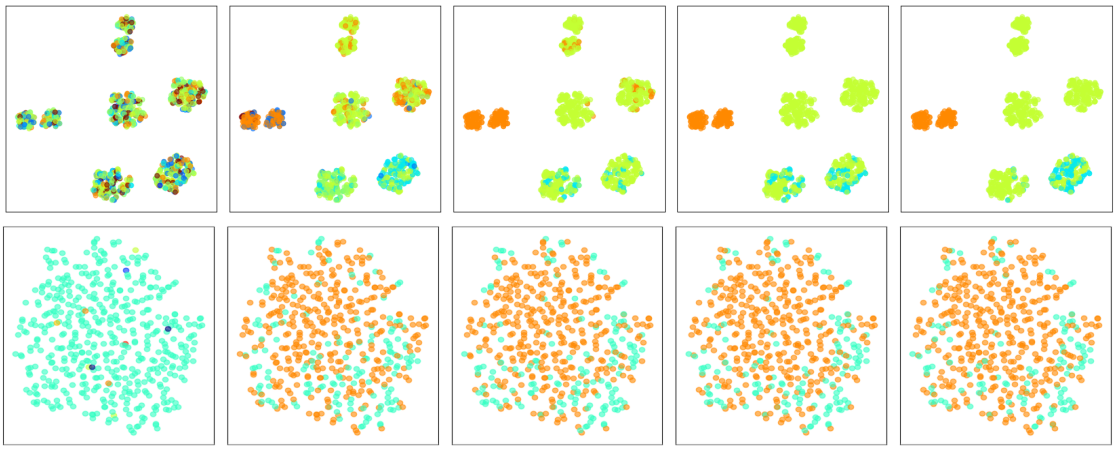


Figure 3.6: tsne plot for s_0 labelled with z (16 categories), initial (left) to final (right), top `3s5z`, bottom `micro_corridor`

Representability The optimal action-value function lies outside of the representation class of the CTDE algorithm used for most interesting problems. One way to tackle this issue is to find local approximations to the optimal value function and choose the best local approximation given the observation. We hypothesise that MAVEN enables application of this principle by mapping the latent space z to local approximations and using the hierarchical policy to choose the best such approximation given the initial state s_0, thus offering better representational capacity while respecting the constraints requiring decentralization. To demonstrate this, we plot the t-SNE [139] of the initial states and colour them according to the latent variable sampled for it using the hierarchical policy at different time steps during training. The top row of Fig. 3.6 gives the time evolution of the plots for `3s5z` which shows that MAVEN learns to associate the initial state clusters with the same latent value, thus partitioning the state-action space with distinct *joint behaviours*. Another interesting plot in the bottom row for `micro_corridor` demonstrates how MAVEN's latent space allows transition to more rewarding joint behaviour which existing methods would struggle to accomplish.

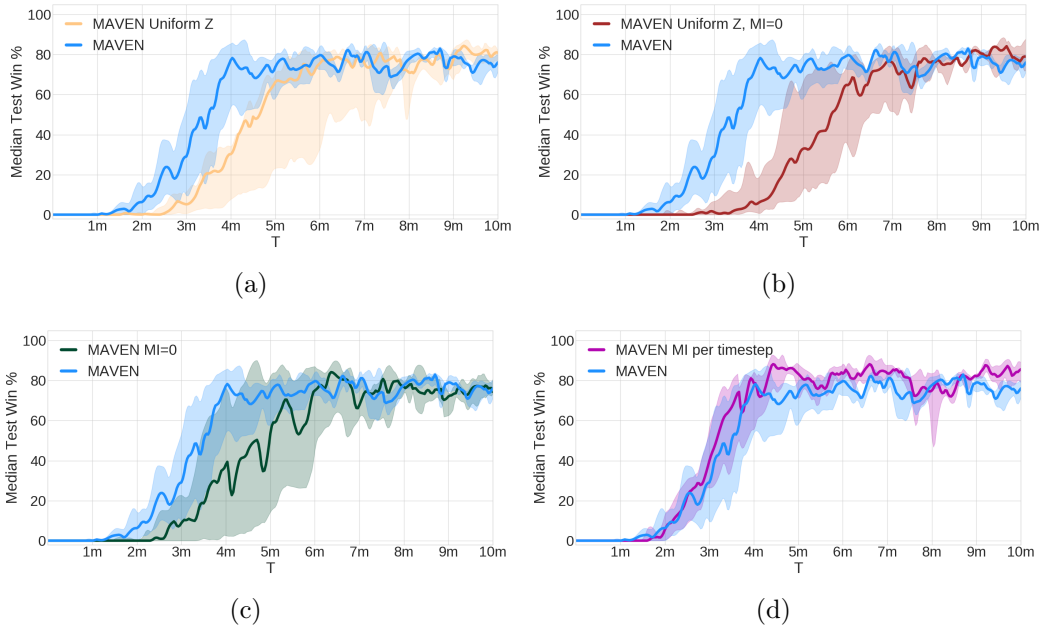


Figure 3.7: (a) & (b) investigate uniform hierarchical policy. (c) & (d) investigate effects of MI loss.

Ablations We perform several ablations on the `micro_corridor` scenario with $k_z = 16$ to try and determine the importance of each component of MAVEN. We first consider using a fixed uniform hierarchical policy over z. Fig. 3.7(a) shows that MAVEN with a uniform policy over z performs worse than a learned policy. Interestingly, using a uniform hierarchical policy and no variational MI loss to encourage diversity results in a further drop in performance, as shown in Fig. 3.7(b). Thus sufficient diversification of the observed trajectories via an explicit agency is important to find good policies ensuring sample efficiency. Fig. 3.7(b) is similar to Bootstrapped-DQN [162], which has no incentive to produce diverse behaviour other than the differing initialisations depending on z. Thus, all the latent variable values can collapse to the same *joint behaviour*. If we are able to learn a hierarchical policy over z, we can focus our computation and environmental samples on the more promising variables, which allows for better final performance. Fig. 3.7(c) shows improved performance relative to Fig. 3.7(b) providing some evidence for this claim. Next, we consider how the different choices of variational MI loss (*per time step, per trajectory*) affect performance in Fig. 3.7(d). Intuitively, the per time step loss promotes a more spread out exploration as it forces the discriminator to learn the inverse map to the latent variable at each step. It thus tends to distribute its exploration budget at each step uniformly, whereas the trajectory loss allows the *joint behaviours* to be similar for extended durations and take diversifying actions at only a few time steps in a trajectory, keeping its spread fairly narrow. However, we found that in most scenarios, the two losses perform similarly. See Appendix A.3.2 for additional plots and ablation results.

3.6 Related Work

Guckelsberge et al. [76] maximise the *empowerment* between one agents actions and the others future state in a competitive setting. Zheng et al. [255] allow each agent to condition their policies on a shared continuous latent variable. In contrast to

our setting, they consider the fully-observable centralised control setting and do not attempt to enforce diversity across the shared latent variable. Aumann [8] proposes the concept of a *correlated* equilibrium in non-cooperative multi-agent settings in which each agent conditions its policy on some shared variable that is sampled every episode.

In the single agent setting, Osband et al.[162] learn an ensemble of Q-value functions (which all share weights except for the final few layers) that are trained on their own sampled trajectories to approximate a posterior over Q-values via the statistical bootstrapping method. MAVEN without the MI loss and a uniform policy over z is then equivalent to each agent using a Bootstrapped DQN. [163] extends the Bootstrapped DQN to include a prior. [51] consider the setting of *concurrent* RL in which multiple agents interact with their own environments in parallel. They aim to achieve more efficient exploration of the state-action space by seeding each agent's parametric distributions over MDPs with different seeds, whereas MAVEN aims to achieve this by maximising the mutual information between z and a trajectory.

Yet another direction of related work lies in defining intrinsic rewards for single agent hierarchical RL that enable learning of diverse behaviours for the low level layers of the hierarchical policy. Florensa et al. [62] use hand designed state features and train the lower layers of the policy by maximising MI, and then tune the policy network's upper layers for specific tasks. Similarly [73, 55] learn a mixture of diverse behaviours using deep neural networks to extract state features and use MI maximisation between them and the behaviours to learn useful skills without a reward function. MAVEN differs from DIAYN [55] in the use case, and also enforces *action diversification* due to MI being maximised jointly with states and actions in a trajectory. Hence, agents jointly learn to solve the task is many different ways; this is how MAVEN prevents suboptimality from representational constraints, whereas DIAYN is concerned only with discovering new states. Furthermore, DIAYN

trains on diversity rewards using RL whereas we train on them via gradient ascent. Haarnoja et al. [83] use normalising flows [177] to learn hierarchical latent space policies using max entropy RL [217, 259, 57], which is related to MI maximisation but ignores the variational posterior over latent space behaviours. In a similar vein [98, 165] use auxiliary rewards to modify the RL objective towards a better tradeoff between exploration and exploitation.

3.7 Conclusion and Future work

In this work, we analysed the effects of representational constraints on exploration under CTDE. We also introduced MAVEN, an algorithm that enables committed exploration while obeying such constraints. As immediate future work, we aim to develop a theoretical analysis similar to QMIX for other CTDE algorithms. We would also like to carry out empirical evaluations for MAVEN when z is continuous. To address the intractability introduced by the use of continuous latent variables, we propose the use of state-of-the-art methods from variational inference [112, 178, 177, 113]. Yet another interesting direction would be to condition the latent distribution on the joint state space at each time step and transmit it across the agents to get a low communication cost, centralised execution policy and compare its merits to existing methods [195, 103].

Chapter 4

Tesseract: Tensorised Actors for Multi-Agent Reinforcement Learning

Contents

4.1 Introduction

As we saw in the previous Chapter 3, MARL introduces several new challenges that do not arise in single-agent reinforcement learning (RL), including exponential growth of the action space in the number of agents. This affects multiple aspects of learning, such as credit assignment [63], gradient variance [138] and exploration [140]. In addition, we also noted how practical constraints on observability and communication during deployment imply that decision making must be decentralised, leading to study of new settings like CTDE.

Recent work in CTDE-MARL can be broadly classified into value-based methods and actor-critic methods. We extensively studied value-based methods [198, 172, 188, 236, 247] in Chapter 3, which typically enforce decentralisability by modelling the joint action Q-value such that the argmax over the joint action space can be tractably computed by local maximisation of per-agent utilities. However, as we saw in Chapter 3, constraining the representation of the Q-function can interfere with exploration, yielding provably suboptimal solutions [140]. Actor-critic methods [138, 63, 239] typically use a centralised critic to estimate the gradient for a set of decentralised policies. In principle, actor-critic methods can satisfy CTDE without incurring suboptimality, this would be our main motivation in the present chapter towards creating actor-critic MARL algorithms. However, in practice the performance of actor-critic methods is limited by the accuracy of the critic, which is hard to learn given exponentially growing action spaces. This can exacerbate the problem of the *lagging* critic [117]. Moreover, unlike the single-agent setting, this problem cannot be fixed by increasing the critic's learning rate and number of training iterations. Similar to these approaches, an exponential blowup in the action space also makes it difficult to choose the appropriate class of models which strike the correct balance between expressibility and learnability for the given task.

In this work, we present new theoretical results that show how the aforementioned

approaches can be improved such that they accurately represent the joint action-value function whilst keeping the complexity of the underlying hypothesis class low. This translates to accurate, sample efficient modelling of long-term agent interactions.

In particular, we propose TESSERACT (derived from "Tensorised Actors"), a new framework that leverages tensors for MARL. Tensors are high dimensional analogues of matrices that offer rich insights into representing and transforming data. The main idea of TESSERACT is to view the output of a joint Q-function as a tensor whose modes correspond to the actions of the different agents. We thus formulate the Tensorised Bellman equation, which offers a novel perspective on the underlying structure of a multi-agent problem. In addition, it enables the derivation of algorithms that decompose the Q-tensor across agents and utilise low rank approximations to model relevant agent interactions.

Many real-world tasks (e.g., robot navigation) involve high dimensional observations but can be completely described by a low dimensional feature vector (e.g., a 2D map suffices for navigation). For value-based TESSERACT methods, maintaining a tensor approximation with rank matching the intrinsic task dimensionality* helps learn a compact approximation of the true Q-function (alternatively MDP-dynamics for model based methods). In this way, we can avoid the suboptimality of the learnt policy while remaining sample efficient. Similarly, for actor-critic methods, TESSERACT reduces the critic's learning complexity while retaining its accuracy, thereby mitigating the lagging critic problem. Thus, TESSERACT offers a natural spectrum for trading off accuracy with computational/sample complexity.

To gain insight into how tensor decomposition helps improve sample efficiency for MARL, we provide theoretical results for model-based TESSERACT algorithms and show that the underlying joint transition and reward functions can be efficiently

*We define intrinsic task dimensionality (ITD) as the minimum number of dimensions required to describe an environment

recovered under a PAC framework (in samples polynomial in accuracy and confidence parameters). We also introduce a tensor-based framework for CTDE-MARL that opens new possibilities for developing efficient classes of algorithms. Finally, we explore the relevance of our framework to rich observation MDPs.

Our main contributions are:

1. A novel tensorised form of the Bellman equation;
2. TESSERACT, a method to factorise the action-value function based on tensor decomposition, which can be used for any factored action space;
3. PAC analysis and error bounds for model based TESSERACT that show an exponential gain in sample efficiency of $O(|U|^{n/2})$; and
4. Empirical results illustrating the advantage of TESSERACT over other methods and detailed techniques for making tensor decomposition work for deep MARL.

4.2 Background

We use the different Multi-Agents settings defined in Chapter 2 . We next cover the specific background required for this work:

Tensor Decomposition Tensors are high dimensional analogues of matrices and tensor methods generalize matrix algebraic operations to higher orders. Tensor decomposition, in particular, generalizes the concept of low-rank matrix factorization. In the rest of this work, we use $\hat{\cdot}$ to represent tensors. Formally, an order n tensor $\hat{T}$ has n index sets $I_j, \forall j \in \{1..n\}$ and has elements $T(e), \forall e \in \times_{\mathcal{I}} I_j$ taking values in a given set $\mathcal{S}$, where $\times$ is the set cross product and we denote the set of index sets by $\mathcal{I}$. Each dimension $\{1..n\}$ is also called a mode. An elegant way of representing tensors and associated operations is via tensor diagrams as shown in Fig. 4.1. Tensor contraction generalizes the concept of matrix with matrix multiplication. For any

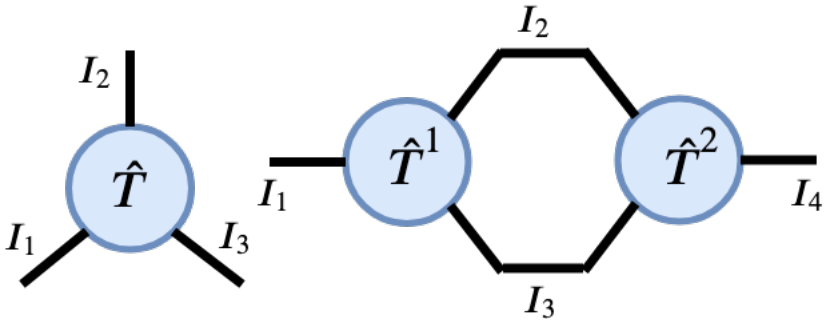


Figure 4.1: Left: Tensor diagram for an order 3 tensor $\hat{T}$. Right: Contraction between $\hat{T}^1$,$\hat{T}^2$ on common index sets I_2, I_3.

two tensors $\hat{T}^1$ and $\hat{T}^2$ with $\mathcal{I}_\cap = \mathcal{I}^1 \cap \mathcal{I}^2$ we define the contraction operation as $\hat{T} = \hat{T}^1 \odot \hat{T}^2$ with $\hat{T}(e_1, e_2) = \sum_{e \in \times_{\mathcal{I}_\cap} I_j} \hat{T}^1(e_1, e) \cdot \hat{T}^2(e_2, e), e_i \in \times_{\mathcal{I}^i \setminus \mathcal{I}_\cap} I_j$. The contraction operation is associative and can be extended to an arbitrary number of tensors. Fig. 4.2 illustrates the contraction operator. Using this building block,

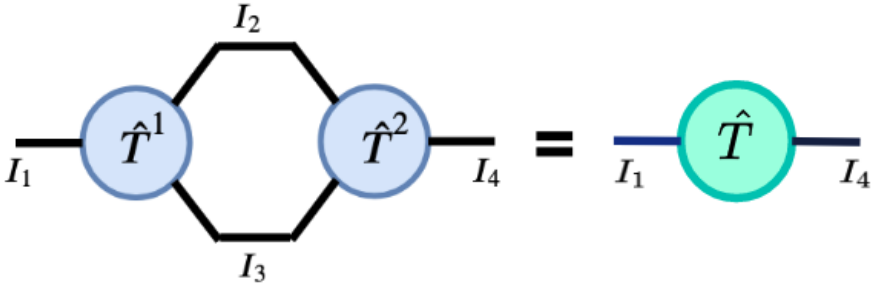


Figure 4.2: Tensor contraction result

we can define tensor decompositions, which factorizes a (low-rank) tensor in a compact form. This can be done with various decompositions [114], such as Tucker, Tensor-Train (also known as Matrix-Product-State), or CP (for Canonical-Polyadic). In this work, we focus on the latter, which we briefly introduce here. Just as a matrix can be factored as a sum of rank-1 matrices (each being an outer product of vectors), a tensor can be factored as a sum of rank-1 tensors, the latter being an outer product of vectors. The number of vectors in the outer product is equal to the rank of the tensor, and the number of terms in the sum is called the *rank of the decomposition* (sometimes also called CP-rank). Formally, a tensor $\hat{T}$ can be factored using a (rank–k) CP decomposition into a sum of k vector outer products

(denoted by $\otimes$), as,

$$\hat{T} = \sum_{r=1}^{k} w_r \otimes^n u_r^i, i \in \{1..n\}, ||u_r^i||_2 = 1. \tag{4.1}$$

4.3 Methodology

4.3.1 Tensorised Bellman equation

In this section, we provide the basic framework for Tesseract. We focus here on the discrete action space. The extension for continuous actions is similar and is deferred to Appendix B.2.2 for clarity of exposition.

Proposition 4.1. *Any real-valued function f of n arguments $(x_1..x_n)$ each taking values in a finite set $x_i \in \mathcal{D}_i$ can be represented as a tensor $\hat{f}$ with modes corresponding to the domain sets $\mathcal{D}_i$ and entries $\hat{f}(x_1..x_n) = f(x_1..x_n)$.*

Given a multi-agent problem $G = \langle S, U, P, r, Z, O, n, \gamma \rangle$, let $\mathcal{Q} \triangleq \{Q : S \times U^n \to \mathbb{R}\}$ be the set of real-valued functions on the state-action space. We are interested in the *curried* [13] form $Q : S \to U^n \to \mathbb{R}, Q \in \mathcal{Q}$ so that $Q(s)$ is an order n tensor (We use functions and tensors interchangeably where it is clear from context). Algorithms in Tesseract operate directly on the curried form and preserve the structure implicit in the output tensor. (Currying in the context of tensors implies fixing the value of some index. Thus, Tesseract-based methods keep action indices free and fix only state-dependent indices.)

We are now ready to present the tensorised form of the Bellman equation shown in Eq. (2.3). Fig. 4.3 gives the equation where $\hat{I}$ is the identity tensor of size $|S| \times |S| \times |S|$. The dependence of the action-value tensor $\hat{Q}^\pi$ and the policy tensor $\hat{U}^\pi$ on the policy is denoted by superscripts π. The novel **Tensorised Bellman equation** provides a theoretically justified foundation for the approximation of the

joint Q-function, and the subsequent analysis (Theorems 1-3) for learning using this approximation.

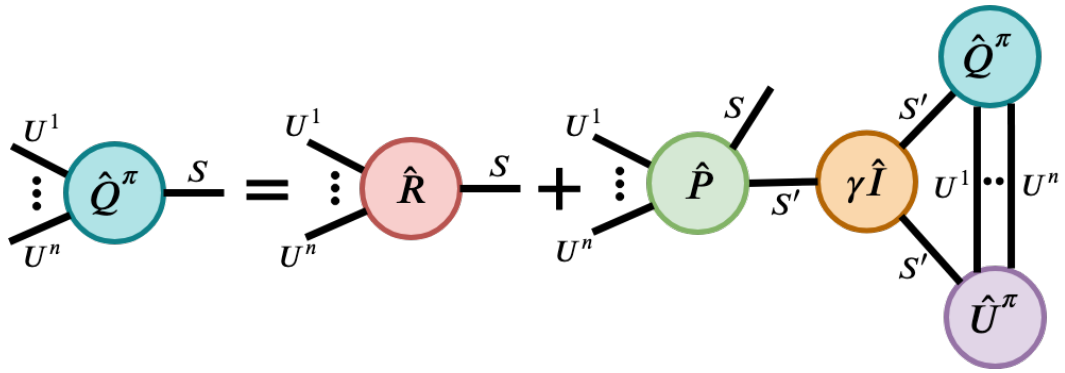


Figure 4.3: **Tensorised Bellman Equation** for n agents. There is an edge for each agent $i \in \mathcal{A}$ in the corresponding nodes $\hat{Q}^\pi, \hat{U}^\pi, \hat{R}, \hat{P}$ with the index set U^i.

4.3.2 TESSERACT Algorithms

For any $k \in \mathbb{N}$ let $\mathcal{Q}_k \triangleq \{Q : Q \in \mathcal{Q} \wedge rank(Q(\cdot, s)) \leq k, \forall s \in S\}$. Given any policy π we are interested in projecting Q^π to $\mathcal{Q}_k$ using the projection operator $\Pi_k(\cdot) = \arg\min_{Q \in \mathcal{Q}_k} ||\cdot - Q||_{\pi,F}$. where $||X||_{\pi,F} \triangleq \mathbb{E}_{s \sim \rho^\pi(s)}[||X(s)||_F]$ is the weighted Frobenius norm w.r.t. policy visitation over states. Thus a simple planning based algorithm for rank k TESSERACT would involve starting with an arbitrary Q_0 and successively applying the Bellman operator $\mathcal{T}^\pi$ and the projection operator Π_k so that $Q_{t+1} = \Pi_k \mathcal{T}^\pi Q_t$.

As we show in Theorem 4.1, constraining the underlying tensors for dynamics and rewards $(\hat{P}, \hat{R})$ is sufficient to bound the CP-rank of $\hat{Q}$. From this insight, we propose a model-based RL version for TESSERACT in Algorithm 2. The algorithm proceeds by estimating the underlying MDP dynamics using the sampled trajectories obtained by executing the behaviour policy $\pi = (\pi^i)_1^n$ (factorisable across agents) satisfying Theorem 4.2. Specifically, we use a rank k approximate CP-Decomposition to calculate the model dynamics R, P as we show in Section 4.4. Next π is evaluated using the estimated dynamics, which is followed by policy improvement, Algorithm 2 gives the pseudocode for the model-based setting. The termination and policy

improvement decisions in Algorithm 2 admit a wide range of choices used in practice in the RL community. Example choices for internal iterations which broadly fall under approximate policy iteration include: 1) Fixing the number of applications of Bellman operator 2) Using norm of difference between consecutive Q estimates etc., similarly for policy improvement several options can be used like ϵ-greedy (for Q derived policy), policy gradients (parametrized policy) [201]

Algorithm 2 Model-based Tesseract

Initialise rank k, $\pi = (\pi^i)_1^n$ and $\hat{Q}$: Theorem 4.2
Initialise model parameters $\hat{P}, \hat{R}$
Learning rate $\leftarrow \alpha$,$\mathcal{D} \leftarrow \{\}$
for each episodic iteration i **do**
  Do episode rollout $\tau_i = \left\{(s_t, \mathbf{u}_t, r_t, s_{t+1})_0^L\right\}$ using π
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_i\}$
  Update $\hat{P}, \hat{R}$ using CP-Decomposition on moments from $\mathcal{D}$ (Theorem 4.2)
  for each internal iteration j **do**
    $\hat{Q} \leftarrow \mathcal{T}^\pi \hat{Q}$
  end for
  Improve π using $\hat{Q}$
end for
Return $\pi, \hat{Q}$

For large state spaces where storage and planning using model parameters is computationally difficult (they are $\mathcal{O}(kn|U||S|^2)$ in number), we propose a model-free approach using a deep network where the rank constraint on the Q-function is directly embedded into the network architecture. Fig. 4.4 gives the general network architecture for this approach and Algorithm 3 the associated pseudo-code. Each agent in Fig. 4.4 has a policy network parameterized by θ which is used to take actions in a decentralised manner. The observations of the individual agents along with the actions are fed through representation function g_ϕ whose output is a set of k unit vectors of dimensionality $|U|$ corresponding to each rank. The output $g_{\phi,r}(s^i)$ corresponding to each agent i for factor r can be seen as an action-wise contribution to the joint utility from the agent corresponding to that factor. The joint utility here is a product over individual agent utilities. For partially observable settings,

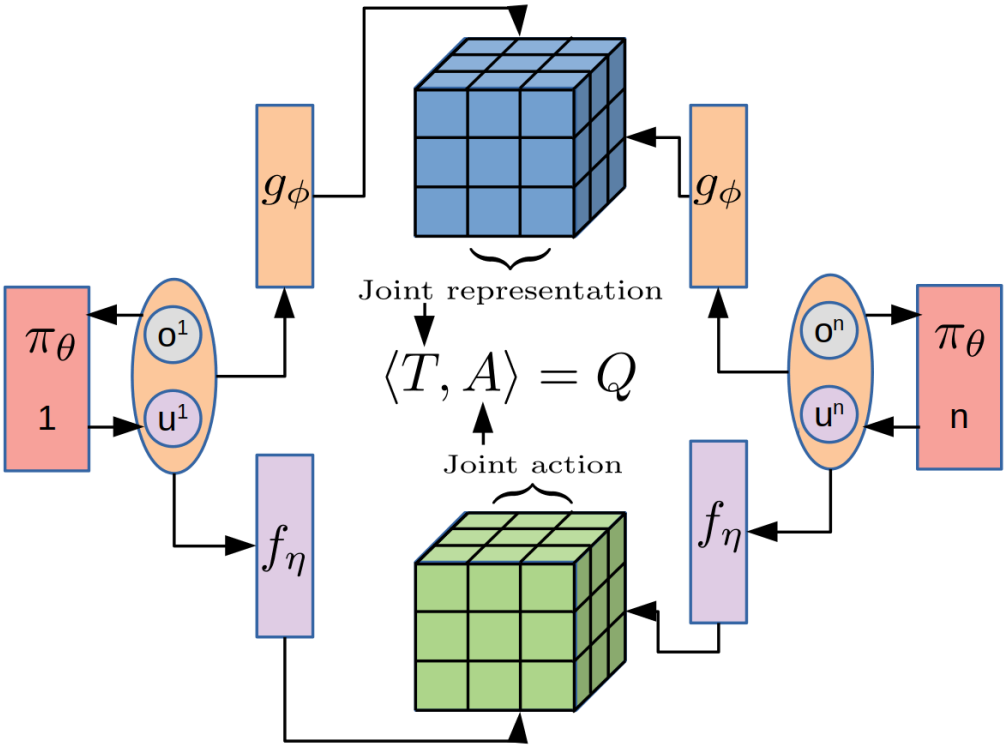


Figure 4.4: Tesseract architecture

an additional RNN layer can be used to summarise agent trajectories. The joint action-value estimate of the tensor $\hat{Q}(s)$ by the centralized critic is:

$$\hat{Q}(s) \approx T = \sum_{r=1}^{k} w_r \otimes^n g_{\phi,r}(s^i), i \in \{1..n\}, \tag{4.2}$$

where the weights w_r are learnable parameters exclusive to the centralized learner. In the case of value based methods where the policy is implicitly derived from utilities, the policy parameters θ are merged with ϕ. The network architecture is agnostic to the type of the action space (discrete/continuous) and the action-value corresponding to a particular joint-action $(u^1..u^n)$ is the inner product $\langle T, A \rangle$ where $A = \otimes^n u^i$ (This reduces to indexing using joint action in Eq. (4.2) for discrete spaces). More representational capacity can be added to the network by creating an abstract representation for actions using f_η, which can be any arbitrary monotonic function (parametrised by η) of vector output of size $m \geq |U|$ and preserves relative

order of utilities across actions; this ensures that the optimal policy is learnt as long as it belongs to the hypothesis space. In this case $A = \otimes^n f_\eta(u^i)$ and the agents also carry a copy of f_η during the execution phase. Furthermore, the inner product $\langle T, A \rangle$ can be computed efficiently using the property

$$\langle T, A \rangle = \sum_{r=1}^{k} w_r \prod_{1}^{n} \langle f_\eta(u^i) g_{\phi,r}(s^i) \rangle, i \in \{1..n\}$$

which is $O(nkm)$ whereas a naive approach involving computation of the tensors first would be $O(km^n)$. Training the Tesseract-based Q-network involves minimising the squared TD loss [201]:

$$\mathcal{L}_{TD}(\phi, \eta) = \mathbb{E}_\pi[(Q(\mathbf{u}_t, s_t; \phi, \eta) \\ -[r(\mathbf{u}_t, s_t) + \gamma Q(\mathbf{u}_{t+1}, s_{t+1}; \phi^-, \eta^-)])^2],$$

where ϕ^-, η^- are target parameters. Policy updates involve gradient ascent w.r.t. to the policy parameters θ on the objective $\mathcal{J}_\theta = \int_S \rho^\pi(s) \int_{\mathbf{U}} \pi_\theta(\mathbf{u}|\mathbf{s}) Q^\pi(s, \mathbf{u}) d\mathbf{u} ds$. More sophisticated targets can be used to reduce the policy gradient variance [72, 253] and propagate rewards efficiently [200]. Note that Algorithm 3 does not require the individual-global maximisation principle [188] typically assumed by value-based MARL methods in the CTDE setting, as it is an actor-critic method. In general, any form of function approximation and compatible model-free approach can be interleaved with Tesseract by appropriate use of the projection function Π_k.

4.3.3 Why Tesseract?

As discussed in Section 4.1, $Q(s)$ is an object of prime interest in MARL. Value based methods [198, 172, 247] that directly approximate the optimal action values Q^* place constraints on $Q(s)$ such that it is a monotonic combination of agent utilities. In terms of Tesseract this directly translates to finding the best projection

Algorithm 3 Model-free Tesseract

Initialise rank k, parameter vectors θ, ϕ, η
Learning rate $\leftarrow \alpha$,$\mathcal{D} \leftarrow \{\}$
for each episodic iteration i **do**
  Do episode rollout $\tau_i = \left\{(s_t, \mathbf{u}_t, r_t, s_{t+1})_0^L\right\}$ using π_θ
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{\tau_i\}$
  Sample batch $\mathcal{B} \subseteq \mathcal{D}$.
  Compute empirical estimates for $\mathcal{L}_{TD}, \mathcal{J}_\theta$
  $\phi \leftarrow \phi - \alpha \nabla_\phi \mathcal{L}_{TD}$ (Rank k projection step)
  $\eta \leftarrow \eta - \alpha \nabla_\eta \mathcal{L}_{TD}$ (Action representation update)
  $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{J}_\theta$ (Policy update)
end for
Return $\pi, \hat{Q}$

constraining $Q(s)$ to be rank one (Appendix B.2.1 for details).

Similarly, the following result demonstrates containment of action-value functions representable by FQL[39] which uses a learnt inner product to model pairwise agent interactions (**proof and additional results in Appendix B.2.1**):.

Proposition 4.2. *The set of joint Q-functions representable by FQL is a subset of that representable by* Tesseract.

MAVEN [140] illustrates how rank 1 projections can lead to insufficient exploration and provides a method to avoid suboptimality by using mutual information (MI) to learn a diverse set of rank 1 projections that correspond to different joint behaviours. In Tesseract, this can simply be achieved by finding the best approximation constraining $Q(s)$ to be rank k. Moreover, the CP-decomposition problem, being a product form (Eq. (4.1)), is well posed, whereas in [140] the problem form is $\hat{T} = \sum_{r=1}^{k} w_r \oplus^n u_r^i, i \in \{1..n\}, ||u_r^i||_2 = 1$, which requires careful balancing of different factors $\{1..k\}$ using MI as otherwise all factors collapse to the same estimate. The above improvements are equally important for the critic in actor-critic frameworks. Note that Tesseract is complete in the sense that every possible joint Q-function is representable by it given sufficient approximation rank. This follows as every possible Q-tensor can be expressed as linear combination of one-hot tensors (which

form a basis for the set).

Many real world problems have high-dimensional observation spaces that are encapsulated in an underlying low dimensional latent space that governs the transition and reward dynamics [10]. For example, in the case of robot navigation, the observation is high dimensional visual and sensory input but solving the underlying problem requires only knowing the 2D position. Standard RL algorithms that do not address modelling the latent structure in such problems typically incur poor performance and intractability. In Section 4.4 we show how Tesseract can be leveraged for such scenarios. Finally, projection to a low rank offers a natural way of regularising the approximate Q-functions and makes them easier to learn, which is important for making value function approximation amenable to multi-agent settings. Specifically for the case of actor-critic methods, this provides a natural way to make the critic learn more quickly. Additional discussion about using Tesseract for continuous action spaces can be found in Appendix B.2.2.

4.4 Analysis

In this section we provide a PAC analysis of model-based Tesseract (Algorithm 2). We focus on the MMDP setting (Section 2.5) for the simplicity of notation and exposition; guidelines for other settings are provided in Appendix B.1.

The objective of the analysis is twofold: Firstly it provides concrete quantification of the sample efficiency gained by model-based policy evaluation. Secondly, it provides insights into how Tesseract can similarly reduce sample complexity for model-free methods. **Proofs for the results stated can be found in Appendix B.1**. We begin with the assumptions used for the analysis:

Assumption 4.1. *For the given MMDP* $G = \langle S, U, P, r, n, \gamma \rangle$*, the reward tensor* $\hat{R}(s), \forall s \in S$ *has bounded rank* $k_1 \in \mathbb{N}$*.*

Intuitively, a small k_1 in Assumption 4.1 implies that the reward is dependent only on a small number of intrinsic factors characterising the actions.

Assumption 4.2. *For the given MMDP $G = \langle S, U, P, r, n, \gamma \rangle$, the transition tensor $\hat{P}(s, s'), \forall s, s' \in S$ has bounded rank $k_2 \in \mathbb{N}$.*

Intuitively a small k_2 in Assumption 4.2 implies that only a small number of intrinsic factors characterising the actions lead to meaningful change in the joint state. Assumption 4.1-2 always hold for a finite MMDP as CP-rank is upper bounded by $\Pi_{j=1}^{n}|U_j|$, where U_j are the action sets.

Assumption 4.3. *The underlying MMDP is ergodic for any policy π so that there is a stationary distribution ρ^{π}.*

Next, we define coherence parameters, which are quantities of interest for our theoretical results: for reward decomposition $\hat{R}(s) = \sum_r w_{r,s} \otimes^n v_{r,i,s}$, let $\mu_s = \sqrt{n} \max_{i,r,j} |v_{r,i,s}(j)|$, $w_s^{\max} = \max_{i,r} w_{r,s}$,$w_s^{\min} = \min_{i,r} w_{r,s}$. Similarly define the corresponding quantities for $\mu_{s,s'}, w_{s,s'}^{\max}, w_{s,s'}^{\min}$ for transition tensors $\hat{P}(s, s')$. A low coherence implies that the tensor's mass is evenly spread and helps bound the possibility of never seeing an entry with very high mass (large absolute value of an entry).

Theorem 4.1. *For a finite MMDP the action-value tensor satisfies rank$(\hat{Q}^{\pi}(s)) \leq k_1 + k_2|S|, \forall s \in S, \forall \pi$.*

Proof. We first unroll the Tensor Bellman equation in Fig. 4.3. The first term $\hat{R}$ has bounded rank k_1 by Assumption 4.1. Next, each contraction term on the RHS is a linear combination of $\{\hat{P}(s, s')\}_{s' \in S}$ each of which has bounded rank k_2 (Assumption 4.2). The result follows from the sub-additivity of CP-rank. □

Theorem 4.1 implies that for approximations with enough factors, policy evaluation converges:

Corollary 4.1.1. *For all $k \geq k_1 + k_2|S|$, the procedure $Q_{t+1} \leftarrow \Pi_k \mathcal{T}^\pi Q_t$ converges to Q^π for all Q_0, π.*

Corollary 4.1.1 is especially useful for the case of M-POMDP and M-ROMDP with $|Z| >> |S|$, i.e., where the intrinsic state space dimensionality is small in comparison to the dimensionality of the observations (like robot navigation Section 4.3.3). In these cases the Tensorised Bellman equation Fig. 4.3 can be augmented by padding the transition tensor $\hat{P}$ with the observation matrix and the lower bound in Corollary 4.1.1 can be improved using the intrinsic state dimensionality.

We next give a PAC result on the number of samples required to infer the reward and state transition dynamics for finite MDPs with high probability using sufficient approximate rank $k \geq k_1, k_2$:

Theorem 4.2 (Model based estimation of $\hat{R}, \hat{P}$ error bounds)**.** *Given any $\epsilon > 0, 1 > \delta > 0$, for a policy π with the policy tensor satisfying $\pi(\mathbf{u}|s) \geq \Delta$, where*

$$\Delta = \max_s \frac{C_1 \mu_s^6 k^5 (w_s^{max})^4 \log(|U|)^4 \log(3k||R(s)||_F/\epsilon)}{|U|^{n/2}(w_s^{min})^4}$$

and C_1 is a problem dependent positive constant. There exists N_0 which is $O(|U|^{\frac{n}{2}})$ and polynomial in $\frac{1}{\delta}, \frac{1}{\epsilon}, k$ and relevant spectral properties of the underlying MDP dynamics such that for samples $\geq N_0$, we can compute the estimates $\bar{R}(s), \bar{P}(s, s')$ such that w.p. $\geq 1 - \delta$, $||\bar{R}(s) - \hat{R}(s)||_F \leq \epsilon, ||\bar{P}(s, s') - \hat{P}(s, s')||_F \leq \epsilon, \forall s, s' \in S$.

Theorem 4.2 gives the relation between the order of the number of samples required to estimate dynamics and the tolerance for approximation. Theorem 4.2 states that aside from allowing efficient PAC learning of the reward and transition dynamics of the multi-agent MDP, Algorithm 2 requires only $O(|U|^{\frac{n}{2}})$ to do so, which is a vanishing fraction of $|U|^n$, the total number of joint actions in any given state. This also hints at why a tensor based approximation of the Q-function helps with sample efficiency. Methods that do not use the tensor structure typically use $O(|U|^n)$

samples. The bound is also useful for off-policy scenarios, where only the behaviour policy needs to satisfy the bound. Given the result in Theorem 4.2, it is natural to ask what is the error associated with computing the action-values of a policy using the estimated transition and reward dynamics. We address this in our next result, but first we present a lemma bounding the total variation distance between the estimated and true transition distributions:

Lemma 4.1. *For transition tensor estimates satisfying* $||\bar{P}(s,s') - \hat{P}(s,s')||_F \leq \epsilon$, *we have for any given state-action pair* (s,a), *the distribution over the next states follows:* $TV(P'(\cdot|s,a), P(\cdot|s,a)) \leq \frac{1}{2}(|1-f|+f|S|\epsilon)$ *where* $\frac{1}{1+\epsilon|S|} \leq f \leq \frac{1}{1-\epsilon|S|}$, *where* TV *is the total variation distance. Similarly for any policy* π, $TV(\bar{P}_\pi(\cdot|s), P_\pi(\cdot|s)) \leq \frac{1}{2}(|1-f|+f|S|\epsilon$ *and* $TV(\bar{P}_\pi(s',a'|s), P_\pi(s',a'|s)) \leq \frac{1}{2}(|1-f|+f|S|\epsilon)$

We now bound the error of model-based evaluation using approximate dynamics in Theorem 4.3. The first component on the RHS of the upper bound comes from the tensor analysis of the transition dynamics, whereas the second component can be attributed to error propagation for the rewards.

Theorem 4.3 (Error bound on policy evaluation)**.** *Given a behaviour policy* π_b *satisfying the conditions in Theorem 4.2 and executed for steps* $\geq N_0$, *for any policy* π *the model based policy evaluation* $Q^\pi_{\bar{P},\bar{R}}$ *satisfies:*

$$|Q^\pi_{P,R}(s,a) - Q^\pi_{\bar{P},\bar{R}}(s,a)| \leq (|1-f|+f|S|\epsilon)\frac{\gamma}{2(1-\gamma)^2} + \frac{\epsilon}{1-\gamma}, \forall (s,a) \in S \times U^n$$

where f *is as defined in Lemma 4.1.*

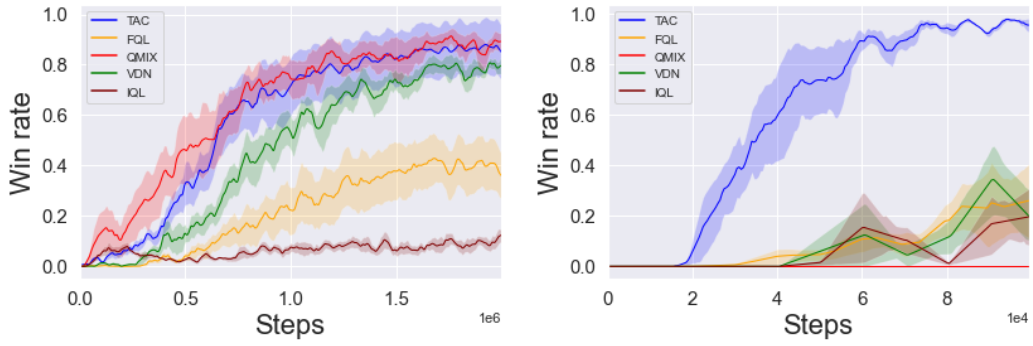
4.4.1 Selecting the CP-rank for approximation

While determining the rank of a fully observed tensor is itself NP-hard [95], we believe we can help alleviate this problem due to two key observations:

- The tensors involved in TESSERACT capture dependence of transition and reward dynamics on the action space. Thus if we can approximately identify (possibly using expert knowledge) the various aspects in which the actions available at hand affect the environment, we can get a rough idea of the rank to use for approximation.
- Our experiments on different domains (Section 4.5, Appendix B.3) provide evidence that even when using a rank insufficient approximation, we can get good empirical performance and sample efficiency. (This is also evidenced by the empirical success of related algorithms like VDN which happen to be specific instances under the TESSERACT framework.)

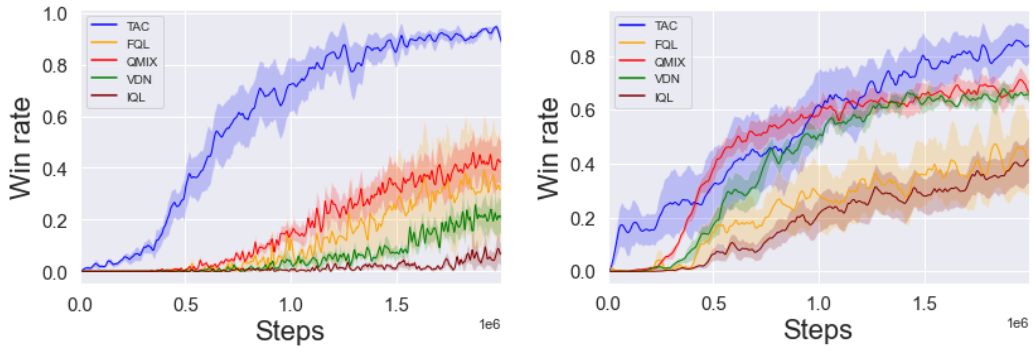
4.5 Experiments

In this section we present the empirical results on the StarCraft domain. Experiments for a more didactic domain of Tensor games can be found in Appendix B.3.3. We use the model-free version of TESSERACT (Algorithm 3) for all the experiments.



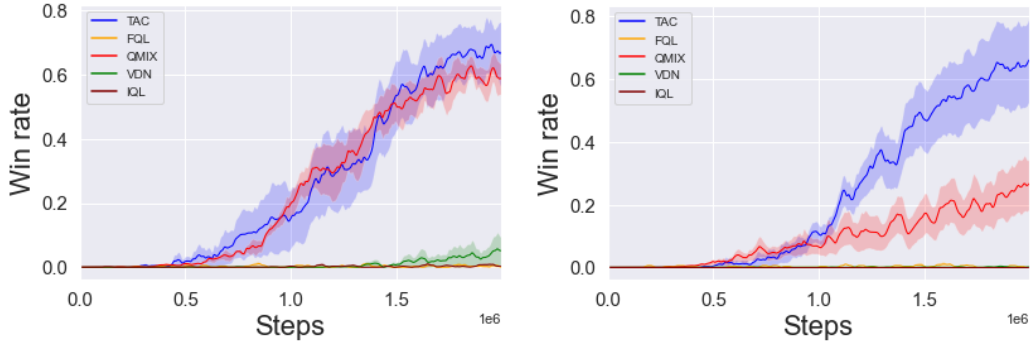
(a) 3s5z **Easy**

(b) 2s_vs_1sc **Easy**



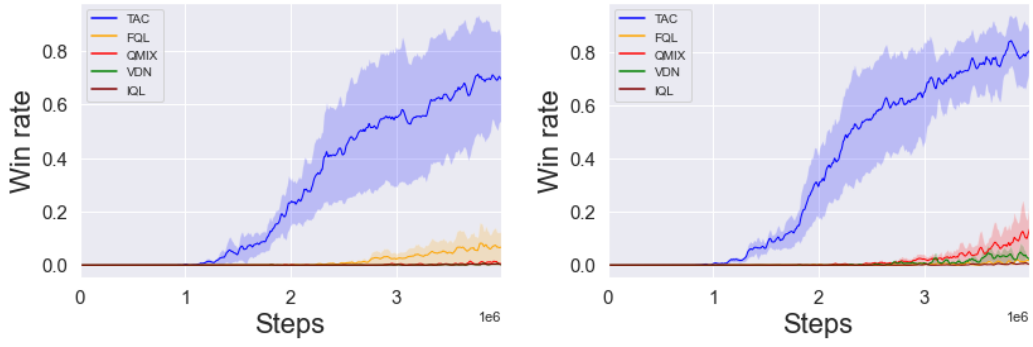
(c) 2c_vs_64zg **Hard**

(d) 5m_vs_6m **Hard**



(e) MMM2 **Super Hard**

(f) 27m_vs_30m **Super Hard**



(g) 6h_vs_8z **Super Hard**

(h) Corridor **Super Hard**

Figure 4.5: Performance of different algorithms on different SMAC scenarios: TAC, QMIX, VDN, FQL, IQL.

StarCraft II We consider a challenging set of cooperative scenarios from the StarCraft Multi-Agent Challenge (SMAC) [181]. Scenarios in SMAC have been classified as **Easy, Hard and Super-hard** according to the performance of exiting algorithms on them. We compare TESSERACT (TAC in plots) to, QMIX [172], VDN [198], FQL [39], and IQL [210]. VDN and QMIX use monotonic approximations for learning the Q-function. FQL uses a pairwise factorized model to capture effects of agent interactions in joint Q-function, this is done by learning an inner product space for summarising agent trajectories. IQL ignores the multi-agentness of the problem and learns an independent per agent policy for the resulting non-stationary problem. Fig. 4.5 gives the win rate of the different algorithms averaged across five random runs. Fig. 4.5(c) features 2c_vs_64zg, a hard scenario that contains two allied agents but 64 enemy units (the largest in the SMAC domain) making the action space of the agents much larger than in the other scenarios. TESSERACT gains a huge lead over all the other algorithms in just one million steps. For the asymmetric scenario of 5m_vs_6m Fig. 4.5(d), TESSERACT, QMIX, and VDN learn effective policies, similar behavior occurs in the heterogeneous scenarios of 3s5z Fig. 4.5(a) and MMM2Fig. 4.5(e) with the exception of VDN for the latter. In 2s_vs_1sc in Fig. 4.5(b), which requires a 'kiting' strategy to defeat the spine crawler, TESSERACT learns an optimal policy in just 100k steps. In the **super-hard** scenario of 27m_vs_30m Fig. 4.5(f) having largest ally team of 27 marines, TESSERACT again shows improved sample efficiency; this map also shows TESSERACT's ability to scale with the number of agents. Finally in the **super-hard** scenarios of 6 hydralisks vs 8 zealots Fig. 4.5(g) and Corridor Fig. 4.5(h) which require careful exploration, TESSERACT is the only algorithm which is able to find a good policy. We observe that IQL doesn't perform well on any of the maps as it doesn't model agent interactions/non-stationarity explicitly. FQL loses performance possibly because modelling just pairwise interactions with a single dot product might not be expressive enough for joint-Q. Finally, VDN and QMIX

are unable to perform well on many of the challenging scenarios possibly due to the monotonic approximation affecting the exploration adversely [140]. Additional plots and experiment details can be found in Appendix B.3.1 with **comparison with other baselines in Appendix B.3.1** including QPLEX[236], QTRAN[188], HQL[149], COMA[63] . We detail the techniques used for stabilising the learning of tensor decomposed critic in Appendix B.3.2.

4.6 Related Work

Policy gradient methods in MARL often utilise the actor-critic framework to cope with decentralisation. MADDPG [138] trains a centralised critic for each agent. COMA [63] employs a centralised critic and a counterfactual advantage function. These actor-critic methods, however, suffer from poor sample efficiency compared to value-based methods and often converge to sub-optimal local minima. While sample efficiency has been an important goal for single agent reinforcement learning methods [144, 146, 107, 127], in this work we shed light on attaining sample efficiency for cooperative multi-agent systems using low rank tensor approximation.

Tensor methods have been used in machine learning, in the context of learning latent variable models [5] and signal processing [186]. Tensor methods provides powerful analytical tools that have been used for various applications, including the theoretical analysis of deep neural networks [44]. Model compression using tensors [40] has recently gained momentum owing to the large sizes of deep neural nets. Using tensor decomposition within deep networks, it is possible to both compress and speed them up [41, 118]. They allow generalization to higher orders [119] and have also been used for multi-task learning and domain adaptation [34]. In contrast to prior work on value function factorisation, TESSERACT provides a natural spectrum for approximation of action-values based on the rank of approximation and provides theoretical guarantees derived from tensor analysis. Multi-view methods utilising

tensor decomposition have previously been used in the context of partially observable single-agent RL [10, 9]. There the goal is to efficiently infer the underlying MDP parameters for planning under rich observation settings [123]. Similarly [33] use four dimensional factorization to generalise across Q-tables whereas here we use them for modelling interactions across multiple agents.

4.7 Conclusions & Future Work

We introduced TESSERACT, a novel framework utilising the insight that the joint action value function for MARL can be seen as a tensor. TESSERACT provides a means for developing new sample efficient algorithms and obtain essential guarantees about convergence and recovery of the underlying dynamics. We further showed novel PAC bounds for learning under the framework using model-based algorithms. We also provided a model-free approach to implicitly induce low rank tensor approximation for better sample efficiency and showed that it outperforms current state of art methods. There are several interesting open questions to address in future work, such as convergence and error analysis for rank insufficient approximation, and analysis of the learning framework under different types of tensor decompositions like Tucker and tensor-train [114].

Chapter 5

Generalization in Cooperative Multi-Agent Systems

Contents

5.1 Introduction

In Chapter 3 and Chapter 4, we studied two different approaches for policy search in cooperative multi-agent systems, which were geared towards tackling the problems arising from an exponentially growing joint action space. In this chapter, we turn our attention to a different kind of multi-agent problem: One that is concerned with the generalization behaviour in a multi-agent systems as its composition changes.

We know that collective intelligence is a important trait shared by several species of living organisms. It has allowed them to thrive in the diverse environmental conditions that exist on our planet. From simple organisations in an ant colony to complex systems in human groups, collective intelligence is vital for solving complex survival tasks. As is commonly observed, such natural systems are flexible to changes in their structure. For instance, imagine attending a football summer camp. The coach decides to split the participating players into random teams for practice. While each player has different capabilities (e.g., defending, dribbling, speed, and pace), they quickly adapt to the other players in the team to facilitate the common objective of outscoring their opponents. Furthermore, they smoothly adjust to unexpected events such as a player getting hurt and retiring with substitution, which forces them to change their behaviours and adjust their roles. Similarly, they rapidly adjust to changes in team size (as a result of a player being sent off or new players joining the team).

Such adaptations are typically possible for two reasons. First, the players understand each others' *capabilities*, including how a change in capabilities affects the underlying environment and chances of success. Second, players have coordination protocols for adapting to the changes, both explicitly (e.g., communicating the game plan) or implicitly (inferring capabilities from observations, e.g., passing the ball to a player going in for an attack). This phenomenon which we term as *Combinatorial Generalization* (CG) is not specific to football or humans, and organisms in general

manifest abilities to adapt in almost every situation requiring team efforts [45, 157, 7].

Towards capturing specific aspects of CG, recent methods in multi-agent reinforcement learning (MARL) utilize advances in deep learning architectures, such as graph neural networks [179] and attention mechanism [100], as well as extensively tuned training regimes, such as a mixture of human and generated data, self-play, and population-based training [233, 160]. While these methods show impressive empirical performance on complex domains, they provide little insight into aspects of when and how much generalization to expect, which is crucial for deploying agents in the real world due to practical considerations like tolerance, minimum expected performance in unseen settings etc. for various deployment scenarios. Additionally, while the problem of sample-efficient generalization is hard for single-agent RL [145, 52, 67, 147], it is particularly exacerbated for the multi-agent case. Specifically, even when the underlying task remains the same, agents in MARL typically need to be trained from scratch for different team compositions. Moreover, across similar tasks with similar team compositions, there is a lack of modularity for sharing knowledge to enable quick learning [237]. Thus, we posit that a theoretical understanding of generalization in multi-agent systems (MAS) can help address both of the above-mentioned issues: it can provide important performance guarantees for practical deployment and can additionally inform better algorithm design to ensure sample efficiency.

We first highlight the key properties that make CG particularly difficult for MAS:

- **P1:** The capabilities of agents can come from infinite sets, e.g., maximum permissible torque for an agent joint which can take values in a continuous set.
- **P2:** Combinatorial blow-up in the number of possible teams (w.r.t. agent capabilities) given a team size.

- **P3:** The capabilities need to be grounded w.r.t. the dynamics of the environment which becomes increasingly hard with team size (similar to credit assignment).
- **P4:** Team sizes can vary across different tasks.
- **P5:** Agents need to infer the capabilities of teammates in settings where it is hidden, in a potentially non-stationary environment.

P2-**P4** particularly distinguish CG from single-agent generalization, highlighting its combinatorial nature. Furthermore, **P5** requires agents to adapt to changing teammate policies making the problem harder.

