Opponent Modelling in Multi-Agent Systems

Zheng Tian

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

November 9, 2021

I, Zheng Tian, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Reinforcement Learning (RL) formalises a problem where an intelligent agent needs to learn and achieve certain goals by maximising a long-term return in an environment. Multi-agent reinforcement learning (MARL) extends traditional RL to multiple agents. Many RL algorithms lose convergence guarantee in non-stationary environments due to the adaptive opponents. Partial observation caused by agents' different private observation introduces high variance during the training which exacerbates the data inefficiency. In MARL, training an agent to perform well against a set of opponents often leads to bad performance against another set of opponents. *Non-stationarity*, *partial observation* and *unclear learning objective* are three critical problems in MARL which hinder agents' learning and they all share a cause which is the lack of knowledge of the other agents. Therefore, in this thesis, we propose to solve these problems with opponent modelling methods. We tailor our solutions by combining opponent modelling with other techniques according to the characteristics of problems we facing. Specifically, we first propose ROMMEO, an algorithm inspired by Bayesian inference, as a solution to alleviate the *non-stationarity* in cooperative games. Then we study the *partial observation* problem caused by agents' private observation and design an implicit communication training method named PBL. Lastly, we investigate solutions to the *non-stationarity* and *unclear learning objective* problems in zero-sum games. We propose a solution named EPSOM which aims for finding safe exploitation strategies to play against non-stationary opponents. We verify our proposed methods by varied experiments and show they can achieve the desired performance. Limitations and future works are discussed in the last chapter of this thesis.

Impact Statement

Multi-agent reinforcement learning (MARL) studies decision-making problems where multiple agents interact with each other in an environment. It has great potential and impact on our life because there are plenty of possible applications for MARL such as traffic control, electricity distribution, pandemic prevention, education, robotic rescue and so on. This thesis studies the *non-stationarity*, *partial observation* and *unclear learning objective* problems in MARL. They are all important challenges for the community because many real-world problems that researchers wish to solve for by MARL share some similarities with these challenges. For example, a successful autonomous driving algorithm is expected to be able to adapt to different driving behaviours of vehicles nearby, infer the intention of pedestrians and have a good balance between efficiency and safety. These skills can be seen as solutions to realisations of those abstract challenges listed above in the driving scenario respectively. Hence, studying and solving these challenges will help the landing of MARL research on real-life applications. We focus on opponent modelling based approaches to these challenges as we observe that a common cause of these problems comes from the lack of knowledge of opponents in the environment. In this thesis, we proposed three novel methods to solve one of or a combination of the above problems under different settings and the empirical results show that our methods can achieve greater performance than many popular or classical baselines. This highlights the importance of opponent modelling based methods in MARL when we need to solve the aforementioned problems. Furthermore, none of the assumptions or restrictions considered in the thesis limit the application of the proposed methods to real-life problems. For example, our PBL algorithm as a solution to *partial observation* problem has been examined in a specialised version of contract bridge, a popular card games for human players. Though to a limited extent, we believe our work assists the MARL community in developing artificial intelligence for solving real-life decision-making problems.

Acknowledgements

I have received a lot of help and support from many people during my Ph.D. study and I would like to take the chance to express my gratitude to them.

First of all, I would like to send my most heartfelt thanks to my supervisor Prof. Jun Wang for his help and supervision. His perseverance and passion for scientific research always inspire me. Weren't it for his patience and kindness, my Ph.D. study would not be this pleasant.

My deep gratitude also goes to Dr. Thomas Anthony, Dr. Aleksandar Botev and Prof. David Barber for offering me a precious opportunity to join my first research project ExIt (Expert Iteration). Their help and mentoring during that journey taught me lots of skills and knowledge beyond the project itself.

Thirdly, I want to thank Dr. Weinan Zhang and Dr. Haifeng Zhang. Their encouragement and support helped me to make two important decisions in my academic career path, none of which I would want to miss.

At the meantime, I would also want to thank all members of Jun's group: Rui Luo, Yixin Wu, Zhicheng Gong, Dr. Shuang Zhao, Dr. Xu Chen, Dr. Yali Du and Dr. Yaodong Yang. In particular, Dr. Ying Wen, Minne Li and Dr. Zhen He, Bowen Zheng, Dr. Qiang Zhang, Jie Li and Fanghua Ye from my office left me countless joyful memories in my Ph.D. life.

I am very grateful to Ian Davies for his tireless effort in helping me with many projects. It is also a great honour to work with these excellent people: Dr. Hang Ren, Dr. Haitham Bou Ammar, Tim Warr, Shihao Zou, Lisheng Wu and Faiz Punakkath and I really appreciate their help and support.

My friends made my stay in London a pleasant journey especially during this global pandemic. My sincere thanks go to Hang An, Rui Xu, Hao Li, Dr. Hantao Lou, Lingyu Liu and Richi He.

Most importantly, I would like to thank my parents Guizhen and Linsheng, my sister Wei, my grandparents Tianrong and Yongan for their unconditional love, support and encouragement.

The list of people who I want to thank may be endless but the acknowledgements have an end. I sincerely apologise to people missed in this list.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Reinforcement Learning

Machine learning is a study which enables machines to identify patterns and rules from data, build models to explain observations and learn without being explicitly programmed. Those functionalities are core aims of artificial intelligence (AI) and machine learning has become the driver of recent AI's developments. Within the broad area of machine learning, reinforcement learning (RL) is a crucial part and has gained increasing attention from both industry and academia. It is a study of decision making and optimal control in real or simulated environments. In a standard RL problem, an agent selects an action from its policy given the current observation, then it receives the corresponding reward and the next observation. This process may repeat for many times depending on the nature of the problem.

If we consider supervised learning (SL) as strong signal learning where human experts tell machines exact expected outputs given inputs and unsupervised learning (UL) as no signal learning where machines need to identify patterns from and learn representations of observed data, RL can be regarded as delayed weak signal learning where an action is accessed by some scalar values only few or many steps after that action being taken. This is challenging as RL algorithms aim at finding optimal policies which an agent uses to make decisions but delayed weak feedback makes it difficult to assign credits to correct actions, a problem called the credit assignment problem in (Minsky, 1961).

Learning and *planning* are two fundamental problem-solving techniques lying at the two extreme ends of the RL continuum. Though they originated from different times and fields, they were integrated by modern RL researchers in the 1980s (Werbos, 1987; Watkins, 1989). The essential idea behind the *learning* technique is trial and error which stemmed from psychology and was first succinctly discussed in (Thorndike, 1911). The *learning* technique aims at optimising the policy for decision making by reinforcing the training agent's behaviours with experiences collected from the agent's real interactions with the environment. Namely, the probability of selecting an action is increased if the action incurs a positive reward and is decreased otherwise. On the other hand, the *planning* technique began in control theory and methods known as dynamic programming (DP) which was first introduced in (Bellman and Corporation, 1957) were typical representatives of *planning* technique. The *planning* technique can plan an optimal policy for decision making. The "planning" term emphasises the fact that it does not require real interactions with the environment in contrast to the *learning* technique which can potentially save the simulation cost. However, the *planning* method requires the full knowledge of the environment.

A distinctive class of approaches known as temporal difference (TD) learning lies in between *learning* and *planning*. It aims at learning an estimate of certain quantities, e.g., the values of actions. It retrieves learning signals from data collected by interacting with the environment (the *learning* technique side) and correct its current estimates as DP methods (the *planning* side). TD learning firstly appeared in (Samuel, 1959) and was further studied and developed by Klopf (Klopf, 1972), Sutton and Barto (1981; 1982) and many other RL researchers. A mature RL method following the prior works known as Q-learning was proposed by Watkins and Dayan (1992) and an enormous amount of classical RL methods in modern RL have been proposed since then.

Classical RL methods often suffer from the curse of dimensionality problem where the computation cost increases exponentially with the complexity of games, e.g., the number of states in a game. Therefore, RL methods were mainly studied on tabular problems (Gronauer and Dieopold, 2021). The revival of neural networks (NNs) (Schmidhuber, 2015) empowered by the significant improvement in graphics processing units (GPUs) computation speed led to the emergence of a plethora of deep RL (DRL) works. As powerful function approximators (Arulkumaran et al., 2017), deep NNs enabled efficient and automatic representation learning. This facilitated state abstraction which is a common dimension reduction approach used in RL but often required human expertise before the advent of DNNs. Consequently, we saw successful applications of DRL works on game playing (Mnih et al., 2015; Silver et al., 2016, 2017, 2018), robotics (Levine et al., 2015; Lillicrap et al., 2015) and biology (Senior et al., 2020).

1.2 Multi-Agent Reinforcement Learning

Multi-Agent System (MAS) generalises the classical decision-making problems by considering multiple intelligent agents in one system (Weiss, 2000). In MAS, each agents has its own observations at each time step. Given their observations, agents need to take actions stochastically or in turns. Then the system will transfer to another state and distribute rewards to each agent based on the joint actions of agents and its current state. In the new state, agents will receive their own new observations and the dynamic repeats. Agents can have the same goal or conflict of interest depending on the nature of the problem and each agent is assumed to maximise its own long-term cumulative rewards. Multi-agent reinforcement learning (MARL) is a branch of RL which focuses on MAS problems.

The origin of the MAS community can date back to the first workshop on distributed artificial intelligence held in June 1980 (Davis, 1980). Although MARL research has been overshadowed by single-agent RL (SARL) since its birth, efforts to address the aforementioned issues have never been stopped (Zhang et al., 2019). Recently, this domain has experienced a resurgence of interest and we have witnessed many significant successes in multi-agent games such as StarCraft II (Vinyals et al., 2017, 2019), Dota2 (OpenAI, 2018) and Texas hold'em (Moravčík et al., 2017; Brown and Sandholm, 2018, 2019). These games are all currently popular multiplayer games with professional tournaments held every year world-wide. Unsurprisingly, the advance in MARL is also accompanied with the development of DNNs due to its strong performance in terms of function approximation. The ongoing resurgence of MARL has not reached an imminent end and significant achievements beyond games are expected to come soon.

1.3 Why Opponent Modelling?

Though MARL studies decision-making problems in a similar vein as SARL, adding extra agents into the environment makes the problem much more challenging in several ways. Firstly, agents' observations normally contain some information about the environment, the history of the play and the opponents but rarely the full knowledge of the whole game including other players. In many cases agents do not share their observations and thus, one agent can have some information that other agents don't. For example, in Poker games, each agent cannot observe others' cards in their hands (Mealing and Shapiro, 2017). Therefore, *partial observation* problem often comes with the introduction of extra agents. When an agent's can not distinguish two states in an environment due to its partial observation, the training feedback will become noisy as the different feedback gained by taking the same action in the two different states will be regarded as the same feedback with noise. This directly increases the variance in the training and cost the agent more samples but only learns a sub-optimal policy.

In addition, an agent's opponents or teammates are often adaptive which means that they will adjust their strategies to play given their interacting experiences with the environment and other players. Consequently, an agent who takes the same action at the same state but at different time steps can receive very different rewards and observations on expectation due to the changes in other players' behaviour patterns, a problem known as *non-stationarity* in MARL (Hernandez-Leal et al., 2017). Many classical value-based methods are popular in RL communities because of their convergence properties and sample efficiency as they could reuse past trajectories for training (Sutton and Barto, 2018). However, in non-stationary environments, these methods lose their convergence properties and it is difficult to reuse the past trajectories due to the changing environments. *Non-stationarity* also challenges policy-based algorithms as it requires them to learn a good policy before the environment changes.

Finally, an agent can encounter different types of opponents or teammates in different tournaments. A strategy that wins against/with one type of opponents/teammates cannot guarantee wins when the agent is facing different types of players. With the concern of not knowing the future opponents/teammates, a safe strategy is to learn a robust policy which can performs not badly with regardless to who the opponents/teammates are. This is effectively the Nash equilibrium solution concept from game theory. However, this safe strategy will forego potential profits if we actually know our opponents/teammates. Hence, defining a *suitable learning objective* for an agent becomes non-trivial (Shoham et al., 2003).

The above challenges induced by the existence of other intelligent agents in the system (*partial observation*, *non-stationarity*, *unclear learning objective*) are deeply related. One critical cause of these issues is that the training agent normally has very little knowledge of its opponents[1]. Specifically, *partial observation* in a MAS is often the direct consequence of the training agent having no or partial knowledge about the observations of its opponents. As for the second problem, it is because the training agent cannot know actions taken by its opponents that the environment it perceives becomes *non-stationary*. Also if we had a powerful oracle foretelling the training agent all the information about the opponents in every tournament, then the *unclear learning objective* would become clear. Namely, the training agent only needs to learn how to take the best actions given the prediction from the oracle. Then, an intuitive way to address these problems will be to help the training agent gain more knowledge about its opponents. Following the example above, an accurate oracle which can predict all information about opponents in fact renders the learning problem a single-agent optimisation which can be solved by typical reinforcement learning.

[1]Hereafter, we will refer to the other players excluding our training agents as opponents regardless of the nature of the games.

In reality, however, there rarely exists such an oracle. Nonetheless, we could learn to build a model which can approximate the oracle's prediction. Opponent modelling is a process where a model is constructed to predict information about the training agent's opponents. A model in this context is essentially a function that takes as input some information observable by the training agent and outputs a prediction of some properties of its opponents including but not limited to the opponents' private information, actions, preferences, goal, etc. This approach is common in the real world. For example, players in many professional sports matches often need to collect information about their opponents in advance to better understand their opponents and be able to predict their strategies (Knottenbelt et al., 2012; Laviers et al., 2009). Sun Tzu, a famous general in Imperial China, has said "If you know both yourself and your enemy, you can win a hundred battles without a single loss." (Tzu, 2002). This highlights the importance of opponent modelling in warfare and the long history of opponent modelling.

In this thesis, we will focus on solving the above problems with opponent modelling based algorithms. Specifically, we propose three novel methods to solve (1) *non-stationarity*, (2) *partial observation* in cooperative environments and (3) *non-stationarity* and the *unclear learning objective* in competitive environments respectively. We examined our methods on different scales of games and show that they all achieve the expected performance and outperforming strong baselines.

1.4 Games as a test bed for algorithms

Formally, a game is a domain of conflict or cooperation between multiple entities including players and the environment. In a game, a player needs to apply an action in one state and then is transferred to another state depending on its current state and the action it chooses. Such a transition can continue for infinitely long or terminate when the player reaches a certain state. In addition to the state transitions, an action often incurs a (delayed) reward and the player's goal is to maximise its long-term cumulative rewards. In this thesis, we characterise games as the dynamic environments which has well pre-defined reward functions and relatively low cost of simulation.

The dynamics where an agent or a group of agents making a sequence of decisions, being transferred to different states and receiving consequences of their decisions can also be seen in many real world problems such as business negotiations, epidemic prevention and control, financial investment, and education, etc. These problems share the same properties with complex games such as high-dimensional state and action space, unknown world dynamics models, delayed feedback and multiple interacting entities. In contrast to real world problems, however, games are normally more cost efficient to simulate, bear less risk and have well-predefined reward functions. Therefore, we mainly investigate the effectiveness of our methods by testing them on games.

1.5 Overview

In this section, we first briefly introduced RL and MARL. We listed three main challenges the MARL community faces and motivated opponent modelling as a solution for them. The language presented in this section may not be mathematically formal but intuitive for an induction purpose. In the following parts of the thesis, we will formally introduce some necessary mathematical preliminaries, survey related works and present a sequence of approaches to address the challenges mentioned above. Specifically, the rest of this thesis is organised as follows:

- In Chapter 2, we review the key concepts of reinforcement learning and multi-agent reinforcement learning.
- In Chapter 3, we provide literature review of challenges in multi-agent reinforcement learning, opponent modelling and relevant techniques we used.
- In Chapter 4, we focus on the *non-stationarity* in cooperative games. Specifically, we formalise the classical multi-agent coordination as a Bayesian inference problem and derive a maximum entropy objective with regularised opponent modelling by variational inference. We propose two off-policy algorithms for optimising this objective and show that they outperform other strong baselines in both tabular and approximate settings.

- In Chapter 5, we consider the *partial observation* problem in cooperative games and assume that agents cannot communicate explicitly. Therefore, we propose an iterative learning style algorithm PBL which can train agents to communicate implicitly by actions. We evaluate our method on simple matrix game and complex non-competitive bridge bidding problems and a modified multi-agent particle environment.

- In Chapter 6, we study the *non-stationarity* and *unclear learning objective* problems in zero-sum games. We propose an algorithm named EPSOM which learns to safely best respond to the predictions of its opponent model and we build the opponent model by Dirichlet process such that we can formalise an opponent's learning process as transitions among different policies.

- In Chapter 7, we discuss common limitations of works present in this thesis and potential solutions for these limitations as our future work. Finally, we briefly discuss why we should continue the study of opponent modelling.

These chapters are based on the following papers and pre-prints:

1. Chapter 4: Zheng Tian, Ying Wen, Zhichen Gong, Faiz Punakkath, Shihao Zou, and Jun Wang. A regularized opponent model with maximum entropy objective. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 602–608. AAAI Press, 2019. ISBN 9780999241141.

2. Chapter 5: Zheng Tian, Shihao Zou, Ian Davies, Tim Warr, Lisheng Wu, Haitham Bou Ammar, and Jun Wang. Learning to communicate implicitly by actions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34: 7261–7268, 04 2020. doi: 10.1609/aaai.v34i05.6217

3. Chapter 6: Zheng Tian, Hang Ren, Yaodong Yang, Yuchen Sun, Xiaohang Tang, Ian Davies, Ziqi Han, and Jun Wang. Learning to safely exploit a non-stationary opponent. *submitted to AAAI 2022 for review*, 05 2021

Chapter 2

Background

In this section we will introduce a shared mathematical foundation across works presented in the thesis. Some preliminaries and formulations specific to a work will be postponed to the chapter where that work is introduced for easier reference and better reading experience.

2.1 Markov Decision Process

Most RL algorithms focus on solving problems which can be formulated as Markov Decision Processes (MDPs) or its variants (Sutton and Barto, 2018).

Definition 1. *An MDP (Bellman, 1957; Puterman, 1994) defines a tuple* $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, p_0, \gamma)$*, where:*

- $\mathcal{S}$ *is a finite set of states;*
- $\mathcal{A}$ *is a finite set of actions;*
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta(\mathcal{S})$*, is a state transition function, a probability distribution over possible next states and* $\Delta(\cdot)$ *denotes a probability simplex;*
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$*, is a reward function, determining the step reward distributed to an agent at a state* $s \in \mathcal{S}$ *taking an action* $a \in \mathcal{A}$*;*
- p_0 *specifies the probability distribution of the initial state* s_0*;*
- *and* $\gamma \in [0, 1]$ *is a discount factor.*

At each time step t, an acting agent takes an action a_t at state s_t, receives a reward $r_t(s_t, a_t) = \mathcal{R}(s_t, a_t)$ and is transited to s_{t+1} following the transition probability $Pr(s_{t+1}|s_t, a_t) = \mathcal{T}(s_t, a_t, s_{t+1})$. In this thesis, we only consider a MDP with bounded rewards. Namely, we always re-scale the rewards to $[-1, 1]$.

A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ is a mapping from state space $\mathcal{S}$ to a distribution over action space $\mathcal{A}$, describing the probability $Pr(a_t|s_t) = \pi(a_t|s_t)$ of the agent selecting an action $a_t \in \mathcal{A}$ at a state $s_t \in \mathcal{S}$. We define the sum of discounted rewards across time the discounted return: $R_t = \sum_{j=t} \gamma^t r_j$. The goal of a training agent is to learn an optimal policy π^* maximising the expectation of the discounted return: $\pi^* = \arg\max_\pi \mathbb{E}_{\pi,\mathcal{T}}[R_t]$, where the expectation is with respect to the stochasticity from the environment and the training agent's policy.

Discounting the future rewards when we calculate the return matches the economic concept that present rewards are valued more than future rewards. However, more importantly, the use of discount factor in MDP is related to the concern with regard to the length of a MDP. We call a MDP episodic if the MDP terminates with a finite number of steps. An episodic MDP can either terminate when an agent reaches some special state we call terminal state or the number of time steps reaches a predefined limit. We define an episode as a sequence of interactions between the agent and the environment starting from the initial state till the current MDP terminates. In episodic MDPs, we are guaranteed to have a finite return $R_t = \sum_{j=t}^{T} \gamma^t r_j$ where the time of termination T can be a random variable whose value varies from an episode to another. However, besides episodic MDPS, we also have the infinite-horizon MDP where interactions between the agent and the environment never terminates. Without discounting the future rewards, we can see that the return $R_t = \sum_{j=t}^{\infty} r_j$ in the infinite-horizon MDP will be infinite which is problematic as the training agent's goal is to maximise the return.

2.1.1 Value Functions

We say a training agent's objective is to maximise the expectation of return in a MDP which can been seen as a function of the agent's policy:

$$\eta(\pi) = \mathbb{E}_{\pi,\mathcal{T}}[R_t].$$

Before taking actions, then an intelligent agent with the mindset of maximising the return may ask questions such as "How good is a state s which I will be transited to?" or "How good is an action a given my current state s?". In RL, we use values functions to answer these questions which can help a training agent make decisions.

We first define the sate value function as:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}^{1}{}_{\pi}[R_t|s_t = s], \\ &= \mathbb{E}_{\pi}[r_t + \gamma R_{t+1}|s_t = s]. \end{aligned}$$

It essentially measures the expectation of return an agent can obtain if it starts from the state s and follows the policy π in the MDP. Similarly, we can also define the state-action value function as:

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[R_t \mid s_t = s, a_t = a\right].$$

It measures the expectation of return an agent obtain if it selects an action a at a state s and then follows the policy π in the MDP.