In this work, we analyse multi-agent generalization by modelling the dependence of underlying environment rewards and transitions on agent capabilities. We first look at generalization bounds for the case when the environment dynamics are linear with respect to the agent capabilities. We elucidate how this generalizes the successor feature (SF) framework [16] to the multi-agent case. We provide theoretical bounds for generalization between team compositions, transfer of optimal policy from one team to another and changes to optimal values arising from agent addition and elimination under this framework. Next, we bound the performance gap as a result of an error in estimating the agent capabilities which covers scenarios such as lossy or inaccurate communication. Further, we provide bounds for optimal value deviation when the dynamics themselves are approximately linear. Finally, we elucidate how the bounds can be extended to Lipschitz rewards (Appendix C.1.6) and then extend this framework to study arbitrary dependence of rewards on capabilities to shed light on when generalization can be difficult (Appendix C.1.7). Our results apply to various training and deployment settings in MAS and are agnostic to the type of algorithm used (MARL or other forms of policy search methods). Finally, we empirically analyse popular methods in MARL on tasks designed to offer varying difficulty in terms of generalization and discuss important desiderata to be met for

better generalization.

5.2 Background and Formulation

We start with the Dec-POMDP formulation and would also assume the various specialization of this setting as illustrated Section 2.5 in this work . Without loss of generality (WLOG), we assume the state is represented as a k-dimensional feature vector $S \subset [0,1]^k$ and similarly observations $Z \subset [0,1]^l$. In this chapter we denote $J^\pi = \sum_{s \in S} \rho(s) V^\pi(s)$ to represent the scalar policy value in addition to value function $V^\pi(\cdot)$ (a vector), we also consider rewards to be only dependent on state for the ease of exposition (similar results hold for using action dependence).

MARL with Agent Capabilities

We now extend the MARL problem setting for generalisation where agents can have different capabilities. To this end, we assume that each agent in the task can be characterised by a d-dimensional *capability vector* $c \in \mathcal{C}$, which governs its contribution to rewards and transition dynamics (and thus its policy/behaviour denoted as $\pi^i(\ .\ ; c)$). Without loss of generality, we assume $\mathcal{C} \subseteq \Delta_{d-1}$ (the $d-1$ dimensional simplex). Intuitively, an agent's capability reflects the abilities of an agent along various properties that may be important for solving the collective task (e.g., an agent's speed, health recovery, and accuracy). We next assume an unknown probability distribution $\mathcal{M} : \mathcal{C}^n \to \mathbb{R}^+$ with support $Sup(\mathcal{M})$ over a subset of the joint capability space $\mathcal{C}^n$. Any $\mathcal{T}$ sampled from $\mathcal{M}$ can be seen as a tuple of capability vectors $\mathcal{T} = (c_i)_{i=1}^n$, one for each agent in the team. We augment the Dec-POMDP with $\mathcal{T}$: $G = \langle S, U, P_\mathcal{T}, r_\mathcal{T}, Z, O, n, \rho, \gamma, \mathcal{T} \rangle$ and call it a *variation* for the MARL setting [*]. Thus $\mathcal{T}$ defines the rewards and transition dynamics of

[*]Agent capabilities can also be interpreted as the contexts, see [88]

the underlying MMDP (ie. $r_{\mathcal{T}}(s) = \langle f(\mathcal{T}) \cdot s \rangle$ where $\langle \cdot \rangle$ is the dot product[†] and $f : \mathcal{C}^n \to \mathbb{R}^k$ and similarly for transitions). Our goal is then to find algorithms, which when trained on a small number of *variations* sampled from $\mathcal{M} : \{\mathcal{T}^j\}_{j=1}^M$, generalise well to unseen team variations in $\mathcal{M}$. i.e., we want to maximise the expected value over the team variation distribution,

$$\max_{\pi} \; \mathbb{E}_{\mathcal{T}\sim\mathcal{M}} \left[\mathbb{E}_{\pi(\cdot;\mathcal{T}),P_{\mathcal{T}},\rho} \left[\sum_{t=0}^{\infty} \gamma^t r_{\mathcal{T}}(s_t) \right] \right], \tag{5.1}$$

where $\pi = \{\pi^i\}_{i=1}^n$ is a group of n agents. The challenge here arises because of two main factors. First, the agents do not have any prior knowledge about what these capability vectors mean, and are thus required to learn their semantics (also called grounding). Second, in the setting where the agents cannot observe the capability vectors (including possibly their own), they have to infer and learn protocols for sharing them with each other in order to generalize in a zero-shot setting.

Successor Features

SF framework assumes that the rewards in an MDP can be decomposed as $r(s) = \boldsymbol{\phi}(s)^\top \mathbf{w}$, where $\boldsymbol{\phi}(s) \in \mathbb{R}^d$ are features of s and $\mathbf{w} \in \mathbb{R}^d$ are weights [‡]. When no assumptions is made about $\boldsymbol{\phi}(s)$, any reward function can be recovered using this representation. The value function then follows

$$\begin{aligned} V^\pi(s) &= \mathrm{E}^\pi \left[r_{t+1} + \gamma r_{t+2} + \ldots \,|\, S_t = s \right] \\ &= \mathrm{E}^\pi \left[\boldsymbol{\phi}_{t+1}^\top \mathbf{w} + \gamma \boldsymbol{\phi}_{t+2}^\top \mathbf{w} + \ldots \,|\, S_t = s \right] \\ &= \boldsymbol{\psi}^\pi(s)^\top \mathbf{w}. \end{aligned}$$

[†]Note that this is still the most general form as states can be encoded as one-hot vectors, see [16].

[‡]Similar formulations hold WLOG for $\boldsymbol{\phi}$(s,a),$\boldsymbol{\phi}$(s,a,s')

Here $\boldsymbol{\psi}^{\pi}(s)$ is called the *successor feature* of s under policy π [47, 16, 15, 17]. The ith component of SF $\boldsymbol{\psi}^{\pi}(s)$ provides the expected discounted sum of ϕ_i when following policy π from s.

5.3 Analysis

Our analysis focuses on the generalisation properties w.r.t. $\mathcal{M}$. We focus on the case of MMDPs for ease of exposition, but similar results for the more general cases can be obtained by suitable assumptions for identifiability of the state (e.g., M-ROMDP in [142]). Our results are applicable irrespective of whether agents can observe the capabilities. They are also agnostic to the training and deployment regimes (e.g., centralized or decentralized) and the algorithm being used to find the policy. **All the proofs can be found in Appendix C.1.**

For the analysis we assume that the rewards and transitions depend linearly on the agents capabilities c_i :

$$r_{\mathcal{T}}(s) = \sum_{i=1}^{n} a_i \langle c_i \cdot W_R s \rangle \tag{5.2}$$

$$P_{\mathcal{T}}(s'|s, \mathbf{u}) = \sum_{i=1}^{n} a_i \langle c_i \cdot W_P(s', s, \mathbf{u}) \rangle \tag{5.3}$$

where $W_R \in \mathbb{R}^{dk}$ is the reward kernel of the MMDP and defines the dependence of the rewards on each capability component. Similarly in Eq. (5.3), $W_P : S \times \mathbf{U} \times S \times \{1..d\} \to [0, 1]$ defines the transition kernel of the MMDP so that $P_j(\cdot|s, \mathbf{u}) \triangleq W_P(s, \mathbf{u}, j) \in \Delta_{|S|-1}, j \in \{1..d\}$ give the next state distribution as directed by the j^{th} component of the capability and agent i's propensity (unweighted) to make the state transition to s' is given by $\left\langle c_i \cdot \left[P_1(s'|s, \mathbf{u}) \dots P_d(s'|s, \mathbf{u}) \right] \right\rangle = \langle c_i \cdot W_P(s', s, \mathbf{u}) \rangle$. Finally $(a_i)_{i=1}^{n} \in \Delta_{n-1}$ are the *influence weights* of agents which quantify the influence of agent i in determining the rewards and transitions. Under the linear setting, given a policy π and capabilities $\mathcal{T}$ we have that policy value satisfies

$J^{\pi}_{\mathcal{T}} = \sum_{i=1}^{n} a_i \langle c_i \cdot W_R \mu^{\pi}_{\mathcal{T}} \rangle$ where $\mu^{\pi}_{\mathcal{T}} = E_{\rho, P_{\mathcal{T}}, \pi}[\gamma^t s_t]$ are the expected discounted state features and similarly for a given state s, $V^{\pi}_{\mathcal{T}}(s) = \sum_{i=1}^{n} a_i \langle c_i^T W_R \cdot \mu^{\pi}_{\mathcal{T}}(s) \rangle$ where $\mu^{\pi}_{\mathcal{T}}(s) = E_{P_{\mathcal{T}}, \pi}[\gamma^t s_t | s_0 = s]$. The linear formulation for dynamics generalizes the successor feature [16] formulation to the MAS setting, this can be seen by noting that when the dependence of transition dynamics on capabilities is dropped (Eq. (5.3)) and only single agent is considered (by considering a one-hot a), we get the successor feature formulation with capability of the non zero a_i interpreted as the task weight in [16](see Section 5.2).

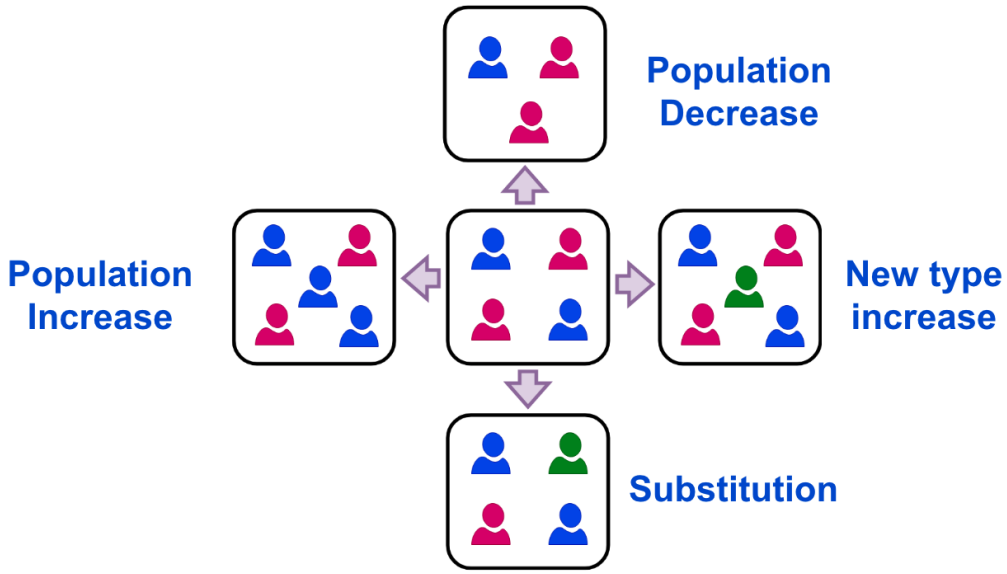


Figure 5.1: Combinatorial Generalization in MAS, various settings.

We now present the first result concerning the difference between the optimal values of two different team compositions:

Theorem 5.1 (Generalisation between team compositions)**.** *Let team compositions* $\mathcal{T}^x, \mathcal{T}^y \in \mathcal{C}^n$ *with influence weights* $a^x, a^y \in \Delta_{n-1}$, $s_{max} = \max_s ||W_R s||_1$, $V_{mid} = \frac{1}{2} \max_s V^*_{\mathcal{T}^y}(s)$, *Then*[§]*:*

$$|J^*_{\mathcal{T}^x} - J^*_{\mathcal{T}^y}| \leq \frac{s_{max} + \gamma d V_{mid}}{\gamma(1-\gamma)} \Psi, \text{ where}$$

[§]for $\gamma \in (0, \frac{\sqrt{5}-1}{2})$ we can replace the factor $\frac{1}{\gamma(1-\gamma)}$ by $\frac{1+\gamma}{1-\gamma}$

$$\Psi = \Big[|\sum_i a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty + |\sum_i (a_i^x - a_i^y)\mathcal{T}_i^y|_\infty \Big] \tag{5.4}$$

Theorem 5.1 gives an interesting decomposition of an upper bound to the difference of the optimal values between the two team compositions. The first terms in the square brackets on the RHS denotes contributions arising purely from substituting the old capacities with the new one. The second term denotes the contribution arising from a change in how much influence the agents have over the dynamics of the MMDP.

Corollary 5.1.1 (Change in optimal value as a result of agent substitution)**.** *Let $\mathcal{T} \in \mathcal{C}^n$ be a team composition with influence weights $a \in \Delta_{n-1}$. If agent i is substituted with i' keeping a_i unchanged such that $|\mathcal{T}_{i'} - \mathcal{T}_i|_\infty \leq \epsilon_C$ then the new team ($\mathcal{T}'$) optimal value follows:*

$$|J^*_{\mathcal{T}'} - J^*_{\mathcal{T}}| \leq \frac{(s_{max} + \gamma d V_{mid}) a_i \epsilon_C}{\gamma(1-\gamma)}$$

We define an important policy concept which captures the absolute optimality for an oracle with access to the capabilities. For the ease of exposition we consider fixed influence weights a and define a metric on the joint capability space as $d_a(\mathcal{T}^x, \mathcal{T}^y) = |\sum_i a_i(\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty$. We similarly generalize this metric to distances between sets by taking the infimum of the distances between pairs of points in the cross product $d_a(\mathcal{M}_x, \mathcal{M}_y) \triangleq \inf_{\mathcal{T}^x \in \mathcal{M}_x, \mathcal{T}^y \in \mathcal{M}_y} d_a(\mathcal{T}^x, \mathcal{T}^y)$.

Definition 5.1 (Absolute Oracle)**.** *Let $\pi^*_{\mathcal{M}}$ be the oracle policy which optimizes Eq.* (5.1) *ie. $\pi^*_{\mathcal{M}}$ is the multiplexer policy which given a team composition $\mathcal{T}$ behaves identically to the optimal policy for $\mathcal{T}^j$ where $\mathcal{T}^j \in \arg\min_{\mathcal{T}^l \in Sup(\mathcal{M})} d_a(\mathcal{T}^l, \mathcal{T})$.*

We now answer the question of what happens when agents are trained on specific capabilities but the learnt policy is used on potentially unseen capabilities (this

could occur e.g. due to changes in hardware components).

Theorem 5.2 (Transfer of optimal policy)**.** *Let* $\mathcal{T}^x, \mathcal{T}^y \in \mathcal{C}^n$, $a^x, a^y \in \Delta_{n-1}$, $s_{max} = \max_s ||W_R s||_1$, $V_{mid} = \frac{1}{2}\max_s V^*_{\mathcal{T}^y}(s)$. *Let* π^*_y *be the optimal policy for the team composed of agents with capabilities* $\mathcal{T}^y$ *and influence weights* a^y*. Then:*

$$J^*_{\mathcal{T}^x} - J^{\pi^*_y}_{\mathcal{T}^x} \leq 2\frac{s_{max} + \gamma d V_{mid}}{\gamma(1-\gamma)}\Psi,$$

where Ψ *is defined as in Eq.* (5.4)*.*

Corollary 5.2.1 (Out of distribution performance)**.** *Let* $\mathcal{T} \notin Sup(\mathcal{M})$ *be an out of distribution task, we then have that the performance of the absolute oracle policy on* $\mathcal{T}$ *satisfies:*

$$J^*_{\mathcal{T}} - J^{\pi^*_{\mathcal{M}}}_{\mathcal{T}} \leq 2\frac{s_{max} + \gamma d V_{mid}}{\gamma(1-\gamma)} d_a(\mathcal{T}, Sup(\mathcal{M})),$$

We now address the scenarios when the team population changes.

Theorem 5.3 (Population decrease bound)**.** *For the team composition* $\mathcal{T} \in \mathcal{C}^n$ *with influence weights* $a \in \Delta_{n-1}$*. If agent* n *is eliminated followed by a re-normalization of influence weights, we have that for the remaining team* ($\mathcal{T}^- \triangleq (\mathcal{T})_{i=1}^{n-1}$)*:*

$$|J^*_{\mathcal{T}^-} - J^*_{\mathcal{T}}| \leq \frac{a_n(s_{max} + \gamma d V_{mid})}{\gamma(1-\gamma)} \Big| \sum_{i=1}^{n-1} \frac{a_i \mathcal{T}_i}{1-a_n} - \mathcal{T}_n \Big|_\infty$$

The special case when $\sum_{i=1}^{n-1} \frac{a_i \mathcal{T}_i}{1-a_n} = \mathcal{T}_n$ for the linear dynamics formulation when an agent-n can in principle be rendered redundant if the rest of the agents in the team can effectively provide a perfect substitute. In fact, this holds true as long as capacity $\mathcal{T}_n$ can be formed from a convex combination of the capabilities $\mathcal{T}_i, i \in \{1..n-1\}$. The latter case however requires using the corresponding convex coefficients instead of re-normalization. A similar bound can be easily constructed for reusing the policy after an agent eliminated to give the corresponding transfer bound along the lines

of Theorem 5.2.

Corollary 5.3.1 (Population increase bound)**.** *For the team composition $\mathcal{T} \in \mathcal{C}^n$ with influence weights $a \in \Delta_{n-1}$. If agent $n+1$ is added with capability $\mathcal{T}_{n+1}$ and weight a_{n+1} (other weights scaled down by $\lambda = 1 - a_{n+1}$) we have that for the new team ($\mathcal{T}^+ \triangleq (\mathcal{T}_1..\mathcal{T}_n, \mathcal{T}_{n+1})$):*

$$|J^*_{\mathcal{T}^+} - J^*_{\mathcal{T}}| \leq \frac{a_{n+1}(s_{max} + \gamma d V_{mid})}{\gamma(1-\gamma)} \Big| \sum_{i=1}^{n} a_i \mathcal{T}_i - \mathcal{T}_{n+1} \Big|_\infty$$

We next extend the generalization bound Theorem 5.1 to include the scenario where the reward and the transition dynamics are not exactly linear but are approximately linear with deviation $\hat{\epsilon}_R$,$\hat{\epsilon}_P$ respectively.

Theorem 5.4 (Approximate $\hat{\epsilon}_R$,$\hat{\epsilon}_P$ dynamics)**.** *Let $\mathcal{T}^x, \mathcal{T}^y \in \mathcal{C}^n$, $a^x, a^y \in \Delta_{n-1}$ and the dynamics be only approximately linear so that $|r_{\mathcal{T}}(s) - \sum_{i=1}^{n} a_i \langle c_i \cdot W_R s \rangle| \leq \hat{\epsilon}_R$ and $|P_{\mathcal{T}}(s'|s, \mathbf{u}) - \sum_{i=1}^{n} a_i \langle c_i \cdot W_P(s', s, \mathbf{u}) \rangle| \leq \hat{\epsilon}_P$. Then:*

$$|J^*_{\mathcal{T}^x} - J^*_{\mathcal{T}^y}| \leq \frac{s_{max} + \gamma d V_{mid}}{\gamma(1-\gamma)} \Psi + \frac{2(\hat{\epsilon}_R + \gamma \hat{\epsilon}_P V_{mid})}{\gamma(1-\gamma)},$$

where Ψ is defined as in Eq. (5.4)*.*

The other bounds for transfer and population change can similarly be obtained for the approximate dynamics case.

We now consider the scenario when the capabilities are not directly observed but inferred using an approximator which in-turn introduces some errors in their estimation (this could happen due to noise in observations, inaccurate implicit or explicit communication protocols, etc.).

Theorem 5.5 (Error from estimation of capabilities)**.** *For the team composition $\mathcal{T} \in \mathcal{C}^n$ with influence weights $a \in \Delta_{n-1}$. If the agent capabilities are inaccurately*

inferred as $\hat{\mathcal{T}}$ *with* $\max_i |\mathcal{T}_i - \hat{\mathcal{T}}_i|_\infty \leq \epsilon_{\mathcal{T}}$ *and agents learn the inexact policy* $\hat{\pi}^*$ *then:*

$$|J^*_{\mathcal{T}} - J^{\hat{\pi}^*}_{\mathcal{T}}| \leq \frac{2\epsilon_{\mathcal{T}}(s_{max} + \gamma d V_{mid})}{\gamma(1-\gamma)}$$

where $V_{mid} = \frac{1}{2}\max_s V^*_{\hat{\mathcal{T}}}(s)$

We note that all our results can be easily extended to the setting where rewards $r_{\mathcal{T}}(s) = \langle f(\mathcal{T}) \cdot W_R s \rangle$, $f(\mathcal{T})$ is not linear in capabilities as in Eq. (5.2) but is Lipschitz with coefficient L_i for $i \in \mathcal{A}$. For eg. Theorem 5.1 becomes:

Theorem 5.6. *For rewards* L_i *Lipschitz in the capabilities with respect to* $|\cdot|_\infty$ *norm, the difference in optimal values between team compositions* $\mathcal{T}^x, \mathcal{T}^y$ *satisfy:*

$$|J^*_{\mathcal{T}^x} - J^*_{\mathcal{T}^y}| \leq \frac{s_{max} \sum_{i=1}^n L_i |\mathcal{T}^x_i - \mathcal{T}^y_i|_\infty}{\gamma(1-\gamma)}$$

See Appendix C.1.6 for proof, which also provides a method for extending the other results in a similar fashion.

We now consider the dependence of rewards on the capabilities in the most general form (as is common for dense capability embeddings). For this, we introduce the notion of (α, K)-rewards where $\alpha \geq 0, K \in \mathbb{N}$.

$$r_{\mathcal{T}}(s) = \left\langle \sum_{K_i \in \mathbb{N}, \sum K_i \leq K} a_{K_1..K_n} \Pi_{i=1}^n c_i^{K_i} \cdot W_R s \right\rangle \tag{5.5}$$

where $\mathbb{N}$ are non negative integers, $|a_{K_1..K_n}| \leq \alpha$ and $c_i^{K_i}$ represents element-wise exponentiation. . Rewards in Eq. (5.2) can be seen as a special case belonging to Eq. (5.5) the choice $\alpha, K = 1$. Similarly the union $\cup_{\alpha \geq 0, K \in \mathbb{N}}(\alpha, K)$-rewards cover all possible reward dependencies on capabilities. We have further relaxed the assumption of influence weights belonging to a simplex here and replaced it with individual bounds on the power series coefficients here. We next see that for

this scenario, even a small change in the capability of a single agent can shift the rewards massively. Let the capability of agent i be changed from $\mathcal{T}_i$ to $\mathcal{T}_{i'}$ such that $|\mathcal{T}_i - \mathcal{T}_{i'}|_\infty \leq \delta$. Then we have

Lemma 5.1. *For substitution $\mathcal{T}_i$ to $\mathcal{T}_{i'}$ such that $|\mathcal{T}_i - \mathcal{T}_{i'}|_\infty \leq \delta$ under the (α, K)-rewards setting we have that*

$$\epsilon_R \in \mathcal{O}(\alpha \delta s_{max} K 2^K)$$

See Appendix C.1.7 for proof. While this is not a lower bound, the above still suggests that even a small change in the capability of an agent can cause the rewards to change by a lot, hence it is natural to expect that generalization becomes harder as the problem start showing the needle in the haystack phenomenon where only the *right combination* of capabilities gives a large optimal value.

We provide experiments elucidating the bounds stated above in Section 5.5.1.

5.4 Experimental Setup

We evaluate the ability of existing MARL algorithms to generalize to novel settings where the capabilities of teammates change during the training. We are interested in evaluating the gap between settings encountered during training and held-out agent configurations reserved for testing. Furthermore, we aim to study how well algorithms ground privileged information about teammate capabilities and use that during unseen settings at test time. Lastly, we evaluate the bounds derived in Section 5.3 on a simple multi-agent problem.

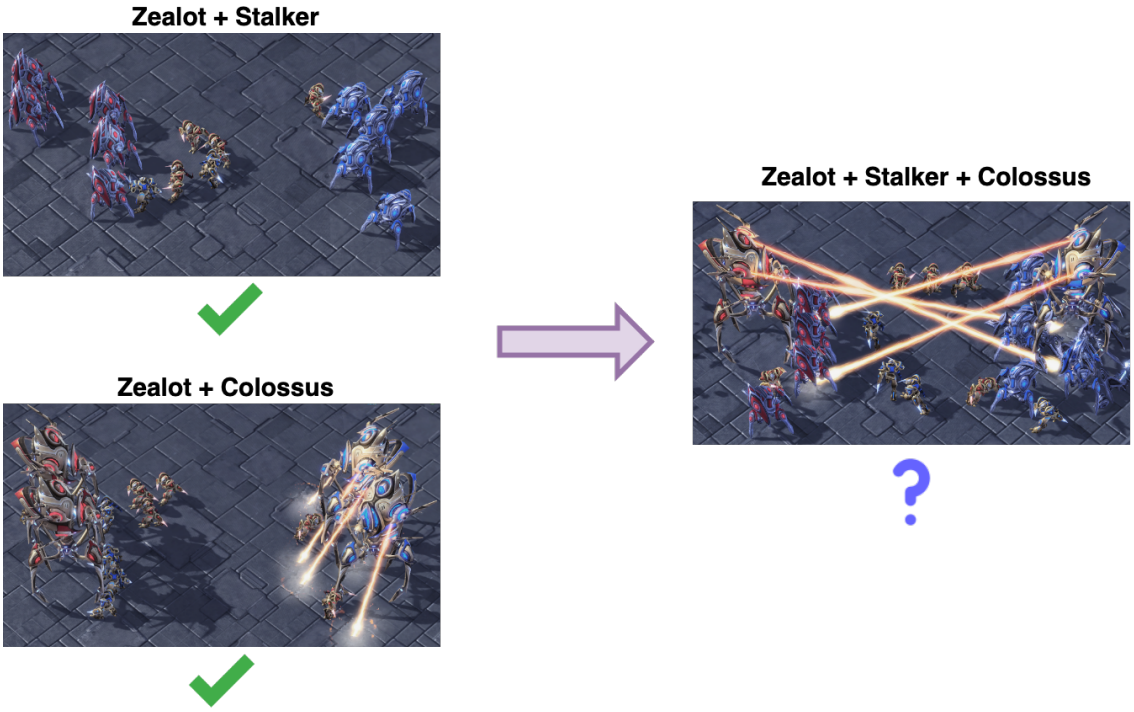


Figure 5.2: Three episodes from the `10_Protoss_Hard` task (a) One featuring only Zealot and Stalkers during training. (b) One featuring only Zealot and Colossus during training. (c) A held-out episode featuring Zealot, Stalker, and Colossus encountered during testing.

5.4.1 Environments

Fruit Forage

We use the fruit forage task on a grid world to empirically demonstrate the generalisation bounds in Section 5.3. On a 8×8 grid world we have n agents and d types of fruit trees. For each agent i, $\mathcal{T}_i(j), j \in \{1..d\}$ represents the utility of fruit j for agent i. The state vector is appended with the d dimensional binary vector representing whether each of the tree types has foraged at a given time step. The details for the team compositions can be found in Appendix C.2.1.

Predator Prey

We consider the grid-world version of the multi-agent Predator Prey task where 4 agents have to hunt 4 prey in an 8×8 grid. Here, both predators and preys have certain capabilities. Specifically, each predator has a parameter describing the hit

point damage it can cause the prey. Similarly, the prey comes with variations in health. For example, a prey with a capability of 5 can only be caught if the total capability of agents taking the capture action simultaneously on it have capabilities ≥ 5 (such as [1,1,1,2]), otherwise, the whole team receives a penalty p. Here, we test for generalization to novel team composition where test tasks contain a team composition which has not been encountered during training (PP Unseen Team in Figure 5.4), and additionally test tasks where novel team compositions can also have agent types with capabilities not encountered during training (PP Unseen Team, Agent in Figure 5.4). More details are provided in the Appendix C.2.1.

StarCraft II

To assess the generalization capabilities of modern MARL approaches, we make use of a modified version of StarCraft II unit micromanagement tasks of the SMAC benchmark [181]. Particularly, we consider novel scenarios featuring three unit types from each race of the game where the team composition changes during training and testing, unlike standard SMAC which is static. The opponent's team is always identical to the ally team which ensures that the optimal win rate is close to 1. In the simple cases (`10_Protoss`, `10_Zerg`, and `10_Terran`), agents are trained on various team formations of 10 units that feature all combinations of one, two, and all three unit types, and is later tested on held out team formations. In the hard cases (`10_Protoss_Hard`, `10_Zerg_Hard`, and `10_Terran_Hard`), agents are exposed to various team formations including two unit types during training. During testing, however, the agents encounter held-out scenarios featuring scenarios with using all three unit types (see Appendix C.2.1 for more details). Fig. 5.2 illustrates three episodes from the `10_Protoss_Hard` environment. In these tasks, agent capabilities are described as a one-hot encoding of agent types.

To test performance on continuously varying capabilities, we also use variants of the environment where either the health or attack accuracy of certain units are

reduced. We randomize these configurations for the allied units during training and later test on held-out team configurations. We evaluate baselines on the `3m`, `2s3z`, `8m` scenarios from the original benchmark with these modifications. The varying team size also helps understand how grounding the capabilities becomes harder as team size increases. Here agent capabilities are described as their accuracy or health coefficients. Further details are provided in the Appendix C.2.1.

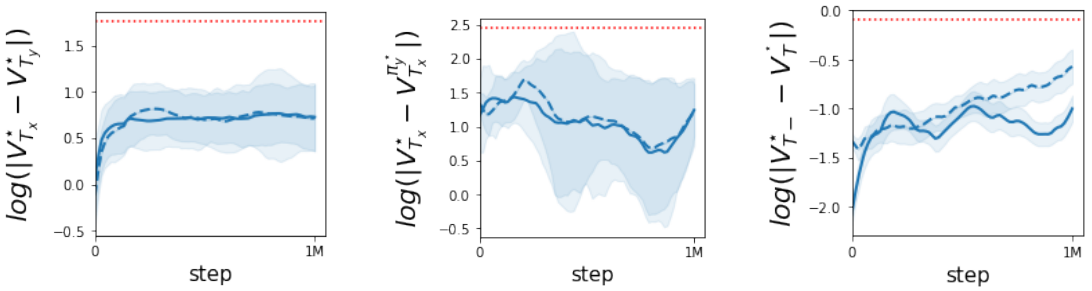
5.4.2 Baselines

Our empirical evaluation is based on various types of MARL algorithms. We make use of two popular value-based approaches, QMIX [172] and VDN [197] that trained fully decentralized policies in a centralized fashion. We also use policy gradient method PPO [184] that recently shown good results on various MARL domains, both with decentralised (Independent PPO) [49] and centralised critics (MAPPO) [248]. We access the performance of all baselines when the information about teammates capabilities are provided as observation (denoted with a 'C' in parentheses) and when it is not. **The evaluation procedure, architectures and training details are presented in Appendix C.2.2.**

5.5 Results and Discussion

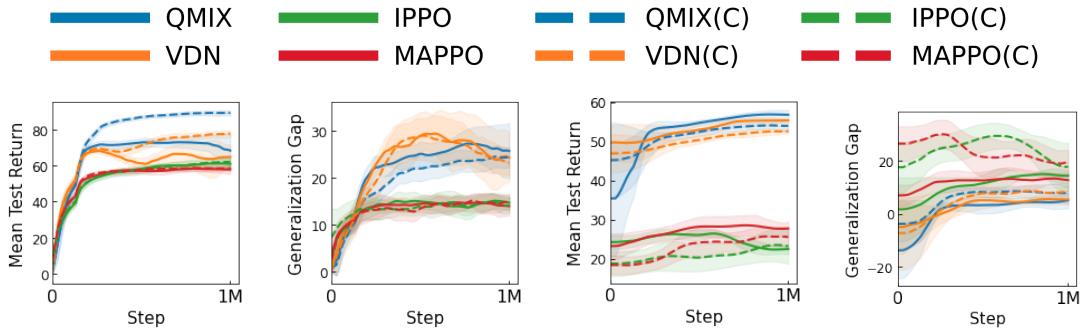
5.5.1 Generalization Bounds

Fig. 5.3 provides empirical evaluation of bounds presented in Section 5.3 in the Fruit Forage domain. We present the plots for training the agents for one million steps of training using QMIX. Fig. 5.3(a) shows that the policies in both the domains converge quickly leading to a stable difference in performance thus comfortably satisfying Theorem 5.1. Fig. 5.3(b) showing the gap between optimal and transferred policy shows interesting variations as training proceeds (we posit this happens



(a) Theorem 1 (b) Theorem 2 (c) Theorem 3

Figure 5.3: Evaluating the bounds for QMIX on Fruit Forage domain. Dashed blue line indicates the setting where agent capabilities are observable. The red dotted line indicates the corresponding upper bound for each theorem.



(a) PP Unseen Team (b) PP Unseen Team (c) PP Unseen Team, Agent (d) PP Unseen Team, Agent

Figure 5.4: Experimental results for the Predator Prey domain. Standard deviation is shaded.

because the transferred policy becomes steadily specialized thus getting less useful for the target task) the bound in Theorem 5.2 gives a tight fit despite these variations. Finally, we see similarly good fit for the agent elimination scenario in Theorem 5.3 in Fig. 5.3(c).

5.5.2 Utilizing Information of Agent Capabilities

Fig. 5.4 presents the results of the baselines on Predator Prey domain. We can observe from Fig. 5.4(a) that providing additional information on agent capabilities improves the test-time performance of the baselines with the maximal effect seen on QMIX and VDN. Furthermore, when capabilities are observable to the agents,

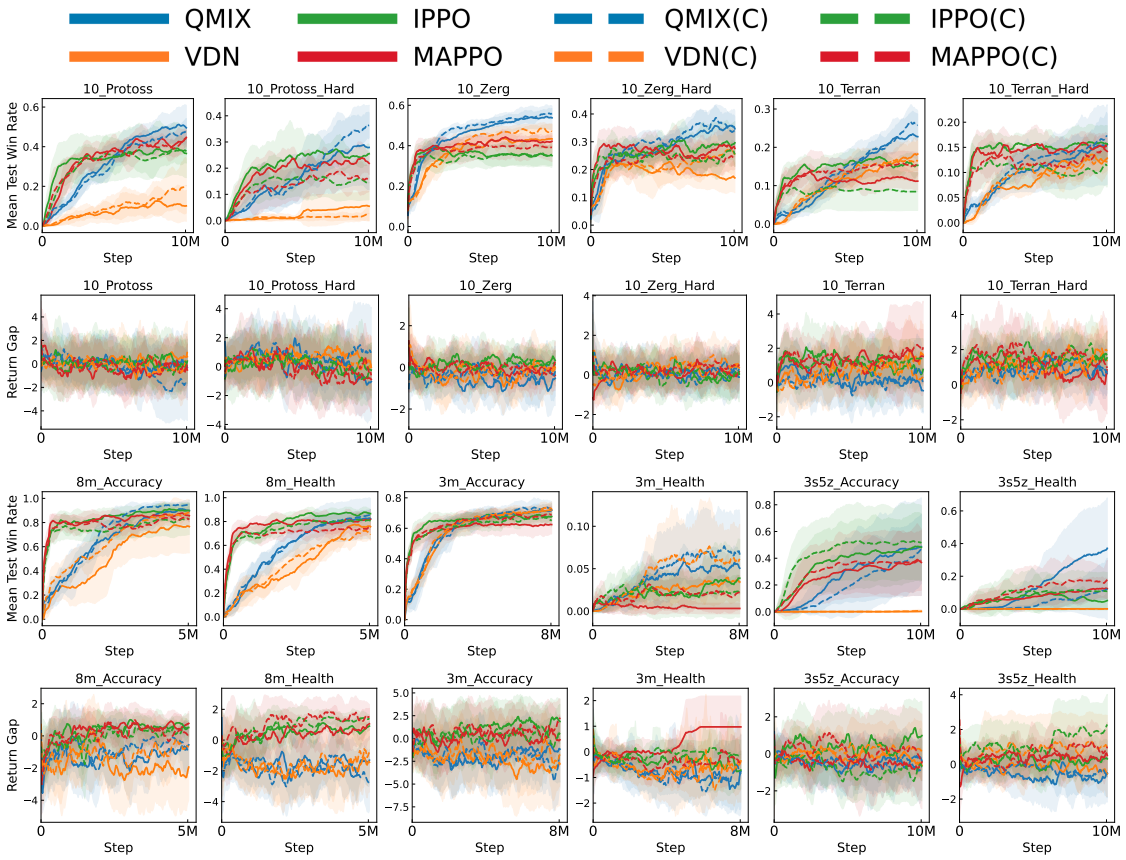


Figure 5.5: Experimental results on the SMAC benchmark. Standard deviation is shaded. Rows show win rates and generalization gaps.

baselines are able to generalize to new team compositions Fig. 5.4(b), thus successfully grounding the additional information. This hypothesis is additionally supported by the fact that knowing agent capabilities result in a lower generalization gap. Finally, the gap between the settings with known vs. unknown capabilities (dashed vs solid) indicates that agents have likely not come up with any appropriate protocol to communicate their capabilities during test time. We also note that the PPO variants do not perform as well as the value-based approaches. Therefore, their low generalization gap Fig. 5.4(b) is unlikely representative of good grounding of capability. We posit that this is just because PPO agents are ignoring the privileged information when available.

For a harder scenario, where both new team composition and agent types appear during evaluation Fig. 5.4(c), we observe that the situation is reversed from the

previous setting as the agents which do not have access to the other's capabilities now perform slightly better. This is strongly indicative of insufficient grounding of the privileged information given to them, which highlights the need for better grounding mechanisms to obtain CG. We see a similar pattern on generalization gap in Fig. 5.4(d) where privileged information hurts the performance and is likely perceived as observation noise.

On the more challenging domain of StarCraft, we see that for easier capability variations of health and accuracy (as they are continuous and more readily usable for an agent's immediate actions), knowing the capabilities is advantageous to the agent during test time. Moreover, the relative gains of knowing the privileged information go down as the task difficulty increases. The accuracy variations tend to be easier as typical joint policies like focus fire remain unchanged. Moreover, health variations on the smaller team make the task much harder than bigger team due to relative loss in team hit points. In this regard, `8m, 3s5z` accuracy versions show good grounding and generalization. This changes as tasks get harder. On the harder tasks which involve swapping unit types within `Protoss, Zerg, Terran` races, we observe that knowing the capabilities of other agents gives little advantage. This is especially noticeable on the `Hard` versions where all unit types are never within a single team during training. Furthermore, with win-rate performances on these maps being low, we hypothesise that the agents do not successfully utilize the capability information. Thus, it is highly unlikely that they learn any meaningful communication protocols for exchanging capability information. **For full StarCraft II results, including `8m_vs_9m` & `10m_vs_11m` scenarios, see Appendix C.3**.

Compared to the relatively simple Predator Prey task, generalization in StarCraft proved to be more difficult for the baselines. Although static versions of SMAC environments are comfortably solved by them [172, 49, 248], changing unit formations or unit health/accuracy makes the tasks significantly difficult, even for configurations

seen during the training. As observed in Fig. 5.5, providing the capability information does not consistently improve the test-time performance. This suggests the poor grounding abilities of the baseline algorithms, which reinforces the need for better grounding mechanisms in the MARL algorithms (e.g., forward dynamics prediction as in [101]). The failure to generalize on index-based privileged information regarding agent types suggests using mechanisms such as latent embeddings to compose and reason about capabilities. Finally, a low test performance gap between agents having privileged information vs those which do not, coupled with a low generalization gap, suggests that these methods do not facilitate information sharing between the agents, which is another desideratum towards attaining CG.

5.6 Related Work

Multi-agent systems offer means to overcome theoretical barriers like exponential blow up in state-action space and compute resource requirements for large problems. [100] regularize value functions to share factors comprised of sub-groups of entities, in order to transfer knowledge across cooperative tasks. In the competitive/general sum MARL space [160] have shown impressive performance on complex tasks. [231] use an options framework to learn agents which generalize against different opponents. [46, 226, 169] explore the structural and theoretical properties of general payoff games.

Ad-hoc coordination was first formalised by [193] by modelling the multi-agent problem as a single-agent task and using competency scores to measure agent compatibility. Methods for using explicit hard-coded protocols for adaptations were explored in [207, 74]. Opponent modelling for general game was explored in [194, 148, 128]. Several approaches to the ad-hoc cooperation problem assume that the behaviour of other agents in the ensemble are fixed [30]. Planning methods like Monte Carlo tree search are used for finding optimal adaptation policy from

a fixed set of choices [18, 3, 4]. [156] develop over this by enabling learning a set of behaviours for the adapting agent while performing the task with human agents instead of assuming that it is given beforehand. Recent methods allow a change in the behaviour of the other agents to ones picked from a fixed set and account for the possible non-stationarities using change point detection [93, 174]. However, these methods do not consider arbitrary learning for other agents. Furthermore, they do not focus on generalization to unseen agent capabilities.

Generalization in RL aims to develop approaches that generalize well to the novel, unseen scenarios after training. Such methods avoid overfitting to seen tasks and can produce robust behaviour when deployed to novel settings. Recent work on generalization in single-agent RL make use of techniques such as data augmentation [171, 121], adaptive task distribution [213, 129], encoding inductive biases [94], and regularization [43]. Methods in contextual MDPs [88, 252] also provide generalization with guarantees. Recent work also elucidate some of the fundamental bounds arising from computational complexity which prevents sample efficient generalization [52, 67, 147].

5.7 Conclusion and Future work

We studied the generalization properties in multi-agent systems (MAS) following Markovian dynamics with a linear dependence of dynamics on the agent capabilities. We showed how the framework extends the successor feature setting to MAS. We explored performance bounds for various interesting scenarios arising in the MAS including generalization, transfer, agent substitutions, approximate inference of capabilities and deviations in environment dynamics. Furthermore, we showed how the bounds can be extended to the Lipschitz reward setting and elucidated the most general form of rewards and how they make generalization difficult. Finally, we extensively tested the popular MARL algorithms on domains presenting a wide

spectrum of hardness for CG. We saw that while some algorithms demonstrated CG on easier domains, all of them are insufficient towards ensuring CG on the challenging domains. We further highlighted how the first step towards ensuring CG should be ensuring proper grounding of agent capabilities. For future work, we aim to provide tighter bounds for CG for more general dynamics and create methods for better grounding of agent capabilities.

Related works for Part I

Previous methods for modelling multi-agent interactions include those that use coordination graph methods for learning a factored joint action-value estimation [79, 80, 14], however typically require knowledge of the underlying coordination graph. To tackle computational intractability from exponential blow-up of state-action space, Guestrin et al. [77, 78] use coordination graphs to factor large MDPs for multi-agent systems and propose inter-agent communication arising from message passing on the graphs. Similarly [195, 103] model inter-agent communication explicitly.

In recent years there has been considerable work extending MARL from small discrete state spaces that can be handled by tabular methods [245, 35] to high-dimensional, continuous state spaces that require the use of function approximators [63, 138, 167]. In CTDE value based setting, [122], [208] extend Independent Q-Learning [209] to use DQN to learn Q-values for each agent independently. [159] tackle the instability that arises from training the agents independently. Lin et al.[136] first learn a centralised controller to solve the task, and then train the agents to imitate its behaviour. Sunehag et al.[197] propose *Value Decomposition Networks* (VDN), which learn the joint-action Q-values by factoring them as the sum of each agent's Q-values. QMIX [172] extends VDN to allow the joint action Q-values to be a monotonic combination of each agent's Q-Values that can vary depending on the state. QTRAN [188] approaches the suboptimality vs. decentralisation tradeoff differently by introducing relaxed L2 penalties in the RL objective. MAVEN [140] learns a diverse ensemble of

monotonic approximations by conditioning agent Q-functions on a latent space which helps overcome the detrimental effects of decentralization constraints on exploration in value based methods. Similarly, Uneven [82] uses universal successor features for efficient exploration in the joint action space. Qatten [246] makes use of a multi-head attention mechanism to decompose Q_{tot} into a linear combination of per-agent terms. RODE [237] learns an action effect based role decomposition for sample efficient learning. Similarly policy based approaches like MADDPG [138] and Tesseract [142] use function approximation for scaling to large problems and allow for native CTDE support by using factored policies.

Part II

Learning in continuous state-action spaces

Chapter 6

Virel: Variational inference framework for reinforcement learning

Contents

6.1 Introduction

In this chapter we turn our attention to the problem of reinforcement learning (RL) in continuous state-action spaces. Towards this, we utilize a perspective of looking at the RL control problem that is very distinct from the classical approaches in RL: namely RL as probabilistic inference. Efforts to combine reinforcement learning and probabilistic inference have a long history, spanning diverse fields such as control, robotics, and RL [222, 221, 168, 175, 90, 258, 257, 256, 132]. Formalising RL as probabilistic inference enables the application of many approximate inference tools to reinforcement learning, extending models in flexible and powerful ways [131] thus also enabling dealing with large continuous spaces. However, existing methods at the intersection of RL and inference suffer from several deficiencies. Methods that derive from the pseudo-likelihood inference framework [48, 222, 168, 87, 155, 1] and use expectation-maximisation (EM) favour risk-seeking policies [130], which can be suboptimal. Yet another approach, the MERL inference framework [131] (which we refer to as MERLIN), derives from maximum entropy reinforcement learning (MERL) [116, 258, 257, 256]. While MERLIN does not suffer from the issues of the pseudo-likelihood inference framework, it presents different practical difficulties. These methods do not naturally learn deterministic optimal policies and constraining the variational policies to be deterministic renders inference intractable [175]. Moreover, these methods rely on soft value functions which, as we demonstrate empirically in Section 6.6, are less suited to capturing complex underlying value structures in higher dimensional MDPs.

Additionally, no framework formally accounts for replacing exact value functions with function approximators in the objective; learning function approximators is carried out independently of the inference problem and no analysis of convergence is given for the corresponding algorithms.

This work addresses these deficiencies. We introduce VIREL, an inference framework

that translates the problem of finding an optimal policy into an inference problem. Given this framework, we demonstrate that applying EM induces a family of actor-critic algorithms, where the E-step corresponds exactly to policy improvement and the M-step corresponds exactly to policy evaluation. Using a simple variational EM algorithm, we derive analytic updates for both the model and variational policy parameters, giving a unified approach to learning parametrised value functions and optimal policies.

We extensively evaluate two algorithms derived from our framework against DDPG [135] and an existing state-of-the-art actor-critic algorithm, soft actor-critic (SAC) [85], on a variety of OpenAI gym domains [32]. While our algorithms perform similarly to SAC and DDPG on simple low dimensional tasks, they outperform them substantially on complex, high dimensional tasks due their ability to better represent multi-modal value structures in higher dimensional MDPs.

The main contributions of this work are: 1) an exact reduction of entropy regularised RL to probabilistic inference using value function estimators; 2) the introduction of a theoretically justified general framework for developing inference-style algorithms for RL that incorporate the uncertainty in the optimality of $\hat{Q}_\omega(h)$ to drive exploration, but that can also learn optimal deterministic policies; and 3) a family of practical algorithms arising from our framework that adaptively balances exploration-driving entropy with the RL objective and outperforms the current state-of-the-art SAC, reconciling existing advanced actor critic methods like A3C [151], MPO [1] and EPG [42] into a broader theoretical approach.

6.2 Background

We assume familiarity with probabilistic inference [105] and also provide a quick review in Section 2.8.

6.2.1 Reinforcement Learning

We start with the Markov decision process (MDP) setup defined by the tuple $\langle \mathcal{S}, U, r, p, \rho, \gamma \rangle$ Section 2.1 . We assume WLOG $U \subseteq \mathbb{R}^n$ is set of available actions for the continuous action space. We use the notation for a state-action pair as $h \in \mathcal{H}$, $h \coloneqq \langle s, a \rangle$. Our goal being maximization of policy value:

$$J^\pi = \mathbb{E}_{h \sim \rho(s)\pi(a|s)} \left[Q^\pi(h) \right]. \tag{6.1}$$

6.2.2 Maximum Entropy RL

The MERL objective supplements each reward in the RL objective with an entropy term [218, 258, 257, 256], $J^\pi_{\text{merl}} \coloneqq \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{t=0}^{T-1} \left(r_t - c \log(\pi(a_t|s_t)) \right) \right]$. The standard RL, undiscounted objective is recovered for $c \to 0$ and we assume $c = 1$ without loss of generality. The MERL objective is often used to motivate the MERL inference framework (which we call MERLIN) [130], mapping the problem of finding the optimal policy, $\pi^*_{\text{merl}}(a|s) = \arg\max_\pi J^\pi_{\text{merl}}$, to an equivalent inference problem. A full exposition of this framework is given by [131] and we discuss the graphical model of MERLIN in comparison to VIREL in Section 6.3.3. The inference problem is often solved using a message passing algorithm, where the log backward messages are called soft value functions due to their similarity to classic (hard) value functions [220, 176, 85, 84, 131]. The soft Q-function is defined as $Q^\pi_{\text{soft}}(h) \coloneqq \mathbb{E}_{\tau \sim q^\pi(\tau|h)} \left[r_0 + \sum_{t=1}^{T-1} (r_t - \log \pi(a_t|s_t)) \right]$ where $q^\pi(\tau|h) \coloneqq p(s_0|h) \prod_{t=0}^{T-1} p(s_{t+1}|h_t) \pi(a_t|s_t)$. The corresponding soft Bellman operator is $\mathcal{T}^\pi_{\text{soft}} \bullet \coloneqq r(h) + \mathbb{E}_{h' \sim p(s'|h)\pi(a'|s')}[\bullet - \log \pi(a'|s')]$. Several algorithms have been developed that mirror existing RL algorithms using soft Bellman equations, including maximum entropy policy gradients [131], soft Q-learning [84], and soft actor-critic (SAC) [85]. MERL is also compatible with methods that use recall traces [71].

A drawback of MERLIN is that optimal deterministic policies are not learnt natu-

rally. Although a deterministic policy can be constructed by approximating $a \in \arg\max_a Q^*_{\text{soft}}(a, s)$ with the mean of the learnt stochastic policy [85], even when this estimate is accurate, there exists no analysis relating $a \in \arg\max'_a Q^*_{\text{soft}}(a, s)$ to the optimal action $a \in \arg\max_{a'} Q^*(a', s)$ under J^π [190]. Constraining the variational policies to the set of delta distributions renders the inference intractable [175, 176].

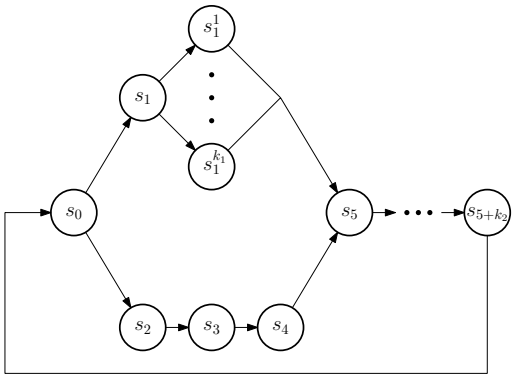


Figure 6.1: A discrete MDP counterexample for optimal policy under maximum entropy.

Next, we demonstrate that the optimal policy under J^π cannot always be recovered from the MAP policy under J^π_{merl}. Consider the discrete state MDP as shown in Fig. 6.1, with action set $\mathcal{A} = \{a_1, a_2, a_1^1, \cdots a_1^{k_1}\}$ and state set $\mathcal{S} = \{s_0, s_1, s_2, s_3, s_4, s_1^1 \cdots s_1^{k_1}, s_5, \cdots s_{5+k_2}\}$. All state transitions are deterministic, with $p(s_1|s_0, a_1) = p(s_1|s_0, a_2) = p(s_1^i|s_1, a_1^i) = 1$. All other state transitions are deterministic and independent of action taken, that is, $p(s_j|\cdot, s_{j-1}) = 1 \; \forall \; j > 2$ and $p(s_5|\cdot, s_1^i) = 1$. The reward function is $r(s_0, a_2) = 1$ and zero otherwise. Clearly the optimal policy under J^π has $\pi^*(a_2|s_0) = 1$. Define a maximum entropy reinforcement learning policy as π_{merl} with $\pi_{\text{merl}}(a_1|s_0) = p_1$, $\pi_{\text{merl}}(a_2|s_0) = (1 - p_1)$ and $\pi_{\text{merl}}(a_1^i|s_1) = p_1^i$. For π_{merl} and $k_2 >> 5$, we can evaluate J^π_{merl} for any scaling constant c and discount factor γ as:

$$J^\pi_{\text{merl}} = (1 - p_1)(1 - c\log(1 - p_1)) - p_1 \left(c\log p_1 + \gamma c \sum_{i=1}^{k} p_1^i \log p_1^i \right). \tag{6.2}$$

We now find the optimal MERL policy. Note that $p_1^i = \frac{1}{k}$ maximises the final term in Eq. (6.2). Substituting for $p_1^i = \frac{1}{k_1}$, then taking derivatives of Eq. (6.2) with

respect to p_1, and setting to zero, we find $p_1^* = \pi^*_{\text{merl}}(a_1|s_0)$ as:

$$1 - c\log(1 - p_1^*) = \gamma c \log(k_1) - c \log p_1^*,$$
$$\implies p_1^* = \frac{1}{{k_1}^{-\gamma} \exp\left(\frac{1}{c}\right) + 1},$$

hence, for any ${k_1}^{-\gamma} \exp\left(\frac{1}{c}\right) < 1$, we have $p_1^* > \frac{1}{2}$ and so π^* cannot be recovered from π^*_{merl}, even using the mode action $a_1 = \arg\max_a \pi^*_{\text{merl}}(a|s_0)$. The degree to which the MAP policy varies from the optimal unregularised policy depends on both the value of c and k_1, the later controlling the number of states with sub-optimal reward. Our counterexample illustrates that when there are large regions of the state-space with sub-optimal reward, the temperature must be comparatively small to compensate, hence algorithms derived from MERLIN become very sensitive to temperature. As we discuss Section 6.3.3, another drawback to MERLIN its its reliance on a variational distribution $q^{\pi_\theta}(\tau)$ to approximate the underlying dynamics of the MDP for entire trajectories; for complex domains, we hypothesise that the expressiveness of the variational distribution $q^{\pi_\theta}(\tau)$ is a bottleneck to performance. We provide evidence for this claim in Section 6.6. Finally, many existing models are defined for finite horizon problems [131, 176]. While it is possible to discount and extend MERLIN to infinite horizon problems, doing so is nontrivial and can alter the objective [215, 85].

6.2.3 Pseudo-Likelihood Methods

A related but distinct approach is to apply Jensen's inequality directly to the RL objective J^π. Firstly, we rewrite Eq. (6.1) as an expectation over τ to obtain $J = \mathbb{E}_{h\sim\rho(s)\pi(a|s)}\left[Q^\pi(h)\right] = \mathbb{E}_{\tau\sim p(\tau)}\left[R(\tau)\right]$, where $R(\tau) = \sum_{t=0}^{T-1} \gamma^t r_t$ and $p(\tau) = \rho(s_0)\pi(a_0|s_o)\prod_{t=0}^{T-1} p(h_{t+1}|h_t)$. We then treat $p(R,\tau) = R(\tau)p(\tau)$ as a joint distribution, and if rewards are positive and bounded, Jensen's inequality can be applied, enabling the derivation of an evidence lower bound (ELBO). Inference

algorithms such as EM can then be employed to find a policy that optimises the pseudo-likelihood objective [48, 222, 168, 87, 155, 1]. Pseudo-likelihood methods can also be extended to a model-based setting by defining a prior over the environment's transition dynamics. [66] demonstrate that the posterior over all possible environment models can be integrated over to obtain an optimal policy in a Bayesian setting.