Recall that we denote an optimal policy π^* as a policy which maximises the expected return from every state in a MDP. Similarly, we can denote value functions

[1]In this thesis, we may omit some super/sub-scripts in notations we have defined before (e.g. $\mathcal{T}$ in $\mathbb{E}_{\pi,\mathcal{T}}[R_t]$) for concise presentation if its absence does not cause ambiguity.

associated with the optimal policy as optimal value functions:

$$
\begin{aligned}
V^*(s) &= \mathbb{E}_{\pi^*}[R_t|s_t = s], \\
&= \max_{\pi} \mathbb{E}_{\pi}[R_t|s_t = s]; \\
Q^*(s,a) &= \mathbb{E}_{\pi^*}[R_t|s_t = s, a_t = a], \\
&= \max_{\pi} \mathbb{E}_{\pi}[R_t|s_t = s, a_t = a].
\end{aligned}
$$

2.1.2 Bellman Equation

The Bellman equation (Bellman and Corporation, 1957) is a critical part of RL and it helps RL researchers to find optimal policies and value functions. Specifically, it decomposes values functions into step rewards and discounted future value functions so that the complex problem of calculating value functions can be divided into simpler and recursive sub-problems. In MDP, a state value $V^{\pi}(s)$ has a Bellman equation as:

$$V^{\pi}(s_t) = \mathbb{E}_{\pi}[r_t + \gamma V^{\pi}(s_{t+1}))].$$

Equivalently, for state-action value functions, we have:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[r(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, a_{t+1}))].$$

2.2 Value-Based Reinforcement Learning

If an agent knows how good a state s is or how good taking an action a at each state s is for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$ in a MDP given its current policy, it can select the best actions in terms of obtaining the greatest expected return. If the new policy resulting from this greedy selection behaviour is different from the agent's current policy, then the agent learns a new policy which is better than the current one. And the process can repeat many times until the agent cannot learn a better policy. This is essentially the intuition behind value-based RL.

Specifically, most value-based RL consists of two parts: policy evaluation and policy improvement (Sutton and Barto, 2018). The former estimates the $V^{\pi}(s)$ and $Q^{\pi}(s,a)$ for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$ in a MDP given its current policy π. The latter improves the current policy by greedily selecting actions with respect to the new value functions. The improvement of the policy is guaranteed by:

Theorem 1. *Policy Improvement Theorem (Bellman and Corporation, 1957): Suppose π and π' are two policies such that:*

$$\mathbb{E}_{a\sim\pi'}\left[Q^{\pi}(s,a)\right] \geq V^{\pi}(s)\forall s \in \mathcal{S}.$$

, then we have:

$$V^{\pi'}(s) \geq V^{\pi}(s)\forall s \in \mathcal{S}.$$

The value functions can be updated in a *full backup* or a *sample backup* fashion depending on the training agent's accessibility to the environment model. At each iteration, the former updates the value functions for all possible states and actions where each update performs a full-width lookahead. As full-width lookahead relies on the knowledge of the transition $\mathcal{T}$ and reward $\mathcal{R}$ functions, the *full backup* requires the training agent to have the model of the environment and thus, we call this type of approaches model-based RL. Without the environment model, we replace the full-width lookahead with sampled interactions from a MDP (including agent's action a, current state s, transited next state s' and reward $r(s,a)$) and update the value functions for states and actions we have visited.

Dynamic programming (DP) was first proposed for solving optimal control problems (Bellman, 1952). Now it has become a general method for solving a complex problem by breaking it into simpler sub-problems and solving these sub-problems in recursive a manner. In RL, full-width update of value functions satisfying Bellman equations is essentially a form of DP. For example, in policy iteration (PI) (Howard, 1960), to evaluate a given policy π, we define the Bellman operator $\mathbf{B}_\pi$ as:

$$\mathbf{B}_\pi(V)(s) = \sum_a \pi(a|s) \sum_{s'} \mathcal{T}(s,a,s')[r(s,a) + \gamma V(s')].$$

Then we can repeatedly apply the operator on our current estimate of value functions till convergence. The above process provides us with a way to evaluate a given policy π which can stand alone by serving as a means to predict how good the policy π is. However, more frequently than PI, the *policy evaluation* is followed by the *policy improvement* where we define a greedy operator $\mathbf{G}$ to return an improved policy given the current value functions:

$$\mathbf{G}(V)(s) = \arg\max_a \sum_a \pi(a|s) \sum_{s'} \mathcal{T}(s,a,s')[r(s,a) + \gamma V(s')].$$

Alternatively, value iteration (VI) (Bellman and Corporation, 1957) combines these two steps into one and directly solves the MDP for the optimal value functions where we have an optimal bellman operator $\mathbf{B}^*$ defined as:

$$\mathbf{B}^*(V)(s) = \max_a \pi(a|s) \sum_{s'} \mathcal{T}(s,a,s')[r(s,a) + \gamma V(s')].$$

Monte Carlo value estimation (MCVE) considers the situation where an agent does not have the environment model thus DP is infeasible. Therefore, we replace the *full backup* with *sample backup*. Together with some other approaches introduced later, MCVE is classified as a model-free RL method which does not require an environment model. Recall the state value function $V_\pi(s)$ given a policy π is the expected return for an agent starting from state s and then following the policy π:

$$V^\pi(s) = \mathbb{E}_\pi[R_t|s_t = s].$$

MCVE methods replace the expectation with empirical average over a set of samples which are often simulated from the environment. Mathematically, given a set of samples over N steps which can spanning multiple episodes, we have:

$$\hat{V}(s) = \frac{\sum_{t=0}^{N} R_t \mathbb{1}_{s_t=s}}{\sum_{t=0}^{N} \mathbb{1}_{s_t=s}},$$

where $\mathbb{1}_{X=x}$ takes value 1 if $X = x$ and 0 otherwise. Therefore, the numerator calculates the sum of all the (discounted) returns resulting from visits to the state s, while the denominator counts those visits. Similarly, for state-action value function, we have:

$$\hat{Q}(s,a) = \frac{\sum_{t=0}^{N} R_t \mathbb{1}_{s_t=s,a_t=a}}{\sum_{t=0}^{N} \mathbb{1}_{s_t=s,a_t=a}}.$$

The above equations assume we have a complete set of training data which often span over many completed episodes before conducting the estimations. However, a more efficient way would be to update these value functions whenever the training agent completes one episode by maintain a running average:

$$\hat{V}(s_t) := \hat{V}(s_t) + \alpha \left(R_t - \hat{V}(s_t)\right), \tag{2.1}$$

where $\alpha \in [0,1]$ is a learning rate.

Temporal difference (TD) learning is an alternative model-free method to MCVE (Samuel, 1959; Sutton, 1988). It is an application of *bootstrapping* on value functions estimations. The core idea of *bootstrapping* is to update an estimate from another estimate so that a training agent can conduct updates before the current episode completes. This is also particularly helpful in terms of variance reduction as lengthy episodes starting from the same state are likely to have dramatically different results due to the randomness from the player's policy and the environment transitions. A basic form of TD learning is TD(0) where we update the state value at the current state s_t as:

$$\hat{V}(s_t) := \hat{V}(s_t) + \alpha \left(r_t + \gamma \hat{V}(s_{t+1}) - \hat{V}(s_t)\right).$$

Comparing the above equation with Equation 2.1, we can see that the Monte Carlo return R_t is replaced with the sum of one-step reward r_t and the discounted estimate of state value for the next state in TD(0). We could regard "0" in TD(0) stands for "no further simulated steps required". Therefore, we have a general recipe for TD(K) as:

$$\hat{V}(s_t) := \hat{V}(s_t) + \alpha \left(\hat{V}_K - \hat{V}(s_t)\right),$$

where $\hat{V}_K = \sum_{k=0}^{K} \gamma^k r_{t+k} + \gamma^{K+1} \hat{V}\left(s_{t+K+1}\right)$ and $K \in \{0,1,2,3,\ldots,T\}$ represents "K extra simulated steps required" for updating the value functions. However, any TD(K) method only considers a fixed value of $K \in \{0,1,2,3,\ldots,T\}$ when calculating the update target $\hat{V}_K$. TD(λ) (Sutton, 1988) proposed to consider a weighted average target as:

$$\hat{V}_\lambda = \frac{1}{1-\lambda^T} \sum_{k=0}^{K} (1-\lambda)\lambda^K \hat{V}_K,$$

where $\lambda \in [0,1]$. The above setting ensures that the recent values are weighted more than the future values and the weights sum to 1.

2.3 Deep Reinforcement Learning

In the above sections, we treat an agent's value functions as look-up tables such that for each state s or state-action combination (s,a) there is an entry which represents its corresponding value. However, in large or continuous environments, it is not possible to maintain a table for each state or state-action combination. Feature extraction and selection (Sutton, 1996; Keller et al., 2006; Parr et al., 2007) consider representing a high-dimension state s with a (combination of) state feature(s) $\phi(s)$ such that the latter dimension is much smaller than the former. State abstraction is another common technique to reduce the state dimension by grouping together similar states while not changing the underlying problem (Bertsekas and Castanon, 1989; Singh et al., 1999; Li et al., 2006). Though these two approaches are similar, one subtle difference is that some different states will have the same state representation in state abstraction but this is not necessarily true in feature extraction and selection.

The above methods, however, normally require domain knowledge or extra training process for obtaining state representations with lower dimensions than the original states. This makes the training of the RL agent economically or timely expensive. The revival of neural networks (NNs) (Schmidhuber, 2015) empowered by the significant improvement in graphics processing units (GPUs) computation speed significantly changed the situation. As powerful function approximators (Arulkumaran et al., 2017), deep NNs enabled end-to-end efficient and automatic feature engineering and state abstraction. Consequently, we saw successful applications of DRL works on game playing (Mnih et al., 2015; Silver et al., 2016, 2017, 2018), robotics (Levine et al., 2015; Lillicrap et al., 2015) and biology (Senior et al., 2020).

2.4 Policy Gradient Reinforcement Learning

The methods introduced in the previous section are classified as value-based RL methods because they rely on value functions for deriving the optimal policies. As our ultimate goal is the optimal decision-making strategy, we can directly solve for the optimal policy. Policy gradient (PG) reinforcement learning optimises a parameterised policy π_θ to maximise the cumulative returns by gradient descent:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R(\tau)] = \mathbb{E}_{\pi_\theta}\left[\sum_{a,s\in\tau} R(\tau)\nabla_\theta \log\left(\pi_\theta(a \mid s)\right)\right],$$

where $R(\tau)$ is the return of the interaction trajectory τ induced by the policy π_θ in the current environment. As we assume no knowledge of the model of the environment, we again use Monte-Carlo samples to replace the expected return $\mathbb{E}_{\pi_\theta}[R(\tau)]$. However, this gives us a gradient estimate with high variance. In addition, once the policy is updated, the corresponding trajectory data cannot be easily reused for future training. Therefore, PG methods normally have low data efficiency (Marbach and Tsitsiklis, 2003).

To reduce the variance and improve the data efficiency, REINFORCE (Williams,

1992) proposed to subtract the Monte Carlo return by a baseline $b(s)$ which is a function of the current state s but does not depend on the action a:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R(\tau)] = \mathbb{E}_{\pi_\theta}\left[\sum_{a,s\in\tau}(R(\tau)-b(s))\nabla_\theta \log\left(\pi_\theta(a \mid s)\right)\right].$$

Similarly, as in value-based RL, we can also replace the high variance Monte Carlo estimate with (estimates of) value functions (Sutton et al., 1999) and have:

$$\nabla_\theta \mathbb{E}_{\pi_\theta}[R(\tau)] = \mathbb{E}_{\pi_\theta}\left[\sum_{a,s\in\tau}(Q^\pi(s,a)-b(s))\nabla_\theta \log\left(\pi_\theta(a \mid s)\right)\right].$$

This type of methods is commonly known as ACTOR-CRITIC algorithms as the policy acts as an actor selecting actions and the value function acts as critic which evaluates how good the action selected by actor is.

Among many types of RL algorithms, PG is a natural candidate which closely matches the trial and error training scheme as it essentially increases the probability of selecting an action if it causes high returns and decreases the probability otherwise. In addition to its straightforward motivation, PG has advantages of: 1. learning stochastic policies; 2. better convergence property; and 3. learning in continuous state and action spaces. As the policy in PG is effectively a parameterised function, DNNs are a natural candidate. The advantages of applying DNNs into value-based RL algorithms are also maintained in PG methods.

2.5 Multi-Agent Markov Decision Process

We consider a set of agents, denoted by $\mathcal{N} = \{1,2,\ldots,N\}$, interacting with an environment by executing actions from a joint set $\mathcal{A} = \{\mathcal{A}^1,\ldots,\mathcal{A}^N\}$, with $\mathcal{A}^i$ denoting the action space of agent i, and N the total number of agents. To enable models that approximate real-world scenarios, we assume private and public information states. Private information states space, jointly (across agents) denoted by $\mathcal{X} = \{\mathcal{X}^1,\ldots,\mathcal{X}^N\}$ are a set of hidden information states where $\mathcal{X}^i$ is information set only observable by agent i, while public states in $o \in \mathcal{O}$ set are observed by all agents. We assume that hidden information states at each time step are sampled from a distribution $p_x : \mathcal{X} \to [0,1]$, while public states evolve from an initial distribution $p_o : \mathcal{O} \to [0,1]$, according to a stochastic transition model $\mathcal{T} : \mathcal{O} \times \mathcal{X} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \times \mathcal{O} \to [0,1]$. Having transitioned to a successor state according to $\mathcal{T}$, agent i receive rewards from $\mathcal{R}^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$, where we have used $\mathcal{S} = \mathcal{O} \times \mathcal{X}$ to denote joint state descriptions that incorporate both public and private information. Finally, rewards are discounted over time by a factor $\gamma \in (0,1]$. With this notation, our problem can be described succinctly by the tuple: $\langle \mathcal{N}, \mathcal{A}, \mathcal{O}, \mathcal{X}, \mathcal{T}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, p_x, p_o, \gamma \rangle$, which we refer to as an imperfect-information Markov decision process (I2MDP). To resolve issues resulting from partial observability, one could use the history of a game $h_t^i = \{o^i_{1:t}, x^i_{1:t}, a^i_{1:t-1}, a^{-i}_{1:t-1}\}$ so far from time step t as the input to a policy $\pi^i(a^i|h^i)$. The observable history h_t^i by agent i defined above is general as it contains all possible information agent i can observe and recall in a distributed setting. However, one could redefine the history h_t^i by only including parts of the information.

When agents have full observation of the environment, I2MDP recovers to the N-agent stochastic game (Shapley, 1953), which formally extends MDP to multiple agents. As stochastic game is a classical formulation of MARL problems, we introduce it formally as bellow.

Definition 2. *An N-agent stochastic game is defined as a tuple* $(\mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \{\mathcal{R}^i\}_{i\in\mathcal{N}}, p_0, \gamma)$*, where:*

- $\mathcal{N}$ *is a finite set of N agents;*
- $\mathcal{S}$ *is a finite set of states;*
- $\mathcal{A}^i$ *is the action space for agent* $i \in \mathcal{N}$*, and* $\mathcal{A} = \times \mathcal{A}^i$ *is the joint action space of all agents;*
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \Delta(\mathcal{S})$*, is a state transition function, a probability distribution over possible next states;*
- $\mathcal{R}^i : \mathcal{S} \times \mathcal{A}^i \to \mathbb{R}$ *is the reward function for agent* $i \in \mathcal{N}$*;*
- p_0 *specifies the probability distribution of the initial state* s_0*;*

- *and* $\gamma \in [0,1]$ *is a discount factor.*

Agent i chooses its action $a^i \in \mathcal{A}^i$ according to the policy $\pi^i(a^i|s)$ conditioning on some given state $s \in \mathcal{S}$. We denote the joint policy (the product of all agents' policies) as $\pi = \pi^1 \times \pi^2 \times \ldots \times \pi^N$ and agent i's opponents joint policy as $\pi^{-i}(a^{-i}|s)$, where a^{-i} is the joint action of all agents except agent i. As in a single-agent MDP, agent i's objective is to find the optimal policy which maximises its expected long term return defined as:

$$\max_{\pi^i} \eta(\pi^i) = \max_{\pi^i} \mathbb{E}_{\pi,\mathcal{T}}[R_t^i].$$

However, the difference is that the state transitions $\mathcal{T}$ and the expected return are both affected by the joint policy π^{-i} of i's opponents.

Chapter 3

Literature Review

As propose in Chapter 1, we focus on opponent modelling based approaches to solving *non-stationarity*, *partial observation* and *unclear learning objective* in MARL. To have a deeper insight into the current progress in related topics, we conduct literature review on these problems and opponent modelling methods in this chapter. Based on our observation of the above problems in different settings, we customise our solutions by combining opponent modelling with different techniques such as Bayesian inference, maximum entropy objective and multi-agent communication etc. Therefore, before we introducing our novel methods, review about related prior works is also provided in the remaining of this chapter.

3.1 Challenges in Multi-Agent Systems

By the nature of the game, we can classify multi-agent problems into *purely cooperative games*, *purely competitive games* and *mixed games*. In *purely cooperative games*, all agents either share the same reward functions $\mathcal{R}^i = \mathcal{R}\ \forall i \in \mathcal{N}$ or joint optimal policies $\arg\max_\pi \eta(\pi^i) = \arg\max_\pi \eta(\pi^j)\ \forall i, j \in \mathcal{N}$ and $i \neq j$. Therefore, agents are incentivised to communicate in some ways for sharing their private information or coordinate their actions during the training and execution. The term "execution" specifically refers to the period that the training agents' policies are fixed and expected to perform well without further training. In *purely competitive games*, agents have conflicting goals such that agent i's maximal return is obtained at the cost of all other agents' returns being minimised $\arg\max_\pi \eta(\pi^i) =$

$\arg\min_\pi \eta(\pi^j)\ \forall i, j \in \mathcal{N}$ and $i \neq j$. In contrast with *purely cooperative games*, players in *purely competitive games* are incentivised to exploit their opponents.

Games which are neither *purely cooperative* nor *purely competitive* are classified as *mixed games*. In these games, not all players have conflicting goals but they do not share the joint optimal policies either. Therefore, a coalition with punishment for cheating between players may be established to guarantee participants with satisfying but not optimal returns. Though each type of game has its own characteristics and assumptions, they all suffer from some common problems due to the existence of extra agents in the environment, among which *partial observation*, *non-stationarity* and *unclear learning objective* are three critical issues (Gronauer and Dieopold, 2021; Nguyen et al., 2018).

Partial observation is a common issue in many real-world decision-making problems where an agent can only have partial observation o_t of the full environment state s_t. As the agent's partial observation does not contain the full information of the current environment state, the environment perceived by the agent is no longer Markovian (Gronauer and Dieopold, 2021), which means the system's future dynamics is irrelevant to the history once the current state is given. The Markov property is critical as it guarantees that a current state s_t in a MDP problem is sufficient to make the optimal decision. To recover the property in partially observable problems, an agent has to maintain the entire history of observations. Therefore, a common approach to this issue is to equip agents with memory such that all history information observed by the agent is stored for future decision making (Hausknecht and Stone, 2015). Research works of applying recurrent neural networks (RNN) on partial observable problems have seen some successes (Omidshafiei et al., 2017; Foerster et al., 2017b; Dibangoye and Buffet, 2018; Gupta et al., 2017). In multi-agent settings, *partial observation* can arise from non-shared observations of agents. A typical example is the poker game where each agent's hand is only observable to itself and the union of all agents' hands (plus some other public information) forms the environment state (Mealing and Shapiro, 2017). As each player cannot observe other players' hands, *Partial observation* occurs. In this setting, when agents intend to cooperate or form a coalition, an alternative to memory-based approaches is for agents to communicate their own private information such that all agents will have the full environment state s_t before making any decisions (Goldman and Zilberstein, 2011).

Non-stationarity [1] is another major problem caused by the existence of multiple adaptive agents interacting in a shared environment (Hernandez-Leal et al., 2017). In contrast to the single agent setting, an agent's rewards and the environment transitions depend not only on the current environment state and the agent's action but also the actions of others. When the actions of others are not observed, the adaptation and learning of other agents therefore induces non-stationarity in the environment dynamics from the perspective of a single agent. Specifically, before and after its opponents update their policies, an agent may face differing transition dynamics and reward functions despite the environment remaining unchanged. This non-stationarity can pose serious problems for value function-based algorithms whose convergence usually relies on the assumption of a stationary environment (Hernandez-Leal et al., 2017; Laurent et al., 2011). Policy search algorithms also struggle in multi-agent settings due to the high variance of policy training induced by the changes in opponents' behaviours over time. Centralised training decentralised execution (CTDE) is the most common solution where information restriction is loosened so that each agent can observe more information to stabilise its training (Iqbal and Sha, 2019; Bono et al., 2019). For example, other agents joint actions a_t^{-i} may be revealed to the training agent i before the agent deciding its optimal action given the current state: $a_t^{i*} = \arg\max_{a_t^i} Q(s_t, a_t^i, a_t^{-i})$ (Lowe et al., 2017).

Unclear learning objective concerns how to define the optimal policy for a training agent in multi-agent problems. Recall that in single-agent case, an optimal policy is the one which maximises the expected long term return as:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi,\mathcal{T}}[R_t].$$

[1]Generally, *non-stationarity* refers to the environment's dynamics changes over time.

If we rewrite the above formula in multi-agent problems, we will have:

$$\begin{aligned}\pi^{i*} &= \arg\max_{\pi^i} \mathbb{E}_{\pi,\mathcal{T}}[R_t^i], \\ &= \arg\max_{\pi^i} \mathbb{E}_{\pi^i,\pi^{-i},\mathcal{T}}[R_t^i].\end{aligned}$$

The optimal policy π^{i*} defined above in MAS depends on the opponents' joint policy π^{-i}, so different π^{-i} will lead to different optimal policies π^{i*}. The changes in π^{-i} can be caused by (1). different opponents in different episodes; (2) updated policies by the same group of opponents. Therefore, training an agent with the above objective can either lead to poor performance during execution or non-converged policy. Hence, finding a reliable learning objective and evaluation method becomes non-trivial (Balduzzi et al., 2019; Czarnecki et al., 2020; Omidshafiei et al., 2019; Yang et al., 2020).

3.2 Opponent Modelling

Though there are other issues unsolved in MAS such as scalability (Zhang et al., 2019; Yang and Wang, 2020), we will focus on methods solving the 3 challenges introduced in the previous section. These challenges are closely related and share a common cause which is the lack of knowledge of opponents. Therefore, the capability of reasoning about other agents' belief, private information, behaviour, strategy and other characteristics is crucial. A reasoning model can be used in many different ways, but the most common case is where an agent utilises its reasoning model to aid it decision making (Brown, 1951; Heinrich and Silver, 2016; He et al., 2016; Raileanu et al., 2018). Normally, this reasoning is built on observation of modelled agents' past actions. In machine learning, we define the process of a modelling agent building a reasoning model to predict information relevant to modelled agents as opponent modelling.

There have been plenty of applications of opponent modelling across different AI problems which demonstrates the significant role of it in AI. For instance, intentions and plans of users need to be understood and predicted in dialogue systems

(Grosz and Sidner, 1986; Litman and Allen, 1984); intelligent tutoring systems can use opponent modelling to identify students' misconceptions (Iida et al., 1996; McCalla et al., 2000; Anderson et al., 1990); user experience can be improved by system with user models (McTear, 1993; Oh et al., 2011); autonomous vehicles need to predict and reason about behaviours of other vehicles or pedestrians (Buehler et al., 2009).