Many pseudo-likelihood methods minimise $\mathrm{KL}(p_{\mathcal{O}} \parallel p_\pi)$, where p_π is the policy to be learnt and $p_{\mathcal{O}}$ is a target distribution monotonically related to reward [131]. Classical RL methods minimise $\mathrm{KL}(p_\pi \parallel p_{\mathcal{O}})$. The latter encourages learning a mode of the target distribution, while the former encourages matching the moments of the target distribution. If the optimal policy can be represented accurately in the class of policy distributions, optimisation converges to a global optimum and the problem is fully observable, the optimal policy is the same in both cases. Otherwise, the pseudo-likelihood objective reduces the influence of large negative rewards, encouraging risk-seeking policies.

6.3 VIREL

Before describing our framework, we state some relevant assumptions.

Definition 6.1 (Unique Maximum and Locally Smooth Function)**.** *Let $f : \mathcal{X} \to \mathcal{Y}$ be a function with a unique maximum $f(x^*) = \sup_x f$ and a bounded domain $\mathcal{X}$ and range $\mathcal{Y}$. Let f be locally smooth about x^*, that is $\exists\ \Delta > 0\ s.t. f(x) \in \mathbb{C}^2\ \forall\ x \in \{x | \|x - x^*\| < \Delta\ \}$.*

Assumption 6.1. *The optimal action-value function for the reinforcement learning problem is finite and strictly positive, i.e. $0 < Q^*(h) < \infty\ \forall\ h \in \mathcal{H}$.*

Any MDP for which rewards are lower bounded and finite, that is $R \subset [r_{\min}, \infty)$ satisfies Assumption 6.1. To see this, we can construct a new MDP by adding $r_{\min}$

to the reward function, ensuring that all rewards are positive and hence the optimal action-value function for the reinforcement learning problem is finite and strictly positive. This does not affect the optimal solution. Now we introduce a function approximator $\hat{Q}_\omega(h) \approx Q^\pi(h)$ parametrised by $\omega \in \Omega$.

Assumption 6.2 (Exact Representability Under Optimisation)**.** *Our function approximator can represent the optimal Q-function, i.e.,* $\exists\ \omega^* \in \Omega$ *s.t.* $Q^*(\cdot) = \hat{Q}_{\omega^*}(\cdot)$.

In Appendix D.5.1, we extend the work of [25] to continuous domains, demonstrating that Assumption 6.2 can be neglected if projected Bellman operators are used.

Assumption 6.3 (Local Smoothness of Q-functions)**.** *For* ω^* *parametrising* $Q^*(h)$ *in Assumption 6.2,* $Q_{\omega^*}(h)$ *has a unique maximum and is locally smooth under Definition 6.1 for actions in any state.*

This assumption is formally required for the strict convergence of a Boltzmann to a Dirac-delta distribution and, as we discuss in Appendix D.5.4, is of more mathematical than practical concern.

6.3.1 Objective Specification

We now define an objective that we motivate by satisfying three desiderata: ① in the limit of maximising our objective, a deterministic optimal policy can be recovered and the optimal Bellman equation is satisfied by our function approximator, ② when our objective is not maximised, stochastic policies can be recovered that encourage effective exploration of the state-action space and ③ our objective permits the application of powerful and tractable optimisation algorithms from variational inference that optimise the risk-neutral form of KL divergence $\mathrm{KL}(p_\pi \parallel p_\mathcal{O})$ introduced in Section 6.2.3.

Firstly, we define the residual error $\varepsilon_\omega \coloneqq \frac{c}{p}\|\mathcal{T}_\omega \hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p$ where $\mathcal{T}_\omega = \mathcal{T}^{\pi_\omega}\bullet \coloneqq$

$r(h) + \gamma \mathbb{E}_{h' \sim p(s'|h)\pi_\omega(a'|s')}[\cdot]$ is the Bellman operator for the Boltzmann policy with temperature ε_ω:

$$\pi_\omega(a|s) \coloneqq \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}. \tag{6.3}$$

We assume $p = 2$ and $c = \frac{1}{|\mathcal{H}|}$ without loss of generality. Our main result in Theorem 6.2 proves finding ω^* that reduces the residual error to zero, i.e. $\varepsilon_{\omega^*} = 0$, is a sufficient condition for learning an optimal Q-function $\hat{Q}_{\omega^*}(h) = Q^*(h)$. Additionally, the Boltzmann distribution $\pi_\omega(a|s)$ tends towards a Dirac-delta distribution $\pi_\omega(a|s) = \delta(a = \arg\max'_a \hat{Q}_{\omega^*}(a', s))$ whenever $\varepsilon_\omega \to 0$ (see Theorem 6.1), which is an optimal policy. The simple objective $\arg\min(\mathcal{L}(\omega)) \coloneqq \arg\min(\varepsilon_\omega)$ therefore satisfies ①. Moreover, when our objective is not minimised, we have $\varepsilon_\omega > 0$ and from Eq. (6.3) we see that $\pi_\omega(a|s)$ is non-deterministic *for all non-optimal* ω. $\mathcal{L}(\omega)$ therefore satisfies ② as any agent following $\pi_\omega(a|s)$ will continue exploring until the RL problem is solved. To generalise our framework, we extend $\mathcal{T}_\omega \cdot$ to any operator from the set of target operators $\mathcal{T}_\omega \cdot \in \mathbb{T}$ in Definition 6.2:

Definition 6.2 (Target Operator Set)**.** *Define* $\mathbb{T}$ *to be the set of target operators such that an optimal Bellman operator for* $\hat{Q}_\omega(h)$ *is recovered when the Boltzmann policy in Eq.* (6.3) *is greedy with respect to* $\hat{Q}_\omega(h)$*, i.e.,* $\mathbb{T} \coloneqq \{\mathcal{T}_\omega \cdot \mid \lim_{\varepsilon_\omega \to 0} \pi_\omega(a|s) \implies \mathcal{T}_\omega \hat{Q}_\omega(h) = \mathcal{T}^* \hat{Q}_\omega(h)\}$.

As an illustration, we prove in Appendix D.2 that the Bellman operator $\mathcal{T}^{\pi_\omega} \cdot$ introduced above is a member of $\mathbb{T}$ and can be approximated by several well-known RL targets. We also discuss how $\mathcal{T}^{\pi_\omega} \cdot$ induces a constraint on Ω. As we show in Section 6.3.2, there exists an ω in the constrained domain that maximises the RL objective under these conditions, so any optimal solution is always feasible. Moreover, we prove in Appendix D.4.3 that $\nabla_\omega \varepsilon_\omega$ still has an analytic solution, facilitating gradient-based optimisation. By definition, the optimal Bellman operator

$\mathcal{T}^{*\bullet}$ is a member of $\mathbb{T}$ and does not constrain Ω. We discuss another member of $\mathbb{T}$ that does not constrain ω in Appendix D.5.2. Soft Bellman operators are not members of $\mathbb{T}$ as the optimal policy under J^{π}_{merl} is not deterministic.

One problem remains: calculating the normalisation constant to sample directly from the Boltzmann distribution in Eq. (6.3) is intractable for many MDPs and function approximators. As such, we look to variational inference to learn an approximate *variational policy* $\pi_\theta(a|s) \approx \pi_\omega(a|s)$, parametrised by $\theta \in \Theta$ with finite variance and the same support as $\pi_\omega(a|s)$. This suggests optimising a new objective that penalises $\pi_\theta(a|s)$ when $\pi_\theta(a|s) \neq \pi_\omega(a|s)$ but still has a global maximum at $\varepsilon_\omega = 0$. A tractable objective that meets these requirements is the evidence lower bound (ELBO) on the unnormalised potential of the Boltzmann distribution, defined as $\{\omega^*, \theta^*\} \in \arg\max_{\omega,\theta} \mathcal{L}(\omega, \theta)$,

$$\mathcal{L}(\omega, \theta) \coloneqq \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] + \mathcal{H}(\pi_\theta(a|s))\right], \tag{6.4}$$

where our variational distribution $q_\theta(h) \coloneqq d(s)\pi_\theta(a|s)$, $\mathcal{H}(\bullet)$ denotes the differential entropy of a distribution and $d(s)$ is any arbitrary sampling distribution with support over $\mathcal{S}$. From Eq. (6.4), maximising our objective with respect to ω is achieved when $\varepsilon_\omega \to 0$ and hence $\mathcal{L}(\omega, \theta)$ satisfies ① and ②. As we show in Lemma 6.1, $\mathcal{H}(\cdot)$ in Eq. (6.4) causes $\mathcal{L}(\omega, \theta) \to -\infty$ whenever $\pi_\theta(a|s)$ is a Dirac-delta distribution for all $\varepsilon_\omega > 0$. This means our objective heavily penalises premature convergence of our variational policy to greedy Dirac-delta policies except under optimality. We discuss a probabilistic interpretation of our framework in Appendix D.1, where it can be shown that $\pi_\omega(a|s)$ characterises our model's uncertainty in the optimality of $\hat{Q}_\omega(h)$.

We now motivate $\mathcal{L}(\omega, \theta)$ from an inference perspective: in Appendix D.3.1, we write $\mathcal{L}(\omega, \theta)$ in terms of the log-normalisation constant of the Boltzmann distribution and the KL divergence between the action-state normalised Boltzmann distribution

$p_\omega(h)$ and the variational distribution $q_\theta(h)$:

$$\mathcal{L}(\omega, \theta) = \ell(\omega) - \mathrm{KL}(q_\theta(h) \parallel p_\omega(h)), \tag{6.5}$$

$$\text{where} \quad \ell(\omega) \coloneqq \log \int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) dh, \quad p_\omega(h) \coloneqq \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) dh}.$$

As the KL-divergence in Eq. (6.5) is always positive, maximising our objective for θ always reduces the KL-divergence between $\pi_\omega(a|s)$ and the variational approximation $\pi_\theta(a|s)$ for any $\varepsilon_\omega > 0$, with $\pi_\theta(a|s) = \pi_\omega(a|s)$ achieved under exact representability (see Theorem 6.3). This yields a tractable way to estimate $\pi_\omega(a|s)$ at any point during our optimisation procedure by maximising $\mathcal{L}(\omega, \theta)$ for θ. From Eq. (6.5) we see that our objective satisfies ③, as we minimise the mode-seeking direction of KL-divergence $\mathrm{KL}(q_\theta(h) \parallel p_\omega(h))$ and our objective is an ELBO, which is the starting point for inference algorithms [105, 20, 64]. When the RL problem is solved and $\varepsilon_\omega = 0$, our objective tends towards ∞ for *any* variational distribution that is non-deterministic (see Lemma 6.1). This is of little consequence however, as whenever $\varepsilon_\omega = 0$, our approximator is the optimal value function $\hat{Q}_{\omega^*}(h) = Q^*(h)$ (Theorem 6.2) and hence $\pi^*(a|s)$ can be inferred exactly by finding $\max_{a'} \hat{Q}_{\omega^*}(a', s)$ or by using the policy gradient $\nabla_\theta \mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_{\omega^*}(h)\right]$ (see Section 6.4.2).

6.3.2 Theoretical Results

We now formalise the intuition behind ①-③. Theorem 6.1 establishes the emergence of a Dirac-delta distribution in the limit $\varepsilon_\omega \to 0$. To the authors' knowledge, this is the first rigorous proof of this result. Theorem 6.2 shows that finding an optimal policy that maximises the RL objective in Eq. (6.1) reduces to finding the Boltzmann distribution associated with the parameters $\omega^* \in \arg\max_\omega \mathcal{L}(\omega, \theta)$. The existence of such a distribution is a sufficient condition for the policy to be optimal. Theorem 6.3 shows that whenever $\varepsilon_\omega > 0$, maximising our objective for θ always reduces the

KL-divergence between $\pi_\omega(a|s)$ and $\pi_\theta(a|s)$, providing a tractable method to infer the current Boltzmann policy.

Theorem 6.1 (Convergence of Boltzmann Distribution to Dirac Delta)**.** *Let* $p_\varepsilon : \mathcal{X} \to [0,1]$ *be a Boltzmann distribution with temperature* $\varepsilon \in \mathbb{R}_{\geq 0}$, $p_\varepsilon(x) = \frac{\exp\left(\frac{f(x)}{\varepsilon}\right)}{\int_{\mathcal{X}} \exp\left(\frac{f(x)}{\varepsilon}\right) dx}$, *where* $f : \mathcal{X} \to \mathcal{Y}$ *is a function that satisfies Definition 6.1. In the limit* $\varepsilon \to 0$, $p_\varepsilon(x) \to \delta(x = \sup_{x'} f(x'))$.

Proof. See Appendix D.3.2 □

Lemma 6.1 (Lower and Upper limits of $\mathcal{L}(\omega, \theta)$)**.** *i) For any* $\varepsilon_\omega > 0$ *and* $\pi_\theta(a|s) = \delta(a^*)$, *we have* $\mathcal{L}(\omega, \theta) = -\infty$. *ii) For* $\hat{Q}_\omega(h) > 0$ *and any non-deterministic* $\pi_\theta(a|s)$, $\lim_{\varepsilon_\omega \to 0} \mathcal{L}(\omega, \theta) = \infty$.

Proof. See Appendix D.3.3. □

Theorem 6.2 (Optimal Boltzmann Distributions as Optimal Policies)**.** *For* ω^* *that maximises* $\mathcal{L}(\omega, \theta)$ *defined in Eq.* (6.4), *the corresponding Boltzmann policy induced must be optimal, i.e.,* $\{\omega^*, \theta^*\} \in \arg\max_{\omega,\theta} \mathcal{L}(\omega, \theta) \implies \pi_{\omega^*}(a|s) \in \Pi^*$.

Proof. See Appendix D.3.3. □

Theorem 6.3 (Maximising the ELBO for θ)**.** *For any* $\varepsilon_\omega > 0$, $\max_\theta \mathcal{L}(\omega, \theta) = \mathbb{E}_{d(s)}\left[\min_\theta \mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s))\right]$ *with* $\pi_\omega(a|s) = \pi_\theta(a|s)$ *under exact representability.*

Proof. See Appendix D.3.4. □

6.3.3 Comparing VIREL and MERLIN Frameworks

To compare MERLIN and VIREL, we consider the probabilistic interpretation of the two models discussed in Appendix D.1; introducing a binary variable $\mathcal{O} \in \{0, 1\}$ defines a graphical model for our inference problem whenever $\varepsilon_\omega > 0$. Comparing

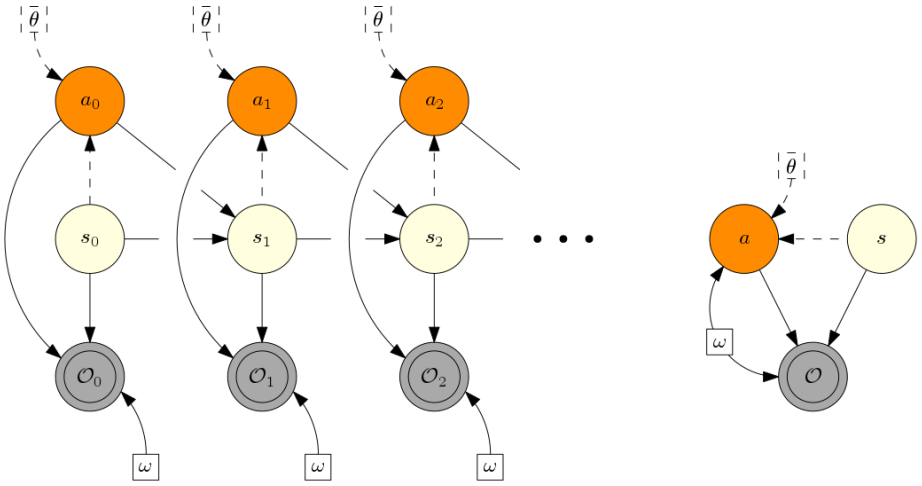


Figure 6.2: Graphical models for MERLIN and VIREL (variational approximations are dashed)

the graphs in Fig. 6.2, we hypothesise that since MERLIN models entire trajectories, its variational distribution takes the onus of representing future dynamics of the system. By contrast, VIREL's variational policy only needs to model a single step and a function approximator is used to model future dynamics, which is more expressive and better suited to capturing essential modes than a parametrised distribution. This effect becomes more pronounced in higher dimensions, which are typically multi-modal. We provide empirical evidence supporting our hypothesis in Section 6.6.

Theorem 6.1 demonstrates that, unlike in MERLIN, VIREL naturally learns optimal deterministic policies directly from the optimisation procedure while still maintaining the benefits of stochastic policies in training. While Boltzmann policies with fixed temperatures have been proposed before [180], as we discuss in Appendix D.1, the adaptive temperature ε_ω in VIREL's Boltzmann policy has a unique interpretation, characterising the model's uncertainty in the optimality of $\hat{Q}_\omega(h)$; both $\pi_\omega(a|s)$ and its variational approximation $\pi_\theta(a|s)$ have an adaptive variance that reduces as $\hat{Q}_\omega(h) \to Q^*(h)$, allowing us to benefit from uncertainty-driven exploration when sampling under $\pi_\theta(a|s)$.

6.4 Actor-Critic and EM

We now apply the expectation-maximisation (EM) algorithm [50, 81] to optimise our objective $\mathcal{L}(\omega, \theta)$. (See Section 2.8 for an exposition of this algorithm). In keeping with RL nomenclature, we refer to $\hat{Q}_\omega(h)$ as the *critic* and $\pi_\theta(a|s)$ as the *actor*. We establish that the expectation (E-) step is equivalent to carrying out policy improvement and the maximisation (M-)step to policy evaluation. This formulation reverses the situation in most pseudo-likelihood methods, where the E-step is related to policy evaluation and the M-step is related to policy improvement, and is a direct result of optimising the forward KL-divergence $\mathrm{KL}(q_\theta(h) \parallel p_\omega(h|\mathcal{O}))$ as opposed to the reverse KL-divergence used in pseudo-likelihood methods. As discussed in Section 6.2.3, this mode-seeking objective prevents the algorithm from learning risk-seeking policies. We now introduce an extension to Assumption 6.2 that is sufficient to guarantee convergence.

Assumption 6.4 (Universal Variational Representability)**.** *Every Boltzmann policy can be represented as $\pi_\theta(a|s)$, i.e. $\forall\ \omega \in \Omega\ \exists\ \theta \in \Theta$ s.t. $\pi_\theta(a|s) = \pi_\omega(a|s)$.*

Assumption 6.4 is strong but, like in variational inference, our variational policy $\pi_\theta(a|s)$ provides a useful approximation when Assumption 6.4 does not hold. As we discuss in Appendix D.5.1, using projected Bellman errors also ensures that our M-step always converges no matter what our current policy is.

6.4.1 Variational Actor-Critic

In the E-step, we keep the parameters of our critic ω_k constant while updating the actor's parameters by maximising the ELBO with respect to θ: $\theta_{k+1} \leftarrow \arg\max_\theta \mathcal{L}(\omega_k, \theta)$. Using gradient ascent with step size α_{actor}, we optimise $\varepsilon_{\omega_k}\mathcal{L}(\omega_k, \theta)$ instead, which prevents ill-conditioning and does not alter the optimal solution, yielding the update (see Appendix D.4.1 for full derivation):

E-Step (Actor): $\theta_{i+1} \leftarrow \theta_i + \alpha_{\text{actor}} \left(\varepsilon_{\omega_k} \nabla_\theta \mathcal{L}(\omega_k, \theta)\right)|_{\theta=\theta_i}$,

$$\varepsilon_{\omega_k} \nabla_\theta \mathcal{L}(\omega_k, \theta) = \mathbb{E}_{s\sim d(s)} \left[\mathbb{E}_{a\sim\pi_\theta(a|s)} \left[\hat{Q}_{\omega_k}(h) \nabla_\theta \log \pi_\theta(a|s) \right] + \varepsilon_{\omega_k} \nabla_\theta \mathcal{H}(\pi_\theta(a|s)) \right]. \tag{6.6}$$

In the M-step, we maximise the ELBO with respect to ω while holding the parameters θ_{k+1} constant. Hence expectations are taken with respect to the variational policy found in the E-step: $\omega_{k+1} \leftarrow \arg\max_\omega \mathcal{L}(\omega, \theta_{k+1})$. We use gradient ascent with step size $\alpha_{\text{critic}}(\varepsilon_{\omega_i})^2$ to optimise $\mathcal{L}(\omega, \theta_{k+1})$ to prevent ill-conditioning, yielding (see Appendix D.4.2 for full derivation):

M-Step (Critic): $\omega_{i+1} \leftarrow \omega_i + \alpha_{\text{critic}}(\varepsilon_{\omega_i})^2 \nabla_\omega \mathcal{L}(\omega, \theta_{k+1})|_{\omega=\omega_i}$,

$$(\varepsilon_{\omega_i})^2 \nabla_\omega \mathcal{L}(\omega, \theta_{k+1}) = \varepsilon_{\omega_i} \mathbb{E}_{d(s)\pi_{\theta_{k+1}}(a|s)} \left[\nabla_\omega \hat{Q}_\omega(h) \right] - \mathbb{E}_{d(s)\pi_{\theta_{k+1}}(a|s)} \left[\hat{Q}_{\omega_i}(h) \right] \nabla_\omega \varepsilon_\omega. \tag{6.7}$$

6.4.2 Discussion

From an RL perspective, the E-step corresponds to training an actor using a policy gradient method [205] with an adaptive entropy regularisation term [243, 151]. The M-step update corresponds to a policy evaluation step, as we seek to reduce the MSBE in the second term of Eq. (6.7). Note that the gradient of this term $\nabla_\omega \varepsilon_\omega$ depends on $(\mathcal{T}_\omega \hat{Q}_\omega(h) - \hat{Q}_\omega(h)) \nabla_\omega \mathcal{T}_\omega \hat{Q}_\omega(h)$, which typically requires evaluating two independent expectations. For convergence guarantees, techniques such as residual gradients [11] or GTD2/TDC [25] need to be employed to obtain an unbiased estimate of this term. If guaranteed convergence is not a priority, dropping gradient terms allows us to use semi-gradient methods [202], which are often simpler to implement. We discuss these methods further in Appendix D.5.3. A key component of our algorithm

is the behaviour when $\varepsilon_{\omega^*} = 0$; under this condition there is no M-step update (both $\varepsilon_{\omega_k} = 0$ and $\nabla_\omega \varepsilon_\omega = 0$) and $Q_{\omega^*}(h) = Q^*(h)$ (see Theorem 6.2), so our E-step reduces exactly to a policy gradient step, $\theta_{k+1} \leftarrow \theta_k + \alpha_{\text{actor}} \mathbb{E}_{h \sim d(s)\pi_\theta(a|s)} [Q^*(h) \nabla_\theta \log \pi_\theta(a|s)]$, recovering the optimal policy $\pi_\theta(a|s) \to \pi^*(a|s)$ in the limit of convergence.

From an inference perspective, the E-step improves the parameters of our variational distribution to reduce the gap between the current Boltzman posterior and the variational policy, $\mathrm{KL}(\pi_\theta(a|s)) \parallel \pi_{\omega_k}(a|s))$ (see Theorem 6.3). This interpretation makes precise the intuition that how much we can improve our policy is determined by how similar $\hat{Q}_{\omega_k}(h)$ is to $Q^*(h)$, limiting policy improvement to the complete E-step: $\pi_{\theta_{k+1}}(a|s) = \pi_{\omega_k}(a|s)$. We see that the common greedy policy improvement step, $\pi_{\theta_{k+1}}(a|s) = \delta(a \in \arg\max_{a'}(\hat{Q}_{\omega_k}(a', s)))$ acts as an approximation to the Boltzmann form in Eq. (6.3), replacing the softmax with a hardmax.

If Assumption 6.4 holds and any constraint induced by $\mathcal{T}_\omega \cdot$ does not prevent convergence to a complete E-step, the EM algorithm alternates between two convex optimisation schemes, and is guaranteed to converge to at least a local optimum of the $\mathcal{L}(\omega, \theta)$ [244]. In reality, we cannot carry out complete E- and M-steps for complex domains, and our variational distributions are unlikely to satisfy Assumption 6.4. Under these conditions, we can resort to the empirically successful variational EM algorithm [105], carrying out partial E- and M-steps instead, which we discuss further in Appendix D.5.3.

6.4.3 Advanced Actor-Critic Methods

A family of actor-critic algorithms follows naturally from our framework: 1) we can use powerful inference techniques such as control variates [75] or variance-reducing baselines by subtracting any function that does not depend on the action [182], e.g., $V(s)$, from the action-value function, as this does not change our objective, 2) we can manipulate Eq. (6.6) to obtain variance-reducing gradient estimators such as

EPG [42], FPG [56], and SVG0 [92], and 3) we can take advantage of $d(s)$ being any general decorrelated distribution by using replay buffers [153] or empirically successful asynchronous methods that combine several agents' individual gradient updates at once [151]. As we discuss in Appendix D.4.4, the manipulation required to derive the estimators in 2) is not strictly justified in the classic policy gradient theorem [205] and MERL formulation [85].

MPO is a state-of-the-art EM algorithm derived from the pseudo-likelihood objective [1]. In its derivation, policy evaluation does not naturally arise from either of its EM steps and must be carried out separately. As we demonstrate in Appendix D.6, under the probabilistic interpretation of our model, including a prior of the form $p_\phi(h) = \mathcal{U}(s)\pi_\phi(a|s)$ in our ELBO and specifying a hyper-prior $p(\omega)$, the MPO objective with an adaptive regularisation constant can be recovered from VIREL:

$$\mathcal{L}^{\text{MPO}}(\omega, \theta, \phi) = \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] - \text{KL}(\pi_\theta(a|s) \parallel \pi_\phi(a|s))\right] + \log p(\omega).$$

We also show in Appendix D.6 that applying the (variational) EM algorithm from Section 6.4 yields the MPO updates with the missing policy evaluation step.

6.5 Using alternate inference frameworks

We now discuss how we can use alternate approximate inference methods, other than variational inference under the VIREL framework. Note that our method differs from most existing methods in that the variable $\mathcal{O} = 1$ is understood to be the event that the agent is acting *softly* optimally [131, 220]. As we are using function approximators in VIREL, we interpret $\mathcal{O} = 1$ as the event that the agent is behaving optimally with respect to the given $\hat{Q}_\omega(h)$. The likelihood for $\mathcal{O} = 1$ under VIREL is

given by:

$$p_\omega(\mathcal{O}=1|h) = \exp\left(\frac{\hat{Q}_\omega(h) - \max_{a'} \hat{Q}_\omega(a', s)}{\varepsilon_\omega}\right), \tag{6.8}$$

Thus our likelihood encodes for a *soft* greedy-Q operator (with varying temperature given by the bellman error) instead of *soft* reward maximizer. Hence it can be seen as a form of soft policy iteration. Further, the likelihood is dependent on ω. Maximum a posteriori(MAP) is guaranteed to improve the policy under our framework (as it corresponds to policy iteration which provably converges to optimal [201]). Our action posterior is given by (see Appendix D.1 for details):

$$p_\omega(a|s, \mathcal{O}=1) = \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}$$

Thus, the action-posterior is exactly the Boltzmann policy introduced in Section 6.3.1 (see Eq. (6.3)). From a Bayesian perspective, the action-posterior $p_\omega(a|s, \mathcal{O}=1)$ characterises the uncertainty we have in deducing the optimal action for a given state s under $\hat{Q}_\omega(h)$. One way to obtain different RL algorithms under VIREL model is to also incorporate uncertainty of the Q approximator under a probabilistic framework, for instance we can modify Eq. (6.8) by multiplicative functional forms which are inversely proportional to variance of Q estimates.

Finding policies satisfying Eq. (6.3) is a constrained optimization problem due to policy bellman operator $\mathcal{T}^{\pi_\omega}$ appearing in the policy expression in the temperature term $\varepsilon_\omega \coloneqq \frac{c}{p}\|\mathcal{T}_\omega \hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p$, thus giving rise to recursive definition. Even when the MDP is known, this is in general a hard problem to solve. Use of the variational inference framework as described in this work helps neatly separate the recursive constraint. However, we can also use other approximate inference techniques like expectation propagation (EP), Markov chain monte carlo (MCMC), cross entropy methods (CEM) towards the goal of sampling from the optimal policy, with some

modifications as we discuss below.

The special form of our likelihood and consequently the action posterior means that we can recover optimal policy by forcing the following additional constraint on our action posterior Eq. (6.3): the function approximator $\hat{Q}$ accurately approximates the action-value function of π_ω (mathematically the constraint is equivalent to requiring $\epsilon_\omega = 0$). This technique works because the policy becomes greedy with respect to its own value function and thus must represent the optimal policy ([201]). It also acts as a stopping criteria for the iterative application of the operator induced by VIREL. We use this property in the VIREL framework while deriving the variational lower bound for our expectation maximization framework (see Appendix D.1.1).

In principle, it is also possible to use alternate approximate inference methods like expectation proposition (EP) which minimizes the reversed form of the KL $\theta^* \in \arg\min_\theta \mathrm{KL}(p_\omega(h|\mathcal{O}) \parallel q_\theta(h))$, although the algorithms so derived may suffer from mean capturing effects and risk seeking behaviour as previously discussed for pseudo likelihood methods (Section 6.2.3). They however would also be able to separate the recursive constraint introduced in Eq. (6.3).

Methods like MCMC which attempt to directly sample for an optimal policy would require inclusion of constraints in an online fashion [68]. While they may be simpler to execute, the computational and sample complexity of such approaches is likely to be much higher. Further, the online computation of the constraints can lead to improper exploration and instabilities arising from not having discovered large parts of the state action space. One way to deal with the recursive policy constraint Eq. (6.3) for such methods would be to combine them with alternate optimization methods [23, 161]. In such a scheme, we would alternate between fixing a temperature target and ϵ_{target} computing the policy π_ω, this can also ameliorate exploration issues while using such methods and can help obtain robust estimate of the underlying dynamics when combined with a model based approaches. In a similar vein, cross

entropy methods can also be modified for directly sampling from the boltzmann class used in the VIREL model with the optimality constraint $\epsilon_\omega = 0$. [240] demonstrate how constrained cross entropy methods can be used in the RL control setting which can be adapted to sampling from the constrained action posterior class in VIREL. Evaluating the effectiveness of these approaches is a promising future direction.

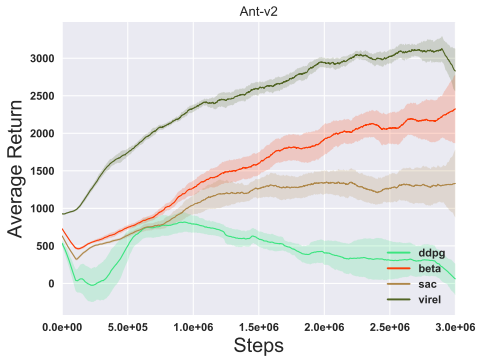
6.6 Experiments

The aim of our experimental evaluation is threefold: Firstly, as explained in Section 6.3.1, algorithms using soft value functions cannot be recovered from VIREL. We therefore demonstrate that using hard value functions does not harm performance. Secondly, we provide evidence for our hypothesis introduced in Section 6.3.3 that using soft value functions can harm performance in higher dimensional tasks. Thirdly, we show that even under all practical approximations discussed in Appendix D.5.3, the algorithm derived in Section 6.4 still outperforms advanced actor-critic methods.

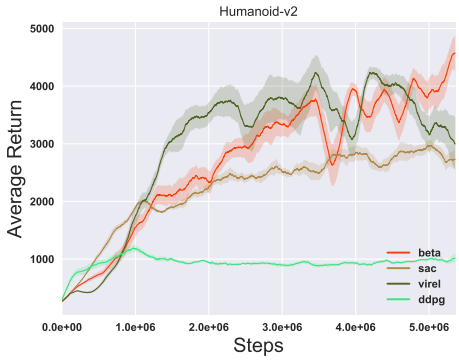
We compare our methods to the state-of-the-art SAC* and DDPG [135] algorithms on MuJoCo tasks in OpenAI gym [32] and in rllab [54]. We use SAC as a baseline because [85] show that it outperforms PPO [185], Soft Q-Learning [84], and TD3 [65]. We compare to DDPG [135] because, like our methods, it can learn deterministic optimal policies. We consider two variants: in the first one, called *virel*, we keep the scale of the entropy term in the gradient update for the variational policy constant α; in the second, called *beta*, we use an estimate $\hat{\varepsilon}_\omega$ of ε_ω to scale the corresponding term in Eq. (D.15). We compute $\hat{\varepsilon}_\omega$ using a buffer to draw a fixed number of samples N_ε for the estimate.

To adjust for the relative magnitude of the first term in Eq. (D.15) with that of ε_ω

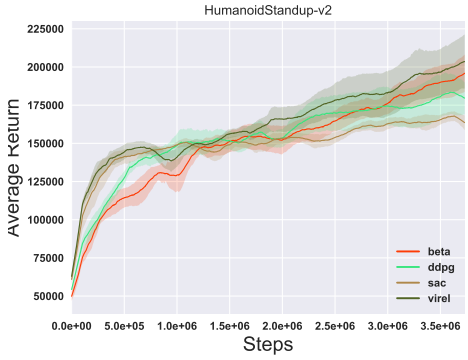
*We use implementations provided by the authors `https://github.com/haarnoja/sac` for v1 and `https://github.com/vitchyr/rlkit` for v2.



(a) Ant



(b) Humanoid



(c) Humanoid standup

Figure 6.3: Training curves on continuous control benchmarks gym-Mujoco-v2 : High dimensional domains

scaling the second term, we also multiply the estimate $\hat{\varepsilon}_\omega$ by a scalar $\lambda \approx \frac{1-\gamma}{r_{avg}}$, where r_{avg} is the average reward observed; λ^{-1} roughly captures the order of magnitude of the first term and allows $\hat{\varepsilon}_\omega$ to balance policy changes between exploration and exploitation. We found performance is poor and unstable without λ. To reduce variance, all algorithms use a value function network $V(\phi)$ as a baseline and a Gaussian policy, which enables the use of the reparametrisation trick. Pseudocode can be found in Appendix D.7. All experiments use 5 random initialisations and parameter values are given in Appendix D.8.1.

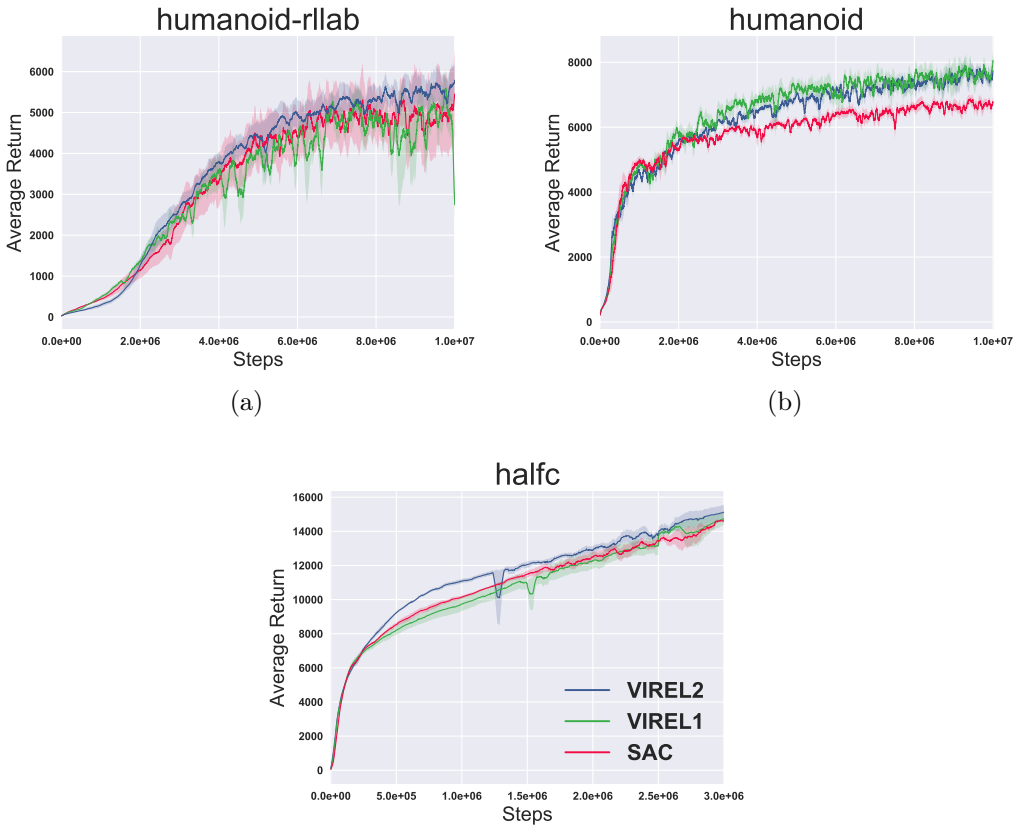
Fig. 6.3 gives the training curves for the various algorithms on high dimensional tasks for on gym-mujoco-v2. In particular, in Humanoid-v2 (action space dimensionality: 17, state space dimensionality: 376) and Ant-v2 (action space dimensionality: 8,

state space dimensionality: 111), DDPG fails to learn any reasonable policy. We believe that this is because the Ornstein-Uhlenbeck noise that DDPG uses for exploration is insufficiently adaptive in high dimensions. While SAC performs better, *virel* and *beta* still substantially outperform it. As hypothesised in Section 6.3.3, we believe that this performance advantage arises because variational policies of VIREL methods do not need to model trajectories and are better predisposed to capturing the multiple modes that are common in higher dimensions. All algorithms learn optimal policies in simple domains, the training curves for which can be found in Fig. D.4 in Appendix D.8.3. Thus as the state-action dimensionality increases, algorithms derived from VIREL outperform SAC and DDPG.

[65] and [228] note that using the minimum of two randomly initialised action-value functions helps mitigate the positive bias introduced by function approximation in policy gradient methods. Therefore, a variant of SAC uses two soft critics. We compare this variant of SAC to two variants of *virel*: *virel1*, which uses two hard Q-functions and *virel2*, which uses one hard and one soft Q-function. We scale the rewards so that the means of the Q-function estimates in *virel2* are approximately aligned. Fig. 6.4 shows the training curves on three gym-Mujoco-v1 domains, with additional plots shown in Fig. D.3 in Appendix D.8.2. Again, the results demonstrate that *virel1* and *virel2* perform on par with SAC in simple domains like Half-Cheetah and outperform it in challenging high dimensional domains like humanoid-gym and -rllab (17 and 21 dimensional action spaces, 376 dimensional state space).

6.7 Conclusion

This work presented VIREL, a novel framework that recasts the reinforcement learning problem as an inference problem using function approximators. We explored the strong theoretical justifications for this framework and compared two simple actor-critic algorithms that arise naturally from applying variational EM on the objective.



(a) (b) (c)

Figure 6.4: Training curves on continuous control benchmarks gym-Mujoco-v1.

Extensive empirical evaluation shows that our algorithms perform on par with the current state of the art on simple domains and substantially outperform them on challenging high dimensional domains.

Part III

Learning to generalize across observation shifts

Chapter 7

Conditional Bisimulation: Generalization to observation shifts

Contents

7.1 Introduction

We now turn our attention to a large state-observation space RL problem: one that admits a special contextual structure in the observation space. Many practical scenarios in reinforcement learning (RL) applications require the agent to be robust to changes in the observations space between training and deployment. These changes can occur due to various practical errors and constraints under which autonomous agents need to be deployed (e.g. variations in sensor position and fitting on automobiles, change in calibration settings of visual input, change in sensor types due to upgrades, calibration changes due to wear and tear etc.). However, existing RL algorithms hardly address this issue. Further, the presence of task irrelevant noise in the environment make it even more difficult for the agent to generalise across the changes in observation space and ignore task irrelavant noise. In this chapter we propose a solution to the aforementioned problem using conditional bisimulation and leveraging the applicability of simulator/specialized setup during train time which help explicitly teach the agent the similarities across changes in observation space. Our methods offers two-fold advantage:

- We can learn representations which are robust to shifts in observation space
- We learn to ignore task irrelavant features as our metric is grounded in rewards

7.2 Background

Our starting point would be the Markov decision process(MDP) formulation (Chapter 2).

7.2.1 Equivalence relations and classes

We first briefly mention some of the concepts from abstract algebra used in motivating state similarity in MDPs. There after, we review state abstractions and metrics for

state similarity followed by an outline of the rich observation parameter context for our setting.

Definition 7.1. *A binary relation $\mathcal{R}$ on a set $\mathcal{S}$ is given by $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$*

Definition 7.2. *$\mathcal{R}$ is symmetric if $\mathcal{R}(a,b) \implies \mathcal{R}(b,a)$*

Definition 7.3. *$\mathcal{R}$ is reflexive if $\mathcal{R}(a,a), \forall a \in \mathcal{S}$*

Definition 7.4. *$\mathcal{R}$ is transitive if $\mathcal{R}(a,b) \wedge \mathcal{R}(b,c) \implies \mathcal{R}(a,c)$*

Definition 7.5. *$\mathcal{R}$ is equivalence if its reflexive, symmetric and transitive.*

Definition 7.6. *$\mathcal{P} \triangleq \{\mathcal{C}_i\}$ is a partition of a set $\mathcal{S}$ if $\mathcal{S} = \cup_i \mathcal{C}_i$ and $\mathcal{C}_i \cap \mathcal{C}_j$ is empty if $i \neq j$.*

Definition 7.7. *If $\mathcal{R}$ is an equivalence relation on $\mathcal{S}$, then $\mathcal{S}$ can be partitioned into equivalence classes with $\mathcal{P}(\mathcal{R}, \mathcal{S}) \triangleq \{\mathcal{C}_i\}$, where $\mathcal{C}_i \subseteq \mathcal{S}, \forall a, b \in \mathcal{C}_i \implies \mathcal{R}(a,b)$ and $\mathcal{C}_i \cap \mathcal{C}_j$ is empty if $i \neq j$.*

Definition 7.8. *For partitions $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1$ is a filtrate of $\mathcal{P}_2$ if $\forall \mathcal{C}_i \in \mathcal{P}_2, \exists \mathcal{D}_j \in \mathcal{P}_1$ s.t. $\mathcal{C}_i = \cup_j \mathcal{D}_j$*

Definition 7.9. *$\mathcal{P}_c$ is the coarsest partition induced by $\mathcal{R}$ if $\forall$ valid partitions $\mathcal{P}$ under $\mathcal{R}$, $\mathcal{P}$ is a filtrate of $\mathcal{P}_c$*

7.2.2 Bisimulation

If for two states s_i and s_j, we find that for any action sequence $a_{0:\infty}$ the sequence of rewards are identical, then s_i and s_j are considered equivalent under the notion of bisimulation. Thus, it is a kind of state abstraction which groups states that are behaviorally equivalent [133]. We can equivalently state the definition recursively by stating that two states are bisimilar if they share both the same immediate reward and equivalent distributions over the next bisimilar states [125, 69].

Definition 7.10 (Bisimulation Relations [69])**.** *Given an MDP $\mathcal{M}$, an equivalence*

relation B between states is a bisimulation relation if, for all states $s_i, s_j \in S$ that are equivalent under B (denoted $s_i \equiv_B s_j$) the following conditions hold:

$$r(s_i, a) = r(s_j, a) \qquad \forall a \in U, \tag{7.1}$$

$$P(G|s_i, a) = P(G|s_j, a) \quad \forall a \in U, \quad \forall G \in S_B, \tag{7.2}$$

where S_B is the partition of S under the relation B (the set of all groups G of states equivalent under B), and $P(G|s, a) = \sum_{s' \in G} P(s'|s, a)$.

Finding the coarsest bisimulation relation is known to be an NP-hard problem [69]. Further, the exact partitioning induced from a bisimulation relation is generally impractical as it a very strict notion of equivalence and seldom leads to meaningful compression of the original MDP, this is especially true in continuous domains, where infinitesimal changes in the reward function or dynamics can break the bisimulation relation but still imply exploitable aggregation. Thus towards addressing this, Bisimulation Metrics [60, 61, 38] relaxes the concept of exact bisimulation, and instead define a pseudometric space (S, d), where a distance function $d : S \times S \mapsto \mathbb{R}_{\geq 0}$ measures the behavioral similarity between two states

Defining a distance metric d between states requires choosing a notion of distance between rewards (towards relaxing Eq. (7.1)), and similay a notion of distance between state transitions (towards relaxing Eq. (7.2)). Prior works use the Wasserstein metric for the latter, originally used in the context of bisimulation metrics by [227]. The Wasserstein-p metric between two probability distributions P_i and P_j is defined as $W_p(P_i, P_j; d) = \inf_{\gamma' \in \Gamma(P_i, P_j)} [\int_{S \times S} d(s_i, s_j)^p \, \mathrm{d}\gamma'(s_i, s_j)]^{1/p}$, where $\Gamma(P_i, P_j)$ is the set of all couplings of P_i and P_j. The metric has intuitive interpretations depending on the exact value of p when viewed from the dual perspective, for example $W_1(P_i, P_j; d)$ denotes the cost of transporting mass from distribution P_i to another distribution P_j where the cost is given by the distance metric d [232]. Thus is known as the earth mover distance. There are several other application of this metric in the optimal

transport theory literature. The bisimulation metric is formally defined as a convex combination of the reward difference added to the Wasserstein distance between transition distributions:

Definition 7.11 (Bisimulation Metric)**.** *From Theorem 2.6 in [60] with* $c \in [0, 1)$*:*

$$d(s_i, s_j) \;=\; \max_{a \in U} (1 - c) \cdot |r^a_{s_i} - r^a_{s_j}| + c \cdot W_1(P^a_{s_i}, P^a_{s_j}; d)$$

The above definition can also be modified to include scenarios involving stochastic rewards, where a similar metric is chosen between reward distributions. To account for state similarities arising from following a particular policy, the π-bisimulation metric [38] is similarly defined by fixing a policy π and replacing the rewards and transitions used by their policy based expectations:

$$d^\pi(s_i, s_j) \;=\; (1 - c) \cdot |r^\pi_{s_i} - r^\pi_{s_j}| + c \cdot W_1(P^\pi_{s_i}, P^\pi_{s_j}; d^\pi) \tag{7.3}$$

In this work we will consider the max entropy RL framework as it ensures a unique optimal policy π^*_{merl}*. Our goal would be to leverage generalization and transfer obtained from informing the agent representation by similarity metric Eq. (7.3) under π^*.

7.2.3 Rich observations and context

We now extend the CMDP framework Section 2.3 to our setting with a parametrized context which defines a functional transformation of the underlying MDP state giving rise to context dependent observations. Formally, we have $\mathcal{M} \triangleq \langle S, U, P, r, \gamma, \rho, \Theta, P_\Theta, Z, f \rangle$, where Θ defines a space of context parameters, P_Θ is a fixed distribution over the contexts, Z is the set of observations emitted as $f : S \times \Theta \to Z$. Thus fixing a particular context θ gives us a richly observed MDP indexed by θ: $\mathcal{M}_\theta$. We

*we will refer to it as simply π^* in this chapter for brevity

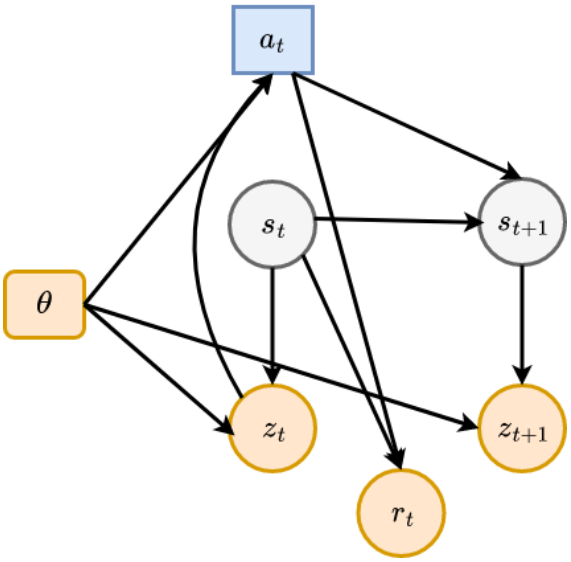


Figure 7.1: PGM for varying observation context setting

assume that the agent observes θ in our setting. Fig. 7.1 illustrates the parametrized observation setting. Without loss of generality, we assume $S \subset [0,1]^n$, $Z \subset [0,1]^l$, where typically $n << l$. We will use $f(s,\theta), f_\theta(s)$ interchangeably to highlight the corresponding (un)-curried versions of the observation function. We will be focusing on functional forms for observations, but the setting can be extended to scenarios with added independent or correlated noise at each step with suitable assumptions about identifiability [250]. .

7.3 Methodology

Algorithm 4 Robust Conditional Bisimulation (RCB)

for Time $t = 0$ to ∞ **do**
  Observe z_t, θ
  Encode observation $y_t = \phi(z_t, \theta)$
  Execute action $a_t \sim \pi(y_t)$
  Record data: $\mathcal{D} \leftarrow \mathcal{D} \cup \{z_t, a_t, z_{t+1}, r_{t+1}\}$
  Sample batch $\mathcal{B} \sim \mathcal{D}$
  Train policy: $\mathbb{E}_{\mathcal{B}}[J^\pi]$
  Train encoder using pairwise loss: $\mathcal{L}_{rep}(\phi)$ {Eq. (7.4)}
  Train dynamics: $J(\hat{P},\phi) = (\hat{P}(\phi(z_t,\theta), a_t) - y_{t+1})^2$
end for

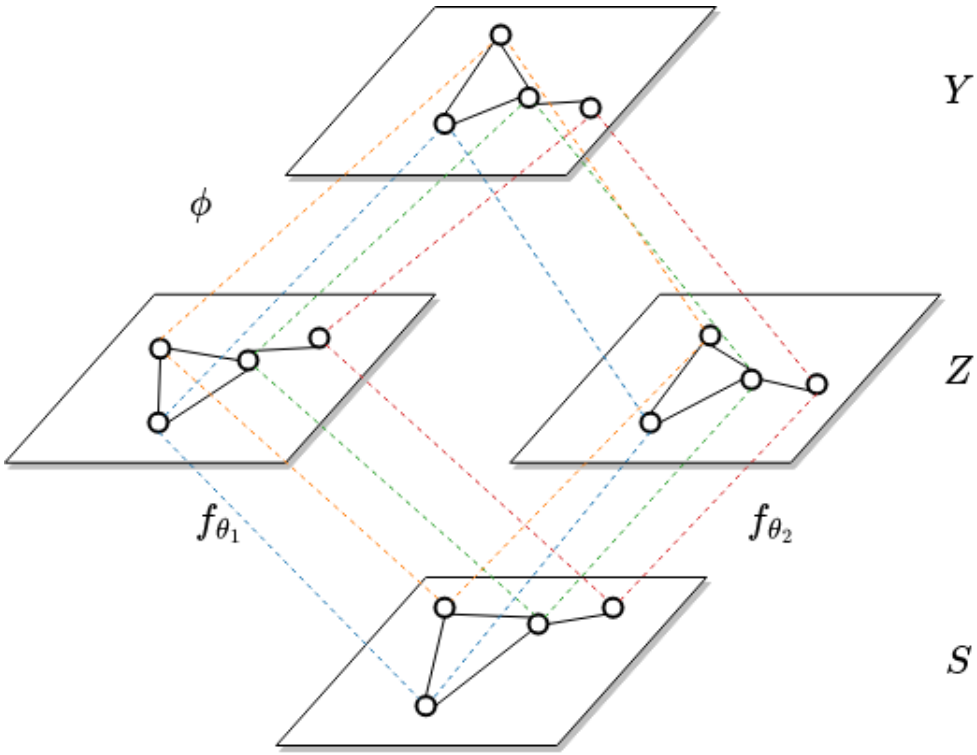


Figure 7.2: Using bisimulation to learn representation invariant to observation shifts. Hollow circles represent states in the space, solid lines depict distances in the corresponding space, dashed lines depict equivalence across spaces tied by the colour.

As previously discussed, it is important that agent policies in RL are robust to observation shifts for deployment in real world scenarios. In this work we wish to learn policies which can generalize well across the support set of the context distribution P_Θ. Towards ensuring this, we propose Robust Conditional Bisimulation (RCB) Algorithm 4, a data-efficient approach to learn control policies from unstructured, high-dimensional observations. Our goal specifically would be to learn effective representation function for the RL task set $\phi : Z \times \Theta \mapsto Y$ which enables robust learning and deployment of autonomous agents to potentially unseen observation shifts (governed by a change in θ), see Fig. 7.2. Towards this, we specify the desiderata which the representation must follow as shown in Fig. 7.3:

- **Base Bisimulation**: Given a fixed $\theta \in \Theta$, the representation should accurately preserve bisimulation distances between states, thus providing robustness to unimportant noise in observations. Concretely $\forall s_i, s_j \in \mathcal{S}$:

$$d(s_i, s_j) = d_Y(\phi(f_\theta(s_i), \theta), \phi(f_\theta(s_j), \theta))$$

  Where d_Y is a metric on Y (we use $Y = \mathbb{R}^m$ and L1 distance for our experiments).

- **Inter-context consistency (ICC)**: The representation should remain invariant under a fixed state as the context changes. Concretely: $\forall s \in \mathcal{S}$ and $\theta_1, \theta_2 \in \Theta$,

$$d_Y(\phi(f_{\theta_1}(s), \theta_1), \phi(f_{\theta_2}(s), \theta_2)) = 0$$

- **Cross consistency (CC)**: This requires that the representation distance

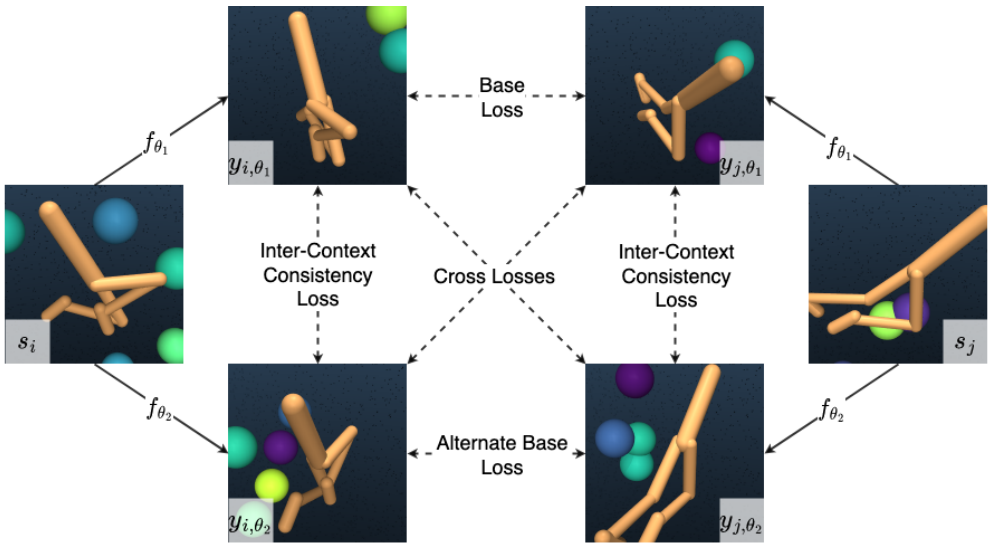


Figure 7.3: Various bisimulation losses. s represents underlying state, f_θ the observation function and y the corresponding observation.

between two states are consistent across observation shifts

$$d(s_i, s_j) = d_Y(\phi(f_{\theta_1}(s_i), \theta_1), \phi(f_{\theta_2}(s_j), \theta_2))$$
$$d(s_i, s_j) = d_Y(\phi(f_{\theta_2}(s_i), \theta_2), \phi(f_{\theta_1}(s_j), \theta_1))$$

Fig. 7.3 depicts the above representation criteria on the Mujoco control domain with 3D background objects acting as noise. We combine the above three representation conditions into a sum of squared loss components. For this we sample pairs of experiences i, j from the buffer along with base context θ_1 (chosen at episode start) and an alternate context θ_2 both sampled from P_Θ. We next compute the embedding of the underlying states under the contexts and finally compute the representation loss term as follows:

$$\begin{aligned}\mathcal{L}_{rep}(\phi) =& \lambda_{base}\big(|\overline{y}_{i,\theta_1} - \overline{y}_{j,\theta_1}|_1 - T_{i,j}\big)^2 + \\ & \lambda_{icc}\big[|\overline{y}_{i,\theta_1} - \overline{y}_{i,\theta_2}|_1^2 + |\overline{y}_{j,\theta_1} - \overline{y}_{j,\theta_2}|_1^2\big] + \\ & \lambda_{cc}\big[\big(|\overline{y}_{i,\theta_1} - \overline{y}_{j,\theta_2}|_1 - T_{i,j}\big)^2 + \big(|\overline{y}_{i,\theta_2} - \overline{y}_{j,\theta_1}|_1 - T_{i,j}\big)^2\big] \end{aligned} \tag{7.4}$$

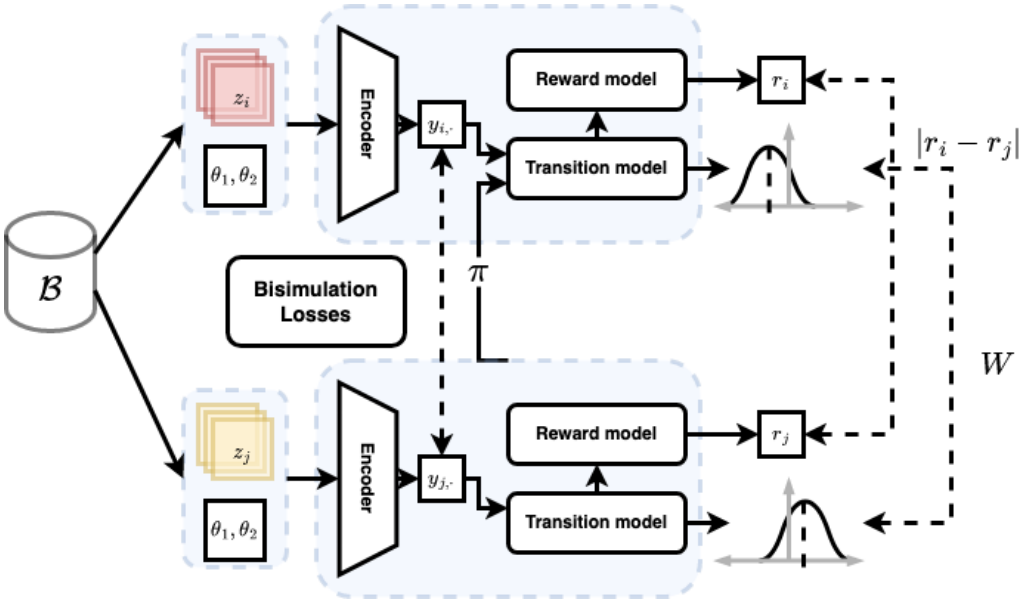


Figure 7.4: Network architecture

where we have use the following notation: $y_{i,\theta} = \phi(f(s_i, \theta), \theta)$ with $\overline{y}_{i,\theta}$ representing embeddings with stopped gradient and the target bisimulation distance $T_{i,j} = |r_i - r_j| + \gamma W_2(\hat{P}(\cdot|y_{i,\theta_1}, a_i), \hat{P}(\cdot|y_{j,\theta_1}, a_j))$. The relative weights for the three loss terms are given by hyper-parameters $\lambda_{base}, \lambda_{icc}, \lambda_{cc}$ respectively. We use a setup similar to [251] where we use a permuted batch of $\mathcal{B}$ for pairwise representation loss computation in step-8 of Algorithm 4. Similarly we a probabilistic dynamics model $\hat{P}$ which outputs a Gaussian distribution. This allows for a simple to compute closed form W_2 metric which is used to replace the W_1 metric in the original formulation: $W_2(\mathcal{N}(\mu_i, \Sigma_i), \mathcal{N}(\mu_j, \Sigma_j))^2 = ||\mu_i - \mu_j||_2^2 + ||\Sigma_i^{1/2} - \Sigma_j^{1/2}||_{\mathcal{F}}^2$, where $||\cdot||_{\mathcal{F}}$ is the Frobenius norm, Fig. 7.4 depicts the overall representation learning process. Finally, for the policy optimization part in step-7, we can use any max entropy policy gradient method. Access to simulator helps us translate a sampled batch from buffer into any randomly sampled contexts from which we can compute the various losses. However, in general this trick can also be extended to non simulator settings like data augmentation, this could be specially promising as the latter approaches currently only minimize representation distance between two views of same input and not the bisimulation distance which is more aligned with solving the task.

7.4 Analysis

We now discuss the important theoretical properties of our approach and study the generalization we can expect from learning representations under the conditional bisimuation framework. Proofs for the results can be found in Appendix E. The first result demonstrates the convergence of the π^*-bisimulation metric Eq. (7.3) on the joint input space $H \triangleq Z \times \Theta$ (We use the notation $h \triangleq (z, \theta)$ for a tuple in this space). We also overload the notion of policy(π) to implicitly contain ϕ so that it can be viewed as operating on the joint space.

Theorem 7.1. *Let* $\mathfrak{met}$ *be the space of bounded pseudometrics on* $Z \times \Theta$ *and* π *a policy that is continuously improving. Define* $\mathcal{F} : \mathfrak{met} \mapsto \mathfrak{met}$ *by*

$$\mathcal{F}(d, \pi)(h_i, h_j) = (1 - c)|r^\pi_{h_i} - r^\pi_{h_j}| + cW(d)(P^\pi_{h_i}, P^\pi_{h_j})$$

Then, $\forall c \in (0, 1)$, $\mathcal{F}$ *has a least fixed point* $\tilde{d}$ *which is a* π^**-bisimulation metric.*

We next discuss an important assumption we need to make towards obtaining generalization results for the observation shifts.

Assumption 7.1 (Block structure)**.** *We assume that* $f_{\theta_1}(s_1) \cap f_{\theta_2}(s_2) \neq \emptyset \implies s_i = s_j, \forall \theta_1, \theta_2$ *so that the observation map is invertible.*

This means that the observation space Z can be partitioned into disjoint blocks, each containing the support for a particular value of $s \in S$ [53]. This also ensures that f_θ^{-1} exists. Relaxing Assumption 7.1 can break any guarantees obtainable on value function similarities arising from state similarity. This is because the same observation can get mapped to entirely different states in the latent MDP each with very different values, making the environment only partially observable. Note however that this requirement is not too restrictive, is possible to consider added noise scenarios (both independent and correlated e.g. see [250]) which maintain identifiability of the state. Finally, many real-world task observations tend to

satisfy this assumption for high dimensional scenarios: e.g. visual projection of non-degenerate objects under different viewing angles.

We next discuss the implications of having learnt an representation ϕ which approximately preserves the π^*-bisimulation metric distances.

Theorem 7.2 (Aggregation value bound)**.** *Given an MDP $\hat{\mathcal{M}}$ constructed by aggregating tuples h of observation, context in an ϵ-neighborhood of the representation space such that $\delta \triangleq \max_{s,s',\theta_i,\theta_j} ||\phi(f_{\theta_i}(s),\theta_i) - \phi(f_{\theta_j}(s'),\theta_j)| - d_S(s,s')|$, where d_S is a π^*-bisimulation metric on S. Further let $\hat{\phi}$ denote the map from any h to these clusters, the optimal value functions for the two MDPs follow:*

$$|V^*(h) - \hat{V}^*(\hat{\phi}(h))| \leq \frac{2(\epsilon+\delta)}{(1-\gamma)(1-c)} \forall h \in Z \times \Theta$$

Note how the value estimate accuracy from aggregation is fundamentally bottlenecked by the representation learning error δ, this means that even the finest partitions (which use small ϵ) using ϕ will give value approximation only as good as the underlying representation.

We now state the lipschitz continuity assumptions we use for further analysis. The first Assumption 7.2 concerns the change in observations z as the context θ changes. Several natural domains like visual projections satisfy this.

Assumption 7.2. *f is lipschitz with coefficient L_θ^f with respect to (w.r.t.) θ.*

Next, we assume that the representation map ϕ and the policy π which conditions on the representations y are also lipshitz w.r.t. the inputs. This can be enforced in practice for example for deep neural networks approximators [234, 70].

Assumption 7.3. *ϕ is lipschitz w.r.t. z and θ with coefficients L_z^ϕ, L_θ^ϕ respectively. Similarly, π is lipschitz with coefficient L_y^π where the distance metric on the policy space is d_{TV}, the total variation metric on space of action distributions $\mathcal{P}(U)$.*

We now discuss the amount of generalization which we can expect when a policy assuming context θ_i is run on observation coming from the context θ_j. This can happen for example in scenarios when a shift in observations happens like change in the calibration settings of an autonomous vehicle's sensors. We introduce the notation $\pi_{\theta_i \leftarrow \theta_j}$ to represent the policy obtained from sampling action w.r.t. the restriction π_{θ_i} but using observation inputs from the context θ_j (ie. $\pi(a|\phi(f_{\theta_j}(s), \theta_i))$).

Theorem 7.3 (Generalization to unseen context)**.** *Under Assumption 7.2, Assumption 7.3 we have that for any two contexts θ_i, θ_j:*

$$|J^{\pi_{\theta_i}} - J^{\pi_{\theta_i \leftarrow \theta_j}}| \leq \frac{1}{1-\gamma} E_{\substack{s \sim f_\theta^{-1} \rho^{\pi_{\theta_i}}, \\ a \sim \pi_{\theta_i \leftarrow \theta_j}}} \Big[A^{\pi_{\theta_i}}(s,a) + \frac{2\gamma A_{max}}{1-\gamma} L_\theta^f L_z^\phi L_y^\pi d_\Theta(\theta_i, \theta_j) \Big]$$

where $A_{max} \triangleq \max_s |E_{a \sim \pi_{\theta_i \leftarrow \theta_j}}[A^{\pi_{\theta_i}}(s,a)]|$ and d_Θ is a metric on the context space.

Thus Theorem 7.3 gives us the upper bound on the deviation of the expected returns when the agent expects an environment with context θ_i but is actually deployed in with an observation context θ_j.

We next discuss the important performance transfer scenarios when the simulator used for training a policy is not exact. These bounds are useful for situations where it is required to access tolerance of agent performance w.r.t. situations like sim to real deployment. Our first result discusses the situation where the simulator dynamics is not exact w.r.t. the real world and instead introduces errors ϵ_R, ϵ_P.

Theorem 7.4 (Simulator fidelity bound)**.** *For an approximately correct simulator $(\hat{r}, \hat{P})$ such that $\max_{s,a} |\hat{r}(s,a) - r(s,a)| \leq \epsilon_R$ and $\max_{s,a} d_{TV}(\hat{P}(s,a), P(s,a)) \leq \epsilon_P$ we have for any policy π:*

$$|J^\pi - \hat{J}^\pi| \leq \frac{\epsilon_R}{(1-\gamma)} + \frac{\gamma \epsilon_P R_{max}}{(1-\gamma)^2}$$

Next, we consider the case where in addition to the latent transition and re-

ward dynamics, the simulator emission function $\hat{f}$ is also approximate. Let $\epsilon_f \triangleq \max_{s,\theta} d_Y(\phi(\hat{f}_\theta(s)), \phi(f_\theta(s)))$. We are interested in what happens when the policy learning happens on the approximate simulator $(\hat{r}, \hat{P}, \hat{f})$ but the resultant learnt policy is deployed in the actual world (R, P, f). Note that this is a common practical setting as most simulators even after knowing the actual underlying state, cannot completely capture the richness in the observations found in the real world. The below result relates the simulator policy performance $(\hat{f})$ to the one obtained by running the simulator policy on real observations (f).

Theorem 7.5 (Complete simulator fidelity bound)**.** *For an approximately correct simulator* $(\hat{r}, \hat{P}, \hat{f})$ *such that* $\max_{s,a} |\hat{r}(s,a) - r(s,a)| \leq \epsilon_R$, $\max_{s,a} d_{TV}(\hat{P}(s,a), P(s,a)) \leq \epsilon_P$ *and* $\epsilon_f \triangleq \max_{s,\theta} d_Y(\phi(\hat{f}_\theta(s)), \phi(f_\theta(s)))$*, we have for any policy* π*:*

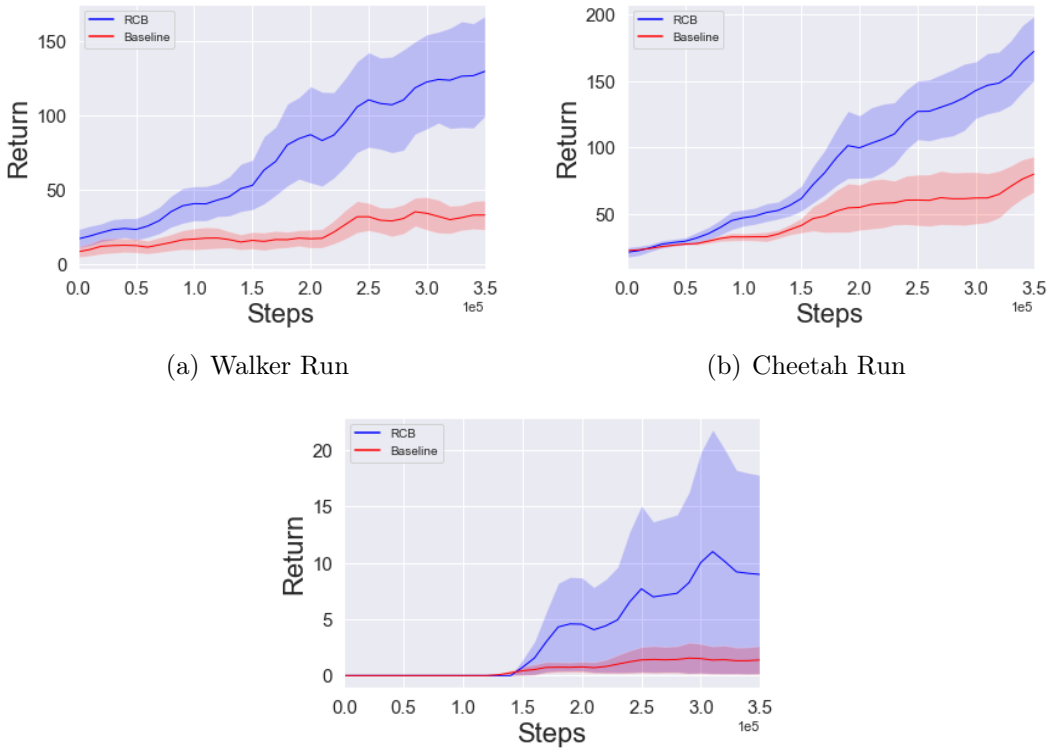
$$|J^{\pi_{\hat{f}\leftarrow f}} - \hat{J}^{\pi_{\hat{f}}}| \leq \frac{\epsilon_R}{(1-\gamma)} + \frac{\gamma \epsilon_P R_{max}}{(1-\gamma)^2} + \frac{1}{1-\gamma} E_{\substack{s\sim \hat{f}^{-1}\rho^{\pi_{\hat{f}}}, \\ a\sim\pi_{\hat{f}\leftarrow f}}} \Big[A^{\pi_{\hat{f}}}(s,a) + \frac{2\gamma A_{max}}{1-\gamma} L_y^\pi \epsilon_f \Big]$$

Where we use the notation $\pi_{\hat{f}\leftarrow f}$ to represent the sampling of actions from $\pi_{\hat{f}}$ but using the observations obtained under the (real world) observation function f. Thus, the above two results Theorem 7.4 and Theorem 7.5 are particularly useful for the realistic scenario where we have imprecise simulation dynamics.

7.5 Experiments

We perform experiments towards understanding whether conditional bisimulation (RCB) helps learn representations which generalize better to observation shifts. Towards this, we use the DeepMind control suite [211] which uses Mujoco [219] as the base simulator. We create new tasks for various agent methodologies where we learn to control the agent using image based input. For testing observation shifts, we use a uniform distribution over the range $P_\Theta = \mathcal{U}(-\pi/4, \pi/4)$ for the camera angle. Further, we also modify the simulator to have 3D spheres randomly bouncing

in the environment, which contribute towards noise. Note that this noise setting is harder than the simple distractor setting in [251] as the agent has to learn to model 3D noise across different visual perspectives (see Fig. 7.3). We use SAC[86] as the base algorithm for optimizing the MERL objective in Algorithm 4. For the baseline we use an agent similar to our architecture but which doesn't use the various bisimulation losses, instead this agent only uses a reward and a emission model to inform the representation. Additional experimental setup details can be found in Appendix E. At the beginning of each episode, we sample a camera angle context from P_Θ, the agents must adapt to changing image perspectives. For evaluation, we use a fixed set of camera angles: $\{-\pi/4, -\pi/8, 0, \pi/8, \pi/4\}$ over which we compute the agent performance during the evaluation phase and report the average across the angles as the performance metric. Fig. 7.5 gives the evaluation performance plots for



(a) Walker Run

(b) Cheetah Run

(c) Hopper Hop

Figure 7.5: Empirical results on modified DMC observation generalization tasks

the two agents on the walker run, cheetah run and hopper hop tasks averaged over

five seeds with one standard error shaded. We see that RCB performs significantly better than the baseline agents on all the scenarios. We also note that generalization for the hopper domain while doing pixel based control is especially hard given the environment stochasticity. For the walker run and cheetah run tasks RCB seems to be fairly robust to the observation shifts, infact its performance is comparable to that obtained when solving the task for only one camera angle.

7.6 Related Work

DBC (Zhang et.al, [251]) use bisimulation metrics to learn task relevant features which are robust to noise in the environment. They learn to tie together states distinguishable only by task irrelevant noise using bisimulation for learning a representation. They further demonstrate its effectiveness over methods like reconstruction for learning a control policy. In this work, we are concerned with the problem of a *functional* shift in the observation space itself arising from a change in an underlying context for the task (irrespective of presence of noise in the environment). Our goal is thus to learn a representation invariant to the changing context in richly observed environments. Towards this, we use the bisimulation framework to learn a representation which can *invert* the change in observation space caused by the varying context and can be seen as abstracting across the group of isomorphic MDPs indexed by the context. We also provide the first generalization bounds for this setting with important practical applications like sim to real transfer.

Robust RL considers rewards maximization under adversarially varying dynamics for the environment. [170] use a two agent zero-sum game to model an adversarial noise towards learning robust policies. Similarly, [192] inject noise in the state space and optimise for a minimax problem for robustness, [214] study the robustness problem under action perturbations. Whereas here we discuss the setting of adapting to potentially unseen deployment scenarios and provide theoretical guarantees for the

policy transfer. Contextual Markov Decision Processes[88] offers a general framework for studying RL problems whose dynamics are structurally dependent on a context space. [154] propose no regret algorithms for generalized linear model based contexts. MDP homomorphism [173, 143] is the principled framework of studying structural similarities across MDPs, this naturally extends the idea of state abstraction and opens the the way to leverage abstract similarities on a much broader scope. [254] compile the various methods used in Sim-to-real settings. Domain randomization, particularly used in robotic vision tasks including object localization [216], object detection [223], pose estimation [196], and semantic segmentation [249], the training data from simulator always have different textures, lighting, and camera positions from the realistic environments. Therefore, domain randomization aims to provide enough simulated variability of the parameters at training time such that at test time the model is able to generalize to potentially unseen, real-world data. However, it doesn't utilize the inherent structure in terms of state similarity and doesn't help ignore the parts of the observation which are not relevant for reward maximization. Data Augmentation [126] use various image transformations on agent observations for data efficient learning of policies for pixel based control. [191] use random crops on image data to be used under a contrastive based framework for representation learning. [120] use random image translations for regularising reinforcement learning from images by using multiple shifts to robustly estimate value function loss and targets.

7.7 Conclusions & Future Work

In this we work we explored how bisimulation can be used to learn representation for RL towards generalization in complex high dimensional environment like visual inputs. We specially focused on learning policies invariant to observation shifts a problem which has several applications in the real world. Further, we analysed the

theory of learning under the framework of conditional bisimualtion and proposed novel bounds characterizing state abstraction and generalization in this setting. Of particular importance were the results relating to performance guarantees across observation shifts when learning on a simulator. Finally, we evaluated our method on the modified DM-contol domain and showed its efficacy in comparison to the baseline approach. For future work we would like to investigate tighter theoretical bounds for performance transfer specially in finite sample setting and ways to speed up learning bisimulation metrics on large state action spaces.

Part IV

Conclusion, References and Appendix

Chapter 8

Conclusion

In this thesis we motivated, proposed and analysed several novel challenges for reinforcement learning in large state action spaces. We covered a variety of RL settings (single and multi-agent systems (MAS) with all the variation in the latter, prediction and control, model-based and model-free methods, value-based and policy-based methods). We were also the first to provide various theoretical and empirical results on several different problems. We now perform an overview of the results in this thesis and discuss some important open challenges and potential directions for future work:

- Chapter 3 gives definite suboptimality bounds arising from insufficient exploration due from constraints on Q function class in cooperative MARL. It covers various joint exploration scenarios. It also demonstrates the successful use of the mutual information(MI) based framework: Maven, towards achieving committed exploration for solving the problem. However, we did not analyse convergence properties for Maven. We believe this can be addressed using soft value function based approaches[86], further this can help find better ways for tuning the MI weight hyper-parameter which we found to be crucial for better performance on complex domains. Another important future direction

to explore is the use of Maven style exploration ideas for general su games. We believe there is great potential in using the MI based exploration framework for avoiding problems like sub-optimal Nash and correlated equilibria.

- Chapter 4 proposes the novel tensorized form of the Bellman equation which allows exponential sample gains in comparison to naive methods for model based leanrning. Similarly, we demonstrated superior performance on the complex StarCraft domain using tensorized critic for actor-critic methods. However, there are currently no approaches which utilize the tensorization for Q-learning approaches. We believe developments on this would be useful as several domains are more effectively solved using value based approach. We think development son approximate tensor maximization could be useful for this. Next, we also provided PAC guarantees for model estimation and policy evaluation under the Tesseract framework for the rank sufficient case. A challenging open problem is to extend these results albeit with approximation errors for the rank deficient case.

- Chapter 5 is the first work to propose a framework for studying combinatorial generalization (CG) in coopertive multi agent systems. It is also the first to extend the notion of successor features to MAS and provides several novel bounds for transfer learning under practical scenarios. However, we found that modern MARL algorithms only demonstrate preliminary generalization on easy scenarios, and are in general are brittle to population changes in complex domains.Creating methods to address this would be a major game changer for application of MARL to real world problems given that the future of automation would be inherently multi-agent. We think it would be promising to try out modern sequence models towards designing MARL algorithms for CG.

- Chapter 6 proposes a new framework: Virel for learning RL policies under a

probabilistic framework. It provides several advantages like ability to learn deterministic policies, mode seeking behaviour and exploration using residual error. Since deep neural network based function approximation can incur large estimation error and sharp variations, we believe Virel could benefit by incorporating alternate ways to the bellman residue for incorporating uncertainty. One promising direction for this would be to explore applicability of state abstraction methods for uncertainity estimation [59].

- Chapter 7 proposes a new conditional bisimulation based framework for generalization across observation shifts. It also discusses several novel transfer bounds for important practical settings like sim to real. One promising direction for this work is exploring the connections to methods like data augmentation [126]. We believe this can help us gain a better understanding about the desiderata for ensuring generalization in RL.

Bibliography

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[2] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International Conference on Machine Learning*, pages 22–31. PMLR, 2017.

[3] Stefano V Albrecht, Jacob W Crandall, and Subramanian Ramamoorthy. Belief and truth in hypothesised behaviours. *Artificial Intelligence*, 235:63–94, 2016.

[4] Stefano V Albrecht and Peter Stone. Reasoning about hypothetical agent behaviours and their parameters. *arXiv preprint arXiv:1906.11064*, 2019.

[5] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

[6] Animashree Anandkumar, Daniel Hsu, and Sham M Kakade. A method of moments for mixture models and hidden markov models. In *Conference on Learning Theory*, pages 33–1, 2012.

[7] Carl Anderson and Elizabeth McMillan. Of ants and men: Self-organized teams in human and insect organizations. *Emergence*, 5(2):29–41, 2003.

[8] Robert J Aumann. Subjectivity and correlation in randomized strategies. *Journal of mathematical Economics*, 1(1):67–96, 1974.

[9] Kamyar Azizzadenesheli. *Reinforcement Learning in Structured and Partially Observable Environments*. PhD thesis, UC Irvine, 2019.

[10] Kamyar Azizzadenesheli, Alessandro Lazaric, and Animashree Anandkumar. Reinforcement learning in rich-observation mdps using spectral methods. *arXiv preprint arXiv:1611.03907*, 2016.

[11] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. *Machine Learning-International Workshop Then Conference-*, (July):30–37, 1995.

[12] David Barber, A Taylan Cemgil, and Silvia Chiappa. *Bayesian time series models*. Cambridge University Press, 2011.

[13] Henk P Barendregt. Introduction to lambda calculus. 1984.

[14] Eugenio Bargiacchi, Timothy Verstraeten, Diederik Roijers, Ann Nowé, and Hado Hasselt. Learning to coordinate with coordination graphs in repeated single-stage multi-agent decision problems. In *International conference on machine learning*, pages 482–490, 2018.

[15] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510. PMLR, 2018.

[16] André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado Van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. *arXiv preprint arXiv:1606.05312*, 2016.

[17] André Barreto, Shaobo Hou, Diana Borsa, David Silver, and Doina Precup. Fast reinforcement learning with generalized policy updates. *Proceedings of the National Academy of Sciences*, 117(48):30079–30087, 2020.

[18] Samuel Barrett, Peter Stone, and Sarit Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *AAMAS*, pages 567–574, 2011.

[19] R.F. Bass. *Real Analysis for Graduate Students*. Createspace Ind Pub, 2013.

[20] Matthew James Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, 2003.

[21] R Bellman. *Dynamic Programming*, volume 70. 1957.

[22] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[23] Dimitri Bertsekas and John Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.

[24] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Athena scientific series in optimization and neural computation. Athena Scientific, 1996.

[25] Shalabh Bhatnagar, Doina Precup, David Silver, Richard S Sutton, Hamid R. Maei, and Csaba Szepesvári. Convergent temporal-difference learning with arbitrary smooth function approximation. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1204–1212. Curran Associates, Inc., 2009.

[26] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[27] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[28] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians, 2017.

[29] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge*, TARK '96, page 195–210. Morgan Kaufmann Publishers Inc., 1996.

[30] Michael Bowling and Peter McCracken. Coordination and adaptation in impromptu teams. In *AAAI*, volume 5, pages 53–58, 2005.

[31] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

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

[33] Stefano Bromuri. A tensor factorization approach to generalization in multi-agent reinforcement learning. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 274–281. IEEE, 2012.

[34] Adrian Bulat, Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. Incremental multi-domain learning with network latent tensor factorization. 2020.

[35] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems,*

Man, and Cybernetics, Part C (Applications and Reviews), 38(2):156–172, 2008.

[36] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An Overview of Recent Progress in the Study of Distributed Multi-agent Coordination. *IEEE Transactions on Industrial Informatics*, 9(1):427–438, 2012.

[37] Anthony R Cassandra, Leslie Pack Kaelbling, and Michael L Littman. Acting optimally in partially observable stochastic domains. In *Aaai*, volume 94, pages 1023–1028, 1994.

[38] Pablo Samuel Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2020.

[39] Yong Chen, Ming Zhou, Ying Wen, Yaodong Yang, Yufeng Su, Weinan Zhang, Dell Zhang, Jun Wang, and Han Liu. Factorized q-learning for large-scale multi-agent systems. *arXiv preprint arXiv:1809.03738*, 2018.

[40] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[41] Andrzej Cichocki, Anh-Huy Phan, Qibin Zhao, Namgil Lee, Ivan Oseledets, Masashi Sugiyama, Danilo P Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives. *Foundations and Trends® in Machine Learning*, 9(6):431–673, 2017.

[42] Kamil Ciosek and Shimon Whiteson. Expected Policy Gradients. *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[43] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR, 09–15 Jun 2019.

[44] Nadav Cohen, Or Sharir, and Amnon Shashua. On the expressive power of deep learning: A tensor analysis. In *Conference on Learning Theory*, pages 698–728, 2016.

[45] Ross H Crozier, Philip S Newey, Ellen A Schluens, Simon KA Robson, et al. A masterpiece of evolution–oecophylla weaver ants (hymenoptera: Formicidae). *Myrmecological News*, 13(5), 2010.

[46] Wojciech Marian Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops, 2020.

[47] Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

[48] Peter Dayan and Geoffrey E. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

[49] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

[50] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[51] Maria Dimakopoulou and Benjamin Van Roy. Coordinated exploration in concurrent reinforcement learning. In *International Conference on Machine Learning*, pages 1270–1278, 2018.

[52] Simon S. Du, Sham M. Kakade, Ruosong Wang, and Lin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning? In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[53] Simon S. Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudík, and John Langford. Provably efficient RL with rich observations via latent state decoding. *Computing Research Repository (CoRR)*, abs/1901.09018, 2019.

[54] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[55] Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

[56] Matthew Fellows, Kamil Ciosek, and Shimon Whiteson. Fourier policy gradients. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1486–1495, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[57] Matthew Fellows, Anuj Mahajan, Tim GJ Rudner, and Shimon Whiteson. Virel: A variational inference framework for reinforcement learning. *arXiv preprint arXiv:1811.01132*, 2018.

[58] Matthew Fellows, Anuj Mahajan, Tim GJ Rudner, and Shimon Whiteson. Virel: A variational inference framework for reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[59] Norm Ferns, Prakash Panangaden, and Doina Precup. Metrics for finite Markov decision processes. In *Uncertainty in Artificial Intelligence (UAI)*, pages 162–169, 2004.

[60] Norm Ferns, Prakash Panangaden, and Doina Precup. Bisimulation metrics for continuous Markov decision processes. *Society for Industrial and Applied Mathematics*, 40(6):1662–1714, December 2011.

[61] Norman Ferns and Doina Precup. Bisimulation metrics are optimal value functions. In *Uncertainty in Artificial Intelligence (UAI)*, pages 210–219, 2014.

[62] Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

[63] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[64] C W Fox and S J Roberts. A tutorial on variational Bayesian inference. *Artificial Intelligence Review*, pages 1–11, 2010.

[65] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[66] Thomas Furmston and D Barber. Variational Methods For Reinforcement Learning. *In AISTATS*, pages 241–248, 2010.

[67] Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability. *arXiv:2107.06277 [cs, stat]*, 2021.

[68] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[69] Robert Givan, Thomas L. Dean, and Matthew Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147:163–223, 2003.

[70] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

[71] Anirudh Goyal, Philemon Brakel, William Fedus, Timothy P. Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. *CoRR*, abs/1804.00379, 2018.

[72] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(9), 2004.

[73] Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016.

[74] Barbara Grosz and Sarit Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 1996.

[75] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E. Turner, and Sergey Levine. Q-Prop: Sample-Efficient Policy Gradient with An Off-Policy Critic. pages 1–13, 2016.

[76] Christian Guckelsberger, Christoph Salge, and Julian Togelius. New and surprising ways to be mean. adversarial npcs with coupled empowerment minimisation. *arXiv preprint arXiv:1806.01387*, 2018.

[77] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent Planning with Factored MDPs. In *Advances in Neural Information Processing Systems*, pages 1523–1530. MIT Press, 2002.

[78] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[79] Carlos Guestrin, Michail Lagoudakis, and Ronald Parr. Coordinated reinforcement learning. In *ICML*, volume 2, pages 227–234. Citeseer, 2002.

[80] Carlos Guestrin, Shobha Venkataraman, and Daphne Koller. Context-specific multiagent coordination and planning with factored mdps. In *AAAI/IAAI*, pages 253–259, 2002.

[81] Asela Gunawardana and William Byrne. Convergence theorems for generalized alternating minimization procedures. *J. Mach. Learn. Res.*, 6:2049–2073, December 2005.

[82] Tarun Gupta, Anuj Mahajan, Bei Peng, Wendelin Böhmer, and Shimon Whiteson. Uneven: Universal value exploration for multi-agent reinforcement learning. *arXiv preprint arXiv:2010.02974*, 2020.

[83] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018.

[84] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In Doina Precup and

Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1352–1361, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[85] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

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

[87] Hirotaka Hachiya, Jan Peters, and Masashi Sugiyama. Efficient sample reuse in em-based policy search. In Wray Buntine, Marko Grobelnik, Dunja Mladenić, and John Shawe-Taylor, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 469–484, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[88] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual markov decision processes, 2015.

[89] Matthew Hausknecht and Peter Stone. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents*, 2015.

[90] Nicolas Heess, David Silver, and Yee Whye Teh. Actor-critic reinforcement learning with energy-based policies. In Marc Peter Deisenroth, Csaba Szepesvári, and Jan Peters, editors, *Proceedings of the Tenth European Work-*

shop on Reinforcement Learning, volume 24 of *Proceedings of Machine Learning Research*, pages 45–58, Edinburgh, Scotland, 30 Jun–01 Jul 2013. PMLR.

[91] Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning Continuous Control Policies by Stochastic Value Gradients. pages 1–13, 2015.

[92] Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2944–2952. Curran Associates, Inc., 2015.

[93] Pablo Hernandez-Leal, Yusen Zhan, Matthew E Taylor, L Enrique Sucar, and Enrique Munoz De Cote. Efficiently detecting switches against non-stationary opponents. *Autonomous Agents and Multi-Agent Systems*, 31(4):767–789, 2017.

[94] Irina Higgins, Arka Pal, Andrei A. Rusu, Loïc Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: improving zero-shot transfer in reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1480–1490. PMLR, 2017.

[95] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

[96] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition:

The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[98] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[99] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. Guided Deep Reinforcement Learning for Swarm Systems. In *AAMAS 2017 Autonomous Robots and Multirobot Systems (ARMS) Workshop*, 2017.

[100] Shariq Iqbal, Christian A. Schroeder de Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning, 2021.

[101] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks, 2016.

[102] Prateek Jain and Sewoong Oh. Provable tensor factorization with missing data. In *Advances in Neural Information Processing Systems*, pages 1431–1439, 2014.

[103] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pages 7254–7264, 2018.

[104] Nan Jiang. Notes on tabular methods. 2018.

[105] Michael I. Jordan, editor. *Learning in Graphical Models*. MIT Press, Cambridge, MA, USA, 1999.

[106] 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.

[107] Sham Machandranath Kakade. *On the sample complexity of reinforcement learning*. PhD thesis, UCL (University College London), 2003.

[108] Michael J Kearns, Umesh Virkumar Vazirani, and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.

[109] John Kelly. *Generalized Functions*, chapter 4, pages 111–124. John Wiley & Sons, Ltd, 2008.

[110] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[111] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes PPT. *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014.

[112] Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *Second International Conference on Learning Representations, ICLR*, 2014.

[113] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

[114] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[115] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[116] Daphne Koller and Ronald Parr. Policy iteration for factored mdps. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, pages 326–334, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[117] Vijaymohan Konda and John N. Tsitsiklis. *Actor-Critic Algorithms*. PhD thesis, USA, 2002. AAI0804543.

[118] Jean Kossaifi, Adrian Bulat, Georgios Tzimiropoulos, and Maja Pantic. T-net: Parametrizing fully convolutional nets with a single high-order tensor. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[119] Jean Kossaifi, Antoine Toisoul, Adrian Bulat, Yannis Panagakis, Timothy Hospedales, and Maja Pantic. Factorized higher-order cnns with an application to spatio-temporal emotion estimation. In *IEEE CVPR*, 2020.

[120] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[121] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels, 2021.

[122] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[123] Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Pac reinforcement learning with rich observations. In *Advances in Neural Information Processing Systems*, pages 1840–1848, 2016.

[124] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classifica-

tion with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[125] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing (preliminary report). In *Symposium on Principles of Programming Languages*, page 344–352. Association for Computing Machinery, 1989.

[126] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33:19884–19895, 2020.

[127] Tor Lattimore, Marcus Hutter, and Peter Sunehag. The sample-complexity of general reinforcement learning. In *International Conference on Machine Learning*, pages 28–36. PMLR, 2013.

[128] Agapito Ledezma, Ricardo Aler, Araceli Sanchis, and Daniel Borrajo. Predicting opponent actions by observation. In *Robot Soccer World Cup*, pages 286–296. Springer, 2004.

[129] Joel Z Leibo, Edgar A Dueñez-Guzman, Alexander Vezhnevets, John P Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charlie Beattie, Igor Mordatch, and Thore Graepel. Scalable evaluation of multi-agent reinforcement learning with melting pot. In *Proceedings of the 38th International Conference on Machine Learning*, pages 6187–6199. PMLR, 2021.

[130] Sergey Levine. *Motor Skill Learning with Trajectory Methods*. PhD thesis, 2014.

[131] Sergey Levine. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. 2018.

[132] Sergey Levine and Vladlen Koltun. Variational policy search via trajectory optimization. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani,

and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 207–215. Curran Associates, Inc., 2013.

[133] Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006.

[134] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, Princeton, NJ, USA, 2011.

[135] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv:1509.02971*, 2015.

[136] Alex Tong Lin, Mark J Debord, Katia Estabridis, Gary Hewer, and Stanley Osher. Cesma: Centralized expert supervises multi-agents. *arXiv preprint arXiv:1902.02311*, 2019.

[137] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6382–6393, 2017.

[138] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[139] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[140] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson.

Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pages 7611–7622, 2019.

[141] Anuj Mahajan, Mikayel Samvelyan, Tarun Gupta, Benjamin Ellis, Mingfei Sun, Tim Rocktäschel, and Shimon Whiteson. Generalization in cooperative multi-agent systems. *arXiv preprint arXiv:2202.00104*, 2022.

[142] Anuj Mahajan, Mikayel Samvelyan, Lei Mao, Viktor Makoviychuk, Animesh Garg, Jean Kossaifi, Shimon Whiteson, Yuke Zhu, and Animashree Anandkumar. Tesseract: Tensorised actors for multi-agent reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 7301–7312. PMLR, 2021.

[143] Anuj Mahajan and Theja Tulabandhula. Symmetry detection and exploitation for function approximation in deep rl. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1619–1621. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[144] Anuj Mahajan and Theja Tulabandhula. Symmetry detection and exploitation for function approximation in deep rl. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1619–1621, 2017.

[145] Anuj Mahajan and Theja Tulabandhula. Symmetry learning for function approximation in reinforcement learning. *arXiv preprint arXiv:1706.02999*, 2017.

[146] Anuj Mahajan and Theja Tulabandhula. Symmetry learning for function approximation in reinforcement learning. *arXiv preprint arXiv:1706.02999*, 2017.

[147] Dhruv Malik, Yuanzhi Li, and Pradeep Ravikumar. When Is Generalizable Reinforcement Learning Tractable? *arXiv:2101.00300 [cs, stat]*, 2021.

[148] Shaul Markovitch and Ronit Reger. Learning and exploiting relative weaknesses of opponent agents. *Autonomous Agents and Multi-Agent Systems*, 10(2):103–130, 2005.

[149] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69. IEEE, 2007.

[150] Andriy Mnih and Karol Gregor. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*, 2014.

[151] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[152] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[153] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[154] Aditya Modi and Ambuj Tewari. Contextual markov decision processes using generalized linear models. 2019.

[155] Gerhard Neumann. Variational inference for policy search in changing situations. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 817–824, USA, 2011. Omnipress.

[156] Stefanos Nikolaidis, Keren Gu, Ramya Ramakrishnan, and Julie Shah. Efficient model learning for human-robot collaborative tasks. arxiv, 2014.

[157] Shervin Nouyan, Roderich Groß, Michael Bonani, Francesco Mondada, and Marco Dorigo. Teamwork in self-organized robot colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711, 2009.

[158] Frans A. Oliehoek and Christopher Amato. *A Concise Introduction to Decentralized POMDPs*. SpringerBriefs in Intelligent Systems. Springer, 2016.

[159] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep Decentralized Multi-task Multi-Agent RL under Partial Observability. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2681–2690, 2017.

[160] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv:1912.06680 [cs, stat]*, 2019.

[161] James M Ortega and Werner C Rheinboldt. *Iterative solution of nonlinear equations in several variables*. SIAM, 2000.

[162] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[163] Ian Osband, Benjamin Van Roy, Daniel Russo, and Zheng Wen. Deep exploration via randomized value functions. *arXiv preprint arXiv:1703.07608*, 2017.

[164] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[165] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

[166] Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6:147–160, 1994.

[167] 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.

[168] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 745–750, New York, NY, USA, 2007. ACM.

[169] Georgios Piliouras, Mark Rowland, Shayegan Omidshafiei, Romuald Elie, Daniel Hennes, Jerome Connor, and Karl Tuyls. Evolutionary dynamics and ϕ-regret minimization in games, 2021.

[170] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust

adversarial reinforcement learning. In *International Conference on Machine Learning*, pages 2817–2826. PMLR, 2017.

[171] Roberta Raileanu, Max Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in deep reinforcement learning, 2021.

[172] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4295–4304, 2018.

[173] Balaraman Ravindran. *An algebraic approach to abstraction in reinforcement learning*. University of Massachusetts Amherst, 2004.

[174] Manish Chandra Reddy Ravula. *Ad-hoc teamwork with behavior-switching agents*. PhD thesis, 2019.

[175] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. Approximate inference and stochastic optimal control. *CoRR*, abs/1009.3958, 2010.

[176] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Robotics: Science and Systems*, 2012.

[177] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

[178] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[179] Heechang Ryu, Hayong Shin, and Jinkyoo Park. Multi-agent actor-critic with hierarchical graph attention network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7236–7243, 2020.

[180] Brian Sallans and Geoffrey E. Hinton. Reinforcement learning with factored states and actions. *J. Mach. Learn. Res.*, 5:1063–1088, December 2004.

[181] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The StarCraft Multi-Agent Challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019.

[182] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3528–3536. Curran Associates, Inc., 2015.

[183] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. 37:1889–1897, 07–09 Jul 2015.

[184] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, 2017.

[185] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[186] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[187] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 387–395, 2014.

[188] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1905.05408*, 2019.

[189] Edward J Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Operations research*, 26(2):282–304, 1978.

[190] Zhao Song, Ronald E. Parr, and Lawrence Carin. Revisiting the softmax bellman operator: Theoretical properties and practical benefits. *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[191] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020.

[192] Samuel Stanton, Rasool Fakoor, Jonas Mueller, Andrew Gordon Wilson, and Alex Smola. Robust reinforcement learning for shifting dynamics during deployment.

[193] Peter Stone, Gal A Kaminka, Sarit Kraus, and Jeffrey S Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[194] Peter Stone, Patrick Riley, and Manuela Veloso. Defining and using ideal teammate and opponent agent models: A case study in robotic soccer. In *Proceedings Fourth International Conference on MultiAgent Systems*, pages 441–442. IEEE, 2000.

[195] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[196] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[197] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[198] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, 2017.

[199] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[200] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[201] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

[202] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 2nd edition, 2017.

[203] Richard S Sutton, Hamid R. Maei, and Csaba Szepesvári. A convergent o(n) temporal-difference algorithm for off-policy learning with linear function approximation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1609–1616. Curran Associates, Inc., 2009.

[204] Richard S. Sutton, Hamid Reza Maei, Doina Precup, Shalabh Bhatnagar, David Silver, Csaba Szepesvári, and Eric Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 993–1000, New York, NY, USA, 2009. ACM.

[205] Richard S. Sutton, David Mcallester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems 12*, pages 1057–1063, 1999.

[206] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[207] Milind Tambe. Towards flexible teamwork. *Journal of artificial intelligence research*, 7:83–124, 1997.

[208] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 2017.

[209] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[210] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.

[211] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. *arXiv:1801.00690 [cs]*, 2018.

[212] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, DeepMind, January 2018.

[213] Open Ended Learning Team, Adam Stooke, Anuj Mahajan, Catarina Barros, Charlie Deck, Jakob Bauer, Jakub Sygnowski, Maja Trebacz, Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Roberta Raileanu, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-ended learning leads to generally capable agents, 2021.

[214] Chen Tessler, Yonathan Efroni, and Shie Mannor. Action robust reinforcement learning and applications in continuous control. *arXiv:1901.09184*, pages 6215–6224, 2019.

[215] Philip Thomas. Bias in natural actor-critic algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 441–448, Bejing, China, 22–24 Jun 2014. PMLR.

[216] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks

from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

[217] Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in neural information processing systems*, pages 1369–1376, 2007.

[218] Emanuel Todorov. Linearly-solvable markov decision problems. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1369–1376. MIT Press, 2007.

[219] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*. IEEE, 2012.

[220] Marc Toussaint. Probabilistic inference as a model of planned behavior. *Kunstliche Intelligenz*, 3, 01 2009.

[221] Marc Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pages 1–8, 2009.

[222] Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state Markov Decision Processes. *Proceedings of the 23rd international conference on Machine learning - ICML '06*, pages 945–952, 2006.

[223] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[224] John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal-difference

learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, 1997.

[225] Richard Eric Turner and Maneesh Sahani. *Two problems with variational expectation maximisation for time series models*, page 104–124. Cambridge University Press, 2011.

[226] Karl Tuyls, Julien Perolat, Marc Lanctot, Edward Hughes, Richard Everett, Joel Z Leibo, Csaba Szepesvári, and Thore Graepel. Bounds and dynamics for empirical game theoretic analysis. *Autonomous Agents and Multi-Agent Systems*, 34(1):1–30, 2020.

[227] Franck van Breugel and James Worrell. Towards quantitative verification of probabilistic transition systems. In Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen, editors, *Automata, Languages and Programming*, pages 421–432. Springer, 2001.

[228] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning. 2015.

[229] Vladimir Vapnik, Isabel Guyon, and Trevor Hastie. Support vector machines. *Mach. Learn*, 20(3):273–297, 1995.

[230] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[231] Alexander Vezhnevets, Yuhuai Wu, Maria Eckstein, Rémi Leblond, and Joel Z Leibo. OPtions as REsponses: Grounding behavioural hierarchies in multi-agent reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, pages 9733–9742. PMLR, 2020.

[232] Cédric Villani. *Topics in optimal transportation.* American Mathematical Society, 01 2003.

[233] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[234] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. *Advances in Neural Information Processing Systems*, 31, 2018.

[235] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[236] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.

[237] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. *arXiv preprint arXiv:2010.01523*, 2020.

[238] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[239] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. Multiagent soft q-learning. In *2018 AAAI Spring Symposium Series*, 2018.

[240] Min Wen and Ufuk Topcu. Constrained cross-entropy method for safe reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[241] Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256, 1992.

[242] Ronald J. Williams, Leemon C. Baird, and III. Analysis of some incremental variants of policy iteration: First steps toward understanding actor-critic learning systems, 1993.

[243] Ronald J. Williams and Jing Peng. Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3(3):241–268, 1991.

[244] C F Jeff Wu. On the Convergence Properties of the EM Algorithm'. *Source: The Annals of Statistics The Annals of Statistics*, 11(1):95–103, 1983.

[245] Erfu Yang and Dongbing Gu. Multiagent reinforcement learning for multi-robot systems: A survey. Technical report, University of Strathclyde, 2004.

[246] Yaodong Yang, Jianye Hao, Ben Lu Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *ArXiv*, abs/2002.03939, 2020.

[247] Xinghu Yao, Chao Wen, Yuhui Wang, and Xiaoyang Tan. Smix: Enhancing centralized value functions for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1911.04094*, 2019.

[248] Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2021.

[249] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[250] Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block mdps. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11214–11224. PMLR, 2020.

[251] Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning Invariant Representations for Reinforcement Learning without Reconstruction. *arXiv:2006.10742 [cs, stat]*, 2021.

[252] Amy Zhang, Shagun Sodhani, Khimya Khetarpal, and Joelle Pineau. Multi-task reinforcement learning as a hidden-parameter block mdp. *arXiv e-prints*, pages arXiv–2007, 2020.

[253] Tingting Zhao, Gang Niu, Ning Xie, Jucheng Yang, and Masashi Sugiyama. Regularized policy gradients: direct variance reduction in policy gradient estimation. In *Asian Conference on Machine Learning*, pages 333–348. PMLR, 2016.

[254] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[255] Stephan Zheng and Yisong Yue. Structured exploration via hierarchical variational policy networks. *openreview*, 2018.

[256] Brian D Ziebart. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy.* PhD thesis, 2010.

[257] Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. Modeling interaction via the principle of maximum causal entropy. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 1255–1262, USA, 2010. Omnipress.

[258] Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1433–1438. AAAI Press, 2008.

[259] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

Appendix A

Appendix for Chapter 3

A.1 Proofs

A.1.1 Uniform visitation

Theorem A.1. *For n player, $k \geq 3$ action matrix games ($|\mathcal{A}| = n, |U| = k$), under uniform visitation; Q_{qmix} learns a δ-suboptimal policy for any time horizon T, for any $0 < \delta \leq R\Big[\sqrt{\frac{a(b+1)}{a+b}} - 1\Big]$ for the payoff matrix given by the template below, where $b = \sum_{s=1}^{k-2} \binom{n+s-1}{s}$, $a = k^n - (b+1)$, $R > 0$:*

$$\begin{bmatrix} R+\delta & 0 & \dots & R \\ 0 & & \iddots & \\ \vdots & \iddots & & \vdots \\ R & \dots & & R \end{bmatrix}$$

Proof. For single state MDPs, under uniform visitation of the joint state-action space, QMIX can be seen as minimising the mean squared error between the actual Q-values and the monotonic projection Q_{qmix}. Using the symmetry of the problem and an exchange argument, it can be shown that only the monotonic projections of

the following form need to be considered:

$$\begin{bmatrix} x_3 & x_2 & \dots & x_1 \\ x_2 & & \iddots & \\ \vdots & \iddots & & \vdots \\ x_1 & \dots & & x_1 \end{bmatrix}$$

where $X \triangleq (x_1, x_2, x_3)$. Consequently, there are two cases for the monotonic approximations. We refer to them as $M1$ and $M2$ corresponding to $x_1 \geq x_2 \geq x_3$ and $x_1 \leq x_2 \leq x_3$ cases respectively. The optimization problem for $M1$ is:

$$\begin{aligned} & M1: \\ & \underset{X}{\text{minimise}} && a(x_1 - R)^2 + bx_2^2 + (x_3 - (R+\delta))^2 \\ & s.t. && x_2 - x_1 \leq 0 \\ & && x_3 - x_2 \leq 0 \end{aligned}$$

where $b = \sum_{s=1}^{k-2} \binom{n+s-1}{s}$, $a = k^n - (b+1)$ are the coefficients corresponding to the number of entries for the general n player, k action game (having k^n entries). It is thus evident that the above problem is a quadratic program and is indeed convex [31] as the Hessian of the objective $diag(a, b, 1)$ is positive definite. The Largrangian is given by:

$$\mathcal{L}(X, \lambda_1, \lambda_2) = a(x_1 - R)^2 + bx_2^2 + (x_3 - (R+\delta))^2 + \lambda_1(x_2 - x_1) + \lambda_2(x_3 - x_2)$$

where λ_1, λ_2 are the dual variables. Moreover, the above problem also satisfies Slater's conditions which implies that KKT conditions are necessary and sufficient

for finding the primal and dual optimal. By setting $\nabla_X \mathcal{L} = 0$, we get:

$$\lambda_1 = 2a(x_1 - R)$$
$$\lambda_2 = 2a(R + \delta - x_3)$$
$$x_2 = \frac{\lambda_2 - \lambda_1}{2b}$$

Using primal and dual feasibility constraints along with complementary slackness, we can see that $x_1 = R, x_2 = x_3 = \frac{R+\delta}{1+b}$ is an optimal solution to $M1$ for $\delta \leq bR$ with the optimal value for the problem as $OPT(M1) = \frac{b(R+\delta)^2}{b+1}$. By solving $M2$ in a similar way for the reversed primal constraints $x_2 - x_1 \geq 0, x_3 - x_2 \geq 0$, we see that an optimal assignment is $x_1 = x_2 = \frac{aR}{b+a}, x_3 = R + \delta$ with the optimal value given by $OPT(M2) = \frac{R^2 ab}{a+b}$. Note that the solution to $M1$ corresponds to the suboptimal policy of picking action corresponding to payoff R, whereas the solution to $M2$ corresponds to that of picking the optimal action with payoff $R+\delta$ (as QMIX picks the action corresponding to the maximal entry of a monotonic projection). For QMIX to learn the suboptimal policy corresponding to $M1$, we require that $OPT(M1) \leq OPT(M2)$. Consequently,

$$\frac{b(R+\delta)^2}{b+1} \leq \frac{R^2 ab}{a+b}$$
$$\implies \delta \leq R\Big[\sqrt{\frac{a(b+1)}{a+b}} - 1\Big] \tag{A.1}$$

□

A.1.2 ϵ-greedy visitation

Theorem A.2. *For n player, $k \geq 3$ action matrix games, under ϵ-greedy visitation $\epsilon(t)$; Q_{qmix} learns a δ-suboptimal policy for any time horizon T with probability $\geq 1 - \Big(\exp(-\frac{T\upsilon^2}{2}) + (k^n - 1)\exp(-\frac{T\upsilon^2}{2(k^n-1)^2})\Big)$, for any $0 < \delta \leq R\Bigg[\sqrt{a\Big(\frac{\upsilon b}{2(1-\upsilon/2)(a+b)} + 1\Big)} -$*

$1\Big]$ *for the payoff matrix given by the template above, where* $b = \sum_{s=1}^{k-2} \binom{n+s-1}{s}$, $a = k^n - (b+1)$, $R > 0$ *and* $\upsilon = \epsilon(T)$.