Opponent modelling has a long history and dates back to the beginning of game theory where opponent models were used in analysing equilibrium solutions of games. A typical example is *fictitious play* where an agent estimates its opponent's strategy from past experience and uses this estimation to choose its best response in the future (Brown, 1951). In early works, because adversarial games were the major focus of relevant research, the term *opponent modelling* became more well known. *Therefore, we use the word "opponents" when referring to other agents in an environment irrespective of the environment's cooperative or adversarial nature.* Equipped with the rich representation power of Neural Networks (NNs), recent works have obtained some progress in opponent modelling. It has been shown empirically that using opponents' features implicitly in one's decision process, parameterised by an NN, improves the performance of a trained agent (He et al., 2016). Also, opponent modelling has been used to handle non-stationary transitions caused by other learners in MARL (Foerster et al., 2017a). The ability of modelling different types of opponents is important in real human society, and some initial research has been conducted in (Rabinowitz et al., 2018).

Opponent modelling has been an active research topic and well documented (Rubin and Watson, 2011; Bakkes et al., 2012; Albrecht and Stone, 2018; Hernandez-Leal et al., 2017). Plenty research works in opponent modelling are conducted in different underlying assumptions and domains, attending various needs in different problems. We list some major factors below which greatly affect how an opponent model is built.

- **Modelled opponents' policies**: When an agent follows a deterministic policy, one can easily predict its action based on the observed trajectory so far if the same trajectory has been encountered in past experience. In this case, we could use deterministic structures such as decision trees and deterministic state automata for modelling opponents' behaviours (Albrecht and Stone, 2018). However, if an agent takes actions from a stochastic policy, it is more difficult to predict the next action it will take and requires more data to build an accurate model.

- **Learning process of the modelled agent**: Early works in opponent modelling often assume the modelled opponents policies are stationary (Hernandez-Leal et al., 2017). However, opponents in many real-world problems often learn and adapt their behaviours while interacting with others in the environment. Therefore, building opponents' models which assumes opponents' policies stationary can lead to poor performance.

- **Number of modelled agents in one environment**: If there is more than one opponent to be modelled in the same environment, the assumption about if those opponents are independent or dependent will greatly affect the approach of constructing models for each of them. Many existing opponent modelling methods assume modelled opponents are independent and thus build models of those agents independently (Wen et al., 2019). However, this is less often the case in real-world problems. When agents interact with each other, one has to take into account this dependency when modelling those opponents.

- **Environmental factors**: Some properties of the environment can also affect the underlying assumptions, such as if agents take turns to play or they make actions simultaneously; if the state space and action space are discrete or continuous respectively; and if the modelling agent has full observation about the modelled agent except the one that is to be predicted (Albrecht and Stone, 2018).

Policy prediction is the most common type in opponent modelling which trains a model to predict the probability distribution of actions modelled agent will take given some observed history in an episode. The predicted distribution is normally used as extra information for the modelling agent's decision making. The classical example of policy prediction is *fictitious play* (Brown, 1951) where agents model each other by a distribution over actions other agents may take. The probability of an action being chosen by a modelled agent is simply the empirical frequency of that action taken by the modelled agent in history. This simple approach has been proven to converge in matrix games (Fudenberg, 1998). However, it is difficult to have an accurate model when modelled agent strategy/policy becomes complex. To make a model more representative, one can use a conditional distribution to predict the modelled agent's policy. This distribution can be conditioned on information such as recent actions which have been taken by the modelled agent (Davison and Hirsh, 1998), recent states which have been visited by the modelled agent or other related features or abstractions of the modelled agent (Chakraborty and Stone, 2013). Tabular style model is not scalable and hard to generalise to unseen situations. Other model representations have been explored to solve these problems such as decision trees (Barrett et al., 2013) or neural networks (Davidson, 1999; Davidson et al., 2000; Anthony et al., 2017), where a model is trained to fit the observed data.

Private information prediction : Information greatly affecting an agent's decision making in multi-agent systems is often private; i.e., not observable by others. This motivates private information prediction modelling. Private information can be in the form of the modelled agent's type (e.g. if the agent is aggressive or defensive?), preference (e.g. if the agent prefers short term reward to long term ones?), goal (e.g. if the agent is heading north or south?) or observation of the world state exclusive to the modelled agent (e.g. the modelled agent's hand in poker games), etc (Albrecht and Stone, 2018).

Type based reasoning can help an agent to identify its opponent or partner's type in the early stage of a game. Therefore the agent can choose corresponding pre-trained policy to better exploit its opponent or cooperate with its partner. This approach normally assumes that there is a fixed number of types of opponents or partners an agent will encounter (Schillo et al., 2000). However this is hardly true in real cases (Rovatsos et al., 2003). However, if the true type of current modelled agent is in the set of considered type, modelling agent can often identify the modelled agent's type and adapt its behaviour to that type quickly. This approach started from game theory (Harsanyi, 1967), but it also has gained interest in the machine learning community. In (Lockett et al., 2007), a neural network is trained to predict a mixture of opponent's types; in (He et al., 2016), several expert networks corresponding to different agent types are used.

The preferences of a modelled agent is often in the form of a utility function or reward function. The motivation of modelling an agent's preferences is that assuming the modelled agent is rational (always tries to maximise its utility/reward), one can infer its action based on the modelled preferences. An advantage of this approach is that when an agent preference is known, one can infer that agent's action in the state even if the state is not explored before in history. In (Carmel and Markovitch, 1993, 1996b), a linear function of features is used to model an opponent's utility function in extensive form games. A human player's utilities are estimated by a linear function of social factors such as fairness and social welfare in (Gal et al., 2004).

Knowledge of an agent's goal can be useful when one interacts with that agent. For instance, a user interface can provide more relevant options and information when it better knows the user's intention (Oh et al., 2011; McTear, 1993) and an intrusion detection system can take countermeasures if it can recognise an agent's intention to attack (Geib and Goldman, 2001). Recently, Raileanu et al. (2018) uses an agent's policy parameterized by a neural network to infer the other agent's goal and update its belief in an online manner. Modelling of an opponent's private observation is also often studied in work on poker (Billings et al., 1998; Korb et al., 2013; Southey et al., 2012). Among these works, either some evaluations of how strong the opponent's hand is or the actual opponent's hand is modelled given the opponents' play.

Recursive reasoning: When an agent models its opponents, the opponents can also model the agent and take into account the fact that they are modelled by the agent when they build their models. This nesting of beliefs can lead to a possibly infinite reasoning process of the form "I believe that you believe that I believe …". The nested beliefs problem is proposed and addressed in game theory (Harsanyi, 1962, 1967) with an assumption that each agent's private information is sampled from a known distribution by all agents. However, this assumption is strong. Instead, to resolve this infinite recursion without the strong assumption, recursive reasoning assumes this belief nesting is down to a fixed recursion depth (Carmel and Markovitch, 1996a). The restriction in recursive reasoning can be further reduced by removing the assumption that each agent knows about other agents' recursion depth, which leads to the Interactive POMDP (I-POMDP) (Gmytrasiewicz and Doshi, 2005). In I-POMDP, an agent not only has belief about the environment state but also belief about models of other agents. One agent's different models mainly vary in the recursion depth. Many solutions to I-POMDP have been studied, including methods based on model equivalence (Rathnasabapathy et al., 2006), value iteration (Doshi and Perez, 2008), policy iteration (Sonu and Doshi, 2015), structural problem reduction (Hoang and Low, 2013) and deep reinforcement learning (Wen et al., 2019, 2020).

In this subsection, we provide a general and brief survey of opponent modelling methods based on our understanding and other related surveys (Rubin and Watson, 2011; Bakkes et al., 2012; Albrecht and Stone, 2018; Hernandez-Leal et al., 2017). We aim to give readers a general idea of opponent modelling, its applications, common assumptions and types. In the following chapters, we will show how to combine opponent modelling with different techniques so that we could solve the three challenges under different conditions. In the remaining part of this chapter, therefore, we will also review related works to techniques we used in the following chapters.

3.3 Control as Bayesian Inference

The Bayesian method is an important approach to decision making problems. It can capture the uncertainties regarding the transition probabilities, the reward functions in the environment or other agents' policies. This distributional information can be used to formulate a more structured exploration/exploitation strategy than those commonly used in classical RL, e.g. ε-greedy. The estimation of gradient performance with respect to parameters of value function and policy can also be done more accurately while using less data. Knowledge and the explicit formulation of domain assumptions can also be naturally encoded in Bayesian approaches as priors.

There have been a plethora of works applying Bayesian inference to solve issues such as partial observation (Furmston and Barber, 2010), data inefficiency (Abdolmaleki et al., 2018) and trade-offs between exploration and exploitation (O'Donoghue et al., 2017). Casting decision making and optimal control as an inference problem has a long history, which dates back to (Kalman, 1960) where Kalman smoothing is used to solve optimal control in linear dynamics with quadratic cost. A common approach in many works for framing RL as an inference problem is by introducing a binary random variable o which represents "optimality" (Toussaint and Storkey, 2006; Rawlik et al., 2013; Levine and Koltun, 2013; Abdolmaleki et al., 2018). However, the literature of Bayesian methods in MARL is limited. Among these are methods performing on cooperative games with prior knowledge on distributions of the game model and the possible strategies of others (Chalkiadakis and Boutilier, 2003) or policy parameters and possible roles of other agents (Wilson et al., 2010).

3.4 Maximum Entropy Objective in MARL

In many single-agent works, maximising entropy is part of a training agent's objective for resolving ambiguities in inverse reinforcement learning (Ziebart et al.), improving the diversity (Florensa et al., 2017), robustness (Fox et al., 2015) and the compositionality (Haarnoja et al., 2018a) of the learned policy. In works where VI is applied, it often presents in the evidence lower bound (ELBO) for the log likelihood of optimality (Haarnoja et al., 2017; Schulman et al., 2017; Haarnoja et al., 2018b), commonly known as maximum entropy objective (MEO), which encourages the optimal policy to maximise the expected return and long term entropy. The idea of preference over the distribution with maximum entropy originates from statistical modelling, and the reasoning behind it is that maximum entropy models have the least assumptions about the unknowns while still matching the observations.

In early works, the maximum entropy principle has been used in policy search in linear dynamics (Todorov, 2010; Toussaint, 2009; Levine and Koltun, 2013) and path integral control in general dynamics (Kappen', 2005; Theodorou et al., 2010). Recently, off-policy methods (Haarnoja et al., 2017; Schulman et al., 2017; Nachum et al., 2017) have been proposed to improve the sample efficiency in optimising MEO. However, in a continuous environment, the complex approximate inference may be needed for sampling actions. To avoid the complex sampling procedure, training a policy in supervised fashion is employed in (Haarnoja et al., 2018b). Variants of these off-policy methods (Wen et al., 2019; Wei et al., 2018; Grau-Moya et al., 2018) haven been applied for solving different problems in MARL. However, they are not derived from MARL problems but simple modifications of methods in single agent problems.

3.5 Multi-Agent Communication

In collaborative multi-agent systems, communication is essential for agents to learn to behave as a collective rather than a collection of individuals. This is particularly important in the imperfect-information setting, where private information becomes crucial to success. In such cases, efficient communication protocols between agents are needed for private information exchange, coordinated joint-action exploration, and true world-state inference. In typical multi-agent reinforcement learning (MARL) settings, designers incorporate explicit communication channels hoping to conceptually resemble language or verbal communication which are known to be important for human interaction (Baker et al., 1999). Though they can be used for facilitating collaboration in MARL, explicit communication channels come at additional computational and memory costs, making them difficult to deploy in decentralised control (Roth et al., 2006).

Environments where explicit communication is difficult or prohibited are common. These settings can be synthetic such as those in games, e.g., bridge and Hanabi, but also frequently appear in real-world tasks such as autonomous driving and autonomous fleet control. In these situations, humans rely upon implicit communication as a means of information exchange (Rasouli et al., 2017) and are effective in learning to infer the implicit meaning behind others' actions (Heider and Simmel, 1944). The ability to perform such inference requires the attribution of a mental state and reasoning mechanism to others. This ability is known as theory of mind (Premack and Woodruff, 1978). In this work, we develop agents that benefit from considering others' perspectives and thereby explore the further development of machine theory of mind (Rabinowitz et al., 2018).

Recently, there has been a surge of interest in using reinforcement learning (RL) approaches to learn communication protocols (Foerster et al., 2016; Lazaridou et al., 2016; Mordatch and Abbeel, 2017; Sukhbaatar et al., 2016). Among these works, Mordatch and Abbeel (2017) observe the emergence of non-verbal communication in collaborative environments without an explicit communication channel, where agents are exclusively either a sender or a receiver. Similar research is also conducted in (de Weerd et al., 2015). We will show in our setting in Chapter 5, we do not restrict agents to be exclusively a sender or a receiver of communications – agents can communicate mutually by actions. Knepper et al. (2017) propose a framework for implicit communication in a cooperative setting and show that various problems can be mapped into this framework.

Dragan et al. (2013) consider how to train agents to exhibit legible behaviour (i.e. behaviour from which it is easy to infer the intention). Their approach is dependent on a hand-crafted cost function to attain informative behaviour. Mutual information has been used as a means to promote coordination without the need for a human engineered cost function. Strouse et al. (2018) use a mutual information objective to encourage an agent to reveal or hide its intention. In a related work, Jaques et al. (2019) utilise a mutual information objective to imbue agents with social influence. While the objective of maximal mutual information in actions can yield highly effective collaborating agents, a mutual information objective in itself is insufficient to necessitate the development of implicit communication by actions. Eccles et al. (2019) introduce a reciprocity reward as an alternative approach to solve social dilemmas.

3.6 Exploitability and Exploitation

In single agent reinforcement learning (SARL), an agent learns to act by iteratively interacting with an environment. In such a setting, an agent's learning objective and its performance evaluation are normally clear and straightforward, e.g., its long-term cumulative rewards gained from the environment. However, in multi-agent reinforcement learning (MARL), one agent's performance greatly depends on the behaviour of other agents. Hence, finding a reliable learning objective and evaluation method become non-trivial (Balduzzi et al., 2019; Czarnecki et al., 2020; Omidshafiei et al., 2019; Yang et al., 2020). Naive solutions of the problem using SARL generalise badly (Lanctot et al., 2017) and optimising the joint policy of all agents does not scale. Recent approaches combining game theoretical analysis with deep RL have seen some success in large zero-sum games (Berner et al., 2019; Vinyals et al., 2019).

Game theory offers a mathematical framework to model strategic interactions among players (Morgenstern and Von Neumann, 1953). Under perfect rationality (Fudenberg and Tirole, 1991), a central solution concept is Nash equilibrium (NE) where no player benefits from deviating from their equilibrium strategy. In a two-player zero-sum game without any inherent advantage for either player (e.g. as a first mover), an NE is a safe strategy to play (i.e., playing not to lose) – NE guarantees a tie in the worst case in expectation.

It is well known that finding an NE is PPAD-hard even in two-player games (Chen and Deng, 2006). An exception are two-player zero-sum games where the NE can be tractably solved by a linear program (LP) in polynomial time (van den Brand, 2020). However, in games with extremely large action spaces, approximate NE solutions, such as fictitious play (FP) (Brown, 1951) and counterfactual regret minimisation (CFR) (Zinkevich et al., 2007b), have to be used. An important design principle that underpins NE approximation is the iterative best-response dynamics. Two representative methods are Double Oracle (DO) (McMahan et al., 2003) and Policy Space Response Oracle (PSRO) (Lanctot et al., 2017). In the dynamics of DO (McMahan et al., 2003), players are initialised with restricted strategy sets; then at each iteration, a NE will be computed over the current restricted sets. These sets will be expanded by adding the best-response strategy to the NE computed over the full strategy sets. The iterative process continues until the best response is in the restricted strategy pool. PSRO approximates DO by interleaving empirical game-theoretic analysis (EGTA) with deep RL. In contrast with DO, the game with restricted strategy sets has to be estimated through simulation. Furthermore, the exact analytical best-response oracle is replaced in PSRO by a deep RL oracle which calculates an approximate best response. PSRO is a general self-play framework for MARL and many approaches built upon it have been proposed to improve its performance (McAleer et al., 2020; Muller et al., 2020; Nieves et al., 2021; Smith et al., 2021).

However, NE is not the most profitable strategy in many cases. In complex competitive games, such as poker, it is common that agents encounter opponents with bounded rationality, in the sense that they may at best play an approximate Nash equilibrium strategy and often play dominated actions (Billings et al., 2003; Ponsen et al., 2014). Therefore, playing a NE can potentially forego significant rewards against sub-optimal opponents. This incentivizes players to deviate from the NE and exploit their opponents' weakness (i.e., playing to win). However, the resulting strategy could render itself exploitable should it overfit to the current opponent. Playing to win can therefore lead to exploitation by other opponent strategies. In the case of deceptive opponents such exploitation is known as the "get taught and exploited" problem (Sandholm, 2007).

To better balance the trade-off between playing to win against the current opponents (exploitation) and not losing to unknown opponents (exploitability), Johanson et al. (2008) proposed a solution concept, named Restricted Nash Response (RNR). RNR and its variants (Johanson and Bowling, 2009; Johanson et al., 2008; Ponsen et al., 2014; Bard et al., 2013) assume stationary opponents, i.e., the strategies they learn to exploit are unknown but fixed. However, in many real-world applications, opponents may adapt and change their strategies on an ongoing basis. For example, in Rock-Paper-Scissors when a player learns to best respond by playing Rock to an opponent's strategy which always plays Scissors, the opponent may then learn to best respond to your best response by playing Paper. Furthermore, prior RNR approaches only provide one-off solutions in the sense that whenever we need to re-adjust the trade-off between exploitation and exploitability or the opponent uses a new fixed policy, we need to re-solve the updated game from scratch.

3.7 Dirichlet Process Opponent Modelling

A fundamental ability of an effective AI agent is the capacity to interact with other intelligent agents. Therefore, the capability of reasoning about other agents is crucial. Classical solutions to resolve the issue of non-stationarity include centralised training (Lowe et al., 2017), self-play (Vinyals et al., 2019), meta-learning (Al-Shedivat et al., 2017) and opponent modelling (Albrecht and Stone, 2018). When specifically applied to the issue of non-stationarity, most previous works focusing on opponent modelling which switches between different opponent models when a change in opponent(s) is detected. A switch of model may be triggered by a drop in opponent model prediction accuracy (Everett and Roberts, 2018) or when performance in terms of reward received for a fixed policy drops (Hernandez-Leal and Kaisers, 2017). Deep BPR+ (Zheng et al., 2018) combines a measure of opponent model accuracy and reward tracking to decide when to learn a new policy. Significantly, most of these works limit the opponents' non-stationarity to periodically changing their policies within a finite pre-defined set of stationary policies.

In this work, we consider non-stationarity during the training stage arising from the opponents' concurrent learning dynamics, rather than drawing stationary opponents from a pre-defined set. The entire lifetime of an opponent can generally be modelled as a mixture of an unknown (possibly infinite) number of policies. This motivates the usage of a Dirichlet Process (DP) mixture model (Blei et al., 2006;

Teh, 2010) which can infer the number of mixture components from data and provide incremental model capacity on demand. Various approximate inference methods are reported for DP mixture models, such as Markov chain Monte Carlo (Ishwaran and James, 2001) and variational inference (Blei et al., 2006; Hughes and Sudderth, 2014; Wang and Blei, 2012). However, these inference methods either do not adapt to an online setting or truncate the number of clusters to a finite value. Recently proposed streaming inference algorithms (Lin, 2013; Tank et al., 2015) enable the DP mixture model to solve online non-stationary problems in a truly non-parametric way. Applications have been reported in task-free continual learning (Lee et al., 2020) and model-based reinforcement learning (Xu et al., 2020). In this work, we adopt this approach to model and simulate a non-stationary opponent for MARL.

Chapter 4

Non-Stationarity in Cooperative Games

One challenge listed in the previous section is the *non-stationarity* in Multi-agent systems where we have adaptive opponents learning and updating their policies as our agent does. This is generally a difficult problem, but simplification can be obtained if we consider cooperative problems. Though we often will not know exactly how the opponents will update their policies, we can reasonably assume that they update their policies towards maximising the same long-term return shared by all agents in a cooperative game. This observation can help us to build a more accurate opponent model of adaptive opponents and alleviate the *non-stationarity* problem.

In this chapter, we consider applying Bayesian inference to incorporate the observation as a prior when we learn our opponent model. We redefine the binary random variable o in a multi-agent setting and formalise multi-agent reinforcement learning (MARL) as probabilistic inference. We derive a variational lower bound of the likelihood of achieving optimality and name it as Regularised Opponent Model with Maximum Entropy Objective (ROMMEO). From ROMMEO, we present a novel perspective on opponent modelling and show how it can improve the performance of training agents theoretically and empirically in cooperative games. To optimise ROMMEO, we first introduce a tabular Q-iteration method ROMMEO-Q with proof of convergence. Then, we extend the exact algorithm to complex environments by proposing an approximate version, ROMMEO-AC. We evaluate these two algorithms on the challenging iterated matrix game and differential game respectively and show that they can outperform strong MARL baselines.

4.1 A Variational Lower Bound

In this work, we consider fully cooperative games where different agents have the same reward function $\mathcal{R}^i(s,a^i,a^{-i}) = \mathcal{R}^{-i}(s,a^i,a^{-i}), \forall i \in 1,\ldots,n$. Our approach can also be extended to the case where agents do not share the same reward function but the same optimal policy, i.e. $\pi^{i*} = \pi^{-i*}$. However, for simplifying the notation, we derive our method in the former setting. Therefore, each agent's objective is to maximise the shared expected return:

$$\max \quad \eta^i(\pi) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^t \mathcal{R}(s_t, a_t^i, a_t^{-i})\right], \ \forall i \in 1,\ldots,n, \tag{4.1}$$

where (a_t^i, a_t^{-i}) is sampled from $\pi = \pi^i \times \pi^{-i}$. In addition, we assume that the solution to the above optimisation objective is unique.

We transform the control problem into an inference problem by introducing a binary random variable o_t^i which serves as the indicator for "optimality" for each agent i at each time step t. In single agent problem, reward $\mathcal{R}(s_t,a_t)$ is bounded, but the achievement of the maximum reward given the action a_t is unknown. Therefore, in the single-agent case, o_t indicates the optimality of achieving the bounded maximum reward r_t^*. It thus can be regarded as a random variable and we have $P(o_t = 1|s_t,a_t) \propto \exp(\mathcal{R}(s_t,a_t))$. Intuitively, this formulation dictates that higher rewards reflect a higher likelihood of achieving optimality, i.e., the case when $o_t = 1$.

In cooperative multi-agent reinforcement learning (CMARL), a single agent's "optimality" o_t^i cannot imply that it obtains the maximum reward because the reward depends on the joint actions of all agents (a^i, a^{-i}). In CMARL, to define agent i's optimality o_t^i, we first introduce the definition of optimum and optimal policy:

Definition 3. *In cooperative multi-agent reinforcement learning, optimum is a strategy profile* $(\pi^{1*},\ldots,\pi^{n*})$ *such that no other strategy profiles can obtain higher re-*

turn:

$$\begin{aligned}&\mathbb{E}_{s\sim p_s, a_t^{i*}\sim\pi^{i*}, a_t^{-i*}\sim\pi^{-i*}}\left[\sum_{t=1}^{\infty}\gamma^t R(s_t, a_t^{i*}, a_t^{-i*})\right]\\ &\geq \mathbb{E}_{s\sim p_s, a_t^{i}\sim\pi^{i}, a_t^{-i}\sim\pi^{-i}}\left[\sum_{t=1}^{\infty}\gamma^t R(s_t, a_t^{i}, a_t^{-i})\right] \forall\pi\in\Pi,\end{aligned} \tag{4.2}$$

where $\pi = \pi^i \times \pi^{-i}$ and Agent i's optimal policy is π^{i*}.

An individual agent cannot obtain the maximum reward alone without coordination of other agents in CMARL, but it can individually play its optimal policy alone. Therefore, we define $o_t^i = 1$ only indicating that *agent i's policy at time step t is optimal.* This definition of "optimality" is subtly different from the one in the single-agent case, but given other players actions a_t^{-i}, the posterior probability of agent i's optimality is still proportional to its exponential reward:

$$P(o_t^i = 1 | s_t, a_t^i, a_t^{-i}) \propto \exp(\mathcal{R}(s_t, a_t^i, a_t^{-i})). \tag{4.3}$$

This assumption is valid because given (s_t, a_t^i, a_t^{-i}), a higher reward can indicate a higher probability that agent i's current policy is optimal. Furthermore, with this new definition, we have that the posterior probability of agent i's optimality given its action a_t^i is the probability that the action is sampled from the optimal policy:

$$P(o_t^i | a_t^i) = P(a_t^i \sim \pi^{i*} | a_t^i) \propto \pi^{i*}(a_t^i). \tag{4.4}$$

For cooperative games, if all agents play optimally, then agents can receive the maximum return, which is the optimum of the game. Given other agents playing their optimal policies $o_{1:T}^{-i} = 1$, the probability that agent i also plays its optimal policy $P(o_{1:T}^i = 1 | o_{1:T}^{-i} = 1)$ is the probability of obtaining the maximum return from agent i's perspective. Therefore, agent i maximising $P(o_{1:T}^i = 1 | o_{1:T}^{-i} = 1)$ is equivalent to agent i's objective defined in Equation 4.1 and we define this new objective as:

$$\max \ \mathcal{J} \triangleq \log P(o_{1:T}^i = 1 | o_{1:T}^{-i} = 1). \tag{4.5}$$

Maximising the above probability conditioning on the assumption that opponents are playing optimally can help us to take into account opponents' adaptive behaviours when we learn our opponent model. This benefit will become more explicit in our following derivation.

As the optimality variables o_t^i indicate whether player i is playing optimally at step t, we assume it is affected by the current state s_t, player's action a_t^i and also opponents' actions a_t^{-i}. Therefore, by the Law of Total Probability, we can expand Equation 4.5 as:

$$\begin{aligned}\log P(o_{1:T}^i|o_{1:T}^{-i}) &= \log \sum_{a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}} P(o_{1:T}^i, a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}|o_{1:T}^{-i}), \\ &= \log \sum_{a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}} P(o_{1:T}^i|a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}, o_{1:T}^{-i}) P(a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}|o_{1:T}^{-i}).\end{aligned} \tag{4.6}$$

Note we can factorise $P(a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}|o_{1:T}^{-i})$ as :

$$P(a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}|o_{1:T}^{-i}) = P(s_1)\prod_t P(s_{t+1}|s_t, a_t)P(a_t^i|a_t^{-i}, s_t, o_t^{-i})P(a_t^{-i}|s_t, o_t^{-i}), \tag{4.7}$$

where $P(s_1)$ is the initial state distribution and $P(s_{t+1}|s_t, a_t)$ is the state transition function. $P(a_t^i|a_t^{-i}, s_t, o_t^{-i})$ is the conditional policy of agent i when other agents $-i$ achieve optimality. In our model we do not presume how an action a_t^i being affected by $(a_t^{-i}, s_t, o_t^{-i})$, so we set $P(a_t^i|a_t^{-i}, s_t, o_t^{-i}) \propto 1$ which is effectively a uniform distribution assuming minimum prior knowledge. Because we have no knowledge of the optimal policies and the model of the environment, we treat them as latent variables. To optimise the observed evidence defined in Equation 4.5, we use variational inference (VI) with an auxiliary distribution over these latent variables $q(a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}|o_{1:T}^i = 1, o_{1:T}^{-i} = 1)$. Without loss of generality, we here derive the solution for agent i.

We factorise $q(a_{1:T}^i, a_{1:T}^{-i}, s_{1:T}|o_{1:T}^i = 1, o_{1:T}^{-i} = 1)$ so as to capture agent i's conditional policy on the current state and opponents actions, and beliefs regarding opponents actions. This way, agent i will learn optimal policy, while also possessing the capability to model opponents actions a^{-i}. Using all modelling assumptions, we may factorise $q(a^i_{1:T}, a^{-i}_{1:T}, s_{1:T} | o^i_{1:T} = 1, o^{-i}_{1:T} = 1)$ as:

$$\begin{aligned}
&q(a^i_{1:T}, a^{-i}_{1:T}, s_{1:T} | o^i_{1:T} = 1, o^{-i}_{1:T} = 1) \\
&= P(s_1) \prod_t P(s_{t+1} | s_t, a_t) q(a^i_t | a^{-i}_t, s_t, o^i_t = o^{-i}_t = 1) \\
&\times q(a^{-i}_t | s_t, o^i_t = o^{-i}_t = 1) \\
&= P(s_1) \prod_t P(s_{t+1} | s_t, a_t) \pi(a^i_t | s_t, a^{-i}_t) \rho(a^{-i}_t | s_t),
\end{aligned}$$

where we have assumed the same initial and states transitions as in the original model. With the above factorisation, we derive a lower bound on the likelihood of optimality of agent i:

$$\begin{aligned}
&\log P(o^i_{1:T} = 1 | o^{-i}_{1:T} = 1) \\
&\geq \mathcal{J}(\pi, \rho) \triangleq \sum_t \mathbb{E}_{(s_t, a^i_t, a^{-i}_t) \sim q} [R(s_t, a^i_t, a^{-i}_t) \\
&+ H(\pi(a^i_t | s_t, a^{-i}_t)) - D_{\mathrm{KL}}(\rho(a^{-i}_t | s_t) || P(a^{-i}_t | s_t))] \qquad (4.8) \\
&= \sum_t \mathbb{E}_{s_t} [\underbrace{\mathbb{E}_{a^i_t \sim \pi, a^{-i}_t \sim \rho} [R(s_t, a^i_t, a^{-i}_t) + H(\pi(a^i_t | s_t, a^{-i}_t))]}_{\text{MEO}} \\
&\underbrace{- \mathbb{E}_{a^{-i}_t \sim \rho} [D_{\mathrm{KL}}(\rho(a^{-i}_t | s_t) || P(a^{-i}_t | s_t))]}_{\text{Regularizer of } \rho}]. \qquad (4.9)
\end{aligned}$$

Written out in full, $\rho(a^{-i}_t | s_t, o^{-i}_t = 1)$ is agent i's opponent model estimating *optimal* policies of its opponents, $\pi(a^i_t | s_t, a^{-i}_t, o^i_t = 1, o^{-i}_t = 1)$ is the agent i's conditional policy at optimum ($o^i_t = o^{-i}_t = 1$) and $P(a^{-i}_t | s_t, o^{-i}_t = 1)$ is the prior of optimal policy of opponents. In our work, we set the prior $P(a^{-i}_t | s_t, o^{-i}_t = 1)$ equal to the observed empirical distribution of opponents' actions given states. As we are only interested in the case where ($o^i_t = 1, o^{-i}_t = 1$), we drop them in π, ρ and $P(a^{-i}_t | s_t)$ here and thereafter. $H(\cdot)$ is the entropy function.

4.2 The Learning of Opponent Model

Our work provides a natural perspective on opponent modelling in coordination problems: biasing one's opponent model towards the optimum from its perspective but regularising it with the empirical distribution of opponent's real behaviour. It is closely related to a series of recent works focusing on maximum entropy objective (MEO) and ROMMEO is an extension of MEO to MARL.

4.2.1 Regularised Opponent Model with Maximum Entropy Objective

For our work, We can further expand Equation 4.8 into Equation 4.9 and we find that it resembles the MEO introduced above. We denote agent i's expectation of reward $\mathcal{R}(s_t, a_t^i, a_t^{-i})$ plus entropy of the conditional policy $H(\pi(a^i|s, a^-i))$ as agent i's *maximum entropy objective (MEO)*. In the multi-agent version, however, it is worthy of noting that optimising the MEO will lead to *the optimisation of* ρ. This can be counter-intuitive at first sight as opponent behaviour models are normally trained with only past state-action data (s, a^{-i}) to predict opponents' actions. However, recall that $\rho(a_t^{-i}|s_t, o_t^{-i} = 1)$ is modelling opponents' *optimal* policies in our work. Given agent i's policy π^i being fixed, optimising MEO with respect to ρ updates agent i's opponent model in the direction of the higher shared reward $\mathcal{R}(s, a^i, a^{-i})$ and the more stochastic conditional policy $\pi^i(a^i|s, a^{-i})$, making it closer to the real optimal policies of the opponents.

Without any regularisation, at iteration d, agent i can freely learn a new opponent model ρ_{d+1}^i which is the closest to the optimal opponent policies π^{-i*} from its perspective given $\pi_d^i(a^i|s, a^{-i})$. Next, agent i can optimise the lower bound with respect to $\pi_{d+1}^i(a^i|s, a^{-i})$ given ρ_{d+1}^i. Then we have an EM-like iterative training and can show it monotonically increases the probability that the opponent model ρ is optimal policies of the opponents. Then, by acting optimally to the converged opponent model $\rho^{i\infty}$, we can recover agent i's optimal policy π^{i*}.

However, it is unrealistic to learn such an opponent model. As the real opponents have no access to agent i's conditional policy $\pi_d^i(a^i|s, a^{-i})$, the learning of its policy can be different from the one of agent i's opponent model. Then the actual opponent policies π_{d+1}^{-i} can be very different from agent i's converged opponent model $\rho^{i\infty}$ learned in the above way given agent i's conditional policy $\pi_d^i(a^i|s,a^{-i})$. Therefore, acting optimally to an opponent model far from the real opponents' policies can lead to poor performance.

The last term in Equation 4.9 can prevent agent i building an unrealistic opponent model. The Kullback-Leibler (KL) divergence between opponent model and a prior $D_{\mathrm{KL}}(\rho(a_t^{-i}|s_t)||P(a_t^{-i}|s_t))$ can act as a regulariser of ρ. By setting the prior to the empirical distribution of opponent past behaviour, the KL divergence penalises ρ heavily if it deviates from the empirical distribution too much. As the objective in Equation 4.9 can be seen as a *Maximum Entropy objective* for one agent's policy and opponent model with *regularisation on the opponent model*, we call this objective *Regularised Opponent Model with Maximum Entropy Objective (ROMMEO).*

4.3 Multi-Agent Soft Actor Critic

To optimise ROMMEO in Equation 4.9 derived in the previous section, we propose two off-policy algorithms. We first introduce an exact tabular Q-iteration method with proof of convergence. For practical implementation in a complex continuous environment, we then propose the ROMMEO actor critic ROMMEO-AC, which is an approximation to this procedure.

4.3.1 Regularised Opponent Model with Maximum Entropy Objective Q-Iteration

In this section, we derive a multi-agent version of Soft Q-iteration algorithm proposed in (Haarnoja et al., 2017) and we name our algorithm as ROMMEO-Q. The derivation follows from a similar logic to (Haarnoja et al., 2017), but the extension of Soft Q-learning to MARL is still nontrivial. From this section, we slightly modify the objective in Equation 4.9 by adding a weighting factor α for the entropy term and the original objective can be recovered by setting $\alpha = 1$.

We first define multi-agent soft Q-function and V-function respectively. Then we can show that the conditional policy and opponent model defined in Equation 4.12 and 4.13 below are optimal solutions with respect to the objective defined in Equation 4.9:

Theorem 2. *We define the soft state-action value function of agent i as*

$$Q^{\pi^*,\rho^*}_{soft}(s_t,a^i_t,a^{-i}_t) = r_t + \mathbb{E}_{(s_{t+l},a^i_{t+l},a^{-i}_{t+l},\ldots)\sim q}[\sum_{l=1}^{\infty}\gamma^l(r_{t+l} + \alpha H(\pi^*(a^i_{t+l}|a^{-i}_{t+l},s_{t+l})) - D_{\mathrm{KL}}(\rho^*(a^{-i}_{t+l}|s_{t+l})||P(a^{-i}_{t+l}|s_{t+l}))], \tag{4.10}$$

and soft state value function as

$$V^*(s) = \log\sum_{a^{-i}}P(a^{-i}|s)\left(\sum_{a^i}\exp(\frac{1}{\alpha}Q^*_{soft}(s,a^i,a^{-i}))\right)^{\alpha}, \tag{4.11}$$

Then the optimal conditional policy and opponent model for Equation 4.8 are

$$\pi^*(a^i|s,a^{-i}) = \frac{\exp(\frac{1}{\alpha}Q^{\pi^*,\rho^*}_{soft}(s,a^i,a^{-i}))}{\sum_{a^i}\exp(\frac{1}{\alpha}Q^{\pi^*,\rho^*}_{soft}(s,a^i,a^{-i}))}, \tag{4.12}$$

and

$$\rho^*(a^{-i}|s) = \frac{P(a^{-i}|s)\left(\sum_{a^i}\exp(\frac{1}{\alpha}Q^*_{soft}(s,a^i,a^{-i}))\right)^{\alpha}}{\exp(V^*(s))}. \tag{4.13}$$

Proof. We first show that the objective in Equation 4.9 can be rewritten in terms of the soft-Q functions. We define the soft state-action value function $Q^{\pi,\rho}_{soft}(s,a,a^{-i})$

of agent i in a stochastic game as:

$$
\begin{aligned}
&Q^{\pi,\rho}_{soft}(s_t, a^i_t, a^{-i}_t) \\
&= r_t + \mathbb{E}_{(s_{t+l}, a^i_{t+l}, a^{-i}_{t+l}, \ldots)\sim q}[\sum_{l=1}^{\infty} \gamma^l (r_{t+l} + \alpha H(\pi(a^i_{t+l}|a^{-i}_{t+l}, s_{t+l})) \\
&- D_{\mathrm{KL}}(\rho(a^{-i}_{t+l}|s_{t+l})||P(a^{-i}_{t+l}|s_{t+l}))] \\
&= \mathbb{E}_{(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1})}[r_t + \gamma(\alpha H(\pi(a^i_{t+1}|s_{t+1}, a^{-i}_{t+1})) \\
&- D_{\mathrm{KL}}(\rho(a^{-i}|s_{t+1})||P(a^{-i}|s_{t+1})) + Q^{\pi,\rho}_{soft}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1}))] \\
&= \mathbb{E}_{(s_{t+1}, a^{-i}_{t+1})}[r_t + \gamma(\alpha H(\pi(\cdot|s_{t+1}, a^{-i}_{t+1})) - D_{\mathrm{KL}}(\rho(a^{-i}|s_{t+1})||P(a^{-i}|s_{t+1})) \\
&+ \mathbb{E}_{a^i_{t+1}\sim\pi}[Q^{\pi,\rho}_{soft}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1})])] \\
&= \mathbb{E}_{(s_{t+1})}[r_t + \gamma(\mathbb{E}_{a^{-i}_{t+1}\sim\rho, a^i_{t+1}\sim\pi}[\alpha H(\pi(a^i_{t+1}|s_{t+1}, a^{-i}_{t+1}))] \\
&- D_{\mathrm{KL}}(\rho(\cdot|s_{t+1})||P(\cdot|s_{t+1}))] + \mathbb{E}_{a^{-i}_{t+1}\sim\rho, a^i_{t+1}\sim\pi}[Q^{\pi,\rho}_{soft}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1})])], \qquad (4.14)
\end{aligned}
$$

Then we can easily see that the objective in Equation 4.9 can be rewritten as:

$$
\begin{aligned}
\mathcal{J}(\pi, \phi) = \mathbb{E}_{(s_t, a^i_t, a^{-i}_t)\sim(p_s, \pi, \rho)}&[Q^{\pi,\rho}_{soft}(s_t, a^i_t, a^{-i}_t) + \alpha H(\pi(a^i_t|s_t, a^{-i}_t)) \\
&- D_{\mathrm{KL}}(\rho(a^{-i}_t|s_t)||P(a^{-i}_t|s_t))], \qquad (4.15)
\end{aligned}
$$

by setting $\alpha = 1$.

Now, we introduce two extra theorem before completing the proof of Theorem 2.

Theorem 3. *(Policy improvement theorem) Given a conditional policy π and opponent model ρ, define a new conditional policy $\tilde{\pi}$ as*

$$
\tilde{\pi}(\cdot|s, a^{-i}) \propto \exp(\frac{1}{\alpha} Q^{\pi,\rho}_{soft}(s, \cdot, a^{-i})), \forall s, a^{-i}. \qquad (4.16)
$$

Assume that throughout our computation, Q is bounded and $\sum_{a^i} Q(s, a^i, a^{-i})$ is bounded for any s and a^{-i} (for both π and $\tilde{\pi}$). Then $Q^{\tilde{\pi},\rho}_{soft}(s, a^i, a^{-i}) \geq Q^{\pi,\rho}_{soft}(s, a^i, a^{-i}) \forall s, a$.

Theorem 4. *(Opponent model improvement theorem) Given a conditional policy π*

and opponent model ρ, define a new opponent model $\tilde{\rho}$ as

$$\tilde{\rho}(\cdot|s) \propto \exp(\sum_{a^i} Q^{\pi,\rho}_{soft}(s,a^i,\cdot)\pi(a^i|\cdot,s) + \alpha H(\pi(s)) + \log P(\cdot|s)), \forall s, a^i. \tag{4.17}$$

Assume that throughout our computation, Q is bounded and

$$\sum_{a^{-i}} \exp(\sum_{a^i} Q(s,a^i,a^{-i})\pi(a^i|s,a^{-i}))$$

is bounded for any s and a^i (for both ρ and $\tilde{\rho}$). Then $Q^{\pi,\tilde{\rho}}_{soft}(s,a^i,a^{-i}) \geq Q^{\pi,\rho}_{soft}(s,a^i,a^{-i}) \forall s, a$.

The proof of Theorem 3 and 4 is based on two observations that:

$$\begin{aligned}&\alpha H(\pi(\cdot|s,a^{-i})) + \mathbb{E}_{a^i\sim\pi}[Q^{\pi,\rho}_{soft}(s,a^i,a^{-i})] \\ &\qquad \leq \alpha H(\tilde{\pi}(\cdot|s,a^{-i})) + \mathbb{E}_{a^i\sim\tilde{\pi}}[Q^{\pi,\rho}_{soft}(s,a^i,a^{-i})],\end{aligned} \tag{4.18}$$

and

$$\begin{aligned}&\mathbb{E}_{a^{-i}_{t+1}\sim\rho, a^i_{t+1}\sim\pi}[\alpha H(\pi(a^i_{t+1}|s_{t+1},a^{-i}_{t+1}))] - D_{\mathrm{KL}}(\rho(\cdot|s_{t+1})||P(\cdot|s_{t+1}))] \\ &+ \mathbb{E}_{a^{-i}_{t+1}\sim\rho, a^i_{t+1}\sim\pi}[Q^{\pi,\rho}_{soft}(s_{t+1},a^i_{t+1},a^{-i}_{t+1})] \\ &\leq \mathbb{E}_{a^{-i}_{t+1}\sim\tilde{\rho}, a^i_{t+1}\sim\pi}[\alpha H(\pi(a^i_{t+1}|s_{t+1},a^{-i}_{t+1}))] - D_{\mathrm{KL}}(\tilde{\rho}(a^{-i}_{t+1}|s_{t+1})||P(\cdot|s_{t+1})) \\ &+ \mathbb{E}_{a^{-i}_{t+1}\sim\tilde{\rho}, a^i_{t+1}\sim\pi}[Q^{\pi,\rho}_{soft}(s_{t+1},a^i_{t+1},a^{-i}_{t+1})].\end{aligned} \tag{4.19}$$

First, we notice that

$$\begin{aligned}&\alpha H(\pi(\cdot|s,a^{-i})) + \mathbb{E}_{a^i\sim\pi}[Q^{\pi,\rho}_{soft}(s,a^i,a^{-i})] \\ &= -\alpha D_{\mathrm{KL}}(\pi(\cdot|s,a^{-i})||\tilde{\pi}(\cdot|s,a^{-i})) + \alpha \log \sum_{a^i} \exp(\frac{1}{\alpha} Q^{\pi,\rho}_{soft}(s,a^i,a^{-i})).\end{aligned} \tag{4.20}$$

Therefore, the LHS is only maximised if the KL-Divergence on the RHS is minimised. This KL-Divergence is minimised only when $\pi = \tilde{\pi}$, which proves the Equation 4.18.

Similarly, we can have

$$\mathbb{E}_{a^{-i}\sim\rho, a^i\sim\pi}[\alpha H(\pi(a^i|s,a^{-i}))] - D_{\mathrm{KL}}(\rho(\cdot|s)||P(\cdot|s))]) + \mathbb{E}_{a^{-i}\sim\rho, a^i\sim\pi}[Q^{\pi,\rho}_{soft}(s,a^i,a^{-i})]$$
$$= -D_{\mathrm{KL}}(\rho(\cdot|s)||\tilde{\rho}(\cdot|s)) + \log\sum_{a^{-i}}\exp(\sum_{a^i} Q^{\pi,\rho}(s,a^i,a^{-i})\pi(a^i|s,a^{-i})$$
$$+ \alpha H(\pi(\cdot|s,a^i)) + \log P(a^{-i|s})), \tag{4.21}$$

which proves the Equation 4.19.