Proof. Given the exploration schedule $\epsilon(t)$, let $\epsilon(T) = \upsilon$ (which is the minimum value since $\epsilon(t)$ is decreasing in T). We reuse the machinery introduced in Appendix A.1.1 and provide an analysis which is agnostic to the actions actually visited by considering the adversarial case for the maximum possible δ for which Q_{qmix} fails. This happens precisely when QMIX is provided with the "best opportunity" for learning the optimal policy (so that it visits the optimal action with probability $1 - \epsilon(t), \forall t$). Therefore, the visitation frequencies we consider are : $\frac{T\upsilon}{k^n-1}$ for any suboptimal action and $T(1-\upsilon)$ for the optimal action. To compute the upper bound on δ, we modify the objective for the quadratic program in Appendix A.1.1 as $X^T diag(a', b', 1))X$ where $a' \leftarrow \frac{a\upsilon}{(1-\upsilon)(a+b)}, b' \leftarrow \frac{b\upsilon}{(1-\upsilon)(a+b)}$ in accordance with our visitations. Next, using the same reasoning as in Eq. (A.1), we get that QMIX learns the suboptimal policy for

$$0 < \delta \leq R\left[\sqrt{a\left(\frac{\upsilon b}{(1-\upsilon)(a+b)} + 1\right)} - 1\right]. \tag{A.2}$$

Note that the upper bound of δ in Eq. (A.2) is probabilistic in nature. Therefore, we provide a lower bound on the probability of this by considering the RHS of Eq. (A.2) with $\upsilon \leftarrow \upsilon/2$ and bounding the probability of deviation from the worst case visitation frequencies. By making use of the Hoeffding's lemma, we derive that:

$$P\Big[\text{empirical frequency of optimal} - \upsilon T \geq \frac{T\upsilon}{2}\Big] \leq \exp(-\frac{T\upsilon^2}{2}),$$
$$P\Big[\text{empirical frequency of suboptimal} - \frac{T\upsilon}{k^n-1} \leq -\frac{T\upsilon}{2(k^n-1)}\Big] \leq \exp(-\frac{T\upsilon^2}{2(k^n-1)^2}).$$

Finally, by using the union bound, we conclude that with probability $\geq 1 - \Big(\exp(-\frac{T\upsilon^2}{2}) + (k^n-1)\exp(-\frac{T\upsilon^2}{2(k^n-1)^2})\Big)$, QMIX fails to learn the optimal policy

for

$$0 < \delta \leq R\left[\sqrt{a\left(\frac{\upsilon b}{2(1-\upsilon/2)(a+b)}+1\right)}-1\right]$$

□

A.1.3 Additional suboptimality results

Theorem A.3 (Uniform visitation VDN)**.** *For n player, $k \geq 3$ action matrix games ($|\mathcal{A}| = n, |U| = k$), under uniform visitation; Q_{vdn} learns a δ-suboptimal policy for any time horizon T, for any $0 < \delta \leq R\left[\binom{k+n-3}{n-1} - 1\right]$ for the payoff matrix (n dimensional) given by the template above, $R > 0$.*

Proof. Once again, for single state MDPs, under uniform visitation of the joint state-action space, VDN can be seen as minimising the mean squared error between the actual Q-values and the sum factored projection $Q_{vdn}(s, \mathbf{u}) = \sum_n q_i(s, u^i)$. Using the symmetry of the given problem and an exchange argument, it can be shown that the problem can be reduced to finding a single vector $q, |q| = k$ such that $q_i = q \forall i$ which minimises the unconstrained quadratic objective:

$$\underset{q}{\text{minimise}} \quad \mathcal{L}(q) = \sum_{\mathbf{u}} \left(M(\mathbf{u}) - \sum_{i=1}^{n} q^{\mathbf{u}(i)}\right)^2$$

where M is the n dimensional payoff matrix, q^j represents the jth entry of q, and with slight abuse of notation we set $\mathbf{u}$ to represent the an n tuple of indices corresponding to the actions ie. $\mathbf{u} \in \{1..k\}^n$. Since the problem is unconstrained we can directly solve for the problem by setting $\nabla_q \mathcal{L} = 0$, which gives rise to system of linear equations: $Aq = b$. The tricky part however lies in identifying the entries in A, b. For this we consider the generating polynomial $\mathbb{P}(n, k) = (\sum_{j=1}^{k} q^j)^n$. It is then

evident that:

$$A_{ii} = 2\frac{\partial}{\partial q^i}\Big(q^i \frac{\partial \mathbb{P}}{\partial q^i}\Big)|_{\mathbf{q}=\mathbf{1}} = 2nk^{n-2}(k+n-1)$$
$$A_{ij} = 2\frac{\partial^2 \mathbb{P}}{\partial q^i \partial q^j}|_{\mathbf{q}=1} = 2nk^{n-2}(n-1), i \neq j$$

Similarly using a careful counting argument via the hockey stick identity it can be shown that :

$$b_1 = 2nR\Big[k^{n-1} - \binom{k+n-3}{n-1}\Big] + 2n(R+\delta)$$
$$b_i = 2nR\Big[k^{n-1} - \binom{k+n-i-2}{n-1}\Big], i > 1$$

Matrix A can be easily inverted given its special structure which then gives analytic solution for q. Note that the optimal action corresponds to the tuple $\mathbf{u} = \mathbf{1}$ where $\mathbf{1} = (1, ..1), n$ times. Finally for the suboptimality of the policy learnt we have that it is sufficient to show:

$$q^1 \leq q^i, \forall i > 1$$
$$\implies 0 < \delta \leq R\Big[\binom{k+n-3}{n-1} - 1\Big]$$

□

Note that the above two upper bounds in Theorems 3.1 and A.3 for the uniform visitation case are tight further the latter bound is $\mathcal{O}(R\max\{k,n\}^{n-2})$ in comparison to the former which is of $\mathcal{O}(R\max\{k,n\}^{\frac{n}{2}})$.

Theorem A.4 (Uniform visitation IQL)**.** *For n player, $k \geq 3$ action matrix games ($|\mathcal{A}| = n, |U| = k$), under uniform visitation; Q_{iql} learns a δ-suboptimal policy for any time horizon T, for any $0 < \delta \leq R\Big[\binom{k+n-3}{n-1} - 1\Big]$ for the payoff matrix (n dimensional) given by the template above, $R > 0$.*

Proof. The case for IQL is simplest to analyse for the matrix game above. Due to the symmetry of the problem, each agent i ends up learning the same vector q of size k that minimises the objective:

$$\underset{q}{\text{minimise}} \quad \mathcal{L}(q) = \mathbb{E}_{\mathbf{u}}\Big[\Big(M(\mathbf{u}) - q^{\mathbf{u}(i)}\Big)^2\Big]$$

The minimiser for the above problem can be found by setting the partial derivative w.r.t. each of the components q_j equal to 0:

$$\begin{aligned}
&\mathcal{L}(q) = \mathbb{E}_{u^i}\Big[\mathbb{E}_{u^{-i}|\mathbf{u}(i)=j}\Big[\Big(M(\mathbf{u}) - q^j\Big)^2|\mathbf{u}(i)=j\Big]\Big] \\
\implies &\frac{\partial \mathcal{L}(q)}{\partial q^j} = -2P(\mathbf{u}(i)=j)\mathbb{E}_{u^{-i}|\mathbf{u}(i)=j}\Big[\Big(M(\mathbf{u}) - q^j\Big)|\mathbf{u}(i)=j\Big]\Big] \\
\implies &q^j = \mathbb{E}_{u^{-i}|\mathbf{u}(i)=j}[M(\mathbf{u})|\mathbf{u}(i)=j]
\end{aligned}$$

which under uniform visitation is just the mean payoff holding the agent i's action fixed to j, this gives $q^1 = \frac{R(1+c)+\delta}{k^{n-1}}, q^k = R$, where $c = k^{n-1} - \binom{k+n-3}{n-1}$ Once again solving for:

$$\begin{aligned}
&q^1 \leq q^i, \forall i > 1 \\
\implies &0 < \delta \leq R\Big[\binom{k+n-3}{n-1} - 1\Big]
\end{aligned}$$

□

A.1.4 Variational Mutual Information lower bound

Let the posterior over z be given by $\log(p(z|\sigma(\mathbf{u}, s)))$ and the variational approximation by $q_v(z|\sigma(\mathbf{u}, s)))$

$$
\begin{aligned}
\mathcal{J}_{MI} &= \mathcal{H}(\sigma(\mathbf{u}, s)) - \mathcal{H}(\sigma(\mathbf{u}, s)|z) \\
&= \mathcal{H}(z) - \mathcal{H}(z|\sigma(\mathbf{u}, s)) && \{\text{MI is symmetric}\} \\
&= \mathcal{H}(z) + \mathbb{E}_{\sigma(\mathbf{u},s)}[\mathbb{E}_z[\log(p(z|\sigma(\mathbf{u}, s)))]] && \{\text{Def. conditional entropy}\} \\
&= \mathcal{H}(z) + \mathbb{E}_{\sigma(\mathbf{u},s)}[\mathbb{E}_z[\log(p(z|\sigma(\mathbf{u}, s))) - \log(q_v(z|\sigma(\mathbf{u}, s)) + \log(q_v(z|\sigma(\mathbf{u}, s))]] \\
&= \mathcal{H}(z) + \mathbb{E}_{\sigma(\mathbf{u},s)}[\mathbb{E}_z[\log(q_v(z|\sigma(\mathbf{u}, s))]] + \mathbb{E}_{\sigma(\mathbf{u},s)}[KL(p(z|\sigma(\mathbf{u}, s))||q_v(z|\sigma(\mathbf{u}, s))] \\
&\geq \mathcal{H}(z) + \mathbb{E}_{\sigma(\mathbf{u},s),z}[\log(q_v(z|\sigma(\mathbf{u}, s)))] && \{\text{KL is non negative}\}
\end{aligned}
$$

A.2 QMIX Architecture

Fig. A.1 gives architecture for QMIX. The components here are (a) Mixing network structure. In red are the hypernetworks that produce the weights and biases for mixing network layers shown in blue. (b) The overall QMIX architecture. (c) Agent network structure.

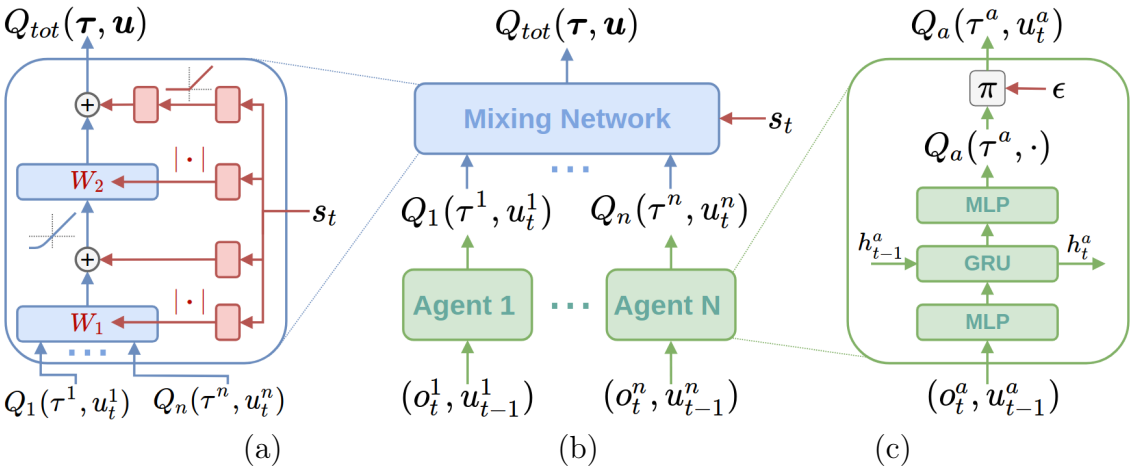


Figure A.1: The overall setup of QMIX

A.3 Experimental Setup

A.3.1 Architecture and Training

All agent are designed as Deep Recurrent Q-Networks [89]. At each time step, each agent network receives a local observation as input, which is fed to a 64-dimensional fully-connected hidden layer, followed by a GRU recurrent layer and a fully-connected layer with $|U|$ outputs. To speed up the learning, all agent networks share the same set of parameters. A one-hot encoded agent id is concatenated to agent observations. The architectures for mixing and utility networks are the same as in [172].

For all experiments we update the target networks after every 200 episodes. We set $\gamma = 0.99$. The optimisation is conducted using RMSprop with a learning rate of 5×10^{-4} and $\alpha = 0.99$ with no weight decay or momentum.

SMAC

Exploration for QMIX is performed during training during which each agent executes ϵ-greedy policy over its own actions. ϵ is annealed from 1.0 to 0.05 or 0.005 over $50k$ time steps and is kept constant afterwards.

We utilise a replay buffer of the most recent 5000 environment steps. A single training step for a batch of size 32 entire episodes is performed after every episodes.

We set $Z = 16$ for all the experiments. We set $\lambda_{MI} = 0.001$ and $\lambda_{QL} = 1$. Unless otherwise mentioned, all MAVEN experiments use the trajectory-based MI loss. We use an entropy regularisation term with a coefficient of 0.001 for the hierarchical policy. We set the final value of ϵ to 0.05 for MAVEN ans QMIX.

All SMAC experiments use the default reward and observation settings of the SMAC benchmark [181].

We run all methods for 10 million environmental steps. This takes approximately 36

hours on a NVIDIA GTX 1080Ti GPU for 12 random initializations.

m-step matrix games

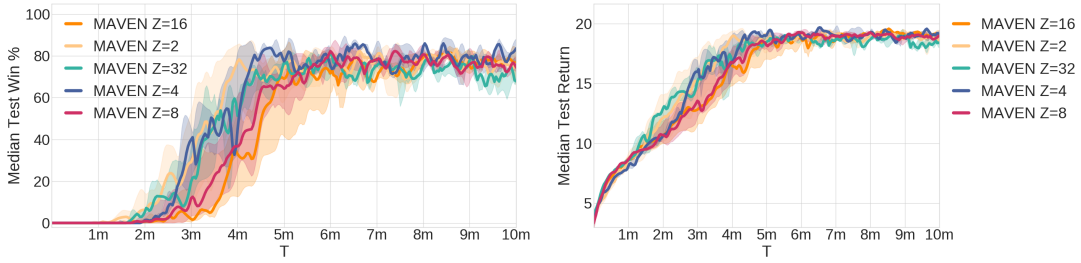
All methods anneal ϵ from 1 to 0.01 over 100 timesteps and keep it constant afterwards.

A single training step for a batch of size 32 is conducted after every episode.

All methods are run for $100k$ timesteps.

For MAVEN we set $Z = 16, \lambda_{MI} = 1, \lambda_{QL} = 1$ and use an entropy regularisation term with a coefficient of 0.001 for the hierarchical policy.

A.3.2 Additional plots & ablations

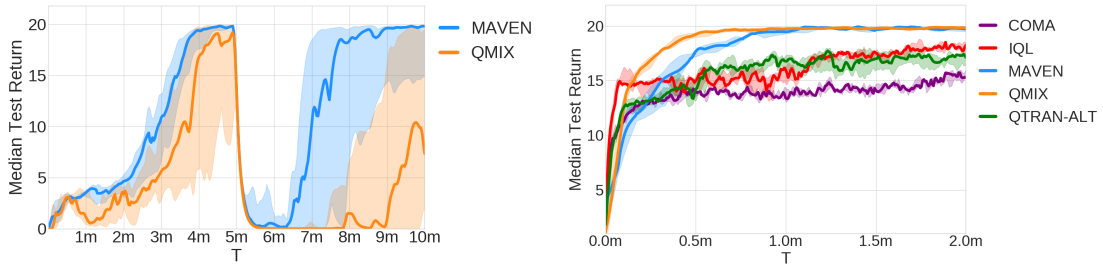


(a) Varying the values for Z

(b) Policy returns for different Z

Figure A.2: Performance with varying the number of latent variable categories

We also consider varying the number of categories for the discrete latent variable Fig. A.2(a). While the number of categories loosely correlates with performance, it was not always the case. For `micro_corridor`, the results are inconclusive because they all use the same budget of gradient updates, yielding two opposing factors that cancel out (more z's vs. less training per z). Fig. A.2(b) gives the returns of the corresponding policies learnt.



(a) `2-corridors` (b) `2s3z`

(c) `micro_corridor` (d) `micro_focus`

Figure A.3: Median test returns on SMAC scenarios.

Appendix B

Appendix for Chapter 4

B.1 Additional Proofs

B.1.1 Proof of Theorem 4.1

Theorem B.1. *For a finite MMDP the action-value tensor satisfies* $rank(\hat{Q}^{\pi}(s)) \leq k_1 + k_2|S|, \forall s \in S, \forall \pi$.

Proof. We first unroll the Tensor Bellman equation in Fig. 4.3. The first term $\hat{R}$ has bounded rank k_1 by Assumption 4.1. Next, each contraction term on the RHS is a linear combination of $\{\hat{P}(s, s')\}_{s' \in S}$ each of which has bounded rank k_2 (Assumption 4.2). The result follows from the sub-additivity of CP-rank. □

B.1.2 Proof of Theorem 4.2

Theorem B.2 (Model based estimation of $\hat{R}, \hat{P}$ error bounds)**.** *Given any* $\epsilon > 0, 1 > \delta > 0$*, for a policy* π *with the policy tensor satisfying* $\pi(\mathbf{u}|s) \geq \Delta$*, where*

$$\Delta = \max_s \frac{C_1 \mu_s^6 k^5 (w_s^{max})^4 \log(|U|)^4 \log(3k||R(s)||_F/\epsilon)}{|U|^{n/2} (w_s^{min})^4} \tag{B.1}$$

and C_1 is a problem dependent positive constant. There exists N_0 which is $O(|U|^{\frac{n}{2}})$ and polynomial in $\frac{1}{\delta}, \frac{1}{\epsilon}, k$ and relevant spectral properties of the underlying MDP dynamics such that for samples $\geq N_0$, we can compute the estimates $\bar{R}(s), \bar{P}(s,s')$ such that w.p. $\geq 1-\delta$, $||\bar{R}(s) - \hat{R}(s)||_F \leq \epsilon, ||\bar{P}(s,s') - \hat{P}(s,s')||_F \leq \epsilon, \forall s, s' \in S$.

Proof. For the simplicity of notation and emphasising key points of the proof, we focus on orthogonal symmetric tensors with $n = 3$. Guidelines for more general cases are provided by the end of the proof.

We break the proof into three parts: Let policy π satisfy $\pi(\mathbf{u}|s) \geq \Delta$ Eq. (B.1). Let ρ be the stationary distribution of π (exists by Assumption 4.3) and let $N_1 = \max_s \frac{1}{\rho(s)} \log\left(\frac{12\sqrt{k}||R(s)||_F}{\epsilon}\right)$. From N_1 samples drawn from ρ by following π, we estimate $\bar{R}$, the estimated reward tensor computed by using Algorithm 1 in [102]. We have by application of union bound along with Theorem 1.1 in [102] for each $s \in S$, w.p. $\geq 1 - |U|^{-5} \log_2\left(\frac{12\sqrt{k}\prod_s ||R(s)||_F}{\epsilon}\right) = p_\epsilon$, $||\bar{R}(s) - \hat{R}(s)||_F \leq \epsilon/3, \forall s \in S$. We now provide a boosting scheme to increase the confidence in the estimation of $\hat{R}(\cdot)$ from p_ϵ to $1 - \delta/3$. Let $\eta = \frac{1}{2}\left(p_\epsilon - \frac{1}{2}\right) > 0$ (for clarity of the presentation we assume $p_\epsilon > \frac{1}{2}$ and refer the reader to [108] for the other more involved case). We compute M independent estimates $\{\bar{R}_i, i \in \{1..M\}\}$ for $\hat{R}(s)$ and find the biggest cluster $\mathcal{C} \subseteq \{\bar{R}_i\}$ amongst the estimates such that for any $\bar{R}_i, \bar{R}_j \in \mathcal{C}, ||\bar{R}_i - \bar{R}_j||_F \leq \frac{2\epsilon}{3}$. We then output any element of $\mathcal{C}$. Intuitively as $p_\epsilon > \frac{1}{2}$, most of the estimates will be near the actual value $\hat{R}(s)$, this can be confirmed by using the Hoeffding Lemma[108]. It follows that for $M \geq \frac{1}{2\eta^2} \ln(\frac{3|S|}{\delta})$ the output of the above procedure satisfies $||\bar{R}(s) - \hat{R}(s)||_F \leq \epsilon$ w.p. $\geq 1 - \frac{\delta}{3|S|}$ for any particular s. Thus MN_1 samples from stationary distribution are sufficient to ensure that for all $s \in S$, w.p. $\geq 1 - \delta/3$, $||\bar{R}(s) - \hat{R}(s)||_F \leq \epsilon$.

Secondly we note that $\hat{P}(s,s')$ for any $s, s' \in S$ is a tensor whose entries are the parameters of a Bernoulli distribution. Under Assumption 4.2, it can be seen as a latent topic model [6] with k factors, $\hat{P}(s,s') = \sum_{r=1}^{k} w_{s,s',r} \otimes^n u_{s,s',r}$. Moreover it

satisfies the conditions in Theorem 3.1 [6] so that $\exists N_2 = \max_{s,s'} \frac{1}{\rho(s)} N_2(s,s')$ where each $N_2(s,s')$ is $\mathcal{O}\Big(\frac{k^{10}|S|^2 \ln^2(3|S|/\delta)}{\delta^2 \epsilon'^2}\Big)$ depending on the spectral properties of $\hat{P}(s,s')$ as given in the theorem and satisfies $||u_{s,s',r}^{-} - u_{s,s',r}||_2 \leq \epsilon'$ on running Algorithm B in [6] w.p. $\geq 1 - \frac{\delta}{3|S|}$. We pick $\epsilon' = \frac{\epsilon}{7n^2 k \mu_{s,s'}^2 (w_{s,s'}^{\max})^2}$ so that $||\bar{P}(s,s') - \hat{P}(s,s')||_F \leq \epsilon, \forall s, s' \in S$. We filter off the effects of sampling from a particular policy by using lower bound constraint in Eq. (B.1) and sampling $\frac{N_2}{\Delta}$ samples.

Finally we account for the fact that there is a delay in attaining the stationary distribution ρ and bound the failure probability of significantly deviating from ρ empirically. Let $\rho' = \min_s \rho(s)$ and $t_{\text{mix},\pi}(x)$ represent the minimum number of samples that need to drawn from the Markov chain formed by fixing policy π so that for the state distribution $\rho_t(s)$ at time step $t = t_{\text{mix},\pi}(x)$ we have $TV(\rho_t - \rho) \leq x$ for any starting state $s \in S$ where $TV(\cdot,\cdot)$ is the total variation distance. We let the policy run for a burn in period of $t' = t_{\text{mix},\pi}(\rho'/4)$. For a sample of N_3 state transitions after the burn in period, let $\bar{\rho}$ represent the empirical state distribution. By applying the Hoeffding lemma for each state, we get: $P(|\bar{\rho}(s) - \rho_{t'}(s)| \geq \rho'/4) \leq 2\exp\Big(\frac{-N_3 \rho'^2}{8}\Big)$, so that for $N_3 \geq \frac{8}{\rho'^2} \ln\Big(\frac{6|S|}{\delta}\Big)$ we have w.p. $\geq 1 - \frac{\delta}{3|S|}$, $|\bar{\rho}(s) - \rho(s)| < \rho'/2, \forall s \in S$.

Putting everything together we get with $t_{\text{mix},\pi}(\rho'/4) + \max\{2MN_1, \frac{2N_2}{\Delta}, N_3\}$ samples, the underlying reward and probability tensors can be recovered such that w.p. $\geq 1 - \delta$, $||\bar{R}(s) - \hat{R}(s)||_F \leq \epsilon, ||\bar{P}(s,s') - \hat{P}(s,s')||_F \leq \epsilon, \forall s, s' \in S$.

For extending the proof to the case of non-orthogonal tensors, we refer the reader to use whitening transform as elucidated in [5]. Likewise for asymmetric, higher order ($n > 3$) tensors methods shown in [102, 5, 6] should be used. Finally for the case of M-POMDP and M-ROMDP, the corresponding results for single agent POMDP and ROMDP should be used, as detailed in [9, 10] respectively. □

B.1.3 Proof of Lemma 4.1

Lemma B.1. *For transition tensor estimates satisfying* $||\bar{P}(s,s') - \hat{P}(s,s')||_F \leq \epsilon$, *we have for any given state and action pair* s, a, *the distribution over the next states follows:* $TV(P'(\cdot|s,a), P(\cdot|s,a)) \leq \frac{1}{2}(|1-f| + f|S|\epsilon)$ *where* $\frac{1}{1+\epsilon|S|} \leq f \leq \frac{1}{1-\epsilon|S|}$. *Similarly for any policy* π, $TV(\bar{P}_\pi(\cdot|s), P_\pi(\cdot|s)), TV(\bar{P}_\pi(s',a'|s), P_\pi(s',a'|s)) \leq \frac{1}{2}(|1-f| + f|S|\epsilon)$

Proof. Let $\bar{P}(\cdot|s,a)$ be the next state probability estimates obtained from the tensor estimates. We next normalise them across the next states to get the (estimated)distribution $P'(\cdot|s,a) = f\bar{P}(\cdot|s,a)$ where $f = \frac{1}{\sum_{s'} \bar{P}(s'|s,a)}$. Dropping the conditioning for brevity we have:

$$\begin{aligned} TV(P', P) &= \frac{1}{2}\sum_{s'} |P(s') - f\bar{P}(s')| \\ &\leq \frac{1}{2}(\sum_{s'} |P(s') - fP(s')| + |fP(s') - \bar{P}(s')|) \\ &= \frac{1}{2}(|1-f| + f|S|\epsilon) \end{aligned}$$

The other two results follow using the definition of TV and Fubini's theorem followed by reasoning similar to above. □

B.1.4 Proof of Theorem 4.3

Theorem B.3 (Error bound on policy evaluation)**.** *Given a behaviour policy* π_b *satisfying the conditions in Theorem 4.2 and being executed for steps* $\geq N_0$, *we have that for any policy* π *the model based policy evaluation* $Q^\pi_{\bar{P},\bar{R}}$ *satisfies:*

$$|Q^\pi_{P,R}(s,a) - Q^\pi_{\bar{P},\bar{R}}(s,a)| \leq (|1-f| + f|S|\epsilon)\frac{\gamma}{2(1-\gamma)^2} + \frac{\epsilon}{1-\gamma}, \forall (s,a) \in S \times U^n$$

where f *is as defined in Lemma 4.1.*

Proof. Let $\bar{P}, \bar{R}$ be the estimates obtained after running the procedure as described in Theorem 4.2 with samples corresponding to error ϵ and confidence $1-\delta$. We will bound the error incurred in estimation of the action-values using $\bar{P}, \bar{R}$. We have for any π by using triangle inequality

$$|Q^{\pi}_{P,R}(s,a) - Q^{\pi}_{\bar{P},\bar{R}}(s,a)| \leq |Q^{\pi}_{P,R}(s,a) - Q^{\pi}_{\bar{P},R}(s,a)| + |Q^{\pi}_{\bar{P},R}(s,a) - Q^{\pi}_{\bar{P},\bar{R}}(s,a)| \tag{B.2}$$

where we use the subscript to denote whether actual or approximate values are used for P, R respectively. We first focus on the first term on the RHS of Eq. (B.2). Let $R_{\pi}(s_t) = \sum_{a_t} \pi(a_t|s_t) R(s_t, a_t)$. We use $P_{t,\pi}(\cdot|s) = (P_{\pi}(\cdot|s))^t$ to denote the state distribution after t time steps. Consider a horizon h interleaving Q estimate given by:

$$Q^{\pi}_h(s,a) = R(s_t,a_t) + \sum_{t=1}^{h-1} \gamma^t \mathbb{E}_{\bar{P}_{t,\pi}(\cdot|s)}[R_{\pi}(s_t)] + \sum_{t=h}^{\infty} \gamma^t \mathbb{E}_{P_{t-h,\pi}(\cdot|s_h)\cdot \bar{P}_{h,\pi}(s_h|s)}[R_{\pi}(s_t)]$$

Where $s_0 = s, a_0 = a$ and the first h steps are unrolled according to $\bar{P}_{\pi}$, the rest are done using the true transition P_{π}. We have that:

$$|Q^{\pi}_{P,R}(s,a) - Q^{\pi}_{\bar{P},\bar{R}}(s,a)| = |Q^{\pi}_0(s,a) - Q^{\pi}_{\infty}(s,a)| \leq \sum_{h=0}^{\infty} |Q^{\pi}_h(s,a) - Q^{\pi}_{h+1}(s,a)|$$

Each term in the RHS of the above can be independently bounded as :

$$\begin{aligned} |Q^{\pi}_h(s,a) - Q^{\pi}_{h+1}(s,a)| =& \gamma^{h+1} \Big| \mathbb{E}_{\bar{P}_{h+1,\pi}(s_{h+1}|s)} \Big[\sum_{a_{h+1}} \pi(a_{h+1}|s_{h+1}) Q^{\pi}_{\infty}(s_{h+1}.a_{h+1}) \Big] \\ & - \mathbb{E}_{P_{\pi}\bar{P}_{h,\pi}(s_{h+1}|s)} \Big[\sum_{a_{h+1}} \pi(a_{h+1}|s_{h+1}) Q^{\pi}_{\infty}(s_{h+1}.a_{h+1}) \Big] \Big| \end{aligned}$$

As the rewards are bounded we get the expression above is $\leq \frac{1}{1-\gamma} \gamma^{h+1} TV(\bar{P}_{\pi}(s',a'|s), P_{\pi}(s',a'|s))$. Finally using Lemma 4.1 we get $\leq (\frac{1}{2}(|1-f| + f|S|\epsilon)) \frac{\gamma^{h+1}}{1-\gamma}$. And plugging in the

original expression:

$$|Q^{\pi}_{P,R}(s,a) - Q^{\pi}_{\bar{P},\bar{R}}(s,a)| \leq (|1-f| + f|S|\epsilon)\frac{\gamma}{2(1-\gamma)^2}$$

Next the second term on the RHS of Eq. (B.2) can easily be bounded by $\frac{\epsilon}{1-\gamma}$ which gives:

$$|Q^{\pi}_{P,R}(s,a) - Q^{\pi}_{\bar{P},\bar{R}}(s,a)| \leq (|1-f| + f|S|\epsilon)\frac{\gamma}{2(1-\gamma)^2} + \frac{\epsilon}{1-\gamma}$$

□

B.2 Discussion

B.2.1 Relation to other methods

In this section we study the relationship between TESSERACT and some of the existing methods for MARL.

FQL

FQL [39] uses a learnt inner product space to represent the dependence of joint Q-function on pair wise agent interactions. The following result shows containment of FQL representable action-value function by TESSERACT :

Proposition B.1. *The set of joint Q-functions representable by FQL is a subset of that representable by* TESSERACT.

Proof. In the most general form, any join Q-function representable by FQL has the form:

$$Q_{fql}(s,\mathbf{u}) = \sum_{i=1:n} q_i(s,u_i) + \sum_{i=1:n,j<i} \langle f_i(s,u_i), f_j(s,u_j)\rangle$$

where $q_i : S \times U \to \mathbb{R}$ are individual contributions to joint Q-function and f_i :

$S \times U \to \mathbb{R}^d$ are the vectors describing pairwise interactions between the agents. There are $\binom{n}{2}$ pairs of agents to consider for (pairwise)interactions. Let $\mathscr{P} \triangleq (i,j)$ be the ordered set of agent pairs where $i > j$ and $i, j \in \{1..n\}$, let $\mathscr{P}_k$ denote the kth element of $\mathscr{P}$. Define membership function $m : \mathscr{P} \times \{1..n\} \to \{0,1\}$ as:

$$m((i,j),x) = \begin{cases} 1 & \text{if } x = i \vee x = j \\ 0 & \text{otherwise} \end{cases}$$

Define the mapping $v_i : S \to \mathbb{R}^{|U| \times D}$ where $D = d\binom{n}{2} + n$ and $v_{i,k}$ represents the kth column of v_i.

$$v_i(s) \triangleq \begin{cases} v_i(s)[j,(k-1)d+1:kd] = f_i(s,u_j) & \text{if } m(\mathscr{P}_k,i) = 1 \\ v_i(s)[j,D-n+i] = q_i(s,u_j) & \\ v_i(s)[j,k] = 1 & \text{otherwise} \end{cases}$$

We get that the tensors:

$$Q_{fql}(s) = \sum_{k=1}^{D} \otimes^n v_{i,k}(s)$$

Thus any Q_{fql} can be represented by TESSERACT, note that the converse is not true ie. any arbitrary Q-function representable by TESSERACT may not be representable by FQL as FQL cannot model higher-order (> 2 agent) interactions. □

VDN

VDN [198] learns a decentralisable factorisation of the joint action-values by expressing it as a sum of per agent utilities $\hat{Q} = \oplus^n u_i, i \in \{1..n\}$. This can be equivalently learnt in TESSERACT by finding the best rank one projection of $\exp(\hat{Q}(s))$. We formalise this in the following result:

Proposition B.2. *For any MMDP, given policy π having Q function representable by VDN ie. $\hat{Q}^\pi(s) = \oplus^n u_i(s), i \in \{1..n\}$, $\exists v_i(s) \forall s \in S$, the utility factorization can*

be recovered from rank one CP-decomposition of $\exp(\hat{Q}^\pi)$

Proof. We have that :

$$\begin{aligned}\exp(\hat{Q}^\pi(s)) &= \exp(\oplus^n u_i(s)) \\ &= \otimes^n \exp(u_i(s))\end{aligned}$$

Thus $(\exp(u_i(s)))_{i=1}^n \in \arg\min_{v_i(s)} ||\exp(\hat{Q}^\pi(s)) - \otimes^n v_i(s)||_F \forall s \in S$ and there always exist $v_i(s)$ that can be mapped to some $u_i(s)$ via exponentiation. In general any Q-function that is representable by VDN can be represented by TESSERACT under an exponential transform (Section 4.3.2). □

B.2.2 Injecting Priors for Continuous Domains

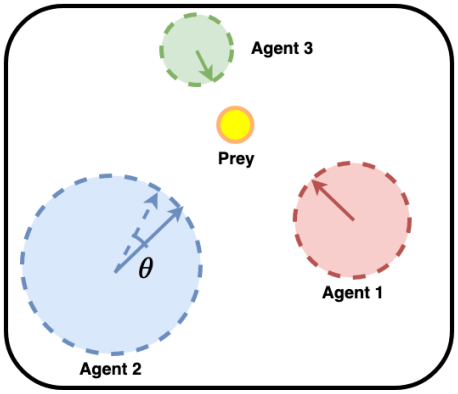


Figure B.1: Continuous actions task with three agents chasing a prey. Perturbing Agent 2's action direction by small amount θ leads to a small change in the joint value.

We now discuss the continuous action setting. Since the action set of each agent is infinite, we impose further structure while maintaining appropriate richness in the hypothesis class of the proposed action value functions. Towards this we present an example of a simple prior for TESSERACT for continuous action domains. WLOG, let $U \triangleq \mathbb{R}^d$ for each agent $\in 1..n$. We are now interested in the function class given by $\mathcal{Q} \triangleq \{Q : S \times U^n \to \mathbb{R}\}$ where each $Q(s) \triangleq \langle T(s, \{||u^i||_2\}), \otimes^n u^i \rangle$, here $T(\cdot) : S \times \mathbb{R}^n \to \mathbb{R}^{d^n}$ is a function that outputs an order n tensor and is invariant

to the direction of the agent actions, $\langle \cdot, \cdot \rangle$ is the dot product between two order n tensors and $|| \cdot ||_2$ is the Euclidean norm. Similar to the discrete case, we define $\mathcal{Q}_k \triangleq \{Q : Q \in \mathcal{Q} \wedge rank(T(\cdot)) = k, \forall s \in S\}$. The continuous case subsumes the discrete case with $T(\cdot) \triangleq Q(\cdot)$ and actions encoded as one hot vectors. We typically use rich classes like deep neural nets for Q and T parametrised by ϕ.

We now briefly discuss the motivation behind the example continuous case formulation: for many real world continuous action tasks the joint payoff is much more sensitive to the magnitude of the actions than their directions, i.e., slightly perturbing the action direction of one agent while keeping others fixed changes the payoff by only a small amount (see Fig. B.1). Furthermore, T_ϕ can be arbitrarily rich and can be seen as representing utility per agent per action dimension, which is precisely the information required by methods for continuous action spaces that perform gradient ascent w.r.t. $\nabla_{u^i} Q$ to ensure policy improvement. Further magnitude constraints on actions can be easily handled by a rich enough function class for T. Lastly we can further abstract the interactions amongst the agents by learnable maps $f^i_\eta(u^i, s) : \mathbb{R}^d \times S \to \mathbb{R}^m$, $m >> d$ and considering classes $Q(s, \mathbf{u}) \triangleq \langle T(s, \{||u^i||\}), \otimes^n f^i_\eta(u^i) \rangle$ where $T(\cdot) : S \times \mathbb{R}^n \to \mathbb{R}^{m^n}$.

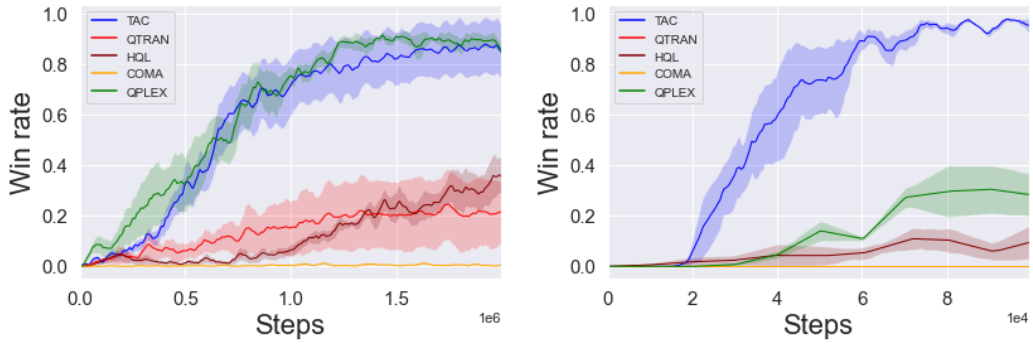
B.3 Additional experiments and details

B.3.1 StarCraft II

Figure B.2: The 2c_vs_64zg scenario in SMAC.

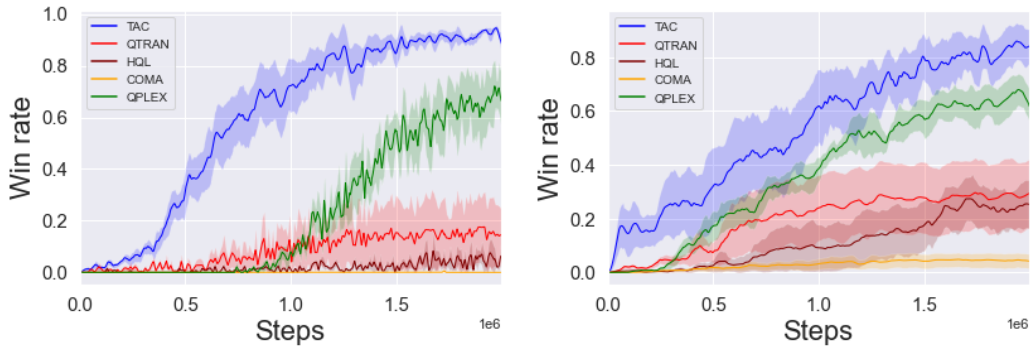
In the SMAC bechmark[181] (https://github.com/oxwhirl/smac), agents can `move` in four cardinal directions, `stop`, take `noop` (do nothing), or select an enemy to `attack` at each timestep. Therefore, if there are n_e enemies in the map, the action space for each ally unit contains $n_e + 6$ discrete actions.

Additional Experiments



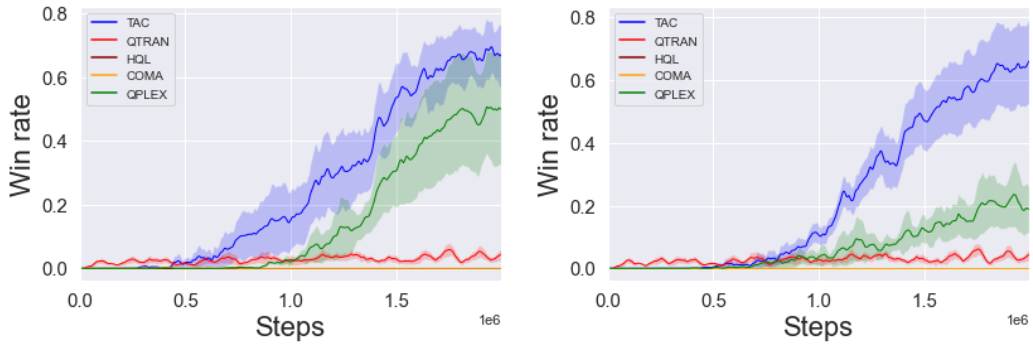
(a) 3s5z **Easy**

(b) 2s_vs_1sc **Easy**



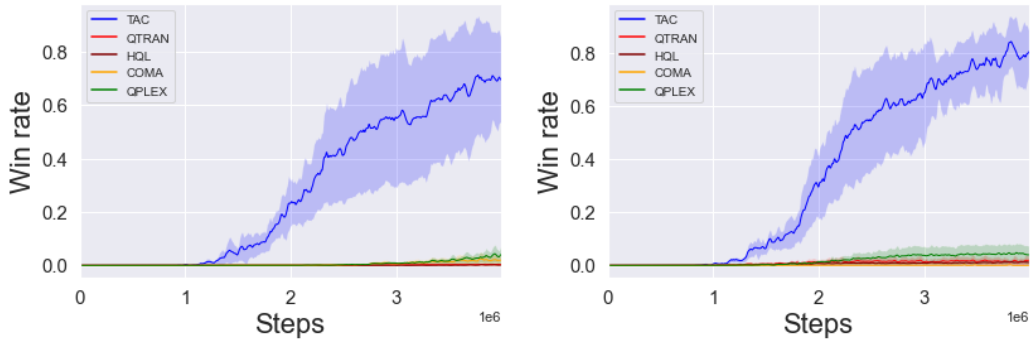
(c) 2c_vs_64zg **Hard**

(d) 5m_vs_6m **Hard**



(e) MMM2 **Super Hard**

(f) 27m_vs_30m **Super Hard**



(g) 6h_vs_8z **Super Hard**

(h) Corridor **Super Hard**

Figure B.3: Performance of different algorithms on different SMAC scenarios: TAC, QTRAN, QPLEX, COMA, HQL.

In addition to the baselines in main text Section 7.5, we also include 4 more baselines: QTRAN [188], QPLEX [236], COMA [63] and HQL. QTRAN tries to avoid the issues arising with representational constraints by posing the decentralised multi agent problem as optimisation with linear constraints, these constraints are relaxed using L2 penalties for tractability [140]. Similarly, QPLEX another recent method uses an alternative formulation using advantages for ensuring the *Individual Global Max* (IGM) principle [188]. COMA is an actor-critic method that uses a centralised critic for computing a counterfactual baseline for variance reduction by marginalising across individual agent actions. Finally, HQL uses the heuristic of differential learning rates on top of IQL [210] to address problems associated with decentralized exploration. **Fig. B.3** gives the average win rates of the baselines on different SMAC scenarios across five random runs (with one standard deviation shaded). We observe that TESSERACT outperforms the baselines by a large margin on most of the scenarios, especially on the **super-hard** ones on which the exiting methods struggle, this validates the sample efficiency and representational gains supported by our analysis. We observe that HQL is unable to learn a good policy on most scenarios, this might be due to uncertainty in the bootstrap estimates used for choosing the learning rate that confounds with difficulties arising from non-stationarity. We also observe that COMA does not yield satisfactory performance on any of the scenarios. This is possibly because it does not utilise the underlying tensor structure of the problem and suffers from a *lagging critic*. While QPLEX is able to alleviate the problems arising from relaxing the IGM constraints in QTRAN, it lacks in performance on the **super-hard** scenarios of Corridor and 6h_vs_8z.

Experimental Setup for SMAC

We use a factor network for the tensorised critic which comprises of a fully connected MLP with two hidden layers of dimensions 64 and 32 respectively and outputs a $r|U|$ dimensional vector. We use an identical policy network for the actors which

outputs a $|U|$ dimensional vector and a value network which outputs a scalar state-value baseline $V(s)$. The agent policies are derived using softmax over the policy network output. Similar to previous work [181], we use two layer network consisting of a fully-connected layer followed by GRU (of 64-dimensional hidden state) for encoding agent trajectories. We used Relu for non-linearities. All the networks are shared across the agents. We use ADAM as the optimizer with learning rate 5×10^{-4}. We use entropy regularisation with scaling coefficient $\beta = 0.005$. We use an approximation rank of 7 for Tesseract ('TAC') for the SMAC experiments. A batch size of 512 is used for training which is collected across 8 parallel environments (additional setup details in Appendix B.3.2). Grid search was performed over the hyper-parameters for tuning.

For the baselines QPLEX, QMIX, QTRAN, VDN, COMA, IQL we use the open sourced code provided by their authors at https://github.com/wjh720/QPLEX and https://github.com/oxwhirl/pymarl respectively which has hyper-parameters tuned for SMAC domain. The choice for architecture make the experimental setup of the neural networks used across all the baselines similar. We use a similar trajectory embedding network as mentioned above for our implementations of HQL and FQL which is followed by a network comprising of a fully connected MLP with two hidden layers of dimensions 64 and 32 respectively. For HQL this network outputs $|U|$ action utilities. For FQL, it outputs a $|U| + d$ vector: first $|U|$ dimension are used for obtaining the scalar contribution to joint Q-function and rest d are used for computing interactions between agents via inner product. We use ADAM as the optimizer for these two baselines. We use differential learning rates of $\alpha = 1 \times 10^{-3}, \beta = 2 \times 10^{-4}$ for HQL searched over a grid of $\{1, 2, 5, 10\} \times 10^{-3} \times \{1, 2, 5, 10\} \times 10^{-4}$. FQL uses the same learning rate 5×10^{-4} with $d = 10$ which was searched over set $\{5, 10, 15\}$.

The baselines use $\epsilon-$greedy for exploration with ϵ annealed from $1.0 \rightarrow 0.05$ over

50K steps. For super-hard scenarios in **SMAC** we extend the anneal time to 400K steps. We use temperature annealing for TESSERACT with temperature given by $\tau = \frac{2T}{T+t}$ where T is the total step budget and t is the current step. Similarly we use temperature $\tau = \frac{4T}{T+3t}$ for super-hard **SMAC** scenarios. The discount factor was set to 0.99 for all the algorithms.

Experiment runs take 1-5 days on a Nvidia DGX server depending on the size of the StarCraft scenario.

B.3.2 Techniques for stabilising TESSERACT critic training for Deep-MARL

- We used a gradient normalisation of 0.5. The parameters exclusive to the critic were separately subject to the gradient normalisation, this was done because the ratio of gradient norms for the actor and the critic parameters can vary substantially across training.
- We found that using multi-step bootstrapping substantially reduced target variance for Q-fitting and advantage estimation (we used the advantage based policy gradient $\int_S \rho^\pi(s) \int_{\mathbf{U}} \nabla \pi_\theta(\mathbf{u}|\mathbf{s}) \hat{A}^\pi(s, \mathbf{u}) d\mathbf{u} ds$ [201]) for **SMAC** experiments. Specifically for horizon T, we used the Q-target as:

$$Q_{target,t} = \sum_{k=1}^{T-t} \lambda^k g_{t,k}$$

$$g_{t,k} = r_t + \gamma r_{t+1} + ... + \gamma^k V(s_{t+k})$$

and similarly for value target. Likewise, the generalised advantage is estimated

as:

$$\hat{A}_t = \sum_{k=0}^{T-t} (\gamma\lambda)^k \delta_{t+k}$$

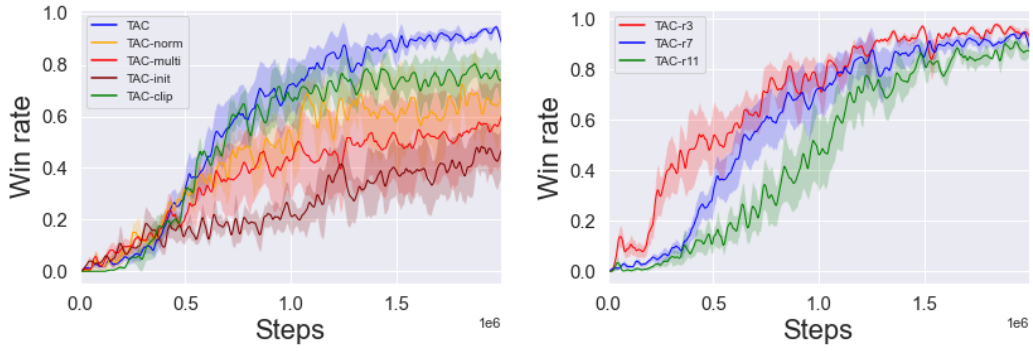
$$\delta_t = r_t + \gamma \hat{Q}(s_{t+1}, \mathbf{u}_{t+1}) - V(s_t)$$

Where $\hat{Q}$ is the tensor network output and the estimates are normalized by the accumulated powers of λ. We used $T = 64, \gamma = 0.99$ and $\lambda = 0.95$ for the experiments.

- The tensor network factors were squashed using a sigmoid for clipping and were scaled by 2.0 for **SMAC** experiments. Additionally, we initialised the factors according to $\mathcal{N}(0, 0.01)$ (before applying a sigmoid transform) so that value estimates can be effectively updated without the gradient vanishing.
- Similarly, we used clipping for the action-value estimates $\hat{Q}$ to prevent very large estimates:

$$clip(\hat{Q}_t) = min\{\hat{Q}_t, Ret_{max}\}$$

we used $Ret_{max} = 40$ for the **SMAC** experiments.



(a) Ablation on stabilisation techniques (b) Ablation on rank

Figure B.4: Variations on TESSERACT

We provide the ablation results on the stabilisation techniques mentioned above on

the 2c_vs_64zg scenario in Fig. B.4(a). The plot lines correspond to the ablations: TAC-multi: no multi-step target and advantage estimation, TAC-clip: no value upper bounding/clipping, TAC-norm: no separate gradient norm, TAC-init: no initialisation and sigmoid squashing of factors. We observe that multi-step estimation of target and advantage plays a very important role in stabilising the training, this is because noisy estimates can adversely update the learn factors towards undesirable fits. Similarly, proper initialisation plays a very important role in learning the Q-tensor as otherwise a larger number of updates might be required for the network to learn the correct factorization, adversely affecting the sample efficiency. Finally we observe that max-clipping and separate gradient normalisation do impact learning, although such effects are relatively mild.

We also provide the learning curves for TESSERACT as the CP rank of Q-approximation is changed, Fig. B.4(b) gives the learning plots as the CP-rank is varied over the set $\{3, 7, 11\}$. Here, we observe that approximation rank makes little impact on the final performance of the algorithm, however it may require more samples in learning the optimal policy. Our PAC analysis Theorem 4.2 also supports this.

B.3.3 Tensor games:

We introduce tensor games for our experimental evaluation. These games generalise the matrix games often used in 2-player domains. Formally, a tensor game is a cooperative MARL scenario described by tuple $(n, |U|, r)$ that respectively defines the number of agents (dimensions), the number of actions per agent (size of index set) and the rank of the underlying reward tensor Fig. B.5. Each agent learns a policy for picking a value from the index set corresponding to its dimension. The joint reward is given by the entry corresponding to the joint action picked by the agents, with the goal of finding the tensor entry corresponding to the maximum reward. We consider the CTDE setting for this game, which makes it additionally

challenging. We compare TESSERACT (TAC) with VDN, QMIX and independent actor-critic (IAC) trained using Reinforce [206]. Stateless games provide are ideal for isolating the effect of an exponential blowup in the action space. The natural difficulty knobs for stateless games are $|n|$ and $|U|$ which can be increased to obtain environments with large joint action spaces. Furthermore, as the rank r increases, it becomes increasingly difficult to obtain good approximations for $\hat{T}$.

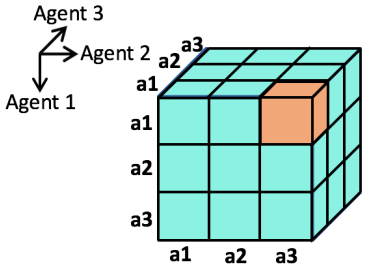
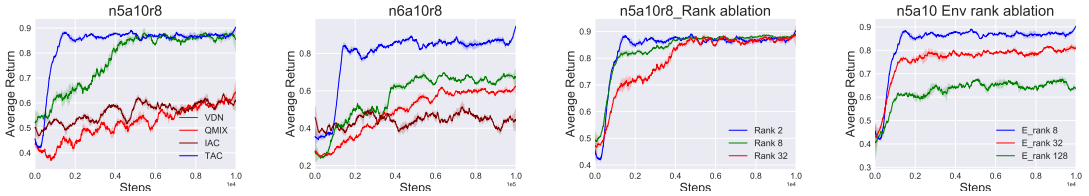


Figure B.5: Tensor games example with 3 agents (n) having 3 actions each (a). Optimal joint-action **(a1, a3, a1)** shown in orange.



(a) n:5 |U|:10 r:8 (b) n:6 |U|:10 r:8 (c) Dependence on approximation rank (d) Effects of approximation

Figure B.6: Experiments on tensor games.

Fig. B.6(a) Fig. B.6(b) present the learning curves for the algorithms for two game scenarios, averaged over 5 random runs with game parameters as mentioned in the figures. We observe that TESSERACT outperforms the other algorithms in all cases. Moreover, while the other algorithms find it increasingly difficult to learn good policies, TESSERACT is less affected by this increase in action space. As opposed to the IAC baseline, TESSERACT quickly learns an effective low complexity critic for scaling the policy gradient. QMIX performs worse than VDN due to the additional challenge of learning the mixing network.

In Fig. B.6(c) we study the effects of increasing the approximation rank of Tesseract

(k in decomposition $\hat{Q}(s) \approx T = \sum_{r=1}^{k} w_r \otimes^n g_{\phi,r}(s^i), i \in \{1..n\},$) for a fixed environment with 5 agents, each having 10 actions and the environment rank being 8. While all the three settings learn the optimal policy, it can be observed that the number of samples required to learn a good policy increases as the approximation rank is increased (notice delay in 'Rank 8', 'Rank 32' plot lines). This again is in-line with our PAC results, and makes intuitive sense as a higher rank of approximation directly implies more parameters to learn which increases the samples required to learn.

We next study how approximation of the actual Q tensors affects learning. In Fig. B.6(d) we compare the performance of using a rank-2 TESSERACT approximation for environment with 5 agents, each having 10 actions and the environment reward tensor rank being varied from 8 to 128. We found that for the purpose of finding the optimal policy, TESSERACT is fairly stable even when the environment rank is greater than the model approximation rank. However performance may drop if the rank mismatch becomes too large, as can be seen in Fig. B.6(d) for the plot lines 'E_rank 32', 'E_rank 128', where the actual rank required to approximate the underlying reward tensor is too high and using just 2 factors doesn't suffice to accurately represent all the information.

Experimental setup for Tensor games

For tensor game rewards, we sample k linearly independent vectors u_r^i from $|\mathcal{N}(0,1)^{|U|}|$ for each agent dimension $i \in \{1..n\}$. The reward tensor is given by $T = \sum_{r=1}^{k} w_r \otimes^n u_r^i, i \in \{1..n\}$. Thus T has roughly k local maxima in general for $k << |U|^n$. We normalise $\hat{T}$ so that the maximum entry is always 1.

All the agents use feed-forward neural networks with one hidden layer having 64 units for various components. Relu is used for non-linear activation.

The training uses ADAM [110] as the optimiser with a $L2$ regularisation of 0.001.

The learning rate is set to 0.01. Training happens after each environment step.

The batch size is set to 32. For an environment with n agents and a actions available per agent we run the training for $\frac{a^n}{10}$ steps.

For VDN [198] and QMIX[172] the ϵ-greedy coefficient is annealed from 0.9 to 0.05 at a linear rate until half of the total steps after which it is kept fixed.

For Tesseract ('TAC') and Independent Actor-Critic ('IAC') we use a learnt state baseline for reducing policy gradient variance. We also add entropy regularisation for the policy with coefficient starting at 0.1 and halved after every $\frac{1}{10}$ of total steps.

We use an approximation rank of 2 for Tesseract ('TAC') in all the comparisons except Fig. B.6(c) where it is varied for ablation.

Appendix C

Appendix for Chapter 5

C.1 Proofs

C.1.1 Generalisation between team compositions

Theorem C.1 (Generalisation between team compositions)**.** *Let team compositions* $\mathcal{T}^x, \mathcal{T}^y \in \mathcal{C}^n$ *with influence weights* $a^x, a^y \in \Delta_{n-1}$*,* $s_{max} = \max_s ||W_R s||_1$ *,* $V_{mid} = \frac{1}{2}\max_s V^*_{\mathcal{T}^y}(s)$*, Then*[*]*:*

$$|J^*_{\mathcal{T}^x} - J^*_{\mathcal{T}^y}| \leq \frac{s_{max} + \gamma d V_{mid}}{\gamma(1-\gamma)}\Psi, \textit{ where}$$

$$\Psi = \Big[|\sum_i a^x_i(\mathcal{T}^x_i - \mathcal{T}^y_i)|_\infty + |\sum_i (a^x_i - a^y_i)\mathcal{T}^y_i|_\infty\Big]$$

Proof. Let $\epsilon_R = \max_s |r_{\mathcal{T}^x}(s) - r_{\mathcal{T}^y}(s)|$ and $\epsilon_P = \max_{s,\mathbf{u}} 2 \cdot D_{TV}\Big(P_{\mathcal{T}^x}(\cdot|s,\mathbf{u}), P_{\mathcal{T}^y}(\cdot|s,\mathbf{u})\Big)$ where D_{TV} is the total variation distance. We have that:

$$\begin{aligned}
&|Q^*_{\mathcal{T}^x}(s,\mathbf{u}) - Q^*_{\mathcal{T}^y}(s,\mathbf{u})| \\
&= |r_{\mathcal{T}^x}(s) - r_{\mathcal{T}^y}(s) + \gamma\Big(\sum_{s'} P_{\mathcal{T}^x}(s'|s,\mathbf{u}) \max_{u'} Q^*_{\mathcal{T}^x}(s',\mathbf{u}') - \sum_{s'} P_{\mathcal{T}^y}(s'|s,\mathbf{u}) \max_{u'} Q^*_{\mathcal{T}^y}(s',\mathbf{u}'))\Big)|
\end{aligned}$$

[*] for $\gamma \in (0, \frac{\sqrt{5}-1}{2})$ we can replace $\frac{1}{\gamma(1-\gamma)}$ by $\frac{1+\gamma}{1-\gamma}$

$$
\begin{aligned}
&\leq |r_{\mathcal{T}^x}(s) - r_{\mathcal{T}^y}(s)| + \gamma\Big\{|\sum_{s'} P_{\mathcal{T}^x}(s'|s,\mathbf{u})\Big[\max_{u'} Q^*_{\mathcal{T}^x}(s',\mathbf{u}') - \max_{u'} Q^*_{\mathcal{T}^y}(s',\mathbf{u}')\Big]| \\
&\quad + |\sum_{s'}\Big[P_{\mathcal{T}^x}(s'|s,\mathbf{u}) - P_{\mathcal{T}^y}(s'|s,\mathbf{u})\Big](\max_{u'} Q^*_{\mathcal{T}^y}(s',\mathbf{u}') - V_{mid})|\Big\} \\
&\leq \epsilon_R + \gamma\Big\{\sum_{s'} P_{\mathcal{T}^x}(s'|s,\mathbf{u})|\max_{u'} Q^*_{\mathcal{T}^x}(s',\mathbf{u}') - \max_{u'} Q^*_{\mathcal{T}^y}(s',\mathbf{u}')| \\
&\quad + \sum_{s'} |P_{\mathcal{T}^x}(s'|s,\mathbf{u}) - P_{\mathcal{T}^y}(s'|s,\mathbf{u})||\max_{u'} Q^*_{\mathcal{T}^y}(s',\mathbf{u}') - V_{mid}|\Big\} \\
&\leq \epsilon_R + \gamma\Big\{\sum_{s'} P_{\mathcal{T}^x}(s'|s,\mathbf{u}) \max_{u'} |Q^*_{\mathcal{T}^x}(s',\mathbf{u}') - Q^*_{\mathcal{T}^y}(s',\mathbf{u}')| \\
&\quad + 2\cdot D_{TV}\Big(P_{\mathcal{T}^x}(s'|s,\mathbf{u}), P_{\mathcal{T}^y}(s'|s,\mathbf{u})\Big)V_{mid}\Big\} \\
&\leq \epsilon_R + \gamma\Big\{\max_{s',u'} |Q^*_{\mathcal{T}^x}(s',\mathbf{u}') - Q^*_{\mathcal{T}^y}(s',\mathbf{u}')| + \epsilon_P V_{mid}\Big\}
\end{aligned}
$$

Next taking max w.r.t. s, u of the above we get:

$$
\max_{s,u} |Q^*_{\mathcal{T}^x}(s,\mathbf{u}) - Q^*_{\mathcal{T}^y}(s,\mathbf{u})| \leq \frac{\epsilon_R + \gamma\epsilon_P V_{mid}}{1-\gamma}
$$

We now bound the deviation quantities appearing above:

$$
\begin{aligned}
\epsilon_R &= \max_s |r_{\mathcal{T}^x}(s) - r_{\mathcal{T}^y}(s)| \\
&= \max_s |\sum_{i=1}^n a_i^x \langle \mathcal{T}_i^x \cdot W_R s\rangle - \sum_{i=1}^n a_i^y \langle \mathcal{T}_i^y \cdot W_R s\rangle| \\
&\leq \max_s \Big[|\sum_{i=1}^n a_i^x \langle (\mathcal{T}_i^x - \mathcal{T}_i^y)\cdot W_R s\rangle| + |\sum_{i=1}^n (a_i^x - a_i^y)\langle \mathcal{T}_i^y \cdot W_R s\rangle|\Big] \\
&\leq \max_s \Big[|\sum_i a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty |W_R s|_1 + |\sum_i (a_i^x - a_i^y)\mathcal{T}_i^y|_\infty |W_R s|_1\Big] \\
&= s_{max}\Big[|\sum_i a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty + |\sum_i (a_i^x - a_i^y)\mathcal{T}_i^y|_\infty\Big]
\end{aligned}
$$

Similarly,

$$
\epsilon_P = \max_{s,\mathbf{u}} 2\cdot D_{TV}\Big(P_{\mathcal{T}^x}(\cdot|s,\mathbf{u}), P_{\mathcal{T}^y}(\cdot|s,\mathbf{u})\Big)
$$

$$
\begin{aligned}
&= \max_{s,\mathbf{u}} \sum_{s'} |P_{\mathcal{T}^x}(s'|s,\mathbf{u}) - P_{\mathcal{T}^y}(s'|s,\mathbf{u})| \\
&= \max_{s,\mathbf{u}} \sum_{s'} |\sum_{i=1}^{n} a_i^x \langle \mathcal{T}_i^x \cdot W_P(s',s,\mathbf{u})\rangle - \sum_{i=1}^{n} a_i^y \langle \mathcal{T}_i^y \cdot W_P(s',s,\mathbf{u})\rangle| \\
&\leq \max_{s,\mathbf{u}} \sum_{s'} \Big[|\sum_{i=1}^{n} a_i^x \langle (\mathcal{T}_i^x - \mathcal{T}_i^y) \cdot W_P(s',s,\mathbf{u})\rangle| + |\sum_{i=1}^{n} (a_i^x - a_i^y) \langle \mathcal{T}_i^y \cdot W_P(s',s,\mathbf{u})\rangle| \Big] \\
&\leq \max_{s,\mathbf{u}} \sum_{s'} \Big[|\sum_{i} a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty |W_P(s',s,\mathbf{u})|_1 + |\sum_{i} (a_i^x - a_i^y) \mathcal{T}_i^y|_\infty |W_P(s',s,\mathbf{u})|_1 \Big] \\
&= \Big[|\sum_{i} a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty + |\sum_{i} (a_i^x - a_i^y) \mathcal{T}_i^y|_\infty \Big] \max_{s,\mathbf{u}} \sum_{s'} |W_P(s',s,\mathbf{u})|_1 \\
&= d \Big[|\sum_{i} a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty + |\sum_{i} (a_i^x - a_i^y) \mathcal{T}_i^y|_\infty \Big]
\end{aligned}
$$

Thus, we get:

$$
|Q^*_{\mathcal{T}^x}(s,\mathbf{u}) - Q^*_{\mathcal{T}^y}(s,\mathbf{u})| \leq \frac{s_{max} + \gamma d V_{mid}}{1-\gamma} \Big[|\sum_{i} a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty + |\sum_{i} (a_i^x - a_i^y) \mathcal{T}_i^y|_\infty \Big]
$$

Finally we get the value difference bound by considering a dummy state $s^\#$ which always transitions according to ρ and then using the Bellman equation. (Note that for $\gamma \in (0, \frac{\sqrt{5}-1}{2})$ we can replace $\frac{1}{\gamma(1-\gamma)}$ by $\frac{1+\gamma}{1-\gamma}$ for a tighter bound without considering a dummy start state) □

Corollary C.1.1 (Change in optimal value as a result of agent substitution)**.** *Let $\mathcal{T} \in \mathcal{C}^n$ be a team composition with influence weights $a \in \Delta_{n-1}$. If agent i is substituted with i' keeping a_i unchanged such that $|\mathcal{T}_{i'} - \mathcal{T}_i|_\infty \leq \epsilon_C$ then the new team ($\mathcal{T}'$) optimal value follows:*

$$
|J^*_{\mathcal{T}'} - J^*_{\mathcal{T}}| \leq \frac{(s_{max} + \gamma d V_{mid}) a_i \epsilon_C}{\gamma(1-\gamma)}
$$

Proof. Applying Theorem 5.1 on original task and a new task with same influence weights and agent i capability replaced with $\mathcal{T}_{i'}$ immediately gives the result. □

C.1.2 Transfer of optimal policy

Theorem C.2 (Transfer of optimal policy)**.** *Let* $\mathcal{T}^x, \mathcal{T}^y \in \mathcal{C}^n$, $a^x, a^y \in \Delta_{n-1}$, $s_{max} = \max_s ||W_R s||_1$, $V_{mid} = \frac{1}{2}\max_s V^*_{\mathcal{T}^y}(s)$. *Let* π^*_y *be the optimal policy for the team composed of agents with capabilities* $\mathcal{T}^y$ *and influence weights* a^y*. Then:*

$$J^*_{\mathcal{T}^x} - J^{\pi^*_y}_{\mathcal{T}^x} \leq 2\frac{s_{max} + \gamma d V_{mid}}{\gamma(1-\gamma)}\Psi,$$

where Ψ *is defined as in Eq.* (5.4)*.*

Proof. We have that:

$$Q^*_{\mathcal{T}^x}(s,\mathbf{u}) - Q^{\pi^*_y}_{\mathcal{T}^x}(s,\mathbf{u}) \leq |Q^*_{\mathcal{T}^x}(s,\mathbf{u}) - Q^*_{\mathcal{T}^y}(s,\mathbf{u})| + |Q^*_{\mathcal{T}^y}(s,\mathbf{u}) - Q^{\pi^*_y}_{\mathcal{T}^x}(s,\mathbf{u})| \quad \text{(C.1)}$$

The first term on the RHS of Eq. (C.1) is taken care of by Theorem 5.1. We now focus on the second term:

$$\begin{aligned}
&|Q^*_{\mathcal{T}^y}(s,\mathbf{u}) - Q^{\pi^*_y}_{\mathcal{T}^x}(s,\mathbf{u})| \\
&= |r_{\mathcal{T}^y}(s) - r_{\mathcal{T}^x}(s) + \gamma\Big(\sum_{s'} P_{\mathcal{T}^y}(s'|s,\mathbf{u}) \max_{u'} Q^*_{\mathcal{T}^y}(s',\mathbf{u}') - \sum_{s'} P_{\mathcal{T}^x}(s'|s,\mathbf{u}) Q^{\pi^*_y}_{\mathcal{T}^x}(s',\pi^*_y(\mathbf{u}'))\Big)| \\
&\leq \epsilon_R + \gamma\Big\{|\sum_{s'} P_{\mathcal{T}^x}(s'|s,\mathbf{u})\Big[\max_{u'} Q^*_{\mathcal{T}^y}(s',\mathbf{u}') - Q^{\pi^*_y}_{\mathcal{T}^x}(s',\pi^*_y(\mathbf{u}')\Big]| \\
&\quad + |\sum_{s'}\Big[P_{\mathcal{T}^y}(s'|s,\mathbf{u}) - P_{\mathcal{T}^x}(s'|s,\mathbf{u})\Big](\max_{u'} Q^*_{\mathcal{T}^y}(s',\mathbf{u}') - V_{mid})|\Big\} \\
&\leq \epsilon_R + \gamma\Big\{\max_{s',u'}|Q^*_{\mathcal{T}^y}(s',\mathbf{u}') - Q^{\pi^*_y}_{\mathcal{T}^x}(s',\pi^*_y(\mathbf{u}')| + \epsilon_P V_{mid}\Big\}
\end{aligned}$$

Once again, taking max w.r.t. $s, \mathbf{u}$ of the above we get:

$$\max_{s,u}|Q^*_{\mathcal{T}^y}(s,\mathbf{u}) - Q^{\pi^*_y}_{\mathcal{T}^x}(s,\mathbf{u})| \leq \frac{\epsilon_R + \gamma\epsilon_P V_{mid}}{1-\gamma}$$

Substituting for deviation expressions and using Theorem 5.1 in Eq. (C.1) we get:

$$|Q^*_{\mathcal{T}^x}(s,\mathbf{u}) - Q^{\pi^*_y}_{\mathcal{T}^x}(s,\mathbf{u})| \leq 2\frac{s_{max}+\gamma dV_{mid}}{1-\gamma}\Big[|\sum_i a_i^x(\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty + |\sum_i (a_i^x - a_i^y)\mathcal{T}_i^y|_\infty\Big]$$

Note the absolute on LHS above can be dropped as $Q^*_{\mathcal{T}^x}$ is optimal. Finally using the same technique as above for Theorem 5.1 we get the statement of the theorem. □

Corollary C.2.1 (Out of distribution performance). *Let $\mathcal{T} \notin Sup(\mathcal{M})$ be an out of distribution task, we then have that the performance of the absolute oracle policy on $\mathcal{T}$ satisfies:*

$$J^*_{\mathcal{T}} - J^{\pi^*_{\mathcal{M}}}_{\mathcal{T}} \leq 2\frac{s_{max}+\gamma dV_{mid}}{\gamma(1-\gamma)} d_a(\mathcal{T}, Sup(\mathcal{M})),$$

Proof. For any task that belongs to $\arg\min_{\mathcal{T}^l \in Sup(\mathcal{M})} d_a(\mathcal{T}^l, \mathcal{T})$, we have by application of Theorem 5.2 that the result immediately holds given definition of $\pi^*_{\mathcal{M}}$. □

C.1.3 Population decrease

Theorem C.3 (Population decrease bound). *For the team composition $\mathcal{T} \in \mathcal{C}^n$ with influence weights $a \in \Delta_{n-1}$. If agent n is eliminated followed by a re-normalization of influence weights, we have that for the remaining team ($\mathcal{T}^- \triangleq (\mathcal{T})_{i=1}^{n-1}$):*

$$|J^*_{\mathcal{T}^-} - J^*_{\mathcal{T}}| \leq \frac{a_n(s_{max}+\gamma dV_{mid})}{\gamma(1-\gamma)}\Big|\sum_{i=1}^{n-1}\frac{a_i\mathcal{T}_i}{1-a_n} - \mathcal{T}_n\Big|_\infty$$

Proof. We use Theorem 5.1 with influence weights $(a_i)_1^n$ and $(\lambda \cdot a_i : i = 1..n-1, a_n = 0)$ where $\lambda = \frac{1}{1-a_n}$ □

Corollary C.3.1 (Population increase bound). *For the team composition $\mathcal{T} \in \mathcal{C}^n$ with influence weights $a \in \Delta_{n-1}$. If agent $n+1$ is added with capability $\mathcal{T}_{n+1}$ and weight a_{n+1} (other weights scaled down by $\lambda = 1 - a_{n+1}$) we have that for the new*

team ($\mathcal{T}^+ \triangleq (\mathcal{T}_1..\mathcal{T}_n, \mathcal{T}_{n+1})$)*:*

$$|J^*_{\mathcal{T}^+} - J^*_{\mathcal{T}}| \leq \frac{a_{n+1}(s_{max} + \gamma d V_{mid})}{\gamma(1-\gamma)} \Big| \sum_{i=1}^{n} a_i \mathcal{T}_i - \mathcal{T}_{n+1} \Big|_\infty$$

Proof. Consider the team compositions $\mathcal{T}^x = (\mathcal{T}_1..\mathcal{T}_n, 0)$ with influence weights $= (a_1..a_n, 0)$ and $\mathcal{T}^y = (\mathcal{T}_1..\mathcal{T}_n, \mathcal{T}_{n+1})$ with influence weights $= (\lambda a_1..\lambda a_n, a_{n+1})$ where $\lambda = 1 - a_{n+1}$, we have that:

$$\begin{aligned}
\Psi &= \Big[|\sum_i a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty + |\sum_i (a_i^x - a_i^y)\mathcal{T}_i^y|_\infty \Big] \\
&= |\sum_{i=1}^{n} (1-\lambda) a_i \mathcal{T}_i^y - a_{n+1}\mathcal{T}_{n+1}^y|_\infty \\
&= a_{n+1} |\sum_{i=1}^{n} a_i \mathcal{T}_i^y - \mathcal{T}_{n+1}^y|_\infty
\end{aligned}$$

which on applying Theorem 5.1 yields the result. □

C.1.4 Approximate $\hat{\epsilon}_R$,$\hat{\epsilon}_P$ dynamics

Theorem C.4 (Approximate $\hat{\epsilon}_R$,$\hat{\epsilon}_P$ dynamics)**.** *Let* $\mathcal{T}^x, \mathcal{T}^y \in \mathcal{C}^n$, $a^x, a^y \in \Delta_{n-1}$ *and the dynamics be only approximately linear so that* $|r_{\mathcal{T}}(s) - \sum_{i=1}^{n} a_i \langle c_i \cdot W_R s \rangle| \leq \hat{\epsilon}_R$ *and* $|P_{\mathcal{T}}(s'|s,\mathbf{u}) - \sum_{i=1}^{n} a_i \langle c_i \cdot W_P(s', s, \mathbf{u}) \rangle| \leq \hat{\epsilon}_P$*. Then:*

$$|J^*_{\mathcal{T}^x} - J^*_{\mathcal{T}^y}| \leq \frac{s_{max} + \gamma d V_{mid}}{\gamma(1-\gamma)} \Psi + \frac{2(\hat{\epsilon}_R + \gamma \hat{\epsilon}_P V_{mid})}{\gamma(1-\gamma)},$$

where Ψ *is defined as in Eq.* (5.4)*.*

Proof. We begin as in proof of Theorem 5.1 to get:

$$\max_{s,u} |Q^*_{\mathcal{T}^x}(s,\mathbf{u}) - Q^*_{\mathcal{T}^y}(s,\mathbf{u})| \leq \frac{\epsilon_R + \gamma \epsilon_P V_{mid}}{1-\gamma}$$

Next we apply the corrections to the relative differences:

$$
\begin{aligned}
\epsilon_R &= \max_s |r_{\mathcal{T}^x}(s) - r_{\mathcal{T}^y}(s)| \\
&\leq \max_s \Big[|r_{\mathcal{T}^x}(s) - \sum_{i=1}^{n} a_i^x \langle \mathcal{T}_i^x \cdot W_R s \rangle| + |\sum_{i=1}^{n} a_i^x \langle \mathcal{T}_i^x \cdot W_R s \rangle - \sum_{i=1}^{n} a_i^y \langle \mathcal{T}_i^y \cdot W_R s \rangle| \\
&\quad + |r_{\mathcal{T}^y}(s) - \sum_{i=1}^{n} a_i^y \langle \mathcal{T}_i^y \cdot W_R s \rangle| \Big] \\
&\leq 2\hat{\epsilon}_R + \max_s \Big[|\sum_{i=1}^{n} a_i^x \langle (\mathcal{T}_i^x - \mathcal{T}_i^y) \cdot W_R s \rangle| + |\sum_{i=1}^{n} (a_i^x - a_i^y) \langle \mathcal{T}_i^y \cdot W_R s \rangle| \Big] \\
&\leq 2\hat{\epsilon}_R + \max_s \Big[|\sum_i a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty |W_R s|_1 + |\sum_i (a_i^x - a_i^y) \mathcal{T}_i^y|_\infty |W_R s|_1 \Big] \\
&= 2\hat{\epsilon}_R + s_{max} \Big[|\sum_i a_i^x (\mathcal{T}_i^x - \mathcal{T}_i^y)|_\infty + |\sum_i (a_i^x - a_i^y) \mathcal{T}_i^y|_\infty \Big]
\end{aligned}
$$

Proceeding similarly with the transition probabilities we get the desired result. □

C.1.5 Error from estimation of capabilities

Theorem C.5 (Error from estimation of capabilities)**.** *For the team composition $\mathcal{T} \in \mathcal{C}^n$ with influence weights $a \in \Delta_{n-1}$. If the agent capabilities are inaccurately inferred as $\hat{\mathcal{T}}$ with $\max_i |\mathcal{T}_i - \hat{\mathcal{T}}_i|_\infty \leq \epsilon_{\mathcal{T}}$ and agents learn the inexact policy $\hat{\pi}^*$ then:*

$$
|J_{\mathcal{T}}^* - J_{\mathcal{T}}^{\hat{\pi}^*}| \leq \frac{2\epsilon_{\mathcal{T}}(s_{max} + \gamma d V_{mid})}{\gamma(1-\gamma)}
$$

where $V_{mid} = \frac{1}{2} \max_s V_{\hat{\mathcal{T}}}^*(s)$

Proof. We have that for the actual and inferred team compositions with same influence weights:

$$
\begin{aligned}
\Psi &= \Big[|\sum_i a_i (\mathcal{T}_i - \hat{\mathcal{T}}_i)|_\infty + |\sum_i (a_i - a_i) \hat{\mathcal{T}}_i|_\infty \Big] \\
&= |\sum_i a_i (\mathcal{T}_i - \hat{\mathcal{T}}_i)|_\infty
\end{aligned}
$$

$$
\begin{aligned}
&\leq \sum_i a_i |\mathcal{T}_i - \hat{\mathcal{T}}_i|_\infty \\
&\leq \sum_i a_i \epsilon_{\mathcal{T}} \\
&= \epsilon_{\mathcal{T}}
\end{aligned}
$$

Now applying Theorem 5.2 gives the result □

C.1.6 Extending to Lipschitz rewards

We demonstrate how to extend the results in Section 5.3 to Lipschitz function of capabilities. For brevity we consider only the setting where the rewards vary with capabilities. Thus, for the reward function form $r_{\mathcal{T}}(s) = \langle f(\mathcal{T}) \cdot W_R s \rangle$ where $f(\mathcal{T})$ is L_i Lipschitz with respect to the capability $\mathcal{T}_i$ for $i \in \mathcal{A}$ for the $|\cdot|_\infty$ norm. We get that for two different team compositions $\mathcal{T}^x, \mathcal{T}^y$

$$
\begin{aligned}
\epsilon_R &= \max_s |r_{\mathcal{T}^x}(s) - r_{\mathcal{T}^y}(s)| \\
&= \max_s |\langle f(\mathcal{T}^x) \cdot W_R s \rangle - \langle f(\mathcal{T}^y) \cdot W_R s \rangle| \\
&= \max_s |\sum_{i=1}^{n} \langle f(\mathcal{T}^i) \cdot W_R s \rangle - \langle f(\mathcal{T}^{i+1}) \cdot W_R s \rangle| \\
&\leq \max_s \sum_{i=1}^{n} |\langle f(\mathcal{T}^i) \cdot W_R s \rangle - \langle f(\mathcal{T}^{i+1}) \cdot W_R s \rangle| \\
&\leq \max_s \sum_{i=1}^{n} |\langle f(\mathcal{T}^i) \cdot W_R s \rangle - \langle f(\mathcal{T}^{i+1}) \cdot W_R s \rangle| \\
&\leq \max_s \sum_{i=1}^{n} |f(\mathcal{T}^i) - f(\mathcal{T}^{i+1})|_\infty |W_R s|_1 \\
&\leq s_{max} \sum_{i=1}^{n} L_i |\mathcal{T}_i^x - \mathcal{T}_i^y|_\infty
\end{aligned}
$$

Where $\mathcal{T}^i$ was the sequence satisfying $\mathcal{T}^1 = \mathcal{T}^x$ and $\mathcal{T}^{n+1} = \mathcal{T}^y$ and changing $\mathcal{T}^x$ one index at a time. We have thus proved that:

Theorem C.6. *For rewards L_i Lipschitz in the capabilities with respect to $|\cdot|_\infty$ norm, the difference in optimal values between team compositions $\mathcal{T}^x, \mathcal{T}^y$ satisfy:*

$$|J^*_{\mathcal{T}^x} - J^*_{\mathcal{T}^y}| \leq \frac{s_{max} \sum_{i=1}^n L_i |\mathcal{T}_i^x - \mathcal{T}_i^y|_\infty}{\gamma(1-\gamma)}$$

C.1.7 General dependence of rewards on capabilities:

Lemma C.1. *For substitution $\mathcal{T}_i$ to $\mathcal{T}_{i'}$ such that $|\mathcal{T}_i - \mathcal{T}_{i'}|_\infty \leq \delta$ under the (α, K)-rewards setting we have that*

$$\epsilon_R \in \mathcal{O}(\alpha \delta s_{max} K 2^K)$$

Proof.

$$\begin{aligned}
\epsilon_R &= \max_{s \in S} \left| \left\langle \sum_{K_i \in \mathbb{N}, \sum K_i \leq K} a_{K_1..K_n} \Pi_{j \neq i} \mathcal{T}_j^{K_j} (\mathcal{T}_i^{K_i} - \mathcal{T}_{i'}^{K_i}) \cdot W_R s \right\rangle \right| \\
&\leq \max_{s \in S} \left| \sum_{K_i \in \mathbb{N}, \sum K_i \leq K} a_{K_1..K_n} \Pi_{j \neq i} \mathcal{T}_j^{K_j} (\mathcal{T}_i^{K_i} - \mathcal{T}_{i'}^{K_i}) \right|_\infty \left| W_R s \right|_1 \\
&\leq \alpha s_{max} \sum_{j=0}^{K} \sum_{l=1}^{j} \binom{l}{j} l |\mathcal{T}_i^{K_i} - \mathcal{T}_{i'}^{K_i}|_\infty \\
&\leq \alpha \delta s_{max} \sum_{j=0}^{K} j 2^{j-1} = \mathcal{O}(\alpha \delta s_{max} K 2^K)
\end{aligned}$$

□

C.2 Experimental Setup

C.2.1 Environments

Fruit Forage

We use the fruit forage task on a grid world to empirically demonstrate the generalisation bounds in Section 5.3. On a $k \times k$ grid world we have n agents and d types of fruit trees. For each agent i, $\mathcal{T}_i(j), j \in \{1..d\}$ represents the utility of fruit j for agent i. The state vector is appended with the d dimensional binary vector representing whether each of the tree types was foraged at a given time step. The details for the team compositions can be found in Appendix C.2.1. We define three team compositions as follows:

- T_x: [[0.05, 0.1, 0.6, 2.8], [0.05, 0.1, 2.1, 0.8], [0.05, 0.1, 1.8, 1.2], [0.05, 0.1, 0.9, 2.4]]
- T_y: [[0.7, 0.4, 0.15, 0.2], [0.2, 1.4, 0.15, 0.2], [0.3, 1.2, 0.15, 0.2], [0.6, 0.6, 0.15, 0.2]]
- T_z: [[0.1, 0.3, 0.6, 0.0], [0.4, 0.1, 0.5, 0.0], [0.05, 0.06, 0.89, 0.0], [0.0, 0.0, 0.0, 1.0]]

For proving bounds on Theorem-1, we compare the mean test returns achieved on tasks T_x and T_y using $J^{\star}_{T_x} - J^{\star}_{T_y}$. For Theorem-2, we compare the mean test returns achieved on tasks T_x and optimal policies of task T_y evaluated on task T_x i.e. $J^{\star}_{T_x} - J^{\pi^{\star}_{T_y}}_{T_x}$. Finally, for Theorem-3, we compare the mean test returns achieved on tasks T_z and optimal policies of task T_z evaluated on task T_z but removing the last agent i.e. $J^{\star}_{T_{z-}} - J^{\star}_{T_z}$.

Predator Prey

We consider a complicated partially observable predator-prey (PP) task in an 8×8 grid involving four agents (predators) and four prey that is designed to test coordination between agents. Specifically, each predator has a parameter describing the hit point damage it can cause the prey. Similarly, the prey comes with variations in health. For example, a prey with a capability of 5 can only be caught if the total capability of agents taking the capture action simultaneously on it have capabilities ≥ 5 (such as [1,1,3]), otherwise, the whole team receives a penalty p. On successful capture, agents get a reward of $+1$. Once prey is captured, another prey is spawned at a random location. Therefore, agents have to collaborate and capture as many preys as possible within 100 time steps.

Each agent can take 6 actions i.e. move in one of the 4 directions (Up, Left, Down, Right), remain still (no-op), or try to catch (capture) any adjacent prey. The prey moves around in the grid with a probability of 0.7 and remains still at its position with the probability of 0.3. Impossible actions for both agents and prey are marked unavailable, for eg. moving into an occupied cell or trying to take a capture action with no adjacent prey.

In this domain, we test for two types of generalization: (1) novel team composition where test tasks contain a team composition which has not been encountered during training (PP Unseen Team in Figure 5.4), and second, (2) test tasks where novel team compositions can also have agent types with capabilities not encountered during training (PP Unseen Team, Agent in Figure 5.4).

For (PP Unseen Team), we train on preys with capabilities [2,2,2,3], and agents with capabilities [2,3,2,3],[1,2,1,2], thereby having agent teams with total hit points of 10 and 6 respectively. We also train on two separate penalties p for miscoordination i.e. $p \in \{0.0, -0.008\}$, this helps inject additional stochasticity in the environment as the agents don't know the penalty value. For test tasks, we create novel team compo-

sitions not encountered during training i.e. agents with capabilities [1,1,2,3],[1,1,1,3] having total hit points of 7 and 6 respectively.

For (PP Unseen Team, Agent) we train on preys with capabilities [1,2,3,4], and agents with capabilities [1, 2, 2, 3], [1, 1, 2, 2], [1, 3, 2, 1], thereby having agent teams with total hit points of 8, 6 and 7 respectively. We also train on two separate penalties p for miscoordination i.e. $p \in \{0.0, -0.008\}$. For test tasks, we create novel team compositions with an unseen agent of capability 4 not encountered during training i.e. agents with capabilities [1, 1, 1, 4], [1, 1, 3, 4], [1, 1, 2, 4] having total hit points of 7, 9, and 8 respectively.

Experimental Setup: For (PP Unseen Team, and PP Unseen Team, Agent) oracle baseline (leftmost), we show the average difference in performance across all test tasks when capability information is included ((c) for each method.

For testing the generalization gap in (PP Unseen Team), we show the difference in returns achieved by training task [1,2,1,2] (hit point 6) and test task [1,1,1,3] (hit point 6). For testing the generalization gap in (PP Unseen Team, Agent), we show the difference in returns achieved by training task [1,3,2,1] (hit point 7) and test task [1,1,1,4] (hit point 7) with a new agent of capability 4. All PP experiments are based on 8 seeds.

StarCraft II

We use the standard set of actions and global state information included as part of the SMAC benchmark [181]. The sight range of the agent units has been increased to the fully observable setting. In the oracle mode, agent capabilities are included as part of individual observations. Each agent always observes its own capabilities. Furthermore, capabilities are always included in the global state.

`10_Terran` and `10_Terran_Hard` environment includes Marine, Maradeur, and Medivac units. `10_Protoss` and `10_Protoss_Hard` environments feature Stalker, Zealot,

and Colossus units. `10_Zerg` and `10_Zerg_Hard` environments include Zergling, Hydralisk and Baneling units.

In `Accuracy` and `Health` tasks, specific values reduced from full unit capabilities are chosen to be equivalent to a loss of a single teammate. For example, if there three agents, their accuracy could be set to 0.75, 0.75 and 0.5 given that $(1 - 0.5) + (1 - 0.75) + (1 - 0.75) = 1$. Consequently, the overall reduction in accuracy would be roughly equivalent to losing one ally unit. This was chosen to ensure that the difficulty of the tasks was not too high.

All SMAC experiments are based on 5 seeds.

Table C.1 gives the training and evaluation distributions used in the terran unit type swapping tasks. For the other two unit classes we use similar distribution with the unit type substitution:

- Zerg: marine $\rightarrow$ zergling, marauder $\rightarrow$ hydralisk, medivac $\rightarrow$ baneling
- Protoss: marine $\rightarrow$ stalker, marauder $\rightarrow$ zealot, medivac $\rightarrow$ collosus

Table C.1: Team formations in `Terran` tasks

`10_Terran`	`10_Terran_Hard`
Training	**Training**
1 marine & 9 marauders	1 marine & 9 marauders
3 marines & 7 marauders	2 marines & 8 marauders
4 marines & 6 marauders	3 marines & 7 marauders
5 marines & 5 marauders	4 marines & 6 marauders
6 marines & 4 marauders	5 marines & 5 marauders
8 marines & 2 marauders	6 marines & 4 marauders
9 marines & 1 marauder	7 marines & 3 marauders
5 marauders & 5 medivacs	8 marines & 2 marauders
7 marauders & 3 medivacs	9 marines & 1 marauder
9 marauders & 1 medivac	5 marauders & 5 medivacs
7 marines & 3 medivacs	6 marauders & 4 medivacs
8 marines & 2 medivacs	7 marauders & 3 medivacs
9 marines & 1 medivac	8 marauders & 2 medivacs
10 marines	9 marauders & 1 medivac
10 marauders	7 marines & 3 medivacs
8 marines & 1 marauder & 1 medivac	8 marines & 2 medivacs
1 marine & 8 marauders & 1 medivac	9 marines & 1 medivac
5 marines & 3 marauders & 2 medivacs	**Testing**
2 marines & 7 marauders & 1 medivac	10 marines
6 marines & 2 marauders & 2 medivacs	10 marauders
2 marines & 6 marauders & 2 medivacs	8 marines & 1 marauder & 1 medivac
4 marines & 4 marauders & 2 medivacs	1 marine & 8 marauders & 1 medivac
Testing	5 marines & 3 marauders & 2 medivacs
2 marines & 8 marauders	3 marines & 5 marauders & 2 medivacs
7 marines & 3 marauders	4 marines & 3 marauders & 3 medivacs
6 marauders & 4 medivacs	3 marines & 4 marauders & 3 medivacs
8 marauders & 2 medivacs	7 marines & 2 marauders & 1 medivac
3 marines & 5 marauders & 2 medivacs	2 marines & 7 marauders & 1 medivac
4 marines & 3 marauders & 3 medivacs	6 marines & 2 marauders & 2 medivacs
3 marines & 4 marauders & 3 medivacs	2 marines & 6 marauders & 2 medivacs
7 marines & 2 marauders & 1 medivac	4 marines & 4 marauders & 2 medivacs

C.2.2 Architecture, Training and Evaluation

The evaluation procedure is similar to the one in [172]. The training is paused after every 30k timesteps during which 16 test episodes are run with agents performing action selection greedily in a decentralised fashion. The percentage of episodes where the agents defeat all enemy units within the permitted time limit is referred to as the test win rate.

To speed up the learning, the agent networks are parameters are shared across all agents. A one-hot encoding of the `agent_id` is concatenated onto each agent's observations. All neural networks are trained using RMSprop without weight decay or momentum.

Value-based baselines

The architecture of all agent networks is a DRQN [89] with a recurrent layer comprised of a GRU with a 64-dimensional hidden state, with a fully-connected layer before and after. We sample batches of 32 episodes uniformly from the replay buffer, and train on fully unrolled episodes, performing a single gradient descent step after 8 episodes.

Table C.2: Hyperparameters of QMIX and VDN

Method	Name	Value
QMIX & VDN	learning rate	5×10^{-4}
	RMSprop α	0.99
	replay buffer size	5000 episodes
	target network update interval	200 episodes
	γ	0.99
	double DQN target	True
	initial ϵ	1
	final ϵ	0.05
	ϵ anneal period	50000 steps
	ϵ anneal rule	linear
QMIX	mixing network hidden layers	1
	mixing network hidden layer units	32
	mixing network non-linearity	ELU
	hypernetwork hidden layers	2
	hypernetwork hidden layer units	64
	hypernetwork non-linearity	ReLU

PPO baselines

We parameterize the actor and critic with two independent recurrent neural networks, each of which is comprised of a GRU with a 64-dimensional hidden state, with a fully-connected layer as the input and output.

Table C.3: Hyperparameters of IPPO and MAPPO

Method	Name	Value
IPPO & MAPPO	critic learning rate	0.001
	actor learning rate	0.99
	γ	0.99
	λ	0.95
	ϵ	0.2
	clip range	0.1
	normalize advantage	True
	normalize inputs	True
	grad norm	0.5
	number of actors	8
	critic coefficient	2
	entropy coefficient	0
	mini epochs for actor update	10
	mini epochs for critic update	10
	mini batch size	64

C.3 Full StarCraft II Results

Complete results for StarCraft II are as shown in Fig. C.1, Fig. C.2, Fig. C.3.

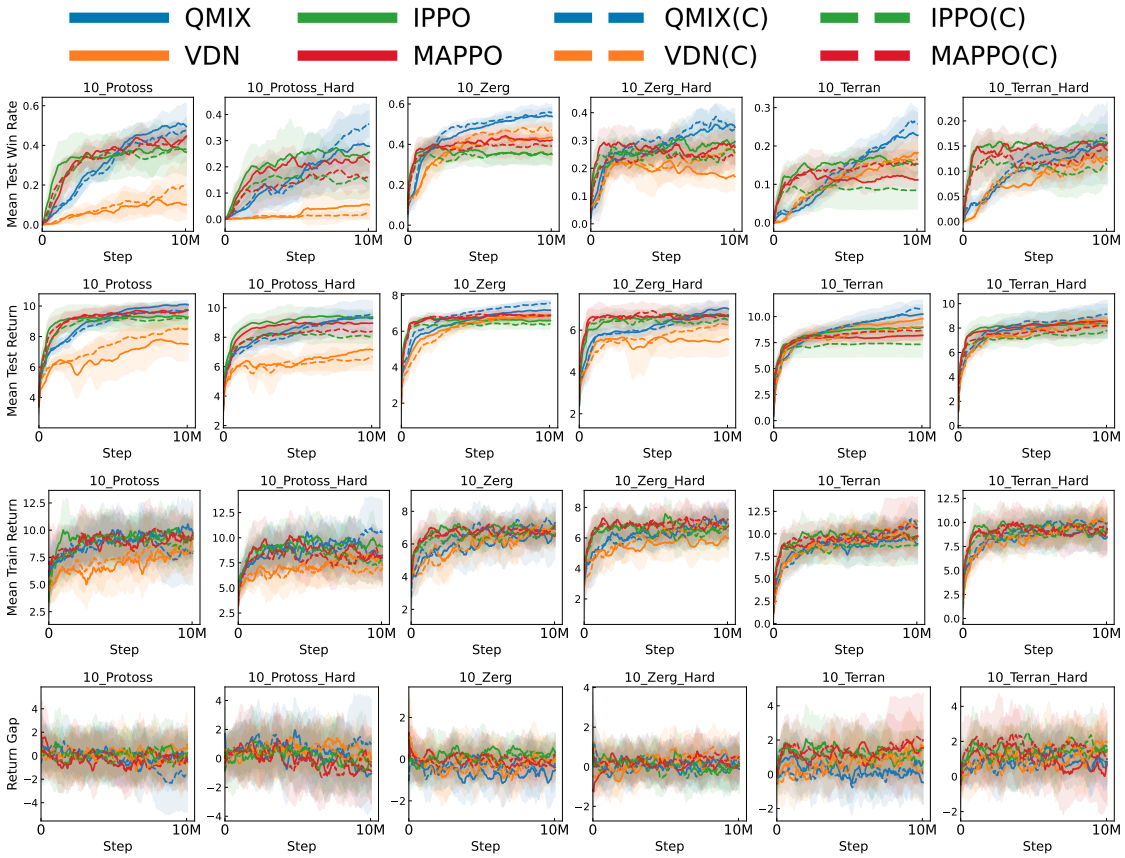


Figure C.1: Experimental results on SMAC unit swapping tasks. Dashed lines indicate the inclusion of information on capabilities as part of the agent observations. Standard deviation is shaded.

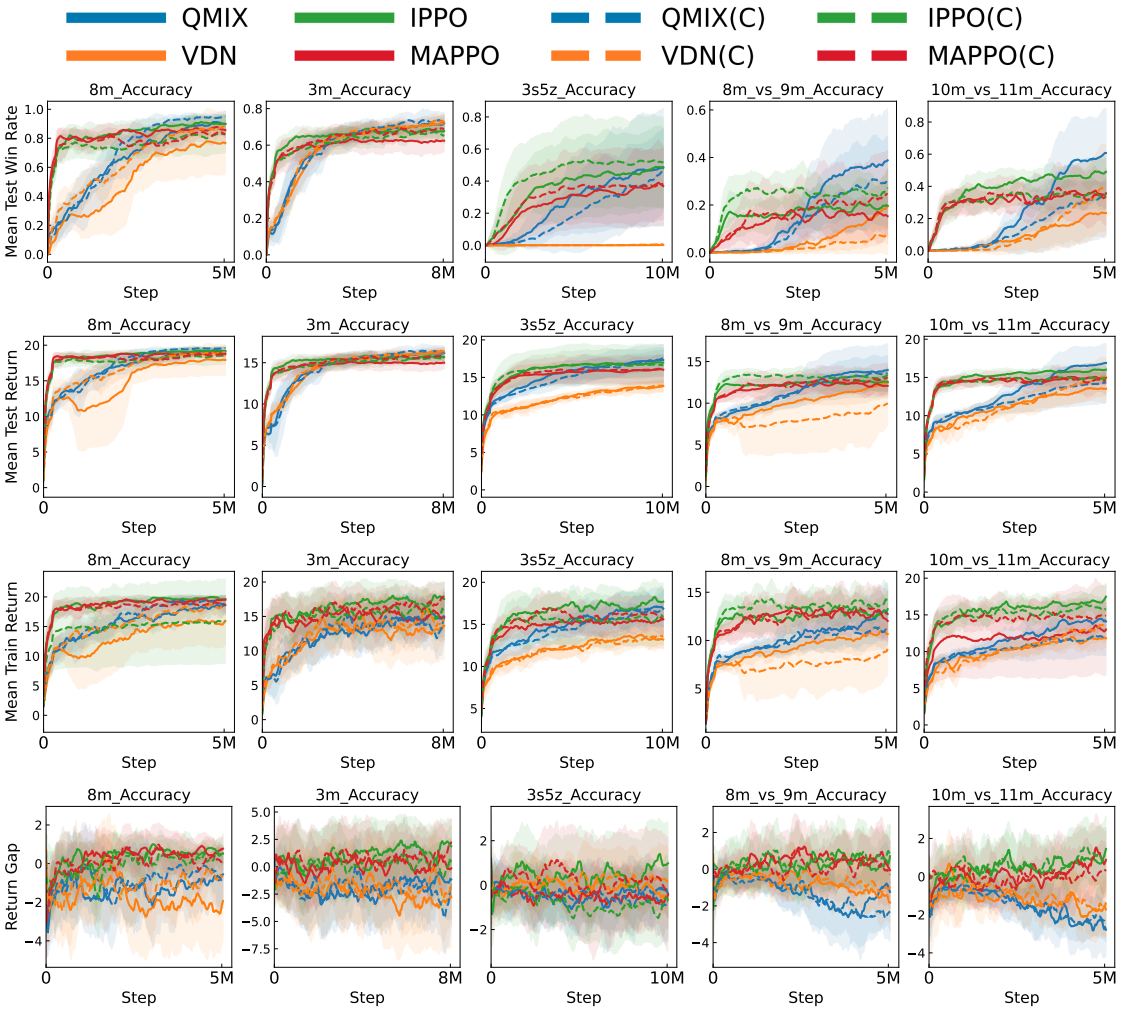


Figure C.2: Experimental results on SMAC unit accuracy tasks. Dashed lines indicate the inclusion of information on capabilities as part of the agent observations. Standard deviation is shaded.

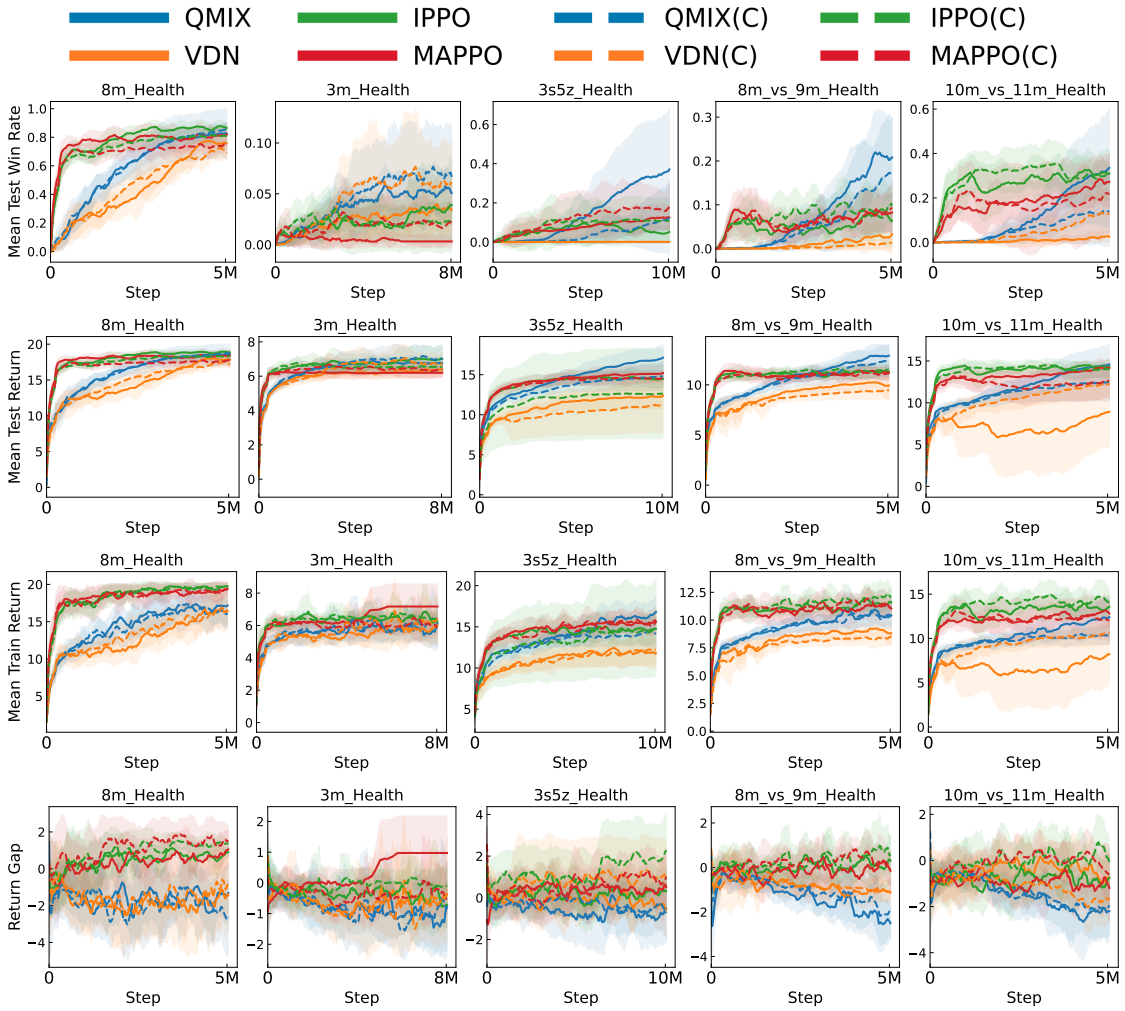


Figure C.3: Experimental results on SMAC unit health tasks. Dashed lines indicate the inclusion of information on capabilities as part of the agent observations. Standard deviation is shaded.

Appendix D

Appendix for Chapter 6

D.1 A Probabilistic Interpretation of VIREL

We now motivate our inference procedure and Boltzmann distribution $\pi_\omega(a|s)$ from a probabilistic perspective, demonstrating that $\pi_\omega(a|s)$ can be interpreted as an action-posterior that characterises the uncertainty our model has in the optimality of $\hat{Q}_\omega(h)$. Moreover, maximising $\mathcal{L}(\omega, \theta)$ for θ is equivalent to carrying our variational inference on the graphical model in Fig. D.1 for any $\varepsilon_\omega > 0$.

D.1.1 Model Specification

Like previous work, we introduce a binary variable $\mathcal{O} \in \{0, 1\}$ in order to define a formal graphical model for our inference problem when $\varepsilon_\omega > 0$. The likelihood of $\mathcal{O}$ therefore takes the form of a Bernoulli distribution:

$$p_\omega(\mathcal{O}|h) = y_\omega(h)^{\mathcal{O}}(1 - y_\omega(h))^{(1-\mathcal{O})},$$

where

$$y_\omega(h) \coloneqq \exp\left(\frac{\hat{Q}_\omega(h) - \max_{a'} Q^*_\omega(a', s)}{\varepsilon_\omega}\right).$$

In most existing frameworks, $\mathcal{O} = 1$ is understood to be the event that the agent is acting optimally [131, 220]. As we are using function approximators in VIREL, $\mathcal{O} = 1$ can be interpreted as the event that the agent is behaving optimally under $\hat{Q}_\omega(h)$. Exploring the semantics of $\mathcal{O}$ further, consider the likelihood when $\mathcal{O} = 1$:

$$p_\omega(\mathcal{O} = 1|h) = \exp\left(\frac{\hat{Q}_\omega(h) - \max_{a'} \hat{Q}_\omega(a', s)}{\varepsilon_\omega}\right),$$

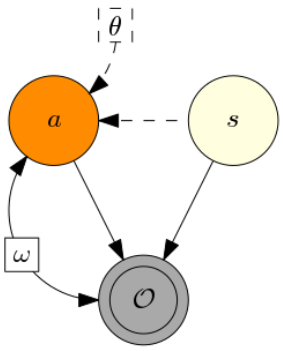


Figure D.1: Graphical model for VIREL (variational approximation dashed)

Observe that $0 \leq p_\omega(\mathcal{O} = 1|\cdot) \leq 1 \;\forall\; \omega \in \Omega \; s.t. \; \varepsilon_\omega > 0$. For any state s, we have $p_\omega(\mathcal{O} = 1|s, a^*) = 1$ for any action a^* that is optimal under $\hat{Q}_\omega(h)$ in the sense that it is the greedy action $a^* \in \arg\max_a \hat{Q}_\omega(h)$. If we find $p_\omega(\mathcal{O} = 1|h) = 1 \;\forall\; h \in \mathcal{H}$, then all observed state-action pairs have been generated from a greedy policy $\pi(a|s) = \delta(a \in \arg\max_{a'} \hat{Q}_\omega(a'|s))$. From Theorem 6.2, the closer the residual error ε_ω is to zero, the closer $\hat{Q}_\omega(h)$ becomes to representing an optimal action-value function. When $\varepsilon_\omega \approx 0$, any a observed such that $p_\omega(\mathcal{O} = 1|a, \cdot) = 1$ will be very nearly an action sampled from an optimal policy, that is $a \sim \pi(a|\cdot) \approx \delta(a \in \arg\max_{a'} Q^*(a'|\cdot))$. We caution readers that in the limit $\varepsilon_\omega \to 0$, our likelihood is not well-defined for

any $a \in \arg\max_{a'} \hat{Q}_\omega(a', s)$. Without loss of generality, we condition on optimality for the rest of this section, writing $\mathcal{O}$ in place of $\mathcal{O} = 1$. Defining the function $y_\omega(s) \coloneqq \exp\left(-\frac{\max_{a'} \hat{Q}_\omega(a', s)}{\varepsilon_\omega}\right)$, our likelihood takes the convenient form:

$$p_\omega(\mathcal{O}|h) = \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s),$$

Defining the prior distribution as the uniform distribution $p(h) = \mathcal{U}(h)$ completes our model, the graph for which is shown in Fig. D.1. Using Bayes' rule, we find our posterior distribution is:

$$\begin{aligned} p_\omega(h|\mathcal{O}) &= \frac{p_\omega(\mathcal{O}|h)p(h)}{p_\omega(\mathcal{O})}, \\ &= \frac{p_\omega(\mathcal{O}|h)p(h)}{\int p_\omega(\mathcal{O}|h)p(h)dh}, \\ &= \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)dh}. \end{aligned} \tag{D.1}$$

We can also derive our action-posterior, $p_\omega(a|s, \mathcal{O})$, which we will find to be equivalent to the Boltzmann policy from Eq. (6.3). Using Bayes' rule, it follows:

$$p_\omega(a|s, \mathcal{O}) = \frac{p_\omega(h|\mathcal{O})}{p_\omega(s|\mathcal{O})}.$$

Now, we find $p_\omega(s|\mathcal{O})$ by marginalising our posterior over actions. Substituting $p_\omega(s|\mathcal{O}) = \int p_\omega(h|\mathcal{O})da$ yields :

$$p_\omega(a|s, \mathcal{O}) = \frac{p_\omega(h|\mathcal{O})}{\int p_\omega(h|\mathcal{O})da}.$$

Substituting for our posterior from Eq. (D.1), we obtain:

$$p_\omega(a|s, \mathcal{O}) = \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)da} \cdot \frac{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)dh}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)dh},$$

$$
\begin{aligned}
&= \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)}{\left(\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da\right) y_\omega(s)}, \\
&= \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}, \\
&= \pi_\omega(a|s),
\end{aligned}
$$

proving that our action-posterior is exactly the Boltzmann policy introduced in Section 6.3.1. From a Bayesian perspective, the action-posterior $p_\omega(a|s, \mathcal{O})$ characterises the uncertainty we have in deducing the optimal action for a given state s under $\hat{Q}_\omega(h)$; whenever $\varepsilon_\omega \approx 0$ and hence $\hat{Q}_\omega(h) \approx Q^*(h)$, the uncertainty will be very small as $p_\omega(a|s, \mathcal{O})$ will have near-zero variance, approximating a Dirac-delta distribution. Our model is therefore highly confident that the maximum-a-posteriori (MAP) action $a \in \arg\max_{a'} \hat{Q}_\omega(a', s)$ is an optimal action, with all of the probability mass being close to this point. In light of this, we can interpret the greedy policy $\pi_\omega(a|s) = \delta(a \in \arg\max_{a'} \hat{Q}_\omega(a', s))$ as one that always selecting the MAP action across all states.