With the above observations, the proof of Theorem 3 and 4 is completed by as follows:

$$
\begin{aligned}
&Q^{\pi,\rho}_{soft}(s_t,a^i_t,a^{-i}_t)\\
&=\mathbb{E}_{(s_{t+1},a^i_{t+1},a^{-i}_{t+1})}[r_t+\gamma(\alpha H(\pi(a^i_{t+1}|s_{t+1},a^{-i}_{t+1}))\\
&-D_{\mathrm{KL}}(\rho(a^{-i}_{t+1}|s_{t+1})||P(a^{-i}_{t+1}|s_{t+1}))+Q^{\pi,\rho}_{soft}(s_{t+1},a^i_{t+1},a^{-i}_{t+1}))]\\
&=\mathbb{E}_{(s_{t+1},a^{-i}_{t+1})}[r_t+\gamma(\alpha H(\pi(\cdot|s_{t+1},a^{-i}_{t+1}))-D_{\mathrm{KL}}(\rho(a^{-i}_{t+1}|s_{t+1})||P(a^{-i}_{t+1}|s_{t+1}))\\
&+\mathbb{E}_{a^i_{t+1}\sim\pi}[Q^{\pi,\rho}_{soft}(s_{t+1},a^i_{t+1},a^{-i}_{t+1})])]\\
&\le\mathbb{E}_{(s_{t+1},a^{-i}_{t+1})}[r_t+\gamma(\alpha H(\tilde{\pi}(\cdot|s_{t+1},a^{-i}_{t+1}))-D_{\mathrm{KL}}(\rho(a^{-i}_{t+1}|s_{t+1})||P(a^{-i}_{t+1}|s_{t+1}))\\
&+\mathbb{E}_{a^i_{t+1}\sim\tilde{\pi}}[Q^{\pi,\rho}_{soft}(s_{t+1},a^i_{t+1},a^{-i}_{t+1})])]\\
&=\mathbb{E}_{(s_{t+1})}[r_t+\gamma(\mathbb{E}_{a^{-i}_{t+1}\sim\rho,a^i_{t+1}\sim\pi}[\alpha H(\tilde{\pi}(a^i_{t+1}|s_{t+1},a^{-i}_{t+1}))]\\
&-D_{\mathrm{KL}}(\rho(\cdot|s_{t+1})||P(\cdot|s_{t+1}))+\mathbb{E}_{a^{-i}_{t+1}\sim\rho,a^i_{t+1}\sim\pi}[Q^{\pi,\rho}_{soft}(s_{t+1},a^i_{t+1},a^{-i}_{t+1})])]\\
&\le\mathbb{E}_{(s_{t+1})}[r_t+\gamma(\mathbb{E}_{a^{-i}_{t+1}\sim\tilde{\rho},a^i_{t+1}\sim\pi}[\alpha H(\tilde{\pi}(a^i_{t+1}|s_{t+1},a^{-i}_{t+1}))]\\
&-D_{\mathrm{KL}}(\tilde{\rho}(\cdot|s_{t+1})||P(\cdot|s_{t+1}))+\mathbb{E}_{a^{-i}_{t+1}\sim\tilde{\rho},a^i_{t+1}\sim\pi}[Q^{\pi,\rho}_{soft}(s_{t+1},a^i_{t+1},a^{-i}_{t+1})])]\\
&=\mathbb{E}_{(s_{t+1},a^i_{t+1},a^{-i}_{t+1})\sim\tilde{q}}[r_t+\gamma(\alpha H(\tilde{\pi}(a^i_{t+1}|s_{t+1},a^{-i}_{t+1}))\\
&-D_{\mathrm{KL}}(\tilde{\rho}(a^{-i}|s_{t+1})||P(a^{-i}|s_{t+1}))+r_{t+1})+\gamma^2\mathbb{E}_{(s_{t+2},a^{-i}_{t+2})}[\alpha H(\pi(\cdot|s_{t+2},a^{-i}_{t+2}))\\
&-D_{\mathrm{KL}}(\rho(a^{-i}_{t+2}|s_{t+2})||P(a^{-i}_{t+2}|s_{t+2}))\\
&+\mathbb{E}_{a^i_{t+2}\sim\pi}[Q^{\pi,\rho}_{soft}(s_{t+2},a^i_{t+2},a^{-i}_{t+2})]]]\\
&\le\mathbb{E}_{(s_{t+1},a^i_{t+1},a^{-i}_{t+1})}[r_t+\gamma(\alpha H(\tilde{\pi}(a^i_{t+1}|s_{t+1},a^{-i}_{t+1}))\\
&-D_{\mathrm{KL}}(\tilde{\rho}(a^{-i}|s_{t+1})||P(a^{-i}|s_{t+1}))+r_{t+1})+\gamma^2\mathbb{E}_{(s_{t+2},a^{-i}_{t+2})}[\alpha H(\pi(\cdot|s_{t+2},a^{-i}_{t+2}))\\
&-D_{\mathrm{KL}}(\rho(a^{-i}_{t+2}|s_{t+2})||P(a^{-i}_{t+2}|s_{t+2}))\\
&+\mathbb{E}_{a^i_{t+2}\sim\tilde{\pi}}[Q^{\pi,\rho}_{soft}(s_{t+2},a^i_{t+2},a^{-i}_{t+2})]]]\\
&\vdots\ \text{(repeating the above process for the next time steps)}\\
&\le r_t+\mathbb{E}_{(s_{t+l},a^i_{t+l},a^{-i}_{t+l},\ldots)\sim\tilde{q}}[\sum_{l=1}^{\infty}\gamma^l(r_{t+l}+\alpha H(\tilde{\pi}(a^i_{t+l}|a^{-i}_{t+l},s_{t+l}))\qquad(4.22)\\
&-D_{\mathrm{KL}}(\tilde{\rho}(a^{-i}_{t+l}|s_{t+l})||P(a^{-i}_{t+l}|s_{t+l}))]\\
&=Q^{\tilde{\pi},\tilde{\rho}}_{soft}(s_t,a^i_t,a^{-i}_t).
\end{aligned}
$$

With Theorem 3 and 4 and the above inequalities, we can see that, if we start from an arbitrary conditional policy π_0 and an arbitrary opponent model ρ_0 and we iterate between policy improvement as

$$\pi_{t+1}(\cdot|s,a^{-i}) \propto \exp(\frac{1}{\alpha}Q^{\pi_t,\rho_t}_{soft}(s,\cdot,a^{-i})), \tag{4.23}$$

and opponent model improvement as

$$\rho_{t+1}(\cdot|s) \propto \exp(\sum_{a^i} Q^{\pi_{t+1},\rho_t}_{soft}(s,a^i,\cdot)\pi_{t+1}(a^i|\cdot,s) + \alpha H(\pi_{t+1}(s)) + \log P(\cdot|s)), \tag{4.24}$$

then $Q^{\pi_t,\rho_t}_{soft}(s,a^i,a^{-i})$ can be shown to increase monotonically and π_t, ρ_t will converge at $\pi_t = \pi_\infty$, $\rho_t = \rho_\infty$ respectively. Similar to (Haarnoja et al., 2017), we can show that any non-optimal policy and opponent model can be improved this way and the algorithm converges when we cannot improve $Q^{\pi_t,\rho_t}_{soft}(s,a^i,a^{-i})$ anymore. At convergence, the equality in Equation 4.22 establishes and Theorem 2 is proved. $\square$

Following from Theorem 2, we can find the optimal solution of Equation 4.9 by learning the soft multi-agent Q-function first and recover the optimal policy π^* and opponent model ρ^* by Equations 4.12 and 4.13. To learn the Q-function, we show that it satisfies a Bellman-like equation, which we name it as multi-agent soft Bellman equation:

Theorem 5. *We define the soft multi-agent Bellman equation for the soft state-action value function $Q^{\pi,\rho}_{soft}(s,a^i,a^{-i})$ of agent i as*

$$Q^{\pi^*,\rho^*}_{soft}(s,a^i,a^{-i}) = r_t + \gamma \mathbb{E}_{(s_{t+1})}[V^*_{soft}(s_{t+1})]. \tag{4.25}$$

With this Bellman equation defined above, we can derive a solution to Equation 4.25 with a fixed point iteration, which we call ROMMEO Q-iteration (ROMMEO-Q). Additionally, We can show that it can converge to the optimal Q^*_{soft} and V^*_{soft} with certain restrictions as stated in (Wen et al., 2019):

Theorem 6. *ROMMEO Q-iteration. In a symmetric game with only one global optimum, i.e.* $\mathbb{E}_{\pi^*}\left[Q_t^i(s)\right] \geq \mathbb{E}_{\pi}\left[Q_t^i(s)\right]$, *where* π^* *is the optimal strategy profile. Let* $Q_{soft}(\cdot,\cdot,\cdot)$ *and* $V_{soft}(\cdot)$ *be bounded and assume*

$$\sum_{a^{-i}} P(a^{-i}|s)\left(\sum_{a^i}\exp(\frac{1}{\alpha}Q^*_{soft}(s,a^i,a^{-i}))\right)^{\alpha} < \infty \tag{4.26}$$

and that $Q^*_{soft} < \infty$ *exists. Then the fixed-point iteration*

$$Q_{soft}(s_t,a_t^i,a_t^{-i}) \leftarrow r_t + \gamma\mathbb{E}_{(s_{t+1})}[V_{soft}(s_{t+1})], \tag{4.27}$$

where $V_{soft}(s_t) \leftarrow \log\sum_{a_t^{-i}} P(a_t^{-i}|s_t)\times\left(\sum_{a_t^i}\exp(\frac{1}{\alpha}Q_{soft}(s_t,a_t^i,a_t^{-i}))\right)^{\alpha}$ $\forall s_t,a_t^i,a_t^{-i}$, *converges to* Q^*_{soft} *and* V^*_{soft} *respectively.*

Proof. As we show above, when the training converges, we have:

$$\pi^*(a^i|s,a^{-i}) = \frac{\frac{1}{\alpha}\exp(Q^*(s,a^i,a^{-i}))}{\sum_{a^i}\exp(\frac{1}{\alpha}Q^*(s,a^i,a^{-i}))}, \tag{4.28}$$

and

$$\begin{aligned}\rho^*(a^{-i}|s) &= \frac{\exp(\sum_{a^i}Q^*(s,a^i,a^{-i})\pi^*(a^i|s,a^{-i}) + \alpha H(\pi^*(a^i|s,a^{-i})) + \log P(a^{-i}|s))}{\sum_{a^{-i}}\exp(\sum_{a^i}Q^*(s,a^i,a^{-i})\pi^*(a^i|s,a^{-i}) + \alpha H(\pi^*(a^i|s,a^{-i})) + \log P(a^{-i}|s))}\\ &= \frac{P(a^{-i}|s)\left(\sum_{a^i}\exp(Q^*_{soft}(s,a^i,a^{-i}))\right)^{\alpha}}{\exp(V^*(s))},\end{aligned} \tag{4.29}$$

where the equality in Equation 4.29 comes from substituting π^* with Equation 4.28 and we define the soft sate value function $V^{\pi,\rho}_{soft}(s)$ of agent i as:

$$V^{\pi,\rho}_{soft}(s_t) = \log\sum_{a_t^{-i}}P(a_t^{-i}|s_t)\left(\sum_{a_t^i}\exp\left(\frac{1}{\alpha}Q^{\pi,\rho}_{soft}(s_t,a_t^i,a_t^{-i})\right)\right)^{\alpha}. \tag{4.30}$$

Then we can show that

$$\begin{aligned}
&Q_{soft}^{\pi^*,\rho^*}(s,a^i,a^{-i}) \\
&= r_t + \gamma \mathbb{E}_{s'\sim p_s}[(\mathbb{E}_{a_{t+1}^{-i}\sim\rho,a_{t+1}^{i}\sim\pi}[\alpha H(\pi(a_{t+1}^i|s_{t+1},a_{t+1}^{-i}))] \\
&- D_{\mathrm{KL}}(\rho(\cdot|s_{t+1})||P(\cdot|s_{t+1}))]\mathbb{E}_{a_{t+1}^{-i}\sim\rho,a_{t+1}^{i}\sim\pi}[Q_{soft}^{\pi,\rho}(s_{t+1},a_{t+1}^i,a_{t+1}^{-i})])] \\
&= r_t + \gamma \mathbb{E}_{s'\sim p_s}[V^*(s')]. \qquad (4.31)
\end{aligned}$$

We define the soft value iteration operator $\mathcal{T}$ as:

$$\begin{aligned}
&\mathcal{T}Q(s,a^i,a^{-i}) = R(s,a^i,a^{-i}) \\
&+ \gamma \mathbb{E}_{s'\sim p_s}\left[\log \sum_{a^{-i\prime}} P(a^{-i\prime}|s') \left(\sum_{a^{i\prime}} \exp\left(\frac{1}{\alpha}Q(s',a^{i\prime},a^{-i\prime})\right)\right)^{\alpha}\right]. \qquad (4.32)
\end{aligned}$$

In a symmetric fully cooperative game with only one global optimum, we can show as done in (Wen et al., 2019), the operator defined above is a contraction mapping. We define a norm on Q-values $\|Q_1^i - Q_2^i\| \triangleq \max_{s,a^i,a^{-i}} \left|Q_1^i\left(s,a^i,a^{-i}\right) - Q_2^i\left(s,a^i,a^{-i}\right)\right|$. Let $\varepsilon = \|Q_1^i - Q_2^i\|$, then we have:

$$\begin{aligned}
&\log \sum_{a^{-i\prime}} P(a^{-i\prime}|s') \left(\sum_{a^{i\prime}} \exp\left(\frac{1}{\alpha}Q_1(s',a^{i\prime},a^{-i\prime})\right)\right)^{\alpha} \\
&\leq \log \sum_{a^{-i\prime}} P(a^{-i\prime}|s') \left(\sum_{a^{i\prime}} \exp\left(\frac{1}{\alpha}Q_2(s',a^{i\prime},a^{-i\prime}) + \varepsilon\right)\right)^{\alpha} \\
&= \log \sum_{a^{-i\prime}} P(a^{-i\prime}|s') \left(\sum_{a^{i\prime}} \exp\left(\frac{1}{\alpha}Q_2(s',a^{i\prime},a^{-i\prime})\right)\exp(\varepsilon)\right)^{\alpha} \\
&= \log \sum_{a^{-i\prime}} P(a^{-i\prime}|s') \exp(\varepsilon)^{\alpha} \left(\sum_{a^{i\prime}} \exp\left(\frac{1}{\alpha}Q_2(s',a^{i\prime},a^{-i\prime})\right)\right)^{\alpha} \\
&= \alpha\varepsilon + \log \sum_{a^{-i\prime}} P(a^{-i\prime}|s') \left(\sum_{a^{i\prime}} \exp\left(\frac{1}{\alpha}Q_2(s',a^{i\prime},a^{-i\prime})\right)\right)^{\alpha}. \qquad (4.33)
\end{aligned}$$

Similarly,

$$\begin{aligned}&\log \sum_{a^{-i\prime}} P(a^{-i\prime}|s') \left(\sum_{a^{i\prime}} \exp\left(\frac{1}{\alpha} Q_1(s', a^{i\prime}, a^{-i\prime})\right)\right)^{\alpha} \\ &\geq -\alpha\varepsilon + \log \sum_{a^{-i\prime}} P(a^{-i\prime}|s') \left(\sum_{a^{i\prime}} \exp\left(\frac{1}{\alpha} Q_2(s', a^{i\prime}, a^{-i\prime})\right)\right)^{\alpha}. \qquad (4.34)\end{aligned}$$

Therefore $\|\mathcal{T}Q_1^i - \mathcal{T}Q_2^i\| \leq \gamma\varepsilon = \gamma\|Q_1^i - Q_2^i\|$, where $\alpha = 1$. □

4.3.2 Regularised Opponent Model with Maximum Entropy Objective Actor Critic

ROMMEO-Q assumes we have the model of the environment and is impractical to implement in high-dimensional continuous problems. To solve these problems, we propose the ROMMEO actor critic (ROMMEO-AC) which is a model-free method. We use neural networks (NNs) as function approximators for the conditional policy, opponent model and Q-function and learn parameters of these functions by stochastic gradient descent. We parameterize the Q-function, conditional policy and opponent model by $Q_\omega(s, a^i, a^{-i})$, $\pi_\theta(a_t^i|s_t, a_t^{-i})$ and $\rho_\phi(a_t^{-i}|s_t)$ respectively.

Without access to the environment model, we first replace the Q-iteration with Q-learning. Therefore, we can train ω to minimise:

$$\begin{aligned}\mathcal{J}_Q(\omega) = \mathbb{E}_{(s_t, a_t^i, a_t^{-i})\sim\mathcal{D}}[\frac{1}{2}(Q_\omega(s_t, a_t^i, a_t^{-i}) \\ - R(s_t, a_t^i, a_t^{-i}) - \gamma\mathbb{E}_{s_{t+1}\sim p_s}[\bar{V}(s_{t+1})])^2], \qquad (4.35)\end{aligned}$$

with

$$\begin{aligned}\bar{V}(s_{t+1}) = Q_{\bar{\omega}}(s_{t+1}, a_{t+1}^i, \hat{a}_{t+1}^{-i}) - \log\rho_\phi(\hat{a}_{t+1}^{-i}|s_{t+1}) \\ - \alpha\log\pi_\theta(a_{t+1}^i|s_{t+1}, \hat{a}_{t+1}^{-i}) + \log P(\hat{a}_{t+1}^{-i}|s_{t+1}), \qquad (4.36)\end{aligned}$$

where $Q_{\bar{\omega}}$ are target functions for providing relatively stable target values. We use $\hat{a}_t^{-i}$ denoting the action sampled from agent i's opponent model $\rho(a_t^{-i}|s_t)$ and it should be distinguished from a_t^{-i} which is the real action taken by agent i's opponent. Equation 4.36 can be derived from Equation 4.12 and 4.13.

To recover the optimal conditional policy and opponent model and avoid intractable inference steps defined in Equation 4.12 and 4.13 in complex problems, we follow the method in (Haarnoja et al., 2018b) where θ and ϕ are trained to minimise the KL-divergence:

$$\begin{aligned}\mathcal{J}_\pi(\theta) = \mathbb{E}_{s_t \sim D, a_t^{-i} \sim \rho} \\ \left[D_{\mathrm{KL}} \left(\pi_\theta(\cdot | s_t, \hat{a}_t^{-i}) \middle\| \frac{\exp(\frac{1}{\alpha} Q_\omega(s_t, \cdot, \hat{a}_t^{-i}))}{Z_\omega(s_t, \hat{a}_t^{-i})} \right) \right],\end{aligned} \tag{4.37}$$

$$\begin{aligned}\mathcal{J}_\rho(\phi) = \mathbb{E}_{(s_t, a_t^i) \sim D} \\ \left[D_{\mathrm{KL}} \left(\rho(\cdot | s_t) \middle\| \frac{P(\cdot | s_t) \left(\frac{\exp(\frac{1}{\alpha} Q(s_t, a_t^i, \cdot))}{\pi_\theta(a_t^i | s_t, \cdot)} \right)^\alpha}{Z_\omega(s_t)} \right) \right].\end{aligned} \tag{4.38}$$

By using the reparameterization trick: $\hat{a}_t^{-i} = g_\phi(\varepsilon_t^{-i}; s_t)$ and $a_t^i = f_\theta(\varepsilon_t^i; s_t, \hat{a}_t^{-i})$, we can rewrite the objectives above as:

$$\begin{aligned}\mathcal{J}_\pi(\theta) = \mathbb{E}_{s_t \sim D, \varepsilon_t^i \sim N, \hat{a}_t^{-i} \sim \rho} [\alpha \log \pi_\theta(f_\theta(\varepsilon_t^i; s_t, \hat{a}_t^{-i})) \\ - Q_\omega(s_t, f_\theta(\varepsilon_t^i; s_t, \hat{a}_t^{-i}), \hat{a}_t^{-i})],\end{aligned} \tag{4.39}$$

$$\begin{aligned}\mathcal{J}_\rho(\phi) = \mathbb{E}_{(s_t, a_t) \sim D, \varepsilon_t^{-i} \sim N} [\log \rho_\phi(g_\phi(\varepsilon_t^{-i}; s_t) | s_t) \\ - \log P(\hat{a}_t^{-i} | s_t) - Q(s_t, a_t^i, g_\phi(\varepsilon_t^{-i}; s_t)) \\ + \alpha \log \pi_\theta(a_t^i | s_t, g_\phi(\varepsilon_t^{-i}; s_t))].\end{aligned} \tag{4.40}$$

The gradient of Equation 4.35, 4.39 and 4.40 with respect to the corresponding

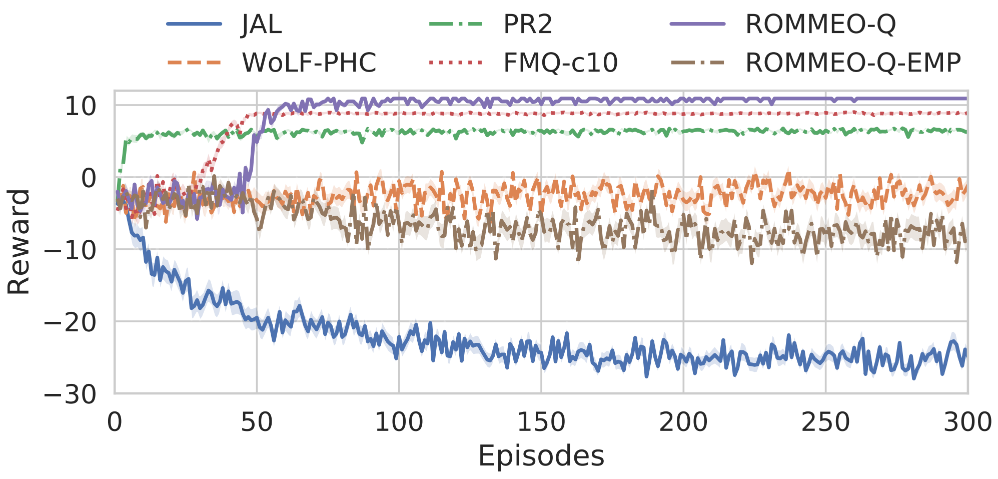


Figure 4.1: Learning curves of ROMMEO and baselines on ICG over 100 episodes.