As our model incorporates the uncertainty in the optimality of $\hat{Q}_\omega(h)$ into the variance of $\pi_\omega(a|s)$, we can benefit directly by sampling trajectories from $\pi_\omega(a|s)$ which drives exploration to gather data that is beneficial to reducing the residual error ε_ω. Unfortunately, calculating the normalisation constant $\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da$ is intractable for most function approximators and MDPs of interest. As such, we resort to variational inference, a powerful technique to infer an approximation to a posterior distribution from a tractable family of variational distributions [105, 20, 28]. As before $\pi_\theta(a|s)$ is known as the variational policy, is parametrised by $\theta \in \Theta$ and with the same support as $\pi_\omega(a|s)$. Like in Section 6.3.1, we define a variational distribution as $q_\theta(h) \coloneqq d(s)\pi_\theta(a|s)$, where $d(s)$ is an arbitrary sampling distribution with support over $\mathcal{S}$. We fix $d(s)$, as in our model-free paradigm we do not learn

the state transition dynamics and only seek to infer the action-posterior.

The goal of variational inference is to find $q_\theta(h)$ closest in KL-divergence to $p_\omega(h|\mathcal{O})$, giving an objective:

$$\theta^* \in \arg\min_\theta \mathrm{KL}(q_\theta(h) \parallel p_\omega(h|\mathcal{O})).$$

This objective still requires the intractable computation of $\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)dh$. Mirroring the analysis in Appendix D.3.1, we can overcome this by writing the KL divergence in terms of the ELBO:

$$\mathrm{KL}(q_\theta(h) \parallel p_\omega(h|\mathcal{O})) = \ell_\omega - \mathcal{L}_\omega(\theta),$$

$$\text{where} \quad \ell(\omega) \coloneqq \log \int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)dh, \quad \mathcal{L}_\omega(\theta) \coloneqq \mathbb{E}_{h\sim q_\theta(h)}\left[\log\left(\frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)}{q_\theta(h)}\right)\right].$$

We see that minimising the KL-divergence for θ is equivalent to maximising the ELBO for θ, which is tractable. This affords a new objective:

$$\theta^* \in \arg\max_\theta \mathcal{L}_\omega(\theta).$$

Expanding the ELBO yields:

$$\begin{aligned}
\mathcal{L}_\omega(\theta) &= \mathbb{E}_{h\sim q_\theta(h)}\left[\log\left(\frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)}{q_\theta(h)}\right)\right], \\
&= \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log\left(\frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) y_\omega(s)}{q_\theta(h)}\right)\right]\right], \\
&= \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] + \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log y_\omega(s)\right] - \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log(\pi_\theta(a|s)d(s))\right]\right], \\
&= \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] + \log y_\omega(s) - \log d(s) - \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log \pi_\theta(a|s)\right]\right],
\end{aligned}$$

$$= \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] - \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log\pi_\theta(a|s)\right]\right] + \mathbb{E}_{s\sim d(s)}\left[\log\left(\frac{y_\omega(s)}{d(s)}\right)\right],$$

$$= \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] + \mathcal{H}(\pi_\theta(a|s))\right] + \mathbb{E}_{s\sim d(s)}\left[\log\left(\frac{y_\omega(s)}{d(s)}\right)\right].$$

As the final term $\mathbb{E}_{s\sim d(s)}\left[\log\left(\frac{y_\omega(s)}{d(s)}\right)\right]$ has no dependency on θ, we can neglect it from our objective, recovering the VIREL objective from Eq. (6.4):

$$\mathcal{L}_\omega(\theta) = \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] + \mathcal{H}(\pi_\theta(a|s))\right].$$

Finally, Theorem 6.3 guarantees that minimising $\mathcal{L}_\omega(\theta)$ always minimises the expected KL divergence between $\pi_\omega(a|s)$ and $\pi_\theta(a|s)$, allowing us to learn a variational approximation for the action-posterior.

D.1.2 Comparison to MERLIN

We now elucidate a key difference in the inference procedure between VIREL and MERLIN that can effect performance. Unlike in previous frameworks including MERLIN, VIREL's hidden variables are state-action pairs h, rather than entire trajectories τ. As is clear from the graphical model in Fig. D.1, the variational distribution $q_\theta(h)$ only needs to represent a single interaction between a and s and the function approximator $\hat{Q}_\omega(h)$ represents all future interactions. Compare this to the graphical model for MERLIN, shown in Fig. D.2 (see [131] for a full exposition of MERLIN's graphical model). The variational distribution for MERLIN $q_\theta(\tau)$ must represent every interaction between s and a for the entire trajectory. As $q_\theta(\tau)$ is restricted to classes of tractable distributions, $q_\theta(\tau)$ may not be expressive enough to capture the underlying dynamics for the MDP in complex

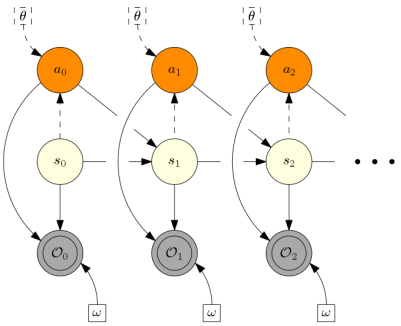


Figure D.2: Graphical model for MERLIN. The variational approximation is shown dashed.

domains where state-action spaces are large and there exist multiple modes. In contrast, VIREL's function approximator $\hat{Q}_\omega(h)$ is typically a neural network, which is much less restricted in representability and can better represent the dynamics of an MDP.

D.2 A Discussion of the Target Set $\mathbb{T}$

We now prove that the Bellman operator for the Boltzmann policy, $\mathcal{T}^{\pi_\omega}\bullet := r(h) + \gamma\mathbb{E}_{h'\sim p(s'|h)\pi_\omega(a'|s')}[\bullet]$, is a member of $\mathbb{T}$. Taking the limit $\varepsilon_\omega \to 0$ of $\mathcal{T}^{\pi_\omega}\hat{Q}_\omega(h)$, we find:

$$\lim_{\varepsilon_\omega\to 0}\mathcal{T}^{\pi_\omega}\hat{Q}_\omega(h) = r(h) + \lim_{\varepsilon_\omega\to 0}\gamma\mathbb{E}_{h'\sim p(s'|h)\pi_\omega(a'|s')}\left[\hat{Q}_\omega(h')\right].$$

From Theorem 6.2, evaluating $\lim_{\varepsilon_\omega\to 0}\gamma\mathbb{E}_{h'\sim p(s'|h)\pi_\omega(a'|s')}[\bullet]$ recovers a Dirac-delta distribution:

$$\begin{aligned}\lim_{\varepsilon_\omega\to 0}\mathcal{T}^{\pi_\omega}\hat{Q}_\omega(h) &= r(h) + \gamma\mathbb{E}_{h'\sim p(s'|h)\delta(a'=\arg\max_a \hat{Q}_\omega(a,s))}\left[\hat{Q}_\omega(h')\right], \\ &= r(h) + \gamma\mathbb{E}_{h'\sim p(s'|h)}\left[\max_{a'}(\hat{Q}_\omega(h'))\right], \\ &= \mathcal{T}^*\hat{Q}_\omega(h).\end{aligned}$$

which is sufficient to demonstrate membership of $\mathbb{T}$.

Observe that using $\mathcal{T}^{\pi_\omega}\bullet$ implies $\hat{Q}_\omega(h)$ cannot represent the true Q-function of any $\pi_\omega(a|s)$ except for the optimal Q-function. To see this, imagine there exists some $\varepsilon_\omega > 0$ such that $Q^{\pi_\omega}(\cdot) = \hat{Q}(\cdot)$. Under these conditions, it holds that $\mathcal{T}^{\pi_\omega}\hat{Q}(\cdot) = \hat{Q}(\cdot) \implies \varepsilon_\omega = 0$, which is a contradiction. More generally, as $\pi_\omega(a|s)$ is defined in terms of ε_ω, which itself depends on $\pi_\omega(a|s)$ from the definition of $\mathcal{T}^{\pi_\omega}\bullet$, any ω satisfying this recursive definition forms a constrained set $\Omega^c \subseteq \Omega$. Crucially, we show in Theorem 6.2 that there always exists some $\omega^* \in \Omega^c$ such that $\hat{Q}_{\omega^*}$ can

represent the action-value function for an optimal policy. Note that there may exist other policies that are not Boltzmann distributions such that $\hat{Q}_\omega(h) = Q^\pi(h)$ for some $\omega \in \Omega^c$. We discuss operators that don't constrain Ω in Appendix D.5.2.

Finally, we can approximate $\mathcal{T}^{\pi_\omega}$ using any TD target sampled from $\pi_\omega(a|s)$ (see [202] for an overview of TD methods). Likewise, the optimum Bellman operator $\mathcal{T}^* \cdot = r(h) + \gamma \mathbb{E}_{h' \sim p(s'|h)} [\max_{a'}(\cdot)]$ is by definition a member of $\mathbb{T}$ and can be approximated using the Q-learning target [238].

D.3 Proofs for Section 6.3

D.3.1 Derivation of Lower Bound in terms of KL Divergence

Recall the definition of $\mathcal{L}(\omega, \theta)$ from Eq. (6.4):

$$\mathcal{L}(\omega, \theta) = \mathbb{E}_{s \sim d(s)} \left[\mathbb{E}_{a \sim \pi_\theta(a|s)} \left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right] + \mathcal{H}(\pi_\theta(a|s)) \right].$$

Expanding the definition of differential entropy:

$$\begin{aligned}
\mathcal{L}(\omega, \theta) &= \mathbb{E}_{s \sim d(s)} \left[\mathbb{E}_{a \sim \pi_\theta(a|s)} \left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right] - \mathbb{E}_{a \sim \pi_\theta(a|s)} [\log \pi_\theta(a|s)] \right]. \\
&= \mathbb{E}_{s \sim d(s)} \left[\mathbb{E}_{a \sim \pi_\theta(a|s)} \left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right] - \mathbb{E}_{a \sim \pi_\theta(a|s)} \left[\log \left(\frac{\pi_\theta(a|s)}{\pi_\omega(a|s)} \cdot \pi_\omega(a|s) \right) \right] \right], \\
&= \mathbb{E}_{s \sim d(s)} \Bigg[\mathbb{E}_{a \sim \pi_\theta(a|s)} \left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right] - \mathbb{E}_{a \sim \pi_\theta(a|s)} \left[\log \left(\frac{\pi_\theta(a|s)}{\pi_\omega(a|s)} \right) \right] \\
&\quad - \mathbb{E}_{a \sim \pi_\theta(a|s)} [\log \pi_\omega(a|s)] \Bigg], \\
&= \mathbb{E}_{s \sim d(s)} \left[\mathbb{E}_{a \sim \pi_\theta(a|s)} \left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right] - \mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s)) - \mathbb{E}_{a \sim \pi_\theta(a|s)} [\log \pi_\omega(a|s)] \right].
\end{aligned}$$

Substituting for the definition of $\pi_\omega(a|s)$ in the final term yields our desired result:

$$\begin{aligned}
\mathcal{L}(\omega, \theta) &= \mathbb{E}_{s\sim d(s)}\Bigg[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] - \mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s)) \\
&\quad - \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\log\left(\frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}\right)\right]\Bigg], \\
&= \mathbb{E}_{s\sim d(s)}\Bigg[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] - \mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s)) \\
&\quad - \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right]\Bigg] + \log\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da, \\
&= \log\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da - \mathbb{E}_{s\sim d(s)}\left[\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s))\right].
\end{aligned}$$

D.3.2 Convergence of Boltzmann Distribution to Dirac-Delta

Theorem D.1 (Convergence of Boltzmann Distribution to Dirac Delta)**.** *Let* $p_\varepsilon : \mathcal{X} \to [0, 1]$ *be a Boltzmann distribution with temperature* $\varepsilon \in \mathbb{R}_{\geq 0}$

$$p_\varepsilon(x) = \frac{\exp\left(\frac{f(x)}{\varepsilon}\right)}{\int_{\mathcal{X}} \exp\left(\frac{f(x)}{\varepsilon}\right) dx},$$

where $f : \mathcal{X} \to \mathcal{Y}$ *is a function with a unique maximum* $f(x^*) = \sup_x f$ *and a bounded domain* $\mathcal{X}$ *and range* $\mathcal{Y}$*. Let* f *be locally smooth about* x^**, that is* $\exists\ \Delta > 0\ s.t. f(x) \in \mathbb{C}^2\ \forall\ x \in \{x | \|x - x^*\| < \Delta\ \}$*. In the limit* $\varepsilon \to 0$*,* $p_\varepsilon(x) \to \delta(x^*)$*, that is:*

$$\lim_{\varepsilon\to 0}\int_{\mathcal{X}} \varphi(x) p_\varepsilon(x) dx = \varphi(x^*), \tag{D.2}$$

for any smooth test function $\varphi \in \mathbb{C}_0^\infty(\mathcal{X})$*.*

Proof. Firstly, we define the auxiliary function to be

$$g(x) \coloneqq f(x) - f(x^*).$$

Note, $g(x) \leq 0$ with equality at $g(x^*) = 0$. Substituting $f(x) = g(x) + f(x^*)$ into $p_\varepsilon(x)$:

$$\begin{aligned} p_\varepsilon(x) =& \frac{\exp\left(\frac{g(x)+f(x^*)}{\varepsilon}\right)}{\int_{\mathcal{X}} \exp\left(\frac{g(x)+f(x^*)}{\varepsilon}\right) dx}, \\ =& \frac{\exp\left(\frac{g(x)}{\varepsilon}\right)\exp\left(\frac{f(x^*)}{\varepsilon}\right)}{\int_{\mathcal{X}} \exp\left(\frac{g(x)}{\varepsilon}\right)\exp\left(\frac{f(x^*)}{\varepsilon}\right) dx}, \\ =& \frac{\exp\left(\frac{g(x)}{\varepsilon}\right)}{\int_{\mathcal{X}} \exp\left(\frac{g(x)}{\varepsilon}\right) dx}. \end{aligned} \tag{D.3}$$

Now, substituting Eq. (D.3) into the limit in Eq. (D.2) yields:

$$\lim_{\varepsilon\to 0} \int_{\mathcal{X}} \varphi(x) p_\varepsilon(x) dx = \lim_{\varepsilon\to 0} \left(\int_{\mathcal{X}} \varphi(x) \frac{\exp\left(\frac{g(x)}{\varepsilon}\right)}{\int_{\mathcal{X}} \exp\left(\frac{g(x)}{\varepsilon}\right) dx} dx \right). \tag{D.4}$$

Using the substitution $u \coloneqq \frac{(x^*-x)}{\sqrt{\varepsilon}}$ to transform the integrals in Eq. (D.4), we obtain

$$\begin{aligned} \lim_{\varepsilon\to 0} \int_{\mathcal{X}} \varphi(x) p_\varepsilon(x) dx &= \lim_{\varepsilon\to 0} \left(\int_{\mathcal{U}} \varphi(x^* - \sqrt{\varepsilon}u) \frac{\exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)}{\int_{\mathcal{U}} \exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right) \sqrt{\varepsilon} du} \sqrt{\varepsilon} du \right), \\ &= \lim_{\varepsilon\to 0} \left(\frac{\int_{\mathcal{U}} \varphi(x^* - \sqrt{\varepsilon}u) \exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right) du}{\int_{\mathcal{U}} \exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right) du} \right). \end{aligned} \tag{D.5}$$

We now find $\lim_{\varepsilon\to 0}\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)$. Denoting the partial derivative $\partial_{\sqrt{\varepsilon}} \coloneqq \frac{\partial}{\partial\sqrt{\varepsilon}}$ and using L'Hôpital's rule to the second derivative with respect to $\sqrt{\epsilon}$, we find the limit

as:

$$\begin{aligned}
\lim_{\varepsilon\to 0}\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right) &= \lim_{\varepsilon\to 0}\left(\frac{\partial_{\sqrt{\varepsilon}}g(x^*-\sqrt{\varepsilon}u)}{\partial_{\sqrt{\varepsilon}}\varepsilon}\right),\\
&= \lim_{\varepsilon\to 0}\left(\frac{\partial_{\sqrt{\varepsilon}}f(x^*-\sqrt{\varepsilon}u)}{\partial_{\sqrt{\varepsilon}}\varepsilon}\right),\\
&= \lim_{\varepsilon\to 0}\left(\frac{-u^\top\nabla f(x^*-\sqrt{\varepsilon}u)}{2\sqrt{\varepsilon}}\right),\\
&= \lim_{\varepsilon\to 0}\left(\frac{-\partial_{\sqrt{\varepsilon}}\left(u^\top\nabla f(x^*-\sqrt{\varepsilon}u)\right)}{\partial_{\sqrt{\varepsilon}}(2\sqrt{\varepsilon})}\right),\\
&= \lim_{\varepsilon\to 0}\left(\frac{u^\top\nabla^2 f(x^*-\sqrt{\varepsilon}u)u}{2}\right),\\
&= \frac{u^\top\nabla^2 f(x^*)u}{2}.
\end{aligned}$$

The integrand in the numerator Eq. (D.5) therefore converges pointwise to $\varphi(x^*)\exp\left(\frac{u^\top\nabla^2 f(x^*)u}{2}\right)$, that is

$$\lim_{\varepsilon\to 0}\left(\varphi(x^*-\sqrt{\varepsilon}u)\exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)\right) = \varphi(x^*)\exp\left(\frac{u^\top\nabla^2 f(x^*)u}{2}\right), \qquad \text{(D.6)}$$

and the integrand in the denominator converges pointwise to $\exp\left(\frac{u^\top\nabla^2 f(x^*)u}{2}\right)$, that is

$$\lim_{\varepsilon\to 0}\left(\exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)\right) = \exp\left(\frac{u^\top\nabla^2 f(x^*)u}{2}\right). \qquad \text{(D.7)}$$

From the second order sufficient conditions for $f(x^*)$ to be a maximum, we have $u^\top\nabla^2 f(x^*)u \leq 0\ \forall\ u\in\mathcal{U}$ with equality only when $u=0$ [134]. This implies that Eq. (D.6) and Eq. (D.7) are both bounded functions.

By definition, we have $g(x^*-\sqrt{\epsilon}u)\leq 0\,\forall\, u\in\mathcal{U}$, which implies that $|\exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)| \leq 1$. Consequently, the integrand in the numerator of Eq. (D.5) is dominated by

$\|\varphi(\cdot)\|_\infty$, that is

$$\left|\varphi(x^* - \sqrt{\varepsilon}u)\exp\left(\frac{g(x^* - \sqrt{\varepsilon}u)}{\varepsilon}\right)\right| \leq \|\varphi(\cdot)\|_\infty, \tag{D.8}$$

and the integrand in the denominator is dominated by 1, that is

$$\left|\exp\left(\frac{g(x^* - \sqrt{\varepsilon}u)}{\varepsilon}\right)\right| \leq 1. \tag{D.9}$$

Together Eqs. (D.6) to (D.9) are the sufficient conditions for applying the dominated convergence theorem [19], allowing us to commute all limits and integrals in Eq. (D.5), yielding our desired result:

$$\begin{aligned}
\lim_{\varepsilon\to 0}\int_{\mathcal{X}} \varphi(x)p_\varepsilon(x)dx &= \lim_{\varepsilon\to 0}\left(\frac{\int_{\mathcal{U}} \varphi(x^* - \sqrt{\varepsilon}u)\exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)du}{\int_{\mathcal{U}}\exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)du}\right), \\
&= \frac{\int_{\mathcal{U}} \lim_{\varepsilon\to 0}\left(\varphi(x^* - \sqrt{\varepsilon}u)\exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)\right)du}{\int_{\mathcal{U}}\lim_{\varepsilon\to 0}\left(\exp\left(\frac{g(x^*-\sqrt{\varepsilon}u)}{\varepsilon}\right)\right)du}, \\
&= \frac{\int_{\mathcal{U}} \varphi(x^*)\exp\left(u^\top\nabla^2 f(x^*)u\right)du}{\int_{\mathcal{U}}\exp\left(u^\top\nabla^2 f(x^*)u\right)du}, \\
&= \varphi(x^*)\frac{\int_{\mathcal{U}}\exp\left(u^\top\nabla^2 f(x^*)u\right)du}{\int_{\mathcal{U}}\exp\left(u^\top\nabla^2 f(x^*)u\right)du}, \\
&= \varphi(x^*).
\end{aligned}$$

□

D.3.3 Optimal Boltzmann Distributions as Optimal Policies

Lemma D.1 (Lower and Upper limits of $\mathcal{L}(\omega,\theta)$)**.** *i) For any $\varepsilon_\omega > 0$ and $\pi_\theta(a|s) = \delta(a^*)$, we have $\mathcal{L}(\omega,\theta) = -\infty$. ii) For $\hat{Q}_\omega(\cdot) > 0$ and any non-deterministic $\pi_\theta(a|s)$,* $\lim_{\varepsilon_\omega\to 0}\mathcal{L}(\omega,\theta) = \infty$.

Proof. To prove i), we substitute $\pi_\theta(a|s) = \delta(a^*)$ into $\mathcal{L}(\omega, \theta)$ from Eq. (6.4), yielding:

$$\begin{aligned}\mathcal{L}(\omega, \theta) &= \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\delta(a^*)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] + \mathcal{H}(\delta(a^*))\right], \\ &= \mathbb{E}_{s\sim d(s)}\left[\frac{\hat{Q}_\omega(a^*, s)}{\varepsilon_\omega} + \mathcal{H}(\delta(a^*))\right], \end{aligned} \tag{D.10}$$

We now prove that $\mathcal{H}(\delta(a^*)) = -\infty$ for any a^*. Let $p : \mathcal{X} \to [0, 1]$ be any zero-mean, unit variance distribution. Using a transformation of variables, we have $U = \sigma\mathcal{X} + a^*$ and hence $p(a) = \frac{1}{\sigma}p(\sigma x - a^*)$. We can therefore write our Dirac-delta distribution as

$$\delta(a^*) = \lim_{\sigma\to 0} p(a) = \lim_{\sigma\to 0} \frac{1}{\sigma}p(\sigma x - a^*).$$

Substituting into the definition of differential entropy, we obtain:

$$\begin{aligned}\mathcal{H}(\delta(a^*)) &= \lim_{\sigma\to 0} \mathcal{H}(p(a)) \\ &= \lim_{\sigma\to 0} \mathcal{H}\left(\frac{1}{\sigma}p(\sigma x - a^*)\right), \\ &= -\lim_{\sigma\to 0} \int \frac{1}{\sigma}p(\sigma x - a^*) \log\left(\frac{1}{\sigma}p(\sigma x - a^*)\right) da, \\ &= -\lim_{\sigma\to 0} \int \frac{1}{\sigma}p(\sigma x - a^*) \log\left(p(\sigma x - a^*)\right) da + \lim_{\sigma\to 0} \int \frac{1}{\sigma}p(\sigma x - a^*) \log\left(\sigma\right) da, \\ &= -\int \delta(a^*) \log\left(p(-a^*)\right) da + \lim_{\sigma\to 0} \log\left(\sigma\right), \\ &= -\log(p(-a^*)) + \lim_{\sigma\to 0} \log\left(\sigma\right), \\ &= -\infty. \end{aligned} \tag{D.11}$$

Substituting for $\mathcal{H}(\delta(a^*))$ from Eq. (D.11) in Eq. (D.10) yields our desired result:

$$\mathcal{L}(\omega, \theta) = \mathbb{E}_{s\sim d(s)}\left[\frac{\hat{Q}_\omega(a^*, s)}{\varepsilon_\omega}\right] + \mathbb{E}_{s\sim d(s)}\left[\mathcal{H}(\delta(a^*))\right],$$

$$= \frac{\mathbb{E}_{s\sim d(s)}\left[\hat{Q}_\omega(a^*, s)\right]}{\varepsilon_\omega} + (\lim_{\sigma\to 0}\log(\sigma) - \log(p(-a^*)))\mathbb{E}_{s\sim d(s)}\left[1\right],$$

$$= -\infty,$$

where our final line follows from the first term being finite for any $\varepsilon_\omega > 0$.

To prove ii), we take the limit $\varepsilon_\omega \to 0$ of $\mathcal{L}(\omega, \theta)$ in Eq. (6.4):

$$\begin{aligned}\lim_{\varepsilon_\omega\to 0}\mathcal{L}(\omega,\theta) &= \lim_{\varepsilon_\omega\to 0}\left(\frac{\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]}{\varepsilon_\omega} + \mathbb{E}_{d(s)}\left[\mathcal{H}(\pi_\theta(a|s))\right]\right),\\ &= \lim_{\varepsilon_\omega\to 0}\left(\frac{\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]}{\varepsilon_\omega}\right) + \mathbb{E}_{d(s)}\left[\mathcal{H}(\pi_\theta(a|s))\right],\\ &= \infty.\end{aligned}$$

where our last line follows from $\mathcal{H}(\pi_\theta(a|s))$ being finite for any non-deterministic $\pi_\theta(a|s)$ and $\hat{Q}_\omega(\cdot) > 0 \implies \mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right] > 0$.

□

Theorem D.2 (Optimal Boltzmann Distributions as Optimal Policies)**.** *For any pair* $\{\omega^*, \theta^*\}$ *that maximises* $\mathcal{L}(\omega, \theta)$ *defined in Eq.* (6.4)*, the corresponding variational policy induced must be optimal, i.e.* $\{\omega^*, \theta^*\} \in \arg\max_{\omega,\theta}\mathcal{L}(\omega, \theta) \implies \pi_{\omega^*}(a|s) \in \Pi^*$*. Moreover, any* θ^* *s.t.* $\pi_{\theta^*}(a|s) = \pi_{\omega^*}(a|s) \implies \theta^* \in \arg\max_{\omega,\theta}\mathcal{L}(\omega, \theta)$.

Proof. Our proof is structured as follows: Firstly, we prove that $\varepsilon_{\omega^*} = 0$ is both a necessary and sufficient condition for any $\omega^* \in \arg\max_{\omega,\theta}\mathcal{L}(\omega, \theta)$ with $\hat{Q}_{\omega^*}(\cdot) > 0$. We then verify that $\hat{Q}_{\omega^*}(\cdot) > 0$ is satisfied by our framework and $\varepsilon_{\omega^*} = 0$ is feasible. Finally, we prove that $\varepsilon_{\omega^*} = 0$ is sufficient for $\pi_{\omega^*}(a|s) \in \Pi^*$.

To prove necessity, assume there exists an optimal ω^* such that $\varepsilon_{\omega^*} \neq 0$. As $\varepsilon_\omega \geq 0$,

it must be that $\varepsilon_{\omega^*} > 0$. Consider $\mathcal{L}(\omega, \theta)$ as defined in Eq. (6.4):

$$\mathcal{L}(\omega, \theta) = \frac{\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]}{\varepsilon_\omega} + \mathbb{E}_{d(s)}\left[\mathcal{H}(\pi_\theta(a|s))\right].$$

As $\pi_\theta(a|s)$ has finite variance, $\mathcal{H}(\pi_\theta(a|s))$ is upper bounded, and as $\hat{Q}_\omega(\cdot)$ is upper bounded, $\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]$ is upper bounded too. Together, this implies that $\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]$ is upper bounded for $\varepsilon_{\omega^*} > 0$. From Assumption 6.2, there exists $\omega^\diamond \in \Omega$ such that $\varepsilon_{\omega^\diamond} = 0$. From Lemma 6.1, there exists θ^* such that $\lim_{\varepsilon_{\omega^*}\to 0} \mathcal{L}(\omega^\diamond, \theta^*) = \infty$, implying $\mathcal{L}(\omega^*, \theta^*) < \mathcal{L}(\omega^\diamond, \theta^*)$ which is a contradiction.

To prove sufficiency, we take $\arg\max_\omega \mathcal{L}(\omega, \theta)$:

$$\begin{aligned}
\arg\max_\omega \mathcal{L}(\omega, \theta) &= \arg\max_\omega \left(\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] + \mathbb{E}_{d(s)}\left[\mathcal{H}(\pi_\theta(a|s))\right] \right), \\
&= \arg\max_\omega \left(\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] \right), \\
&= \arg\max_\omega \left(\frac{\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]}{\varepsilon_\omega} \right).
\end{aligned}$$

Assume that ⓘ $\hat{Q}_{\omega^*}(\cdot) > 0$. It then follows:

$$\begin{aligned}
\arg\max_\omega \mathcal{L}(\omega, \theta) &= \arg\max_\omega \left(\frac{\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]}{\varepsilon_\omega} \right), \\
&\quad \arg\min_\omega \left(\frac{\varepsilon_\omega}{\mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]} \right), \\
&= \arg\min_\omega \varepsilon_\omega,
\end{aligned}$$

which, as $\varepsilon_\omega \geq 0$, is satisfied for any $\omega^* \in \Omega$ *s.t.* $\varepsilon_{\omega^*} = 0$, proving sufficiency.

Assume now ⓘⓘ $\hat{Q}_{\omega^*}(\cdot)$ is locally smooth with a unique maximum over actions according to Definition 6.1. Under this condition we can apply Theorem D.1 and our Boltzmann distribution tends towards a Dirac-delta function:

$$\pi_{\omega^*}(a|s) = \lim_{\varepsilon_\omega \to 0} \left(\frac{\exp\left(\frac{\hat{Q}_{\omega^*}(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_{\omega^*}(h)}{\varepsilon_\omega}\right) da} \right) = \delta(a = \arg\max_{a'} \hat{Q}_{\omega^*}(s, a')), \tag{D.12}$$

which is a greedy policy w.r.t. $\hat{Q}_{\omega^*}(\cdot)$. From Definition 6.2, when $\lim_{\varepsilon_\omega \to 0} \pi_\omega(a|s)$ we have $\mathcal{T}_\omega \hat{Q}_\omega(h) = \mathcal{T}^* \hat{Q}_\omega(h)$. Substituting into $\varepsilon_{\omega^*} = 0$ shows our our function approximator must satisfy an optimal Bellman equation:

$$\begin{aligned} \varepsilon_{\omega^*} &= \frac{c}{p} \|\mathcal{T}^* \hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p = 0, \\ \implies \mathcal{T}^* \hat{Q}_{\omega^*}(\cdot) &= \hat{Q}_{\omega^*}(\cdot), \end{aligned}$$

hence $\hat{Q}_{\omega^*}(\cdot) = Q^*(\cdot)$. Under Assumption 6.2, we see that there exists $\omega^* \in \Omega$ *s.t.* $\varepsilon_{\omega^*} = 0$ for $\hat{Q}_{\omega^*}(\cdot) = Q^*(\cdot)$, hence $\varepsilon_{\omega^*} = 0$ is feasible. Moreover, our assumptions ⓘ and ⓘⓘ are satisfied for $\hat{Q}_{\omega^*}(\cdot) = Q^*(\cdot)$ under Assumptions 6.2 and 6.3 respectively. Substituting for $\hat{Q}_{\omega^*}(\cdot) = Q^*(\cdot)$ into $\pi_{\omega^*}(a|s)$ from Eq. (D.12) we recover our desired result:

$$\begin{aligned} \omega^* &\in \arg\max_{\omega} \mathcal{L}(\omega, \theta) \\ \implies \pi_{\omega^*}(a|s) &= \delta(a = \arg\max_{a'} Q^*(s, a')) \in \Pi^*. \end{aligned}$$

From Lemma 6.1, we have that $\mathcal{L}(\omega, \theta) \to \infty = \max_{\omega,\theta} \mathcal{L}(\omega, \theta)$ when $\varepsilon_\omega = 0$ for any $\theta^* \in \Theta$ such that the variational policy is non-deterministic, hence

$$\{\omega^*, \theta^*\} \in \arg\max_{\omega,\theta} \mathcal{L}(\omega, \theta) \implies \pi_{\omega^*}(a|s) \in \Pi^*,$$

as required. □

D.3.4 Maximising the ELBO for θ

Theorem D.3 (Maximising the ELBO for θ)**.** *Maximsing $\mathcal{L}(\omega, \theta)$ for θ with $\varepsilon_\omega > 0$ is equivalent to minimising the expected KL divergence between $\pi_\omega(a|s)$ and $\pi_\theta(a|s)$, i.e. for any $\varepsilon_\omega > 0$, $\max_\theta \mathcal{L}(\omega, \theta) = \min_\theta \mathbb{E}_{d(s)} \left[\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s))\right]$ with $\pi_\omega(a|s) = \pi_\theta(a|s)$ under exact representability.*

Proof. Firstly, we write $\mathcal{L}(\omega, \theta)$ in terms of $\ell(\omega)$ and $\mathrm{KL}(q_\theta(h) \parallel p_\omega(h))$ from Eq. (6.5):

$$\mathcal{L}(\omega, \theta) = \ell(\omega) - \mathrm{KL}(q_\theta(h) \parallel p_\omega(h)),$$

which implies

$$\begin{aligned} \max_\theta \mathcal{L}(\omega, \theta) &= \max_\theta \left(\ell(\omega) - \mathrm{KL}(q_\theta(h) \parallel p_\omega(h))\right), \\ &= \min_\theta \left(\mathrm{KL}(q_\theta(h) \parallel p_\omega(h))\right). \end{aligned} \tag{D.13}$$

for any $\varepsilon_\omega > 0$. Define

$$p_\omega(s) \coloneqq \frac{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) dh}.$$

We now decompose $p_\omega(h)$ as $p_\omega(h) \coloneqq \pi_\omega(a|s) p_\omega(s)$:

$$\begin{aligned} p_\omega(h) &= \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) dh}, \\ &= \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) dh} \cdot \frac{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}, \\ &= \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da} \cdot \frac{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) dh}, \end{aligned}$$

$$= \pi_\omega(a|s)p_\omega(s).$$

Substituting for $p_\omega(h) = \pi_\omega(a|s)p_\omega(s)$ and $q_\theta(h) = d(s)\pi_\theta(a|s)$ into the KL divergence from Eq. (D.13) yields:

$$\begin{aligned}
\mathrm{KL}(q_\theta(h) \parallel p_\omega(h)) &= \mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\log\left(\frac{d(s)\pi_\theta(a|s)}{p_\omega(s)\pi_\omega(a|s)}\right)\right], \\
&= \mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\log\left(\frac{d(s)}{p_\omega(s)}\right)\right] + \mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\log\left(\frac{\pi_\theta(a|s)}{\pi_\omega(a|s)}\right)\right], \\
&= \mathbb{E}_{d(s)}\left[\log\left(\frac{d(s)}{p_\omega(s)}\right)\right]\mathbb{E}_{\pi_\theta(a|s)}\left[1\right] + \mathbb{E}_{d(s)\pi_\theta(a|s)}\left[\log\left(\frac{\pi_\theta(a|s)}{\pi_\omega(a|s)}\right)\right], \\
&= \mathbb{E}_{d(s)}\left[\log\left(\frac{d(s)}{p_\omega(s)}\right)\right] + \mathbb{E}_{d(s)}\left[\mathbb{E}_{\pi_\theta(a|s)}\left[\log\left(\frac{\pi_\theta(a|s)}{\pi_\omega(a|s)}\right)\right]\right], \\
&= \mathrm{KL}(d(s) \parallel p_\omega(s)) + \mathbb{E}_{d(s)}\left[\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s))\right]. \qquad \text{(D.14)}
\end{aligned}$$

Observe that the first term in Eq. (D.14) does not depend on θ, hence taking the minimum yields our desired result:

$$\begin{aligned}
\max_\theta \mathcal{L}(\omega, \theta) &= \min_\theta \left(\mathrm{KL}(d(s) \parallel p_\omega(s)) + \mathbb{E}_{d(s)}\left[\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s))\right]\right), \\
&= \min_\theta \mathbb{E}_{d(s)}\left[\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s))\right].
\end{aligned}$$

Since $\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s)) \geq 0$, it follows that under exact representability, that is there exists $\theta \in \Theta$ s.t. $\pi_\theta(a|s) = \pi_\omega(a|s)$ and hence $\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s)) = 0$, we have $\min_\theta \mathbb{E}_{d(s)}\left[\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\omega(a|s))\right] = 0$. □

D.4 Deriving the EM Algorithm

D.4.1 E-Step

Here we provide a full derivation of our E-step of our variational actor-critic algorithm. The ELBO for our model from Eq. (6.4) with ω_k fixed is:

$$\mathcal{L}(\omega_k, \theta) = \mathbb{E}_{s\sim d(s)}\left[\frac{\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\hat{Q}_{\omega_k}(h)\right]}{\varepsilon_{\omega_k}} + \mathcal{H}(\pi_\theta(a|s))\right].$$

Taking derivatives of the with respect to θ yields:

$$\begin{aligned}\nabla_\theta\mathcal{L}(\omega_k, \theta) =& \mathbb{E}_{s\sim d(s)}\left[\frac{\nabla_\theta\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\hat{Q}_{\omega_k}(h)\right]}{\varepsilon_{\omega_k}}\right] + \nabla_\theta\mathcal{H}(\pi_\theta(a|s)),\\ =& \mathbb{E}_{s\sim d(s)}\left[\frac{\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\hat{Q}_{\omega_k}(h)\nabla_\theta\log\pi_\theta(a|s)\right]}{\varepsilon_{\omega_k}}\right] + \nabla_\theta\mathcal{H}(\pi_\theta(a|s)),\end{aligned}$$

where we have used the log-derivative trick [205] in deriving the final line. Note that in this form, when $\varepsilon_{\omega_k} \approx 0$, our gradient signal becomes very large. To prevent ill-conditioning, we multiply our objective by the constant ε_{ω_k}. As $\varepsilon_{\omega_k} > 0$ for all non-optimal ω_k (see Theorem 6.2), this will not change the solution to the E-step optimisation. Our gradient becomes:

$$\varepsilon_{\omega_k}\nabla_\theta\mathcal{L}(\omega_k, \theta) = \mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\hat{Q}_{\omega_k}(h)\nabla_\theta\log\pi_\theta(a|s)\right] + \varepsilon_{\omega_k}\nabla_\theta\mathcal{H}(\pi_\theta(a|s))\right], \tag{D.15}$$

as required.

D.4.2 M-Step

Here we provide a full derivation of our M-step of our variational actor-critic algorithm. The ELBO for our model from Eq. (6.4) with θ_{k+1} fixed is:

$$\mathcal{L}(\omega, \theta_{k+1}) = \mathbb{E}_{d(s)} \left[\frac{\mathbb{E}_{\pi_{\theta_{k+1}}(a|s)} \left[\hat{Q}_\omega(h) \right]}{\varepsilon_\omega} + \mathcal{H}(\pi_{\theta_{k+1}}(a|s)) \right]$$

Taking derivatives of the with respect to ω yields:

$$\begin{aligned}
\nabla_\omega \mathcal{L}(\omega, \theta_{k+1}) &= \mathbb{E}_{d(s)\pi_{\theta_{k+1}}(a|s)} \left[\nabla_\omega \left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right) \right], \\
&= \mathbb{E}_{d(s)\pi_{\theta_{k+1}}(a|s)} \left[\frac{\nabla_\omega \hat{Q}_\omega(h)}{\varepsilon_\omega} - \frac{\hat{Q}_\omega(h)}{(\varepsilon_\omega)^2} \nabla_\omega \varepsilon_\omega \right], \\
&= \frac{1}{\varepsilon_\omega} \mathbb{E}_{d(s)\pi_{\theta_{k+1}}(a|s)} \left[\nabla_\omega \hat{Q}_\omega(h) \right] - \frac{1}{(\varepsilon_\omega)^2} \mathbb{E}_{d(s)\pi_{\theta_{k+1}}(a|s)} \left[\hat{Q}_\omega(h) \right] \nabla_\omega \varepsilon_\omega,
\end{aligned}$$

where we note that ε_ω does not depend on h, which allowed us to move it in and out of the expectation in deriving the final line. The gradient depends on terms up to $\frac{1}{(\varepsilon_\omega)^2}$, and so we multiply our objective by $(\varepsilon_{\omega_i})^2$ to prevent ill-conditioning when $\varepsilon_\omega \approx 0$. As $(\varepsilon_{\omega_i})^2 > 0$ for all non-convergent ω^*, this does not change the solution to our M-step optimisation and can be seen as introducing an adaptive step size which supplements α_{critic}. Observe that $\frac{\varepsilon_{\omega_i}}{\varepsilon_\omega}\Big|_{\omega=\omega_i} = 1$, which, with a slight abuse of notation, yields our desired result:

$$(\varepsilon_{\omega_i})^2 \nabla_\omega \mathcal{L}(\omega, \theta_{k+1}) = \varepsilon_{\omega_i} \mathbb{E}_{d(s)\pi_{\theta_{k+1}}(a|s)} \left[\nabla_\omega \hat{Q}_\omega(h) \right] - \mathbb{E}_{d(s)\pi_{\theta_{k+1}}(a|s)} \left[\hat{Q}_\omega(h) \right] \nabla_\omega \varepsilon_\omega.$$

In general, calculating the exact gradient of ε_ω is non trivial. We now derive this update for three important cases:

D.4.3 Gradient of the Residual Error

We define $\beta_\omega(h) \coloneqq \mathcal{T}_\omega \hat{Q}_\omega(h) - \hat{Q}_\omega(h)$ and use the notation $\mathbb{E}[\cdot] \triangleq \mathbb{E}_{h\sim\mathcal{U}(h)}[\cdot]$. Taking the derivative yields:

$$\begin{aligned}
\nabla_\omega \varepsilon_\omega &= \frac{1}{2|\mathcal{H}|}\nabla_\omega \|\beta_\omega(h)^2\|_2^2, \\
&= \frac{1}{2}\nabla_\omega \mathbb{E}\left[\beta_\omega(h)^2\right], \\
&= \mathbb{E}\left[\beta_\omega(h)\nabla_\omega \beta_\omega(h)\right]. \qquad \text{(D.16)}
\end{aligned}$$

For targets that do not depend on $\pi_\omega(a|s)$, the gradient of $\nabla_\omega \beta_\omega(h)$ can be computed directly. As an example, consider the update for the Q-learning target:

$$\nabla_\omega \beta_\omega(h) = \mathbb{E}_{s'\sim p(s'|h)}\left[\nabla_\omega \hat{Q}_\omega(a^*, s')\right] - \nabla_\omega \hat{Q}_\omega(h),$$

where $a^* = \arg\max_a \hat{Q}(a, s')$.

For convenience, we denote the expectation $\mathbb{E}_{h'\sim p(s'|h)\pi_\omega(a'|s')}[\cdot]$ as $\mathbb{E}_\omega[\cdot]$. For the Bellman operator target $\mathcal{T}^{\pi_\omega}\hat{Q}_\omega(h) = r(h) + \gamma \mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right]$ that depends on $\pi_\omega(a|s)$, we must solve a recursive equation for $\nabla_\omega \pi_\omega(a|s)$. Consider the gradient of $\beta_\omega(h)$ using $\mathcal{T}^{\pi_\omega}\bullet$:

$$\begin{aligned}
\nabla_\omega \beta_\omega(h) &= \nabla_\omega \left(r(h) + \gamma \mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right] - \hat{Q}_\omega(h)\right), \\
&= \nabla_\omega \gamma \mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right] - \nabla_\omega \hat{Q}_\omega(h), \\
&= \gamma \mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h') + \nabla_\omega \hat{Q}_\omega(h')\right] - \nabla_\omega \hat{Q}_\omega(h), \\
&= \gamma \mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right] + \gamma \mathbb{E}_\omega\left[\nabla_\omega \hat{Q}_\omega(h')\right] - \nabla_\omega \hat{Q}_\omega(h), \\
&= \gamma \mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right] + \Gamma_\omega(h), \qquad \text{(D.17)}
\end{aligned}$$

where $\Gamma_\omega(h) \coloneqq \gamma \mathbb{E}_\omega\left[\nabla_\omega \hat{Q}_\omega(h')\right] - \nabla_\omega \hat{Q}_\omega(h)$. Substituting Eq. (D.17) into Eq. (D.16),

we obtain:

$$\begin{aligned}\nabla_\omega \varepsilon_\omega &= \mathbb{E}\left[\beta_\omega(h)\nabla_\omega \beta_\omega(h)\right], \\ &= \gamma\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right]\right] + \mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right] \end{aligned} \tag{D.18}$$

To find an analytic expression for the first term of Eq. (D.18), we rely on the following theorem:

Theorem D.4 (Analytic Expression for Derivative of Boltzmann Policy Under Expectation)**.** *If $\pi_\omega(a|s)$ is the Boltzmann policy defined in Eq.* (6.3)*, it follows that:*

$$\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right]\right] = \frac{\varepsilon_\omega \mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right]\mathcal{E}_\omega \hat{Q}_\omega(h) + \mathcal{E}_\omega\left[\nabla_\omega \hat{Q}_\omega(h)\right]}{(\varepsilon_\omega)^2\left(1+\gamma\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right]\right]\right)},$$

where $\mathcal{E}_\omega$ is the operator $\mathcal{E}_\omega \bullet \coloneqq \mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\mathcal{M}_\omega \bullet\right]\right]$ and $\mathcal{M}_\omega$ denotes the operator $\mathcal{M}_\omega[\bullet] \coloneqq \bullet - \mathbb{E}_{a\sim\pi_\omega(a|s)}[\bullet]$

Proof. consider the derivative $\pi_\omega(a|s)\nabla_\omega \log \pi_\omega(a|s)$:

$$\begin{aligned}\pi_\omega(a|s)\nabla_\omega \log \pi_\omega(a|s) &= \nabla_\omega \pi_\omega(a|s), \\ &= \nabla_\omega \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}, \\ &= \nabla_\omega \left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)\frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da} \\ &\quad - \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}\cdot\frac{\int \nabla_\omega\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right) da}, \\ &= \nabla_\omega \left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)\pi_\omega(a|s) - \pi_\omega(a|s)\int \nabla_\omega\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)\pi_\omega(a|s)da, \\ &= \nabla_\omega \left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)\pi_\omega(a|s) - \pi_\omega(a|s)\mathbb{E}_{a\sim\pi_\omega(a|s)}\left[\nabla_\omega\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)\right], \end{aligned}$$

$$= \pi_\omega(a|s) \left(\nabla_\omega \left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right) - \mathbb{E}_{a\sim\pi_\omega(a|s)} \left[\nabla_\omega \left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right) \right] \right). \tag{D.19}$$

Finding an expression for $\nabla_\omega \left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right)$, we have:

$$\nabla_\omega \left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right) = \frac{1}{(\varepsilon_\omega)^2} \left(\varepsilon_\omega \nabla_\omega \hat{Q}_\omega(h) - \hat{Q}_\omega(h) \nabla_\omega \varepsilon_\omega \right).$$

Substituting into Eq. (D.19), we obtain:

$$\begin{aligned}
\pi_\omega(a|s)\nabla_\omega \log \pi_\omega(a|s) &= \frac{\pi_\omega(a|s)}{(\varepsilon_\omega)^2} \Big(\varepsilon_\omega \nabla_\omega \hat{Q}_\omega(h) - \hat{Q}_\omega(h)\nabla_\omega \varepsilon_\omega \\
&\quad - \mathbb{E}_{a\sim\pi_\omega(a|s)} \left[\varepsilon_\omega \nabla_\omega \hat{Q}_\omega(h) - \hat{Q}_\omega(h)\nabla_\omega \varepsilon_\omega \right] \Big), \\
&= \frac{\pi_\omega(a|s)}{(\varepsilon_\omega)^2} \Big(\varepsilon_\omega \left(\nabla_\omega \hat{Q}_\omega(h) - \mathbb{E}_{a\sim\pi_\omega(a|s)} \left[\nabla_\omega \hat{Q}_\omega(h) \right] \right) \\
&\quad + \nabla_\omega \varepsilon_\omega \left(\mathbb{E}_{a\sim\pi_\omega(a|s)} \left[\hat{Q}_\omega(h) \right] - \hat{Q}_\omega(h) \right) \Big), \\
&= \frac{\pi_\omega(a|s)}{(\varepsilon_\omega)^2} \left(\varepsilon_\omega \mathcal{M}_\omega \left[\nabla_\omega \hat{Q}_\omega(h) \right] - \nabla_\omega \varepsilon_\omega \mathcal{M}_\omega \hat{Q}_\omega(h) \right),
\end{aligned}$$

where $\mathcal{M}_\omega$ denotes the operator $\mathcal{M}_\omega[\cdot] \coloneqq \cdot - \mathbb{E}_{a\sim\pi_\omega(a|s)}[\cdot]$. Dividing both sides by $\pi_\omega(a|s)$ yields:

$$\nabla_\omega \log \pi_\omega(a|s) = \frac{1}{(\varepsilon_\omega)^2} \left(\varepsilon_\omega \mathcal{M}_\omega \left[\nabla_\omega \hat{Q}_\omega(h) \right] - \nabla_\omega \varepsilon_\omega \mathcal{M}_\omega \hat{Q}_\omega(h) \right).$$

Now, substituting for $\nabla_\omega \varepsilon_\omega = \mathbb{E}\left[\beta_\omega(h) \nabla_\omega \beta_\omega(h) \right]$ from Eq. (D.16) yields:

$$\nabla_\omega \log \pi_\omega(a|s) = \frac{1}{(\varepsilon_\omega)^2} \left(\varepsilon_\omega \mathcal{M}_\omega \left[\nabla_\omega \hat{Q}_\omega(h) \right] - \mathbb{E}\left[\beta_\omega(h) \nabla_\omega \beta_\omega(h) \right] \mathcal{M}_\omega \hat{Q}_\omega(h) \right).$$

Now substituting for $\nabla_\omega \beta_\omega(h) = \gamma \mathbb{E}_\omega \left[(\nabla_\omega \log \pi_\omega(a'|s')) \hat{Q}_\omega(h') \right] + \Gamma_\omega(h)$ from Eq. (D.17),

and re-arranging for $\nabla_\omega \log \pi_\omega(a|s)$:

$$\nabla_\omega \log \pi_\omega(a|s) = \frac{1}{(\varepsilon_\omega)^2}\Bigg(\varepsilon_\omega \mathcal{M}_\omega\left[\nabla_\omega \hat{Q}_\omega(h)\right] - \gamma \mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right]\right] + \mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right]\mathcal{M}_\omega \hat{Q}_\omega(h)\Bigg),$$

$$\nabla_\omega \log \pi_\omega(a|s) + \gamma \mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right]\right] = \frac{1}{(\varepsilon_\omega)^2}\Bigg(\varepsilon_\omega \mathcal{M}_\omega\left[\nabla_\omega \hat{Q}_\omega(h)\right] + \mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right]\mathcal{M}_\omega \hat{Q}_\omega(h)\Bigg).$$

Now, to obtain our desired result, we first multiply both sides by $\hat{Q}_\omega(h)$, take the expectation $\mathbb{E}_\omega$, multiply by $\beta_\omega(h)$ and finally take the expectation $\mathbb{E}$:

$$\begin{aligned}
&\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right]\right]\left(1+\gamma\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right]\right]\right) \\
&\quad = \frac{1}{(\varepsilon_\omega)^2}\Bigg(\varepsilon_\omega \mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\mathcal{M}_\omega\left[\nabla_\omega \hat{Q}_\omega(h')\right]\right]\right] \\
&\qquad + \mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right]\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\mathcal{M}_\omega \hat{Q}_\omega(h')\right]\right]\Bigg). \\
&\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right]\right] = \frac{\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\mathcal{M}_\omega\left[\nabla_\omega \hat{Q}_\omega(h')\right]\right]\right]}{\varepsilon_\omega\left(1+\gamma\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right]\right]\right)} \\
&\qquad + \frac{\mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right]\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\mathcal{M}_\omega \hat{Q}_\omega(h')\right]\right]}{(\varepsilon_\omega)^2\left(1+\gamma\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right]\right]\right)}, \\
&\quad = \frac{\varepsilon_\omega \mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right]\mathcal{E}_\omega \hat{Q}_\omega(h) + \mathcal{E}_\omega\left[\nabla_\omega \hat{Q}_\omega(h)\right]}{(\varepsilon_\omega)^2\left(1+\gamma\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right]\right]\right)},
\end{aligned}$$

as required. □

Using Theorem D.4 to substitute for $\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[(\nabla_\omega \log \pi_\omega(a'|s'))\hat{Q}_\omega(h')\right]\right]$ into Eq. (D.17), we obtain the result:

$$\nabla \varepsilon_\omega = \frac{\varepsilon_\omega \mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right]\mathcal{E}_\omega \hat{Q}_\omega(h) + \mathcal{E}_\omega\left[\nabla_\omega \hat{Q}_\omega(h)\right]}{(\varepsilon_\omega)^2\left(1+\gamma\mathbb{E}\left[\beta_\omega(h)\mathbb{E}_\omega\left[\hat{Q}_\omega(h')\right]\right]\right)} + \mathbb{E}\left[\beta_\omega(h)\Gamma_\omega(h)\right]. \tag{D.20}$$

The second term of Eq. (D.20) is the standard policy evaluation gradient and the first term changes $\pi_\omega(a|s)$ in the direction of increasing ε_ω. We see that all expectations in Eq. (D.20) can be approximated by sampling from our variational policy $\pi_\theta(a|s) \approx \pi_\omega(a|s)$. After a complete E-step, and under Assumption 6.4, we have $\pi_\theta(a|s) = \pi_\omega(a|s)$ and the gradient is exact.

While the first term in Eq. (D.20) is certainly tractable, it presents a formidable challenge for the programmer to implement, especially if unbiased estimates are required; several expressions which involve the multiplication of more than one expectation $\mathbb{E}_\omega$ need to be evaluated. In all of these cases, expectations approximated using the same data will introduce bias, however it is infeasible to sample more than once from the same state in the environment. Like in [203], a solution to this problem is to learn a function approximator for one of the expectations that is updated at a slower rate than the other expectation. Alternatively, these function approximators can be updated using separate data batches from a replay buffer.