parameters are listed as below:

$$\nabla_\omega \mathcal{J}_Q(\omega) = \nabla_\omega Q_\omega(s_t, a_t^i, a_t^{-i})(Q_\omega(s_t, a_t^i, a_t^{-i}) \\ - R(s_t, a_t^i, a_t^{-i}) - \gamma \bar{V}(s_{t+1})), \tag{4.41}$$

$$\nabla_\theta \mathcal{J}_\pi(\theta) = \nabla_\theta \alpha \log \pi_\theta(a_t^i | s_t, \hat{a}_t^{-i}) + \nabla_\theta f_\theta(\varepsilon_t^i; s_t, \hat{a}_t^{-i}) \\ (\nabla_{a_t^i} \alpha \log \pi_\theta(a_t^i | s_t, \hat{a}_t^{-i}) - \nabla_{a_t^i} Q_\omega(s_t, a_t^i, \hat{a}_t^{-i})), \tag{4.42}$$

$$\nabla_\phi \mathcal{J}_\rho(\phi) = \nabla_\phi \log \rho_\phi(\hat{a}_t^{-i} | s_t) + (\nabla_{\hat{a}_t^{-i}} \log \rho_{\phi^i}(\hat{a}_t^{-i} | s_t) \\ - \nabla_{\hat{a}_t^{-i}} \log P(\hat{a}_t^{-i} | s_t) - \nabla_{\hat{a}_t^{-i}} Q_{\omega^i}(s_t, a_t^i, \hat{a}_t^{-i}) \\ + \nabla_{\hat{a}_t^{-i}} \alpha \log \pi_\theta(a^i | s_t, \hat{a}_t^{-i})) \nabla_\phi g_\phi(\varepsilon_t^{-i}; s_t). \tag{4.43}$$

To have an overview of the proposed methods, we list the pseudo-code of ROMMEO-Q and ROMMEO-AC as follows:

Algorithm 1 Multi-agent Soft Q-learning

```
**Result:** policy $\pi^i$, opponent model $\rho^i$
**Initialisation**:
Initialise replay buffer $\mathcal{M}$ to capacity $M$.
Initialise $Q_{\omega^i}(s, a^i, a^{-i})$ with random parameters $\omega^i$, $P(a^{-i}|s)$ arbitrarily, set $\gamma$ as the discount factor.
Initialise target $Q_{\bar{\omega}^i}(s, a^i, a^{-i})$ with random parameters $\bar{\omega}^i$, set $C$ the target parameters update interval.
**while** not converge **do**
    **Collect experience**
    For the current state $s_t$ compute the opponent model $\rho^i(a_t^{-i}|s_t)$ and conditional policy $\pi^i(a_t^i|s_t, a_t^{-i})$ respectively from:
        $$\rho^i(a_t^{-i}|s_t) \propto P(a_t^{-i}|s_t)\left(\sum_{a_t^i} \exp(\frac{1}{\alpha} Q_{\omega^i}(s_t, a_t^i, a_t^{-i}))\right)^{\alpha},$$
        $$\pi^i(a_t^i|s_t, \hat{a}_t^{-i}) \propto \exp(\frac{1}{\alpha} Q_{\omega^i}(s_t, a_t^i, \hat{a}_t^{-i})).$$
    Compute the marginal policy $\pi^i(a_t^i|s_t)$ and sample an action from it:
        $$a_t^i \sim \pi^i(a_t^i|s_t) = \sum_{a^{-i}} \pi^i(a_t^i|s_t, a_t^{-i}) \rho(a_t^{-i}|s_t).$$
    Observe next state $s_{t+1}$, opponent action $a_t^{-i}$ and reward $r_t^i$, save the new experience in the reply buffer:
        $$\mathcal{M} \leftarrow \mathcal{M} \cup \{(s_t, a_t^i, a_t^{-i}, s_{t+1}, r_t^i)\}.$$
    Update the prior from the replay buffer:
        $$P(a_t^{-i}|s_t) = \frac{\sum_{m=1}^{|\mathcal{M}|} \mathbb{I}(s = s_t, a^{-i} = a_t^{-i})}{\sum_{m=1}^{|\mathcal{M}|} \mathbb{I}(s = s_t)} \forall s_t, a_t^{-i} \in \mathcal{M}.$$
    **Sample a mini-batch from the replay buffer**:
        $$\{s_t^{(n)}, a_t^{i,(n)}, a_t^{-i,(n)}, s_{t+1}^{(n)}, r_t^{(n)}\}_{n=1}^N \sim \mathcal{M}.$$
    **Update** $Q_{\omega^i}(s, a^i, a^{-i})$:
    **for** each tuple $(s_t^{(n)}, a_t^{i,(n)}, a_t^{-i,(n)}, s_{t+1}^{(n)}, r_t^{(n)})$ **do**
        Sample $\{a^{-i,(n,k)}\}_{k=1}^K \sim \rho$, $\{a^{i,(n,k)}\}_{k=1}^K \sim \pi$.
        Compute empirical $\bar{V}^i(s_{t+1}^{(n)})$ as:
            $$\bar{V}^i(s_{t+1}^{(n)}) = \log\left(\frac{1}{K}\sum_{k=1}^K \frac{\left(P^{\frac{1}{\alpha}}(a^{-i,(n,k)}|s_{t+1}^{(n)}) \exp(\frac{1}{\alpha} Q_{\bar{\omega}^i}(s_{t+1}^{(n)}, a^{i,(n,k)}, a^{-i,(n,k)}))\right)^{\alpha}}{\pi(a^{i,(n,k)}|s_{t+1}^{(n)}, a^{-i,(n,k)}) \rho(a^{-i,(n,k)}|s_{t+1}^{(n)})}\right).$$
        Set
            $$y^{(n)} = \begin{cases} r_t^{(n)} & \text{for terminal } s_{t+1}^{(n)} \\ r_t^{(n)} + \gamma \bar{V}^i(s_{t+1}^{(n)}) & \text{for non-terminal } s_{t+1}^{(n)} \end{cases}$$
        Perform gradient descent step on $(y^{(n)} - Q_{\omega^i}(s_{t+1}^{(n)}, a^{i,(n)}, a^{-i,(n)}))^2$ with respect to parameters $\omega^i$
        Every $C$ gradient descent steps, reset target parameters:
            $$\bar{\omega}^i \leftarrow \omega$$
    **end for**
**end while**
**Compute converged $\pi^i$ and $\rho^i$**
```

Algorithm 2 Multi-agent Variational Actor Critic

```
Result: policy $\pi_{\theta^i}$, opponent model $\rho_{\phi^i}$
Initialisation:
Initialise parameters $\theta^i$, $\phi^i$, $\omega^i$, $\psi^i$ for each agent $i$ and the random process $\mathcal{N}$ for action exploration.
Assign target parameters of joint action Q-function: $\bar{\omega}^i \leftarrow \omega$.
Initialise learning rates $\lambda_V, \lambda_Q, \lambda_\pi, \lambda\phi, \alpha$, and set $\gamma$ as the discount factor.
for Each episode $d = (1, \dots, D)$ do
    Initialise random process $\mathcal{N}$ for action exploration.
    for each time step $t$ do
        For the current state $s_t$, sample an action and opponent's action using:
        $\hat{a}_t^{-i} \leftarrow g_{\phi^{-i}}(\varepsilon^{-i}; s_t)$, where $\varepsilon_t^{-i} \sim \mathcal{N}$,
        $a_t^i \leftarrow f_{\theta^i}(\varepsilon^i; s_t, \hat{a}_t^{-i})$, where $\varepsilon_t^i \sim \mathcal{N}$.
        Observe next state $s_{t+1}$, opponent action $a_t^{-i}$ and reward $r_t^i$, save the new experience in the replay buffer:
            $$\mathcal{D}^i \leftarrow \mathcal{D}^i \cup \{(s_t, a_t^i, a_t^{-i}, \hat{a}_t^{-i}, s_{t+1}, r_t^i)\}.$$
        Update the prior from the replay buffer:
            $$\psi^i = \arg\max \mathbb{E}_{\mathcal{D}^i}[-P(a^{-i}|s) \log P_{\psi^i}(a^{-i}|s)].$$
        Sample a mini-batch from the reply buffer:
            $$\{s_t^{(n)}, a_t^{i,(n)}, a_t^{-i,(n)}, \hat{a}_t^{-i,(n)}, s_{t+1}^{(n)}, r_t^{(n)}\}_{n=1}^N \sim \mathcal{M}.$$
        For the state $s_{t+1}^{(n)}$, sample an action and opponent's action using:
        $\hat{a}_{t+1}^{-i,(n)} \leftarrow g_{\phi^{-i}}(\varepsilon^{-i}; s_{t+1}^{(n)})$, where $\varepsilon_{t+1}^{-i} \sim \mathcal{N}$,
        $a_{t+1}^{i,(n)} \leftarrow f_{\bar{\theta}^i}(\varepsilon^i; s_{t+1}^{(n)}, \hat{a}_{t+1}^{-i,(n)})$, where $\varepsilon_{t+1}^i \sim \mathcal{N}$.
        $\bar{V}^i(s_{t+1}^{(n)}) = Q_{\bar{\omega}}(s_{t+1}^{(n)}, a_{t+1}^{i,(n)}, \hat{a}_{t+1}^{-i,(n)}) - \alpha \log \pi_{\theta^i}(a_{t+1}^{i,(n)}|s_{t+1}^{(n)}, \hat{a}_{t+1}^{-i,(n)}) - \log \rho_{\phi^i}(\hat{a}_{t+1}^{-i,(n)}|s_{t+1}^{(n)}) + \log P_{\psi^i}(\hat{a}_{t+1}^{-i,(n)}|s_{t+1}^{(n)})$.
        Set
            $$y^{(n)} = \begin{cases} r_t^{(n)} & \text{for terminal } s_{t+1}^{(n)} \\ r_t^{(n)} + \gamma \bar{V}^i(s_{t+1}^{(n)}) & \text{for non-terminal } s_{t+1}^{(n)} \end{cases}$$
            $$\nabla_{\omega^i} \mathcal{J}_Q(\omega^i) = \nabla_{\omega^i} Q_{\omega^i}(s_t^{(n)}, a_t^{i,(n)}, a_t^{-i,(n)})(Q_{\omega^i}(s_t^{(n)}, a_t^{i,(n)}, a_t^{-i,(n)}) - y^{(n)})$$
            $$\nabla_{\theta^i} \mathcal{J}_\pi(\theta^i) = \nabla_{\theta^i} \alpha \log \pi_{\theta^i}(a_t^{i,(n)}|s_t^{(n)}, \hat{a}_t^{-i,(n)})$$
            $$+ (\nabla_{a_t^{i,(n)}} \alpha \log \pi_{\theta^i}(a_t^{i,(n)}|s_t^{(n)}, \hat{a}_t^{-i,(n)}) - \nabla_{a_t^{i,(n)}} Q_\omega(s_t^{(n)}, a_t^{i,(n)}, \hat{a}_t^{-i,(n)})) \nabla_\theta f_{\theta^i}(\varepsilon_t^i; s_t^{(n)}, \hat{a}_t^{-i,(n)})$$
            $$\nabla_{\phi^i} \mathcal{J}_\rho(\phi^i) = \nabla_{\phi^i} \log \rho_{\phi^i}(\hat{a}_t^{-i,(n)}|s_t^{(n)})$$
            $$+ (\nabla_{\hat{a}_t^{-i,(n)}} \log \rho_{\phi^i}(\hat{a}_t^{-i,(n)}|s_t^{(n)}) - \nabla_{\hat{a}_t^{-i,(n)}} \log P(\hat{a}_t^{-i,(n)}|s_t^{(n)}) - \nabla_{\hat{a}_t^{-i,(n)}} Q_{\omega^i}(s_t^{(n)}, a_t^{i,(n)}, \hat{a}_t^{-i,(n)})$$
            $$+ \nabla_{\hat{a}_t^{-i,(n)}} \alpha \log \pi_{\theta^i}(a^{i,(n)}|s_t^{(n)}, \hat{a}_t^{-i,(n)})) \nabla_{\phi^i} g_{\phi^i}(\varepsilon_t^{-i}; s_t^{(n)})$$
        Update parameters:
        $\omega^i = \omega^i - \lambda_Q \nabla_{\omega^i} \mathcal{J}_Q(\omega^i)$
        $\theta^i = \theta^i - \lambda_\pi \nabla_{\theta^i} \mathcal{J}_\pi(\theta^i)$
        $\phi^i = \phi^i - \lambda_{\phi^i} \nabla_{\phi^i} \mathcal{J}_\rho(\phi^i)$
    end for
    Every $C$ gradient descent steps, reset target parameters:
        $$\overline{\omega^i} = \beta \omega^i + (1 - \beta) \overline{\omega^i}$$
end for
```

4.4 Experiments & Results

4.4.1 Iterated Matrix Games

	A	B	C
A	(11, 11)	(-30, -30)	(0 , 0)
B	(-30, -30)	(7, 7)	(6, 6)
C	(0, 0)	(0, 0)	*(5, 5)*

Table 4.1: Payoff matrix of Iterated Climbing Game

We first present the proof-of-principle result of ROMMEO-Q on iterated matrix games where players need to cooperate to achieve the shared maximum reward. To this end, we study the iterated climbing games (ICG) which is a classic purely cooperative two-player stateless iterated matrix games. Climbing game (CG) is a fully cooperative game proposed in (Claus and Boutilier, 1998) whose payoff matrix is summarised in Table 4.1. It is a challenging benchmark because of the difficulty of convergence to its global optimum. There are two Nash equilibrium (A,A) and (B,B) but one global optimal (A,A). The punishment of miscoordination by choosing a certain action increases in the order of $C \rightarrow B \rightarrow A$. The safest action is C and the miscoordination punishment is the most severe for A. Therefore it is very difficult for agents to converge to the global optimum in ICG.

We compare our method to a series of strong baselines in MARL, including Joint Action Learner (JAL) (Claus and Boutilier, 1998), WoLF Policy Hill Climbing (WoLF-PHC) (Bowling and Veloso, 2001), Frequency Maximum Q (FMQ) (Kapetanakis and Kudenko, 2002) and Probabilistic Recursive Reasoning (PR2) (Wen et al., 2019). ROMMEO-Q-EMP is an ablation study to evaluate the effectiveness of our proposed opponent model learning process, where we replace our opponent model with empirical frequency. Figure 4.1 shows the learning curves on ICG for different algorithms. The difference of rewards between ROMMEO-Q and FMQ-c10 may seem small because of the small reward margin between the global optimum and the local one. However, ROMMEO-Q actually outperforms all baselines significantly in terms of converging to the global optimum, which is shown in Figure 4.2. To further analyse the opponent modelling described in Section 4.2, we

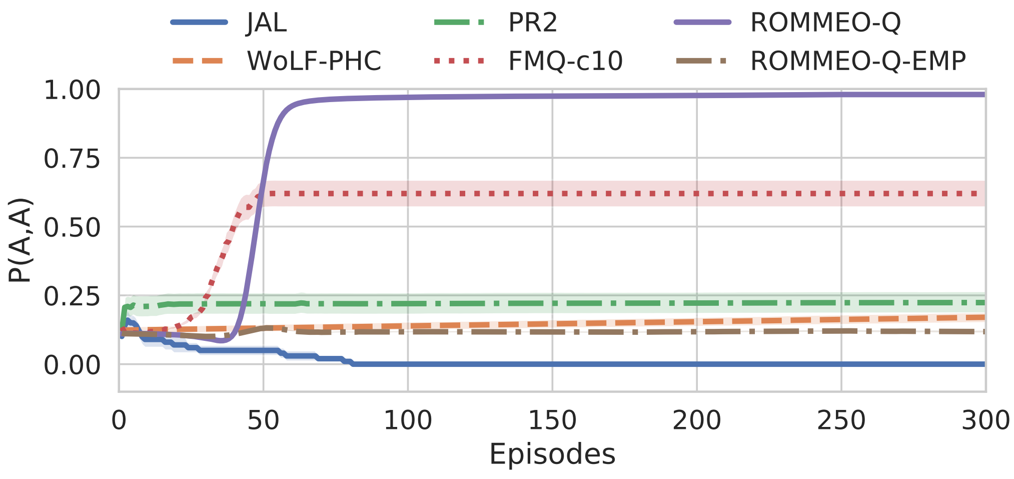


Figure 4.2: Probability of convergence to the global optimum for ROMMEO and baselines on ICG over 100 episodes. The vertical axis is the joint probability of taking actions A for both agents.

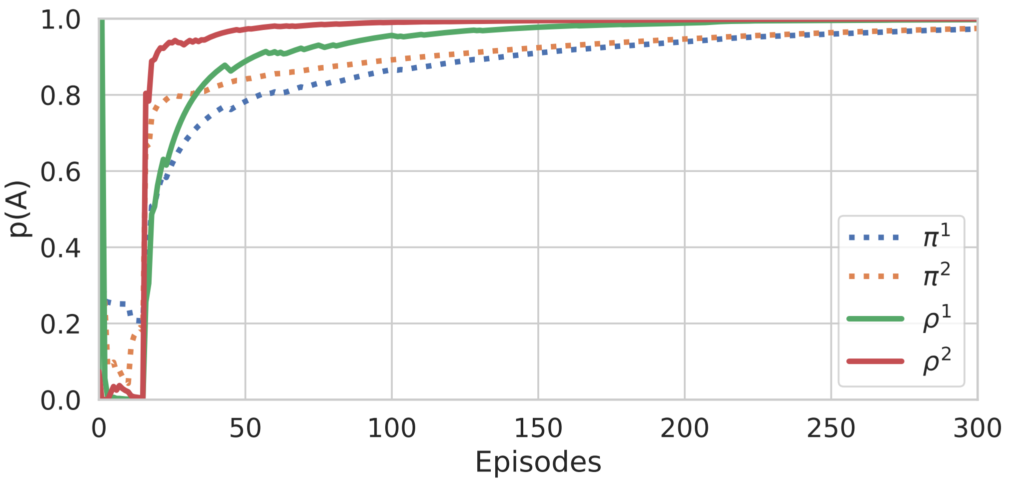


Figure 4.3: Probability of taking A estimated by agent i's opponent model ρ^i and observed empirical frequency P^i in one trail of training, $i \in \{1,2\}$.

visualise the probability of agent $-i$ taking the optimal action A estimated by agent i's opponent model ρ^i and the corresponding true policy π^{-i} in Figure 4.3. Agent i's opponent model "thinks ahead" of agent $-i$ and converges to agent $-i$'s optimal policy before agent $-i$ itself converges to the optimal policy. This helps agent i to respond to its opponent model optimally by choosing action A, which in turn leads to the improvement of agent $-i$'s opponent model and policy. Therefore, the game converges to the global optimum. To note, the big drop of $P(A)$ for both policies and opponent models at the beginning of the training comes from the severe punishment of miscoordination associated with action A.

4.4.2 Differential Games

We adopt the differential Max of Two Quadratic Game (Wei et al., 2018) for continuous case. The agents have continuous action space of $[-10, 10]$. Each agent's reward depends on the joint action following the equations: $r^1\left(a^1, a^2\right) = r^2\left(a^1, a^2\right) = \max\left(f_1, f_2\right)$, where $f_1 = 0.8 \times [-(\frac{a^1+5}{3})^2 - (\frac{a^2+5}{3})^2], f_2 = 1.0 \times [-(\frac{a^1-5}{1})^2 - (\frac{a^2-5}{1})^2] + 10$. We compare the algorithm with a series of baselines including PR2 (Wen et al., 2019), MASQL (Wei et al., 2018; Grau-Moya et al., 2018), MADDPG (Lowe et al., 2017) and independent learner via DDPG (Lillicrap et al., 2015). To compare against traditional opponent modelling methods, similar to (Rabinowitz et al., 2018; He et al., 2016), we implement an additional baseline of DDPG with an opponent module that is trained online with supervision in order to capture the latest opponent behaviours, called DDPG-OM. We trained all agents for 200 iterations with 25 steps per iteration.

The design of reward function above makes the problem a challenging task to most continuous gradient based RL algorithms because gradient update tends to direct the training agent to the sub-optimal point. The reward surface is provided in Figure 4.4 ; there is a local maximum 0 at $(-5, -5)$ and a global maximum 10 at $(5, 5)$, with a deep valley staying in the middle. If the agents' policies are initialised to $(0, 0)$ (the red starred point) that lies within the basin of the left local maximum, the gradient-based methods would tend to fail to find the global maximum equilibrium point due to the valley blocking the upper right area.

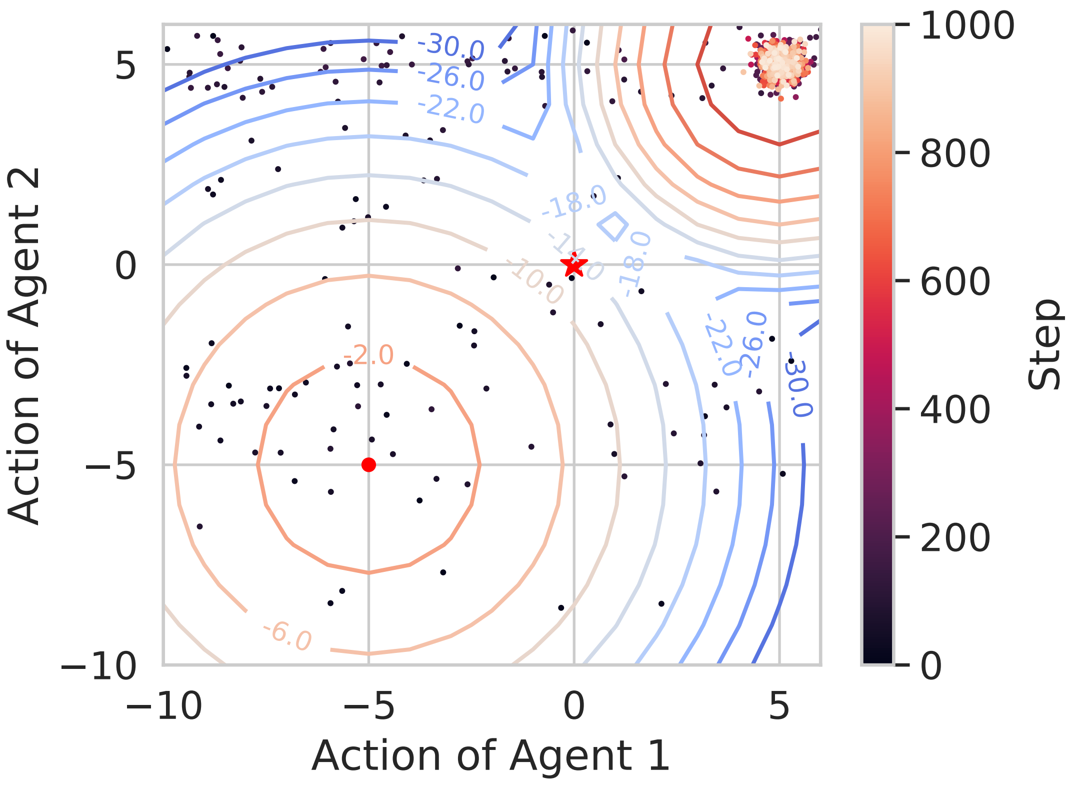


Figure 4.4: Reward surface and learning path of agents. Scattered points are actions taken at each step.

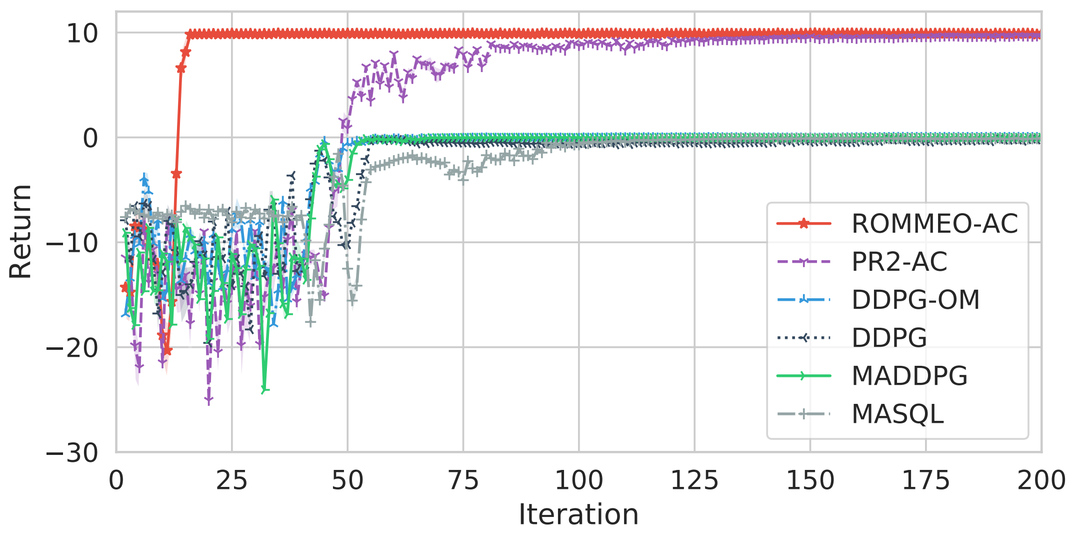


Figure 4.5: Learning curve of ROMMEO and baselines.

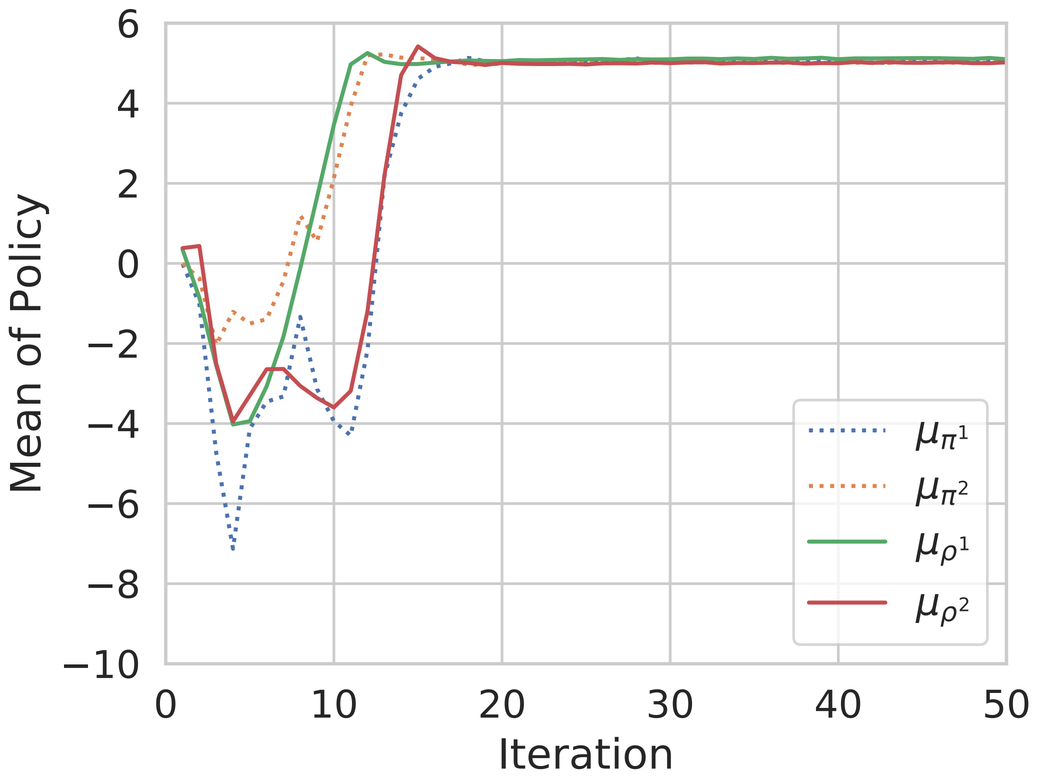


Figure 4.6: Mean of agents' policies π and opponent models ρ.

A learning path of ROMMEO-AC is summarised in Figure 4.4 and the solid bright circle on the right corner implies the convergence to the global optimum. The learning curve is presented in Figure 4.5, ROMMEO-AC shows the capability of converging to the global optimum in a limited amount of steps, while most of the baselines can only reach the sub-optimal point. PR2-AC can also achieve the global optimum but requires many more steps to explore and learn. Additionally, fine tuning on the exploration noise or separate exploration stage is required for deterministic RL methods (MADDPG, DDPG, DDPG-OM, PR2-AC), and the learning outcomes of energy-based RL method (MASQL) are extremely sensitive to the annealing scheme for the temperature. In contrast, ROMMEO-AC employs a stochastic policy and controls the exploration level by the weighting factor α. It does not need a separate exploration stage at the beginning of the training or a delicately designed annealing scheme for α.

Furthermore, we analyse the learning path of policy π and modelled opponent policy ρ during the training, the results are shown in Figure 4.6. The red and orange lines are mean of modelled opponent policy ρ, which always learn to approach the optimal ahead of the policy π (in dashed blue and green lines). This helps the agents to establish the trust and converge to the optimum quickly, which further justifies the effectiveness and benefits of conducting a regularised opponent model proposed in Section 4.2.

4.5 Conclusion

In this chapter, we use Bayesian inference to formulate MARL problem and derive a novel objective ROMMEO which gives rise to a new perspective on opponent modelling. We design an off-policy algorithm ROMMEO-Q with complete convergence proof for optimising ROMMEO. For better generality, we also propose ROMMEO-AC, an actor critic algorithm powered by NNs to solve complex and continuous problems. We give an insightful analysis of the effect of the new learning process of the opponent modelling on agent's performance in MARL. We evaluate our methods on the challenging matrix game and differential game and show that they can outperform a series of strong base lines. It is worthy of noting that Theorems 2, 5 and 6 only guarantees the convergence to optimal solutions with respect to ROMMEO objective but not the optimum in the game. The achievement of the optimum in the game relies on the opponent learning algorithm. In our work, we demonstrate that ROMMEO-Q/AC's convergence to the optimum of the game in self-play setting.

Chapter 5

Partial Observation in Cooperative Games

In an environment where agents intend to cooperate, communication of private information between agents is an efficient way to solve the *partial observation* issue. Most prior works focus on explicit communication where information is conveyed by an established channel such as speech, codified gestures and bluetooth, etc. On the contrary, implicit communication conveys information by actions rather than language or codified gestures. It is important not to mistake non-verbal explicit cues with implicit communication. The former often relates to gestures or body language that have developed an explicit meaning over time, such as the "okay" hand sign, but the information conveyed in the latter needs to be independently inferred by other agents (Gildert et al., 2018).

In this chapter, we propose a generic framework, titled policy belief learning (PBL), for learning to cooperate in imperfect information multi-agent games. Our work combines opponent modelling with a policy that considers that it is being modelled. PBL consists of a *belief module*, which models other agents' private information by considering their previous actions, and a *policy module* which combines the agent's current observation with their beliefs to return a distribution over actions. We also propose a novel auxiliary reward for encouraging communication by actions, which is integrated into PBL. Our experiments show that agents trained using PBL can learn collaborative behaviours more effectively than a number of meaningful baselines without requiring any explicit communication. We conduct a complete ablation study to analyse the effectiveness of different components within PBL in our bridge experiment. A distinguishing feature of our work in relation to previous works in multi-agent communication is that we do not have a predefined explicit communication protocol or learn to communicate through an explicit channel. Information exchange can only happen via actions. In contrast to previous works focusing on unilaterally making actions informative, we focus on bilateral communication by actions where information transmission is directed to a specific party with potentially limited reasoning ability. Our agents learn to communicate through iterated policy and belief updates such that the resulting communication mechanism and belief models are interdependent. The development of a communication mechanism therefore requires either direct access to the mental state of other agents (via centralised training) or the ability to mentalize, commonly known as theory of mind. We investigate our proposed algorithm in both settings.

5.1 Policy Belief Learning

In this chapter, we consider the problem as I2MDP introduced in Chapter 2. We simplify the problem by assuming that hidden information states $\mathcal{X}$ are temporally static and are given at the beginning of the game. Applying naive single agent reinforcement learning (SARL) algorithms to our problem will lead to poor performance. One reason for this is the partial observability of the environment. To succeed in a partially observable environment, an agent is often required to maintain a belief state. Recall that, in our setting, the environment state is formed from the union of the private information of all agents and the publicly observable information, $s_t = [x_t^i, x_t^{-i}, o_t]$. We therefore learn a belief module $\Phi^i(x_t^{-i})$ to model other agents' private information x_t^{-i} which is the only hidden information from the perspective of agent i in our setting. We assume that an agent can model x_t^{-i} given the history of public information and actions executed by other agents $h_t^i = \{o_{1:t-1}, a_{1:t-1}^{-i}\}$. We use a NN to parameterize the belief module which takes in the history of public information and produces a belief state $b_t^i = \Phi^i(x_t^{-i}|h_t^i)$. The belief state together with information observable by agent i forms a sufficient statistic, $\hat{s}_t^i = [x_t^i, b_t^i, o_t]$, which contains all the information necessary for the agent to act optimally (Åström, 1965). We use a separate NN to parameterize agent i's policy $\pi^i(a_t^i|\hat{s}_t^i)$ which takes in the estimated environment state $\hat{s}_t^i$ and outputs a distribution over actions. As we assume hidden information is temporally static, we will drop the time script for it in the rest of the paper.

The presence of multiple learning agents interacting with the environment renders the environment non-stationary. This further limits the success of SARL algorithms which are generally designed for environments with stationary dynamics. To solve this, we adopt centralised training and decentralised execution, where during training all agents are recognised as one central representative agent differing only by their observations. Under this approach, one can imagine belief models $\Phi^i(x^{-i}|h_t^i)$ and $\Phi^{-i}(x^i|h_t^{-i})$ sharing parameters ϕ. The input data, however, varies across agents due to the dependency on both h_t^i and h_t^{-i}. In a similar fashion, we let policies share the parameters θ. Consequently, one may think of updating θ and ϕ using one joint data set aggregated across agents. This centralised training is essentially a self-play training scheme where an agent plays against itself and learns to improve its performance from the experience. Without loss of generality, in the remainder of this section, we discuss the learning procedure from the point of view of a single agent, agent i.

We first present the learning procedure of our belief module. At iteration k, we use the current policy $\pi_{[k]}(a^i|\hat{s})$ to generate a data set of size M, $\Omega_{[k]} = \{(x_j^{-i}, h_j^i)_{j=1}^M\}$, using self-play and learn a new belief module by minimising:

$$\phi_{[k]} := \arg\min_{\phi} \mathbb{E}_{(x^{-i},h^i)\sim\Omega_{[k-1]}} \left[\mathrm{KL}(x_j^{-i} || b_j^i(h_j^i;\phi)) \right], \tag{5.1}$$

where $\mathrm{KL}(\cdot||\cdot)$ is the Kullback–Leibler(KL) divergence and we use a one-hot vector to encode the ground truth, x_j^{-i}, when we calculate the relevant KL-divergence.

With updated belief module $\Phi_{[k]}$, we learn a new policy for the next iteration, $\pi_{[k+1]}$, via a policy gradient algorithm. Sharing information in multi-agent cooperative games through communication reduces intractability by enabling coordinated behaviour. Rather than implementing expensive protocols (Heider and Simmel, 1944), we encourage agents to *implicitly communicate* through actions by introducing a novel auxiliary reward signal. To do so, notice that in the centralised setting agent i has the ability to consult its opponent's belief model $\Phi^{-i}(x^i|h_t^{-i})$ thereby exploiting the fact that other agents hold beliefs over its private information x_i. In fact, comparing b_t^{-i} to the ground-truth x^i enables agent i to learn which actions bring these two quantities closer together and thereby learn informative behaviour. This can be achieved through an auxiliary reward signal devised to encourage informative action communication:

$$r_{c,t}^i = \mathrm{KL}(x^i||b^{-i,*}) - \mathrm{KL}(x^i||b_{t+1}^{-i}), \tag{5.2}$$

where $b^{-i,*} = \Phi_{[k]}^{-i}(x^i|h_{t,*}^{-i})$ is agent $-i$'s best belief (so-far) about agent i's private information:

$$b^{-i,*} = \arg\min \ \mathrm{KL}(x^i||b_u^{-i}) \ \forall \ u \leq t.$$

In other words, $r_{c,t}^i$ encourages communication as it is proportional to the improvement in the opponent's belief (for a fixed belief model $\Phi_{[k]}^{-i}(x^i|h_{t+1}^{-i})$), measured by its proximity to the ground-truth, resulting from the opponent observing agent i's action a_t^i. Hence, during the policy learning step of PBL, we apply a policy gradient algorithm with a shaped reward of the form:[1]

$$r = r_e + \alpha r_c, \tag{5.3}$$

where r_e is the reward from the environment, r_c is the communication reward and $\alpha \geq 0$ balances the communication and environment rewards.

Initially, in the absence of a belief module, we pre-train a policy $\pi_{[0]}$ naively by ignoring the existence of other agents in the environment. As an agent's reasoning ability may be limited, we may then iterate between Belief and Policy learning multiple times until either the allocated computational resources are exhausted or

[1]Please note, we omit the agent index i in the reward equation, as we shape rewards similarly for all agents.

Algorithm 3 Per-Agent Policy Belief Learning (PBL)

```
1: Initialise: Randomly initialise policy $\pi_0$ and belief $\Phi_0$
2: Pre-train $\pi_0$
3: for $k = 0$ to max_iterations do
4:     Sample episodes for belief training using self-play forming the data set $\Omega_{[k]}$
5:     Update belief model using data from $\Omega_{[k]}$ solving Equation 5.1
6:     Given updated beliefs $\Phi_{[k+1]}(\cdot)$, update policy $\pi(\cdot)$ (policy gradients with
       rewards from Equation 5.3)
7: end for
8: Output: Final policy, and belief model
```

the policy and belief modules converge. We summarise the main steps of PBL in Algorithm 3. Note that, although information can be leaked during training, as training is centralised, distributed test-phase execution ensures hidden-private variables during execution.

5.1.1 Machine Theory of Mind

In PBL, we adopt a centralised training and decentralised execution scheme where agents share the same belief and policy models. In reality, however, it is unlikely that two people will have exactly the same reasoning process. In contrast to requiring everyone to have the same reasoning process, a person's success in navigating social dynamics relies on their ability to attribute mental states to others. This attribution of mental states to others is known as theory of mind (Premack and Woodruff, 1978). Theory of mind is fundamental to human social interaction which requires the recognition of other sensory perspectives, the understanding of other mental states, and the recognition of complex non-verbal signals of emotional state (Lemaignan and Dillenbourg, 2015). In collaboration problems without an explicit communication channel, humans can effectively establish an understanding of each other's mental state and subsequently select appropriate actions. For example, a teacher will reiterate a difficult concept to students if she infers from the students' facial expressions that they have not understood. The effort of one agent to model the mental state of another is characterised as Mutual Modelling (Dillenbourg, 1999).

In our work, we also investigate whether the proposed communication reward can be generalised to a distributed setting which resembles a human application of theory of mind. Under this setting, we train a separate belief model for each agent so that $\Phi^i(x^{-i}|h_t^i)$ and $\Phi^{-i}(x^i|h_t^{-i})$ do not share parameters ($\phi^i \neq \phi^{-i}$). Without centralisation, an agent can only measure how informative its action is to others with its own belief model. Assuming agents can perfectly recall their past actions and observations, agent i computes its communication reward as:[2]

$$r_{c,t}^i = \mathrm{KL}(x^i||\tilde{b}_t^{i,*}) - \mathrm{KL}((x^i||\tilde{b}_{t+1}^i),$$

where $\tilde{b}_t^{i,*} = \Phi^i(x^i|h_{t,*}^{-i})$ and $\tilde{b}_{t+1}^i = \Phi^i(x^i|h_{t+1}^{-i})$. In this way, an agent essentially establishes a mental state of others with its own belief model and acts upon it. We believe this could be a step towards machine theory of mind where algorithmic agents learn to attribute mental states to others and adjust their behaviour accordingly.

5.2 Contract Bridge

As we will test PBL on bridge, which is a complex game. Therefore, we introduce the rules of bridge before we presenting our results. In bridge, two teams of two (North-South vs East-West) are situated in opposing positions and play a trick-taking game using a standard 52-card deck. Following a deal, bidding and playing phases can be effectively separated. During the bidding phase, players sequentially bid for a contract until a final contract is reached. A *PASS* bid retains previously proposed contracts and a contract is considered final if it is followed by three consecutive *PASS* bids. A non-*PASS* bid proposes a new contract of the form $\langle$`integer,suit`$\rangle$, where `integer` takes integer values between one and seven, and `suit` belongs to $\{\clubsuit, \diamondsuit, \heartsuit, \spadesuit, \mathrm{NT}\}$. The number of tricks needed to achieve a contract are $6+$`integer`, and an NT suit corresponds to bidding to win tricks without trumps. A contract-declaring team achieves points if it fulfils the contract, and if not, the points for the contract go to the opposing team. Bidding must be non-decreasing, meaning `integer` is non-decreasing and must increase if the

[2]Note the difference of super/sub-scripts of the belief model and its parameters when compared to Equation 5.2.

newly proposed trump suit precedes or equals the currently bid suit in the ordering $\clubsuit < \diamondsuit < \heartsuit < \spadesuit < NT$.

Playing Phase in Bridge: After the final contract is decided, the player from the declaring side who first bid the trump suit named in the final contract becomes Declarer. Declarer's partner is Dummy. The player to the left of the declarer becomes the first leading player. Then Dummy lays his cards face up on the table and then play proceeds clockwise. On each trick, the leading player shows one card from their hand and other players need to play the same suit as the leading player if possible; otherwise, they can play a card from another suit. Trump suit is superior to all other suits and, within a suit, a higher rank card is superior to lower rank one. A trick is won by the player who plays the card with the highest priority. The winner of the hand becomes the leading player for the next trick.

Non-competitive bidding: In this work, we focus on non-competitive bidding in bridge, an imperfect-information game that requires information exchange between agents to agree high-quality contracts. Hence, such a game serves as an ideal test-bed for our algorithm. We consider North (N) and South (S) bidding in the game, while East (E) and West (W) always bid *PASS*. Hence, the declaring team never changes. Thus, each deal can be viewed as an independent episode of the game. The private information of player $i \in \{N,S\}$, x^i, is its hand. x^i is a 52-dimensional binary vector encoding player i's 13 cards. In Bridge, an agent does not have public state o. In each episode, Players N and S are dealt hands x^N, x^S respectively. Their hands, together, describe the full state of the environment $s = \{x^N, x^S\}$, which is *not fully observed* by either of the two players.

Double Dummy Analysis (DDA): Since rolling out via self-play for every contract is computationally expensive, we resort to double dummy analysis (DDA) (Haglund, 2010) for score estimation. Double Dummy Analysis assumes that, for a particular deal, one player's hand is fully observed by other players and players always play cards to their best advantage. However, given a set $s = \{x^N, x^S\}$, the distribution of remaining cards for the two non-bidding players East and West is still unknown. To reduce the variance of the estimate, we repeatedly sample a deal U times by allocating the remaining cards randomly to East and West and then estimate $r_e(s)$ by taking the average of their DDA scores,

$$r_e(s) = \frac{1}{U}\sum_{u=1}^{U} r_e(x^N, x^S, x_u^E, x_u^W), \tag{5.4}$$

where x_u^E, x_u^W are hands for East and West from the u^{th} sampling respectively. For a specific contract a_t, the corresponding score is given by $r_e(a_t|s) = r_e(x^N, x^S, a_t)$. In our work, we set $U = 20$. Interested readers are referred to (Haglund, 2010) for further details.

Scoring: We use standard Duplicate bridge scoring rules (League, 2017) to score games and normalise scores by dividing them by the maximum absolute score.

Algorithm 4 Bridge Duplicate Scoring

```
Score ($tricks\_made, bid\_level, trump\_suit$)
$T \leftarrow trump\_suit$
$\delta \leftarrow tricks\_made - (bid\_level + 6)$
$score \leftarrow 0$
if $\delta \geq 0$ then
    $score \leftarrow score + bid\_level * scale_T + bias_T$ {Contract tricks}
    if score $\geq 100$ then
        $score \leftarrow 300$ {Game Bonus}
    else
        $score \leftarrow 50$ {PARTSCORE}
    end if
    if $\delta = 6$ then
        $score \leftarrow score + 500$ {Slam bonus}
    else if $\delta = 7$ then
        $score \leftarrow score + 1000$ {Grand Slam bonus}
    end if
    if $\delta > 0$ then
        $score \leftarrow score + \delta * scale_T$ {Over-tricks}
    end if
else
    $score \leftarrow score - bid\_level * 50$ {Under-tricks}
end if
```

Algorithm 4 shows how we score a game under Duplicate Bridge Scoring rules. We obtain the average of *tricks_made* using Double Dummy Analysis (Haglund, 2010) given the hands of players North and South $\{x^N, x^S\}$, the declarer and the

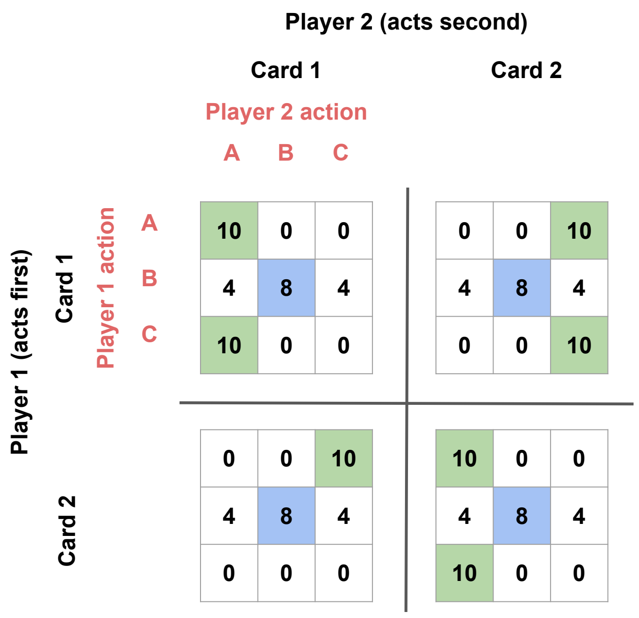


Figure 5.1: Payoff for the matrix game

trump suit. The score function above has a scale and bias for each trump suit. The scale is 20 for ♣ and ♢ and 30 for all others. Bias is zero for all trumps, except NT which has a bias of 10. Note that *Double* is only a valid bid in response to a contract proposed by one's opponents. Also, a *Redouble* bid must be preceded by a *Double* bid. In the non-competitive game, opponents do not propose contracts, so these options are naturally not included.