A radical approach is simply to neglect this gradient term, which we discuss in Appendix D.5.3. A more considered approach is to use an operator that does not constraint Ω. Consider the operator introduced in Appendix D.5.2,

$$\mathcal{T}_{\omega,k}\bullet = r(h) + \gamma \mathbb{E}_{\omega,k}\left[\bullet\right],$$

where we have used the shorthand for expectation $\mathbb{E}_{\omega,k}\left[\bullet\right] \coloneqq \mathbb{E}_{h' \sim p(s'|h)p_{\omega,k}(a'|s')}\left[\bullet\right]$ and the Boltzmann distribution is defined as

$$p_{\omega,k}(a|s) \coloneqq \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_k}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_k}\right) da}.$$

The incremental residual error is defined as $\varepsilon_{\omega,k} \coloneqq \frac{1}{2|\mathcal{H}|}\|\beta_{\omega,k}(h)\|_2^2 + \varepsilon_k$ and $\beta_{\omega,k}(h) \coloneqq$

$\mathcal{T}_{\omega,k}\hat{Q}_\omega(h) - \hat{Q}_\omega(h)$. Taking gradients of $\varepsilon_{\omega,k}$ directly yields:

$$\nabla_\omega \varepsilon_{\omega,k} = \mathbb{E}\left[\beta_{\omega,k}(h)\nabla_\omega \beta_{\omega,k}(h)\right].$$

where

$$\begin{aligned}
\nabla_\omega \beta_{\omega,k}(h) &= \nabla_\omega \mathbb{E}_{\omega,k}\left[\hat{Q}_\omega(h')\right] - \nabla_\omega \hat{Q}_\omega(h), \\
&= \nabla_\omega \mathbb{E}_{\omega,k}\left[\hat{Q}_\omega(h')\right] - \nabla_\omega \hat{Q}_\omega(h), \\
&= \mathbb{E}_{\omega,k}\left[\nabla_\omega \log p_{\omega,k}(a'|s') + \nabla_\omega \hat{Q}_\omega(h')\right] - \nabla_\omega \hat{Q}_\omega(h). \qquad \text{(D.21)}
\end{aligned}$$

Now, $\nabla_\omega \log p_{\omega,k}(a'|s')$ can be computed directly as:

$$\begin{aligned}
\nabla_\omega \log p_{\omega,k}(a'|s') &= \nabla_\omega \left(\frac{\hat{Q}_\omega(h')}{\varepsilon_k} - \log \int \exp\left(\frac{\hat{Q}_\omega(h')}{\varepsilon_k}\right) da \right), \\
&= \frac{\nabla_\omega \hat{Q}_\omega(h')}{\varepsilon_k} - \int \frac{\nabla_\omega \hat{Q}_\omega(h')}{\varepsilon_k} \frac{\exp\left(\frac{\hat{Q}_\omega(h')}{\varepsilon_k}\right) da}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_k}\right) da} da, \\
&= \frac{\nabla_\omega \hat{Q}_\omega(h')}{\varepsilon_k} - \int \frac{\nabla_\omega \hat{Q}_\omega(h)}{\varepsilon_k} p_{\omega,k}(a'|s') da, \\
&= \frac{\nabla_\omega \hat{Q}_\omega(h')}{\varepsilon_k} - \mathbb{E}_{a' \sim p_{\omega,k}(a'|s')}\left[\frac{\nabla_\omega \hat{Q}_\omega(h)}{\varepsilon_k}\right], \\
&= \mathcal{M}_{\omega,k}\left[\frac{\nabla_\omega \hat{Q}_\omega(h')}{\varepsilon_k}\right],
\end{aligned}$$

where where $\mathcal{M}_{\omega,k}$ denotes the operator $\mathcal{M}_{\omega,k}[\cdot] \coloneqq \cdot - \mathbb{E}_{a \sim p_{\omega,k}(a|s)}[\cdot]$. Substituting into Eq. (D.21) yields:

$$\nabla_\omega \beta_{\omega,k}(h) = \mathbb{E}_{\omega,k}\left[\mathcal{M}_{\omega,k}\left[\frac{\nabla_\omega \hat{Q}_\omega(h')}{\varepsilon_k}\right] + \nabla_\omega \hat{Q}_\omega(h')\right] - \nabla_\omega \hat{Q}_\omega(h).$$

D.4.4 Discussion of E-step

We now explore the relationship between classical actor-critic methods and the E-step. The policy gradient theorem [205] derives an update for the derivative of the RL objective (6.1) with respect to the policy parameters

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^\pi(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[Q^\pi(h)\nabla_\theta \log \pi_\theta(a|s)\right]\right],$$

where $\rho^\pi(s)$ is the discounted-ergodic occupancy, defined formally in [42], and in general not a normalised distribution. To obtain practical algorithms, we collect rollouts and treat them as samples from the steady-state distribution instead.

By contrast, the VIREL policy update in Eq. (D.15) involves an expectation over $d(s)$, which can be any sampling distribution decorrelated from π ensuring all states are visited infinitely often. As $\hat{Q}_\omega(h)$ is also independent of $\pi_\theta(a|s)$, we can move the gradient operator ∇_θ out of the inner integral to obtain

$$\mathbb{E}_{s\sim d(s)}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\nabla_\theta \log \pi_\theta(a|s)\right]\right] = \mathbb{E}_{s\sim d(s)}\left[\nabla_\theta \mathbb{E}_{a\sim\pi_\theta(a|s)}\left[\hat{Q}_\omega(h)\right]\right]$$

This transformation is essential in deriving powerful policy gradient methods such as Expected and Fourier Policy Gradients [42, 56] and holds for deterministic polices [187]. However, unlike in VIREL, it is not strictly justified in the classic policy gradient theorem [205] and MERL formulation [85].

D.5 Relaxations and Approximations

D.5.1 Relaxation of Representability of Q-functions

In our analysis, Assumption 6.2 is required by Theorem 6.2 to ensure that a maximum to the optimisation problem exists, however it can be completely neglected provided that projected Bellman operators are used; moreover, if projected Bellman operators

are used, our M-step is also always guaranteed to converge, even if our E-step does not. Consequently, we can terminate the algorithm by carrying out a complete M-step at any time using our variational approximation and still be guaranteed convergence to a sub-optimal point.

We now introduce the assumption that our action-value function approximator is three-times differentiable over Ω, which is required for convergence guarantees.

Assumption D.1 (Universal Smoothness of $\hat{Q}_\omega(h)$)**.** *We require that* $\hat{Q}_\omega(h) \in \mathbb{C}^3(\Omega)$ *for all* $h \in \mathcal{H}$*,*

We now extend the analysis of [25] to continuous domains. Consider the local linearisation of the function approximator $\hat{Q}_\omega(h) \approx b_\omega^\top(h)\omega$, where $b_\omega(h) \coloneqq \nabla_\omega \hat{Q}_\omega(h)$. We define the projection operator $\mathcal{P}_\omega Q(\cdot) \coloneqq b_\omega^\top(h)\omega'$ where $\tilde{\omega}$ are the parameters that minimise the difference between the action-value function and the local linearisation:

$$\tilde{\omega} \coloneqq \arg\min_{\omega'} \frac{1}{2|\mathcal{H}|}\|Q(h) - b_\omega^\top(h)\omega'\|_2^2. \tag{D.22}$$

Using the notation $\mathbb{E}[\cdot] \triangleq \mathbb{E}_{h\sim\mathcal{U}(h)}[\cdot]$ and taking derivatives of Eq. (D.22) with respect to ω' yields:

$$\begin{aligned}
\nabla_{\omega'} \frac{1}{2|\mathcal{H}|}\|Q(h) - {\omega'}^\top\|_2^2 &= \frac{1}{2}\nabla_{\omega'}\mathbb{E}\left[(Q(h) - b_\omega^\top(h)\omega')^2\right], \\
&= \frac{1}{2}\mathbb{E}\left[\nabla_{\omega'}(Q(h)^2 - 2b_\omega^\top(h)\omega' Q(h) + b_\omega^\top(h)\omega' b_\omega^\top(h)\omega'\right], \\
&= \mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\omega' - b_\omega(h)Q(h)\right].
\end{aligned}$$

Equating to zero and solving for $\tilde{\omega}$, we obtain:

$$\tilde{\omega} = \mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}\mathbb{E}\left[b_\omega(h)Q(h)\right].$$

Substituting into our operator yields:

$$\mathcal{P}_\omega \bullet = b_\omega^\top(h)\mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}\mathbb{E}\left[b_\omega(h)\bullet\right].$$

We can therefore interpret $\mathcal{P}$ as an operator that projects an action-value function onto the tangent space of $\hat{Q}_\omega(h)$ at ω. For linear function approximators of the form $\hat{Q}_\omega(h) = b^\top(h)\omega$, the projection operator is independent of ω and projects Q directly onto the nearest function approximator and the operator [204].

We now replace the residual error in Section 6.3.1 with the projected residual error,

$$\varepsilon_\omega \coloneqq \frac{1}{2|\mathcal{H}|}\left\|\mathcal{P}_\omega\left(\mathcal{T}_\omega\hat{Q}_\omega(h) - \hat{Q}_\omega(h)\right)\right\|_2^2. \tag{D.23}$$

By definition, there always exists fixed point $\omega \in \Omega$ for which $\varepsilon_\omega = 0$, which means that ε_ω now satisfies all requirements in Theorem 6.2 without Assumption 6.2. We can also carry out a complete partial variational M-step by minimising the surrogate ε_ω, keeping $\pi_\omega(a|s) = \pi_\theta(a|s)$ in all expectations. At convergence, we have $\varepsilon_\omega = 0$ in this case.

We now derive the more convenient form of ε_ω from Lemma 1 in [25], extending this result to continuous domains. Let $\beta_\omega(h) \coloneqq \mathcal{T}_\omega\hat{Q}_\omega(h) - \hat{Q}_\omega(h)$. Substituting into Eq. (D.23), we obtain:

$$\begin{aligned}
2\varepsilon_\omega &= \frac{1}{|\mathcal{H}|}\left\|\mathcal{P}_\omega\beta_\omega(h)\right\|_2^2, \\
&= \frac{1}{|\mathcal{H}|}\left\|b_\omega^\top(h)\mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}\mathbb{E}\left[b_\omega(h)\beta_\omega(h)\right]\right\|_2^2, \\
&= \mathbb{E}\left[\mathbb{E}\left[b_\omega^\top(h)\beta_\omega(h)\right]\mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}b_\omega(h)b_\omega^\top(h)\mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}\mathbb{E}\left[\beta_\omega(h)b_\omega(h)\right]\right], \\
&= \mathbb{E}\left[b_\omega^\top(h)\beta_\omega(h)\right]\mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}\mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]\mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}\mathbb{E}\left[\beta_\omega(h)b_\omega(h)\right], \\
&= \mathbb{E}\left[b_\omega^\top(h)\beta_\omega(h)\right]\mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}\mathbb{E}\left[\beta_\omega(h)b_\omega(h)\right].
\end{aligned}$$

Denoting $\zeta_\omega \coloneqq \mathbb{E}\left[b_\omega(h)b_\omega^\top(h)\right]^{-1}\mathbb{E}\left[\beta_\omega(h)b_\omega(h)\right]$ following the analysis in [25], we find the derivative of ε_ω as:

$$\nabla_\omega \varepsilon_\omega = \mathbb{E}\left[(\nabla_\omega \beta_\omega(h))b_\omega^\top(h)\zeta_\omega\right] + \mathbb{E}\left[(\beta_\omega(h) - b_\omega^\top(h)\zeta_\omega)\nabla_\omega^2 \hat{Q}_\omega(h)\zeta_\omega\right].$$

Following the method of [166], the multiplication between the Hessian and ζ_ω can be calculated in $O(n)$ time, which bounds the overall complexity of our algorithm. To avoid bias in our estimate, we learn a set of weights $\hat{\zeta} \approx \zeta_\omega$ on a slower timescale, which we update as:

$$\hat{\zeta}_{k+1} \leftarrow \hat{\zeta}_k + \alpha_{\zeta k}\left(\beta_\omega(h) - b_\omega^\top(h)\zeta_k\right)b_\omega(h), \tag{D.24}$$

where $\alpha_{\zeta k}$ is a step size chosen to ensure that $\alpha_{\zeta k} < \alpha_{\text{critic}}$. The weights are then used to find our gradient term:

$$\nabla_\omega \varepsilon_\omega = \mathbb{E}\left[(\nabla_\omega \beta_\omega(h))b_\omega^\top(h)\hat{\zeta}\right] + \mathbb{E}\left[(\beta_\omega(h) - b_\omega^\top(h)\hat{\zeta})\nabla_\omega^2 \hat{Q}_\omega(h)\zeta_\omega\right].$$

In our framework, the term $\nabla\beta_\omega(h)$ is specific to our choice of operator. In [25], a TD-target is used and parameter updates for ω are given as:

$$\begin{aligned}\omega_{k+1} &= \mathfrak{P}\left(\omega_k + \alpha_{\omega k}(b_k - \gamma b_k')b_k^\top \hat{\zeta}_k - q_k\right), \\ q_k &\coloneqq \left(\beta_{\omega_k}(h_k) - b_k^\top \hat{\zeta}_k\right)\nabla_\omega^2 \hat{Q}_{\omega_k}(h_k)\hat{\zeta}_k\end{aligned} \tag{D.25}$$

where $b_k \coloneqq b_{\omega_k}(h_k)$ and $\mathfrak{P}(\cdot)$ is an operator that projects ω_k into any arbitrary compact set with a smooth boundary, $\mathcal{C}$. The projection $\mathfrak{P}(\cdot)$ is introduced for mathematical formalism and, provided $\mathcal{C}$ is large enough to contain all solutions $\left\{\omega | \mathbb{E}\left[\beta_\omega(h)\nabla_\omega \hat{Q}_\omega(h)\right] = 0\right\} \subseteq \mathcal{C}$, has no bearing on the updates in practice. Under Assumption D.1, provided the step size conditions $\sum_k^\infty \alpha_{\zeta k} = \sum_k^\infty \alpha_{\omega k} = \infty$, $\sum_k^\infty \alpha_{\zeta k}^2 <, \sum_k^\infty \alpha_{\omega k}^2 < \infty$ and $\lim_{k\to\infty} \frac{\alpha_{\zeta k}}{\alpha_{\omega k}} = 0$ hold and $\mathbb{E}[b_\omega(h)b_\omega^\top(h)]$ is non-singular

$\forall \omega \in \Omega$, the analysis in Theorem 2 of [25] applies and the updates in Eqs. (D.24) and (D.25) are guaranteed to converge to the TD fixed point. This demonstrates using data sampled from any variational policy $\pi_\theta(a|s)$ to update ω_k as Eqs. (D.24) and (D.25), ω_k will converge to a fixed point.

D.5.2 Relaxation of Constraints on Ω

As discussed in Section 6.3.1, using the Bellman operator $\mathcal{T}^{\pi_\omega}\bullet$ induces a constraint on the set of parameters Ω. While this constraint can be avoided using the optimal Bellman operator $\mathcal{T}^*\bullet \coloneqq r(h) + \gamma \mathbb{E}_{h' \sim p(s'|h)} \left[\max_{a'}(\bullet)\right]$, evaluating $\max_{a'}(\hat{Q}_\omega(h'))$ may be difficult in large continuous domains. We now make a slight modification to our model in Section 6.3.1 to accommodate a Bellman operator that avoids these two practical difficulties.

Firstly, we introduce a new Boltzmann distribution $p_{\omega,k}(a|s)$:

$$p_{\omega,k}(a|s) \coloneqq \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_k}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_k}\right) da},$$

where $\{\varepsilon_k\}$ is a sequence of positive constants $\varepsilon_k \geq 0$, $\lim_{k\to\infty} \varepsilon_k = 0$. We now introduce a new operator $\mathcal{T}_{\omega,k}\bullet$, defined as is the Bellman operator for $p_{\omega,k}(a|s)$:

$$\mathcal{T}_{\omega,k}\bullet \coloneqq \mathcal{T}^{p_{\omega,k}}\bullet = r(h) + \gamma \mathbb{E}_{h' \sim p(s'|h)p_{\omega,k}(a'|s')} \left[\bullet\right]. \tag{D.26}$$

Let $\pi_{\omega,k}(a|s)$ be the Boltzmann policy:

$$\pi_{\omega,k}(a|s) \coloneqq \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_{\omega,k}}\right)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_{\omega,k}}\right) da},$$

where the residual error $\varepsilon_{\omega,k} \coloneqq \frac{c}{p}\|\mathcal{T}_{\omega,k}\hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p + \varepsilon_k$. It is clear that $\mathcal{T}_{\omega,k}\bullet$ does not constrain Ω as ε_k has no dependency on ω and $\pi_{\omega,k}(a|s)$ is well defined for

all $\omega \in \Omega$.

We now formally prove that $\min_\omega \lim_{k\to\infty} \varepsilon_{\omega,k} = \min_\omega \varepsilon_\omega$, and so minimising $\varepsilon_{\omega,k}$ is the same as minimising the objective ε_ω from Section 6.3.1 and that $\mathcal{T}_{\omega,k^\bullet} \in \mathbb{T}$. We also prove that $\min_\omega \lim_{k\to\infty} \varepsilon_{\omega,k} = \lim_{k\to\infty} \min_\omega \varepsilon_{\omega,k}$ (i.e. that min and lim commute), which allows us to minimise our objective incrementally over sequences $\varepsilon_{\omega,k}$.

Theorem D.5 (Incremental Optimisation of $\varepsilon_{\omega,k}$)**.** *Let* $\varepsilon_{\omega,k} \coloneqq \frac{c}{p}\|\mathcal{T}_{\omega,k}\hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p + \varepsilon_k$ *and* $\mathcal{T}_{\omega,k}$ *be the Bellman operator defined in Eq.* (D.26)*. It follows that i)* $\mathcal{T}_{\omega,k^\bullet} \in \mathbb{T}$*, ii)* $\min_\omega \lim_{k\to\infty} \varepsilon_{\omega,k} = \min_\omega \varepsilon_\omega$ *and iii)* $\min_\omega \lim_{k\to\infty} \varepsilon_{\omega,k} = \lim_{k\to\infty} \min_\omega \varepsilon_{\omega,k}$

Proof. To prove i), we take the limit $\lim_{k\to\infty} \mathcal{T}_{\omega,k}\hat{Q}_\omega(h) = \mathcal{T}^*\hat{Q}_\omega(h)$:

$$\lim_{k\to\infty} \mathcal{T}_{\omega,k}\hat{Q}_\omega(h) = r(h) + \lim_{k\to\infty} \gamma\mathbb{E}_{h'\sim p(s'|h)p_{\omega,k}(a'|s')}\left[\hat{Q}_\omega(h)\right].$$

Observe that from Theorem 6.1, we have

$$\lim_{\varepsilon_k\to\infty} \gamma\mathbb{E}_{h'\sim p(s'|h)p_{\omega,k}(a'|s')}\left[\hat{Q}_\omega(h)\right] = \gamma\mathbb{E}_{h'\sim p(s'|h)\delta(a=\arg\max_{a'}(\hat{Q}_\omega(a',s))}\left[\hat{Q}_\omega(h)\right],$$

hence:

$$\begin{aligned}
\lim_{k\to\infty} \mathcal{T}_{\omega,k}\hat{Q}_\omega(h) &= r(h) + \lim_{k\to\infty} \gamma\mathbb{E}_{h'\sim p(s'|h)p_{\omega,k}(a'|s')}\left[\hat{Q}_\omega(h)\right], \\
&= r(h) + \gamma\mathbb{E}_{h'\sim p(s'|h)\delta(a=\arg\max_{a'}(\hat{Q}_\omega(a',s))}\left[\hat{Q}_\omega(h)\right], \\
&= r(h) + \gamma\mathbb{E}_{s'\sim p(s'|h)}\left[\max_{a'}(\hat{Q}_\omega(h))\right], \\
&= \mathcal{T}^*\hat{Q}_\omega(h).
\end{aligned}$$

Our operator is therefore constructed such that in the limit $k \to \infty$, we recover the optimal Bellman operator. Observe too that as $\frac{c}{p}\|\mathcal{T}_{\omega,k}\hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p \geq$

0, we have $\varepsilon_{\omega,k} > 0$ for all $\varepsilon_k > 0$. From Theorem 6.1, we have $\pi_{\omega,k}(a|s) \to \delta(a = \arg\max_{a'}(\hat{Q}_\omega(a', s))$ when $\varepsilon_{\omega,k} = 0$, which therefore can only occur when $\lim_{k\to\infty} \varepsilon_k = 0$. Under this limit, we have $\lim_{k\to\infty} \mathcal{T}_{\omega,k} = \mathcal{T}^*$ and so $\mathcal{T}_{\omega,k} \in \mathbb{T}$, as required for i).

To prove ii), consider taking the limit of $\varepsilon_{\omega,k}$ directly:

$$\begin{aligned}
\lim_{k\to\infty} \varepsilon_{\omega,k} &= \lim_{k\to\infty} \left(\frac{c}{p} \|\mathcal{T}_{\omega,k}\hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p + \varepsilon_k \right), \\
&= \lim_{k\to\infty} \left(\frac{c}{p} \|\mathcal{T}_{\omega,k}\hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p \right) + \varepsilon_\infty, \\
&= \frac{c}{p} \| \lim_{k\to\infty} \mathcal{T}_{\omega,k}\hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p, \\
&= \frac{c}{p} \|\mathcal{T}^*\hat{Q}_\omega(h) - \hat{Q}_\omega(h)\|_p^p, \\
&= \varepsilon_\omega, \qquad \text{(D.27)}
\end{aligned}$$

as required.

To prove iii), let $\tilde{\omega}_k$ be the minimiser of $\varepsilon_{\omega,k}$, that is $\tilde{\omega}_k = \arg\min_\omega \varepsilon_{\omega,k}$. Let $\tilde{\omega}$ be the limit of all such sequences $\tilde{\omega} = \lim_{k\to\infty} \tilde{\omega}_k$ and let $\omega^* = \arg\min_\omega \varepsilon_\omega$. By definition, we have $\varepsilon_{\tilde{\omega}_k,k} \leq \varepsilon_{\omega,k}$. Taking the limit $k \to \infty$ and then the min, we have:

$$\begin{aligned}
&\min \lim_{k\to\infty} \varepsilon_{\tilde{\omega}_k,k} \leq \min \lim_{k\to\infty} \varepsilon_{\omega,k}, \\
&\implies \varepsilon_{\tilde{\omega},\infty} \leq \min \lim_{k\to\infty} \varepsilon_{\omega,k}. \qquad \text{(D.28)}
\end{aligned}$$

Using Assumption 6.2 and Eq. (D.27), it follows that the right hand side of Eq. (D.28) is $\min \lim_{k\to\infty} \varepsilon_{\omega,k} = \min \varepsilon_\omega = 0$, hence $\varepsilon_{\tilde{\omega},\infty} \leq 0$. By definition, $\varepsilon_{\tilde{\omega},\infty} \geq 0$, and so equality must hold. It therefore follows $\lim_{k\to\infty} \min_\omega \varepsilon_{\omega,k} = \varepsilon_{\tilde{\omega},\infty} = 0$, which implies $\min_\omega \lim_{k\to\infty} \varepsilon_{\omega,k} = \lim_{k\to\infty} \min_\omega \varepsilon_\omega = 0$ as required. □

Overall, this result permits us to carry out separate optimisations over $\varepsilon_{\omega,k}$ while gradually increasing $k \to \infty$ to obtain the same result as minimising ε_ω directly.

The advantage to this method is that each minimisation $\varepsilon_{\omega,k}$ involves the operator $\mathcal{T}_{\omega,k}$, which is tractable, mathematically convenient and does not constrain Ω. Note too that, as calculated in Appendix D.4.3, the gradient $\nabla_\omega \varepsilon_{\omega,k}$ is straightforward to implement in comparison with $\nabla_\omega \varepsilon_\omega$ using $\mathcal{T}^{\pi_\omega}$. We save investigating this operator further for future work.

D.5.3 Approximate Gradient Methods and Partial Optimisation

A common trick in policy evaluation is to use a semi-gradient method [202]. Like in supervised methods [27], semi-gradient treats the term $\mathcal{T}_\omega \hat{Q}_\omega(h)$ as a fixed target, rather than a differential function. Introducing the notation $\mathbb{E}[\cdot] \triangleq \mathbb{E}_{h\sim\mathcal{U}(h)}[\cdot]$, the semi-gradient can easily be derived as:

$$\begin{aligned}\nabla_\omega \varepsilon_\omega &= \frac{1}{2}\nabla_\omega \mathbb{E}\left[\left(\vdash\left[\mathcal{T}_\omega \hat{Q}_\omega(h)\right] - \hat{Q}_\omega(h)\right)^2\right], \\ &= -\mathbb{E}\left[\left(\hat{Q}_\omega(h) - \mathcal{T}_\omega \hat{Q}_\omega(h)\right)\nabla_\omega \hat{Q}_\omega(h)\right]\end{aligned}$$

where $\vdash[\cdot]$ is the stopgrad operator, which sets the gradient of its operand to zero, $\vdash[\cdot] = \cdot$, $\nabla \vdash[\cdot] = 0$. The semi-gradient method has no convergence guarantees, and indeed there exist several famous examples of divergence when used with classic RL targets [24, 224, 242], however its ubiquity in the RL community is testament to its ease of implementation and empirical success [153, 202]. We therefore see no reason why it should not be successful for VIREL, a claim which we verify in Section 6.6. In our setting, we replace our M-step with the simplified objective $\omega_{k+1} \leftarrow \arg\min_\omega \varepsilon_\omega$. This is justified because $\arg\min_\omega \varepsilon_\omega$ was the original objective motivated in Section 6.3.1 assuming we have access to as good enough variational policy $\pi_\omega(a|s) \approx \pi_\theta(a|s)$. More formally, our objective $\mathcal{L}(\omega, \theta)$ is maximised for any $\varepsilon_\omega \to 0$, so $\arg\min_\omega \varepsilon_\omega$ can be considered a surrogate objective for $\mathcal{L}(\omega, \theta)$. Using semi-gradients, M-step update becomes:

M-Step (Critic) Semi-gradient: $\omega_{i+1} \leftarrow \omega_i - \alpha_{\text{critic}} \nabla_\omega \varepsilon_\omega|_{\omega=\omega_i}$,

$$\nabla_\omega \varepsilon_\omega = \mathbb{E}\left[\left(\hat{Q}_\omega(h) - \mathcal{T}_\omega \hat{Q}_\omega(h)\right) \nabla_\omega \hat{Q}_\omega(h)\right].$$

We can approximate $\mathcal{T}_\omega \hat{Q}_\omega(h)$ by sampling from the variational distribution $\pi_\theta(a|s)$ and by using any appropriate RL target. Another important approximation that we make is that we perform only partial E- and M-steps, halting optimisation before convergence. From a practical perspective, convergence can often only occur in a limit of infinite time steps anyway, and if good empirical performance result from taking partial E- and M-steps, computation may be wasted carrying out many sub-optimisation steps for little gain.

As analysed by [81], such algorithms fall under the umbrella of the generalised alternating maximisation (GAM) framework, and convergence guarantees are specific to the form of function approximator and MDP. Like in many inference settings, we anticipate that most function approximators and MDPs of interest will not satisfy the conditions required to prove convergence, however variational EM procedures are known to be to empirically successful even when convergence properties are not guaranteed [81, 225]. We demonstrate in Section 6.6 that taking partial EM steps does not hinder our performance.

D.5.4 Local Smoothness of $\hat{Q}_{\omega^*}(\cdot)$

For Theorem 6.1 to hold, we require that $\hat{Q}_{\omega^*}(\cdot)$ is locally smooth about its maximum. Our choice of function approximator may prevent this condition from holding, for example, a neural network with ReLU elements can introduce a discontinuity in gradient at $\max_h \hat{Q}_{\omega^*}(h)$. In practice, a formal Dirac-delta function can only ever emerge in the limit of convergence $\varepsilon_\omega \to 0$. In finite time, we obtain, at best, a nascent delta function; that is a function with very small variance that is 'on the way to convergence' (see, for example, [109] for a formal definition). The mode of

a nascent delta function therefore approximates the true Dirac-delta distribution. When $\hat{Q}_{\omega^*}(\cdot)$ is not locally smooth, functions that behave similarly to nascent delta functions will still emerge at finite time, the mode of which we anticipate provides an approximation to the hardmax behaviour we require for most RL settings.

We also require that $\hat{Q}_{\omega^*}(\cdot)$ has a single, unique global maximum for any state. In reality, optimal Q-functions may have more than one global maxima for a single state corresponding to the existence of multiple optimal policies. To ensure Assumption 6.3 strictly holds, we can arbitrarily reduce the reward for all but one optimal policy. We anticipate that this is unnecessary in practice, as our risk-neutral objective means that a variational policy will be encouraged fit to a single mode anyway. In addition, these assumptions are required to characterise behaviour under convergence to a solution and will not present a problem in finite time where $\hat{Q}_\omega(h)$ is very unlikely to have more than one global optimum anyway.

D.6 Recovering MPO

We now derive the MPO objective from our framework. Under the probabilistic interpretation in Appendix D.1, the objective can be derived using the prior $p_\phi(h) = \mathcal{U}(s)\pi_\phi(a|s)$ instead of the uniform distribution. Following the same analysis as in Appendix D.1, this yields an action-posterior:

$$p_{\omega,\phi}(a|s,\mathcal{O}) = \frac{\exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)\pi_\phi(a|s)}{\int \exp\left(\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right)\pi_\phi(a|s)da}.$$

Again, following the same analysis as in Appendix D.1, our ELBO objective is:

$$\mathcal{L}(\omega,\theta,\phi) = \mathbb{E}_{d(s)}\left[\mathbb{E}_{\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] - \mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\phi(a|s))\right]. \tag{D.29}$$

Including a hyper-prior $p(\phi)$ over ϕ adds an additional term to $\mathcal{L}(\omega, \theta, \phi)$:

$$\mathcal{L}(\omega, \theta, \phi) = \mathbb{E}_{d(s)}\left[\mathbb{E}_{\pi_\theta(a|s)}\left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega}\right] - \mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\phi(a|s))\right] + \log p(\phi).$$

which is exactly the MPO objective, with an adaptive scaling constant ε_ω to balance the influence of $\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_\phi(a|s))$. Without loss of generality, we ignore the hyperprior and analyse Eq. (D.29) instead.

As discussed by [1], the MPO objective is similar to the PPO [185] objective with the KL-direction reversed. In our E-step, we find a new variational distribution $\pi_{\theta_{k+1}}(a|s)$ that maximises the ELBO with ω_k fixed: Doing so yields an identical E-step to MPO. In parametric form, we can use gradient ascent and apply the same analysis as in Appendix D.4.1, obtaining an update

$$\textbf{E-Step (MPO):} \quad \theta_{i+1} \leftarrow \theta_i + \alpha_{\text{actor}}\left(\varepsilon_{\omega_k} \nabla_\theta \mathcal{L}(\omega_k, \phi_k, \theta)|_{\theta=\theta_i}\right),$$

$$\varepsilon_{\omega_k} \nabla_\theta \mathcal{L}(\omega_k, \phi_k, \theta) = \mathbb{E}_{d(s)}\left[\mathbb{E}_{\pi_\theta(a|s)}\left[\hat{Q}_{\omega_k}(h) \nabla_\theta \log \pi_\theta(a|s)\right] - \varepsilon_{\omega_k} \nabla_\theta \mathrm{KL}(\pi_\theta(a|s) \parallel \pi_{\phi_k}(a|s))\right]. \tag{D.30}$$

As a point of comparison, [1] motivate the update in Eq. (D.30) by carrying out a partial E-step, maximising the "one-step" KL-regularised pseudo-likelihood objective. In our framework, maximising Eq. (D.30) constitutes a full E-step, without requiring approximation.

In our M-step, we maximise the LML using the posterior derived from the E-step, yielding the update:

M-Step (MPO): $\omega_{k+1}, \phi_{k+1} \leftarrow \arg\max_{\omega,\phi} \mathcal{L}(\omega, \phi, \theta_{k+1})$,

$$\arg\max_{\omega,\phi} \mathcal{L}(\omega, \phi, \theta_{k+1}) = \arg\max_{\omega,\phi} \left(\mathbb{E}_{d(s)} \left[\mathbb{E}_{\pi_{\theta_{k+1}}} \left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right] - \mathrm{KL}(\pi_{\theta_{k+1}} \parallel \pi_\phi(a|s)) \right] \right).$$

Maximising for ϕ can be achieved exactly by setting $\pi_\phi(a|s) = \pi_{\theta_{k+1}}(a|s)$, under which $\mathrm{KL}(\pi_{\theta_{k+1}} \parallel \pi_\phi(a|s)) = 0$. Maximising for ω is equivalent to finding $\arg\max_\omega \mathbb{E}_{d(s)\pi_{\theta_{k+1}}} \left[\frac{\hat{Q}_\omega(h)}{\varepsilon_\omega} \right]$, which accounts for the missing policy evaluation step, and can be implemented using the gradient ascent updates from Eq. (6.7). Setting $\pi_\phi(a|s) = \pi_{\theta_{k+1}}(a|s)$ is exactly the M-step update for MPO and, like in TRPO [183], means that $\pi_\phi(a|s)$ can be interpreted as the old policy, which is updated only after policy improvement. The objective in Eq. (D.29) therefore prevents policy improvement from straying too far from the old policy, adding a penalisation term $\mathrm{KL}(\pi_\theta(a|s) \parallel \pi_{\text{OLD}}(a|s))$ to the classic RL objective.

D.7 Variational Actor-Critic Algorithm Pseudocode

Algorithms 5 and 6 show the pseudocode for the variational actor-critic algorithms *virel* and *beta* described in Section 6.6. The respective objectives are:

$$\begin{aligned}
J^V(\phi) =& \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\phi(s_t) - \mathbb{E}_{a_t \sim \pi_\theta} \left[Q_\omega(s_t, a_t) \right] \right)^2 \right], \\
J^Q(\omega) =& \mathbb{E}_{(h_t, r_t, s_{t+1}) \sim \mathcal{D}} \left[\frac{1}{2} \left(r_t + \gamma V_{\bar{\phi}}(s_{t+1}) - Q_\omega(h_t) \right)^2 \right], \\
J^{\pi^q}_{virel}(\theta) =& \mathbb{E}_{h_t \sim \mathcal{D}} \left[\log \pi_\theta(a_t|s_t) (\alpha - (Q_\omega(h_t) - V_{\bar{\phi}}(s_t))) \right], \\
J^{\pi^q}_{beta}(\theta) =& \mathbb{E}_{h_t \sim \mathcal{D}} \left[\log \pi_\theta(a_t|s_t) \left(\frac{1-\gamma}{r_{avg}} \varepsilon_\omega - (Q_\omega(h_t) - V_{\bar{\phi}}(s_t)) \right) \right].
\end{aligned}$$

Note that the derivative of the policy objectives can be found using the reparametrisation trick [111, 91], which we use for our implementation.

Algorithm 5 Variational Actor-Critic: *virel*

Initialize parameter vectors ϕ, $\bar{\phi}$, θ, ω, $\mathcal{D} \leftarrow \{\}$

for each iteration **do**

for each environment step **do**

$a_t \sim \pi^q(a|s;\theta)$

$s_{t+1} \sim p(s_{t+1}|s_t, a_t)$

$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$

end for

for each gradient step **do**

$\phi \leftarrow \phi - \lambda_V \hat{\nabla}_\phi J^V(\phi)$ (M-step)

$\omega \leftarrow \omega - \lambda_Q \hat{\nabla}_\omega J^Q(\omega)$ (M-step)

$\theta \leftarrow \theta - \lambda_{\pi^q} \hat{\nabla}_\theta J^{\pi^q}_{virel}(\theta)$ (E-step)

$\bar{\phi} \leftarrow \tau\bar{\phi} + (1-\tau)\bar{\phi}$

end for

end for

Algorithm 6 Variational Actor-Critic: *beta*

Initialize parameter vectors ϕ, $\bar{\phi}$, θ, ω, $\mathcal{D} \leftarrow \{\}$

for each iteration **do**

for each environment step **do**

$a_t \sim \pi^q(a|s;\theta)$

$s_{t+1} \sim p(s_{t+1}|s_t, a_t)$

$\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r(s_t, a_t), s_{t+1})\}$

end for

for each gradient step **do**

$\varepsilon_\omega \leftarrow \mathbb{E}_\mathcal{D}\left[\left(r_t + \gamma V_{\bar{\phi}}(s_{t+1}) - Q_\omega(h_t)\right)^2\right]$

$\phi \leftarrow \phi - \lambda_V \hat{\nabla}_\phi J^V(\phi)$ (M-step)

$\omega \leftarrow \omega - \lambda_Q \hat{\nabla}_\omega J^Q(\omega)$ (M-step)

$\theta \leftarrow \theta - \lambda_{\pi^q} \hat{\nabla}_\theta J^{\pi^q}_{beta}(\theta)$ (E-step)

$\bar{\phi} \leftarrow \tau\bar{\phi} + (1-\tau)\bar{\phi}$

end for

end for

D.8 Experimental details

D.8.1 Parameter Values

Table D.1: Summary of Experimental Parameter Values: Virel

PARAMETER	VALUE
Steps per evaluation	1000
Path Length	999
Discount factor	0.99
Mujoco-v2 Experiments:	
Batch size	128
Net size	300
$\lambda_\beta \approx \frac{1-\gamma}{r_{avg}}$	Humanoid 4e-4 All other 4e-3
Reward scale	Hopper, Half-Cheetah 5 Walker 3 All other 1
Value function learning rate	3e-4
Policy learning rate	3e-4
MLP layout as given in `https://github.com/vitchyr/rlkit`	
Mujoco-v1 Experiments:	
Values as used by [85] in `https://github.com/haarnoja/sac`	

D.8.2 Additional MuJoCo-v1 Experiments

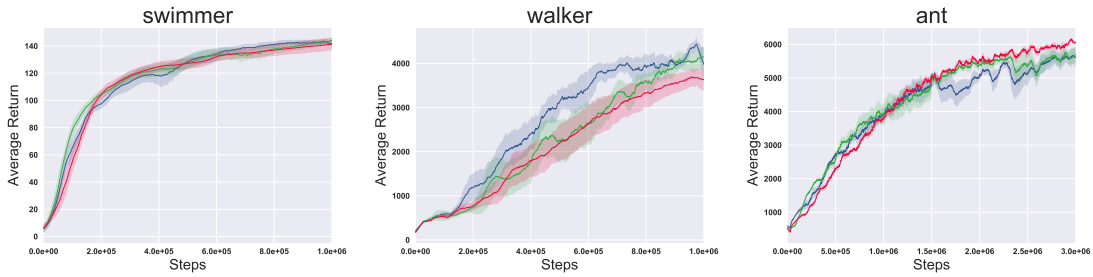


Figure D.3: Training curves on additional continuous control benchmarks Mujoco-v1.

D.8.3 Additional MuJoCo-v2 Experiments

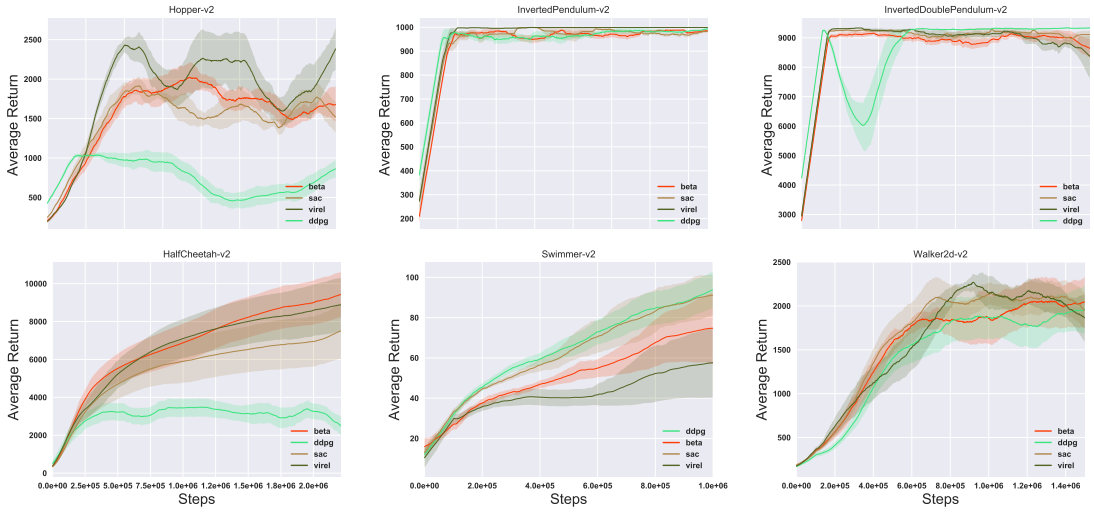


Figure D.4: Training curves on additional continuous control benchmarks gym-Mujoco-v2.

Appendix E

Appendix for Chapter 7

E.1 Additional Definitions and Proofs

We first revisit the concept of MDP homomorphisms [173, 143] which we will use for establishing important results concerning the conditional bisimulation framework.

Definition E.1 (MDP homomorphism [173]). *Let $\Psi \subset S \times U$ is the set of admissible state-action pairs. MDP homomorphism $\mathcal{H}$ from $M = \langle S, U, \Psi, P, r, \gamma, \rho \rangle$ to $M' = \langle S', U', \Psi', P', r', \gamma, \rho' \rangle$ is defined as a surjection $\mathcal{H} : \Psi \to \Psi'$, which is itself defined by a tuple of surjections $\langle f, \{g_s, s \in S\} \rangle$. In particular, $\mathcal{H}((s,a)) := (f(s), g_s(a))$, with $f : S \to S'$ and $g_s : A_s \to A'_{f(s)}$, which satisfies two requirements: Firstly it preserves the reward function:*

$$r'(f(s), g_s(a)) = r(s,a)$$

and secondly it commutes with transition dynamics of M:

$$P'(f(s), g_s(a), f(s')) = P(s, a, [s']_{B_{\mathcal{H}|S}})$$

Here we use the notation $[\cdot]_{B_{\mathcal{H}|S}}$ to denote the *projection* of equivalence classes B that partition Ψ under the relation $\mathcal{H}((s,a)) = (s',a')$ on to S. Isomorphisms $\chi : \Psi \to \Psi'$ can then be formally defined as homomorphisms between $M.M'$ that completely preserve the system dynamics with the underlying functions f, g_s being bijective.

Theorem E.1. *Let* $\mathfrak{met}$ *be the space of bounded pseudometrics on* $Z \times \Theta$ *and* π *a policy that is continuously improving. Define* $\mathcal{F} : \mathfrak{met} \mapsto \mathfrak{met}$ *by*

$$\mathcal{F}(d,\pi)(h_i,h_j) = (1-c)|r^\pi_{h_i} - r^\pi_{h_j}| + cW(d)(P^\pi_{h_i}, P^\pi_{h_j})$$

Then, $\forall c \in (0,1)$, $\mathcal{F}$ *has a least fixed point* $\tilde{d}$ *which is a* π^**-bisimulation metric.*

Proof. First, consider the super-MDP over the unified state space $H \triangleq Z \times \Theta$, $\mathcal{M}_{super} \triangleq \langle H, U, P_H, r_H, \gamma, \rho_H \rangle$, where the H subscripted distributions implicitly account for f, P_Θ, ρ. Similarly, let $\mathcal{M}_\theta$ be the MDP obtained by restricting the context to a particular value θ and $\mathcal{M}_{base} \triangleq \langle S, U, P, r, \gamma, \rho \rangle$. We have that under Assumption 7.1 $\mathcal{M}_\theta$ and $\mathcal{M}_{base}$ are isomorphic and all of $\mathcal{M}_{super}$, $\mathcal{M}_\theta$ and $\mathcal{M}_{base}$ are homomorphic [173, 143]. Thus we can map the policy dynamics in the super-MDP exactly to the base MDP with states S. We now directly apply metric convergence result of Theorem 1 in [251] on the representation space Y, thus showing that the π bisimulation metric converges after repeated applications of the operator $\mathcal{F}$. □

Theorem E.2 (Aggregation value bound)**.** *Given an MDP* $\hat{\mathcal{M}}$ *constructed by aggregating tuples* h *of observation, context in an* ϵ*-neighborhood of the representation space such that* $\delta \triangleq \max_{s,s',\theta_i,\theta_j} ||\phi(f_{\theta_i}(s),\theta_i) - \phi(f_{\theta_j}(s'),\theta_j)| - d_S(s,s')|$*, where* d_S *is a* π^**-bisimulation metric on* S*. Further let* $\hat{\phi}$ *denote the map from any* h *to these clusters, the optimal value functions for the two MDPs follow:*

$$|V^*(h) - \hat{V}^*(\hat{\phi}(h))| \leq \frac{2(\epsilon+\delta)}{(1-\gamma)(1-c)} \forall h \in Z \times \Theta$$

Proof. We use a proof strategy similar to [251]. We have that every θ restriction of $\mathcal{M}_{super}$ is isomorphic to $\mathcal{M}_{base}$ from the above proof. By direct application of Theorem 5.2 in [59] on the MDP $\mathcal{M}_{super}$ for any $h \in Z \times \Theta$:

$$(1-c)|V^*(h) - \hat{V}^*(\hat{\phi}(h))| \leq g(s, \tilde{d}) + \frac{\gamma}{1-\gamma} \max_{s' \in \mathcal{S}} g(s', \tilde{d})$$

where g is the average distance between a state and all other states in its equivalence class under the bisimulation metric $\tilde{d}$. Substituting g with the ϵ-neighborhood ball, and accounting for δ, the error of the representation w.r.t. the metric for each cluster gives us:

$$\begin{aligned}(1-c)|V^*(h) - \hat{V}^*(\hat{\phi}(h))| &\leq 2(\epsilon+\delta) + \frac{\gamma}{1-\gamma} 2(\epsilon+\delta) \\ |V^*(h) - \hat{V}^*(\hat{\phi}(h))| &\leq \frac{1}{1-c}\Big(2(\epsilon+\delta) + \frac{\gamma}{1-\gamma} 2(\epsilon+\delta)\Big) \\ &= \frac{2(\epsilon+\delta)}{(1-\gamma)(1-c)}.\end{aligned}$$

□

Lemma E.1. *Let $f : X \to Y$, $g : Y \to Z$ be two functions with lipschitz constants L_1 and L_2 respectively, then $g(f(\cdot))$ is lipschitz with $L_1 \cdot L_2$*

Proof. Computing deviations for the various functions and using the definition of lipschitzness, we have that:

$$\begin{aligned} df &\leq L_1 dx \\ dg &\leq L_2 dy = L_2 df \\ \implies dg &\leq L_2 L_1 dx \end{aligned}$$

Thus $g(f(\cdot))$ is lipschitz with $L_1 \cdot L_2$ w.r.t. X. □

Theorem E.3 (Generalization to unseen context)**.** *Under Assumption 7.2, Assumption 7.3 we have that for any two contexts θ_i, θ_j:*

$$|J^{\pi_{\theta_i}} - J^{\pi_{\theta_i \leftarrow \theta_j}}| \leq \frac{1}{1-\gamma} E_{\substack{s \sim f_\theta^{-1} \rho^{\pi_{\theta_i}}, \\ a \sim \pi_{\theta_i \leftarrow \theta_j}}} \Big[A^{\pi_{\theta_i}}(s,a) + \frac{2\gamma A_{max}}{1-\gamma} L_\theta^f L_z^\phi L_y^\pi d_\Theta(\theta_i, \theta_j) \Big]$$

where $A_{max} \triangleq \max_s |E_{a \sim \pi_{\theta_i \leftarrow \theta_j}}[A^{\pi_{\theta_i}}(s,a)]|$ and d_Θ is a metric on the context space.

Proof. We have that $d_{TV}(\pi_{\theta_i}, \pi_{\theta_j}) \leq L_\theta^f L_o^\phi L_y^\pi d_\Theta(\theta_i, \theta_j)$ by repeated application of Lemma E.1. We next apply Corollary 2 from [2] that uses the bound for performance difference as a function of policy TV distance giving us the result. □

Theorem E.4 (Simulator fidelity bound)**.** *For an approximately correct simulator $(\hat{r}, \hat{P})$ such that $\max_{s,a} |\hat{r}(s,a) - r(s,a)| \leq \epsilon_R$ and $\max_{s,a} d_{TV}(\hat{P}(s,a), P(s,a)) \leq \epsilon_P$ we have for any policy π:*

$$|J^\pi - \hat{J}^\pi| \leq \frac{\epsilon_R}{(1-\gamma)} + \frac{\gamma \epsilon_P R_{max}}{(1-\gamma)^2}$$

Proof. We proceed similar to [104] for proving the policy value bound. Let us consider the base MDP $\mathcal{M}_{base}$ as defined above. For any projected policy π here, the value function satisfies $\forall s \in S$:

$$\begin{aligned}
&|\hat{V}^\pi(s) - V^\pi(s)| \\
&\leq |(\hat{r}(s,\pi) + \gamma \langle \hat{P}(s,\pi), \hat{V}^\pi \rangle) - (r(s,\pi) + \gamma \langle P(s,\pi), V^\pi \rangle)| \\
&\leq \epsilon_R + \gamma |\langle \hat{P}(s,\pi), \hat{V}^\pi \rangle - \langle P(s,\pi), V^\pi \rangle| \\
&\leq \epsilon_R + \gamma |\langle \hat{P}(s,\pi), \hat{V}^\pi \rangle - \langle P(s,\pi), \hat{V}^\pi \rangle + \langle P(s,\pi), \hat{V}^\pi \rangle - \langle P(s,\pi), V^\pi \rangle| \\
&\leq \epsilon_R + \gamma [|\langle \hat{P}(s,\pi) - P(s,\pi), \hat{V}^\pi \rangle| + |\hat{V}^\pi - V^\pi|_\infty] \\
&\leq \epsilon_R + \gamma [|\langle \hat{P}(s,\pi) - P(s,\pi), \hat{V}^\pi - \frac{R_{max}}{2(1-\gamma)} \mathbf{1} \rangle| + |\hat{V}^\pi - V^\pi|_\infty] \\
&\leq \epsilon_R + \gamma [|\hat{P}(s,\pi) - P(s,\pi)|_1 |\hat{V}^\pi - \frac{R_{max}}{2(1-\gamma)} \mathbf{1}|_\infty + |\hat{V}^\pi - V^\pi|_\infty]
\end{aligned}$$

$$\leq \epsilon_R + \gamma[\frac{\epsilon_P R_{max}}{(1-\gamma)} + |\hat{V}^\pi - V^\pi|_\infty]$$

Here we have viewed the probability transitions and values as vectors. The use of baseline $\frac{R_{max}}{2(1-\gamma)}$ helps tighten the bound by centering the values. We use the definition of TV in last step. As the bound for all $s \in S$ we get after rearranging:

$$|V^\pi - \hat{V}^\pi|_\infty \leq \frac{\epsilon_R}{1-\gamma} + \frac{\gamma \epsilon_P R_{max}}{(1-\gamma)^2}$$

Finally, as J^π is a convex combination of V^π w.r.t. ρ, we can use the above bound to prove the result. □

Theorem E.5 (Complete simulator fidelity bound)**.** *For an approximately correct simulator* $(\hat{r}, \hat{P}, \hat{f})$ *such that* $\max_{s,a} |\hat{r}(s,a) - r(s,a)| \leq \epsilon_R$, $\max_{s,a} d_{TV}(\hat{P}(s,a), P(s,a)) \leq \epsilon_P$ *and* $\epsilon_f \triangleq \max_{s,\theta} d_Y(\phi(\hat{f}_\theta(s)), \phi(f_\theta(s)))$*, we have for any policy* π*:*

$$|J^{\pi_{\hat{f}\leftarrow f}} - \hat{J}^{\pi_{\hat{f}}}| \leq \frac{\epsilon_R}{(1-\gamma)} + \frac{\gamma \epsilon_P R_{max}}{(1-\gamma)^2} + \frac{1}{1-\gamma} E_{\substack{s\sim \hat{f}^{-1}\rho^{\pi_{\hat{f}}}, \\ a\sim\pi_{\hat{f}\leftarrow f}}} \Big[A^{\pi_{\hat{f}}}(s,a) + \frac{2\gamma A_{max}}{1-\gamma} L_y^\pi \epsilon_f \Big]$$

Proof. We consider an intermediate simulator $(\hat{r}, \hat{P}, f)$ which has the same reward and transition as the original simulator $(\hat{R}, \hat{P})$ but uses an exact observation function f (we use $\tilde{}$ to represent quantities associated with this simulator). We can now decompose the difference bound as:

$$|J^{\pi_{\hat{f}\leftarrow f}} - \hat{J}^{\pi_{\hat{f}}}| \leq |J^{\pi_{\hat{f}\leftarrow f}} - \tilde{J}^{\pi_{\hat{f}\leftarrow f}}| + |\tilde{J}^{\pi_{\hat{f}\leftarrow f}} - \hat{J}^{\pi_{\hat{f}}}|$$

Next we have that the $d_{TV}(\pi_{\hat{f}\leftarrow f}, \pi_{\hat{f}}) \leq L_y^\pi \epsilon_f$. Reasoning similarly to Theorem 7.3 for the right term of RHS which gives an upper bound using the TV difference. Finally also applying Theorem 7.4 on the left term of RHS we get the theorem's result. □

E.2 Additional experimental details

E.2.1 Architecture details

We use separate deep networks for actor, critic, transition and reward models. The encoder network for each used 32 filters and a 50 feature dimensions. The actor and critic models each used an MLP trunk of 4 layers and 1024 hidden dimensions on top of the encoder. The reward model used MLP trunk of 2 MLP layers and 512 hidden dimensions on top of the encoder. The transition model type used was a mixture of Gaussians of ensemble size 5. Each component in the transition ensemble uses a 2 MLP layers of 768 hidden dimensions on top of the encoder with the final layer bifurcating for a value for mean and standard deviation per feature dimension. Layer normalization was used for the reward and transition models. Target networks were used for value estimates and were updated every 4 epochs. Relu non-linearity was used for the networks. We exponentially anneal the representation loss with weight $(1.8 - 0.8 * 2^{\frac{\text{steps}}{\text{total steps}}})$. We use identical architectures for the overlapping components of the baseline. The reconstruction agent uses an image decoder with an MLP followed by 2 deconvolution layers with the intermediate layer using 32 filters. Adam optimizer was used for training the parameters of the networks used. Grid search was used for tuning the hyperparameters. Our code is based on implementation by [251] for their work. Each seed takes around 4 days to run on an Nvidia V100 GPU.

E.2.2 Hyper-parameters used: Conditional bisimulation

Table E.1: Hyper-parameters used: Conditional bisimulation

PARAMETER	**VALUE**
λ_{base}	0.24
λ_{icc}	0.32
λ_{cc}	0.24
Initial steps	1000
Batch size	512
Action repeat	2
Encoder learning rate	10^{-3}
Encoder τ	$5 \cdot 10^{-3}$
Decoder learning rate	10^{-3}
Frames	1000
Actor learning rate	10^{-3}
Critic learning rate	10^{-3}
Critic τ	10^{-2}
α learning rate	10^{-4}
γ	0.99
Total Steps	$3.5 \cdot 10^{5}$
Temperature	0.1