5.3 Experiments & Results

We test our algorithms in three experiments. In the first, we validate the correctness of the PBL framework which integrates our communication reward with iterative belief and policy module training in a simple matrix game. In this relatively simple experiment, PBL achieves near optimal performance. Equipped with this knowledge, we further apply PBL to the non-competitive bridge bidding problem to verify its scalability to more complex problems. Lastly, we investigate the efficacy of the proposed communication reward in a distributed training setting.

5.3.1 Matrix Game

We test our PBL algorithm on a matrix card game where an implicit communication strategy is required to achieve the global optimum. This game is first proposed in (Foerster et al., 2018). There are two players and each player receives a card drawn from $\{\text{card } 1, \text{card } 2\}$ independently at the beginning of the game. Player 1 acts first and Player 2 responds after observing Player 1's action. Neither player can see the other's hand. By the design of the payoff table (shown in Figure. 5.1), Player 1 has to use actions C and A to signify that it holds Cards 1 and 2 respectively so that Player 2 can choose its actions optimally with the given information. We compare PBL with algorithms proposed in (Foerster et al., 2018) and vanilla policy gradient. As can be seen from Figure 5.2, PBL performs similarly to BAD and BAD-CF on this simple game and outperforms vanilla policy gradient significantly. This demonstrates a proof of principle for PBL in a multi-agent imperfect information coordination game.

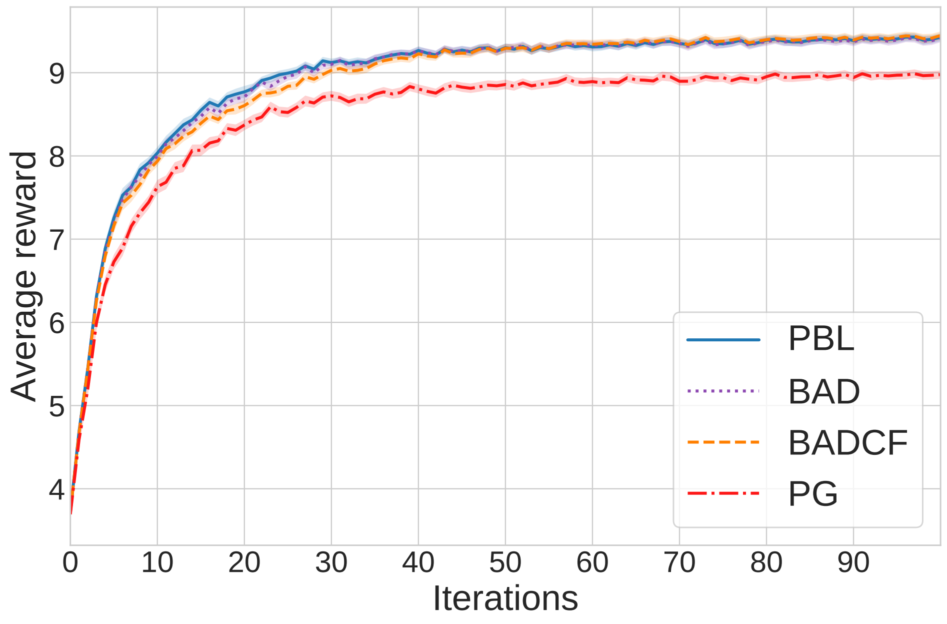


Figure 5.2: Learning curves of PBL and baselines over 100 runs

5.3.2 Contract Bridge Case-Study

Next, we test PBL on the non-competitive bidding problem. As this experiment includes a lot of technical details, we will discuss them first in the following paragraphs.

Policy Pre-training: In the first PBL iteration $(k = 0)$, to have a good initial policy and avoid one with low-entropy, we train our policy to predict a distribution formed using Softmax on $r^e(s)$ with temperature τ as a warm start. The loss for pre-train policy given s is:

$$\mathcal{L}_{k0} = \mathrm{KL}(\pi(a|x)||\mathrm{Softmax}(r^e(s), \tau)),$$

where KL is the KL-Divergence. To have a fair comparison with other benchmarks, all our benchmarks are initialised with this pre-trained policy. Supervising a bidding policy to predict pre-calculated scores for all actions only provides it with a basic understanding of its hand.

Policy Training: We utilise Proximal Policy Optimisation (PPO) (Schulman et al., 2017) for policy training. Optimisation is performed with the Adam optimiser (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$. The initial learning rate is 10^{-4} which we decay exponentially with a rate of decay of 0.95 and a decaying step of 50. The distance measure used in communication rewards r^c is cross entropy and we treat each dimension of belief and ground truth as an independent Bernoulli distribution. We train PBL with 8 PBL iterations, we do policy gradient 200 times per iteration and we sample 5000 episodes each time. Each mini-batch is then a sub-sample of 2048 episodes. The initial communication weight α is 5, and we decay it gradually. We train all other baselines with the same PPO hyper-parameters.

Learning A Belief Model: When a player tries to model its partner's hand based on the observed bidding history, we assume it can omit the restriction that its partner can only hold 13 cards. Therefore, we take the prediction of the partner's hand given the observed bidding history as a 52-label classification problem, where each label represents one corresponding card being in the partner's hand. In other words, we treat each card from a 52-card deck being in the partner's hand as an independent Bernoulli distribution and we train a belief model by maximising the joint likelihood of these 52 Bernoulli distributions given a bidding history h. This gives the loss for belief model as:

$$\mathcal{L}_\Phi = -\sum_{i=1}^{52} x^{-i} \log(b^i) + (1 - x^{-i}) \log(1 - b^i),$$

where x^{-i} and b^i are elements of one-hot encoding vectors of a partner's hand x^{-i} and one agent's belief b^i. The reasoning behind this assumption is we think it is more important to have a more accurate prediction over an invalid distribution than a less accurate one over a valid distribution as the belief itself is already an approximation. For each iteration of belief training, we generate 300,000 data episodes with the current policy to train the belief model. Optimisation is performed with the Adam optimiser with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$ and a decay rate of 0.95. The initial learning rate is 10^{-3}. The batch size is 1024. We split the data set such that 90% of it is used for training and 10% is for early stopping check to prevent overfitting.

Benchmarking & Ablation Studies: PBL introduces several building blocks, each affecting performance in its own right. We conduct an ablation study to better understand the importance of these elements and compare against a state-of-the-art method in PQL (Yeh and Lin, 2016). We introduce the following baselines:

1. **Independent Player (IP)**: A player bids independently without consideration of the existence of the other player.

2. **No communication reward (NCR)**: One important question to ask is how beneficial the additional communication auxiliary reward r_c is in terms of learning a good bidding strategy. To answer this question, we implement a baseline using the same architecture and training schedule as PBL but setting the communication reward weighting to zero, $\alpha = 0$.

3. **No PBL style iteration (NPBI)**: To demonstrate that multiple iterations between policy and belief training are beneficial, we compare our model to a baseline policy trained with the same number of weight updates as our model but no further PBL iterations after training a belief model Φ_0 at PBL iteration $k = 0$.

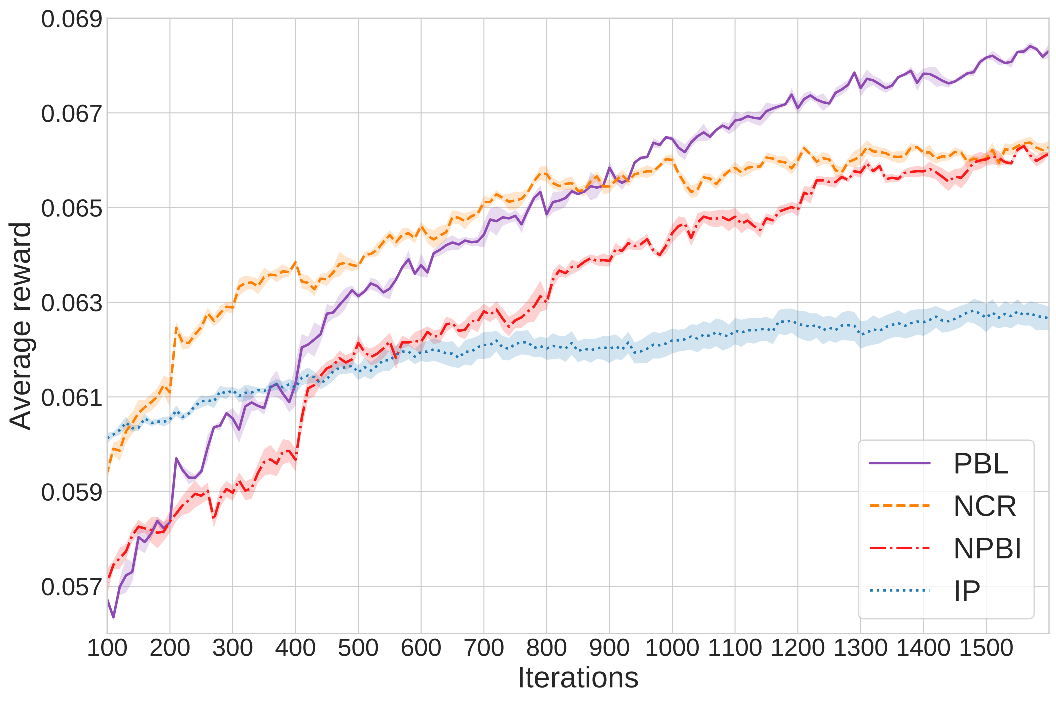


Figure 5.3: Learning curves for non-competitive bridge bidding with a warm start from a model trained to predict the score distribution (average reward at warm start: 0.038).

4. **Penetrative Q-Learning (PQL)**: PQL as proposed by Yeh and Lin (2016) as the first bidding policy for non-competitive bridge bidding without human domain knowledge.

Figure 5.3 shows the average learning curves of our model and three baselines for our ablation study. We obtain these curves by testing trained algorithms periodically on a pre-generated test data set which contains 30,000 games. Each point on the curve is an average score computed by Duplicate bridge scoring rules (League, 2017) over 30,000 games and 6 training runs. As can been seen, IP and NCR both initially learn faster than our model. This is reasonable as PBL spends more time learning a communication protocol at first. However, IP converges to a local optimum very quickly and is surpassed by PBL after approximately 400 learning iterations. NCR learns a better bidding strategy than IP with a belief module. However, NCR learns more slowly than PBL in the later stage of training because it has no guidance on how to convey information to its partner. PBL outperforming NPBI demonstrates the importance of iterative training between policy and belief

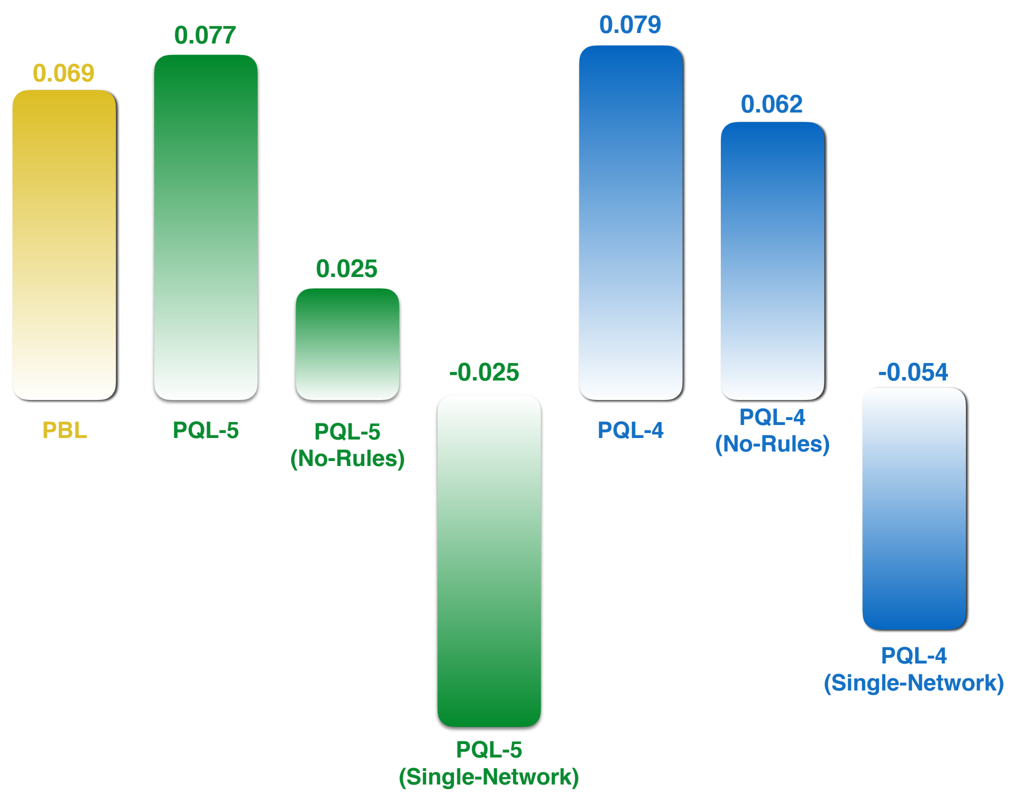


Figure 5.4: Bar graph comparing PBL to variants of PQL, with the full version of PQL results as reported in (Yeh and Lin, 2016)

modules.

Restrictions of PQL: PQL (Yeh and Lin, 2016) is the first algorithm trained to bid in bridge without human engineered features. However, its strong bidding performance relies on heavy adaption and heuristics for non-competitive Bridge bidding. First, PQL requires a predefined maximum number of allowed bids in each deal, while using different bidding networks at different times. Our results show that it will fail when we train a single NN for the whole game, which can been seen as a minimum requirement for most DRL algorithms. Second, PQL relies on a rule-based function for selecting the best contracts at test time. In fact, removing this heuristic significantly reduces PQL's performance as reported in Figure 5.4. In addition, without pre-processing the training data as in (Yeh and Lin, 2016), we could not reproduce the original results. To achieve state-of-the-art performance, we could use these (or other) heuristics for our bidding algorithm. However, this deviates from the focus of our work which is to demonstrate that PBL is a general framework for learning to communicate by actions.

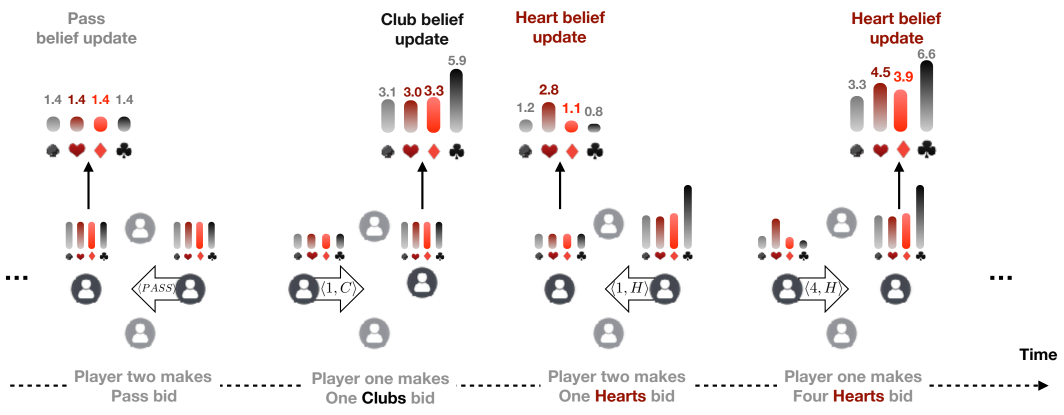


Figure 5.5: An example of a belief update trace showing how PBL agents use actions for effective communication. Upon observing $\langle PASS \rangle$ from East, West decreases its HCP belief in all suits. When West bids $\langle 1, \mathrm{C} \rangle$, East improves belief in clubs. Next, East bids $\langle 1, \mathrm{H} \rangle$. West recalculates its belief from last time step and increases its HCP belief in hearts.

Belief Update Visualisation: To understand how agents update their beliefs after observing a new bid, we visualise the belief update process (Figure 5.5). An agent's belief about its opponent's hand is represented as a 52-dimensional vector with real values which is not amenable to human interpretation. Therefore, we use *high card points* (HCPs) to summarise each agent's belief. For each suit, each card is given a point score according to the mapping: **A=4, K=3, Q=2, J=1, else=0**. Note that while agents' beliefs are updated based on the entire history of its opponent's bids, the difference between that agent's belief from one round to the next is predominantly driven by the most recent bid of its opponent, as shown in Figure 5.5.

Learned Bidding Convention: Whilst our model's bidding decisions are based entirely on raw card data, we can use high card points as a simple way to observe and summarise the decisions which are being made. For example, we observe our policy opens the bid with 1♠ if it has HCPs of spade 4.5 or higher but lower HCPs of any other suits. We run the model on the unseen test set of 30,000 deals and summarise the learned bidding convention. Table 5.1 shows the average HCPs (aHCPs) present in a hand for each of the opening bidding decisions made by North. Once an opening bid is observed, South updates their belief; Table 5.2 shows the effect which each opening bid has on South's belief. We show in Table 5.3 the responding

Bid	♣	♢	♡	♠	Total
PASS	1.4	1.3	1.4	1.4	5.4
1♠	2.3	2.1	2.3	**4.5**	11.2
1♡	2.3	2.2	**4.7**	1.8	11.0
1♢	2.2	**4.4**	2.1	2.0	10.7
1♣	**4.8**	1.9	2.1	2.1	10.9
3NT	4.4	4.4	4.9	4.6	18.3
4♢	3.0	**6.5**	3.1	3.4	16.1
4♣	**6.6**	5.3	2.3	2.3	16.5

Table 5.1: Opening bid - **own** aHCPs.

Bid	♣	♢	♡	♠	Total
PASS	1.4	1.3	1.3	1.3	5.3
1♠	2.3	2.1	2.2	**4.6**	11.1
1♡	2.2	2.2	**4.7**	1.9	11.0
1♢	2.3	**4.5**	2.0	2.0	10.8
1♣	**4.8**	2.0	2.2	2.2	11.1
3NT	4.4	4.6	4.8	4.7	18.5
4♢	2.8	**6.8**	3.3	3.1	16.0
4♣	**6.2**	4.0	3.0	2.8	16.0

Table 5.2: **Belief** HCPs after observing opening bid.

Bid	♣	♢	♡	♠	Total
PASS	4.4	4.5	4.4	4.5	17.8
1♠	4.4	4.3	4.5	**7.1**	20.3
1♡	4.6	4.4	**7.0**	4.9	20.9
1♢	4.3	**6.7**	4.8	4.9	20.7
1♣	**7.2**	4.3	4.7	4.9	21.1
2♠	4.8	4.7	5.2	**8.5**	23.2
2♡	4.9	5.1	**8.6**	5.1	23.7
3NT	6.6	6.6	7.1	7.2	27.6
4♢	5.2	**8.5**	5.5	5.6	24.9
4♣	**8.4**	5.6	5.4	5.4	24.8
6♡	5.7	6.2	**9.9**	6.9	28.7
6♢	6.4	**9.2**	7.3	7.1	30.0
6♣	**10.8**	4.0	10.0	6.5	31.3
6NT	7.5	7.0	8.7	9.0	32.2

Table 5.3: Responding bid - **own + belief** aHCPs.

bidding decisions made by South; aHCP values in Table 5.3 are the sum of HCPs in South's hand and South's belief over HCPs in North's hand. The values highlighted in bold for each row are the maximum values for the respective row. This is only done for rows where the bid has a specified trump suit. We also present some interesting bidding examples by PBL agents in Figure 5.6.

***Double Pass* Analysis** At the beginning of bidding, when two players both bid *Pass* (*Double Pass*), all players' hands are re-dealt and a new episode starts. If we ignore these episodes in training, a naive strategy emerges where a player always bids *Pass*

NORTH
6♠ 10♠ 2♥ 4♥ 6♥ A♥ 2♦ 3♣ 5♣ J♣ Q♣ K♣ A♣
SOUTH
8♠ K♠ A♠ 8♥ 3♦ 4♦ 7♦ 9♦ J♦ A♦ 2♣ 6♣ 7♣

BIDS: 1♣ 1♢ 2♣ 2♢ 3NT PASS
REWARD: 0.283

(a) North has a very strong hand in clubs holding all of the picture cards and the ace. North also holds the ace of hearts. South has strength in diamonds and spaces. The bidding pattern shows that North initially signals their strength in clubs and South their strength in diamonds. Both parties then reiterate their strengths in their next turns and a contract of 3NT is agreed.

NORTH
8♠ Q♠ K♠ A♠ 8♥ Q♥ 2♦ 3♦ 4♦ 9♦ J♦ Q♦ A♦
SOUTH
3♠ 4♠ 3♥ 7♥ 9♥ 10♥ J♥ K♥ 5♦ 6♦ K♦ 4♣ A♣

BIDS: 1♢ 1♡ 3NT 4♡ 6♡ PASS
REWARD: 0.301

(b) North opens bidding with diamonds to show their strength in the suit. South then responds with hearts to show their strongest suit. North then bids no trumps to signal an inferred collective strength in multiple suits. South has little strength outside of hearts and continues to bid hearts and ultimately a contract of 6♡ is reached as bid by North who only has two hearts. This is a bid to win all but one hand with hearts as trumps which reflects the fact that between them the pair have from 7 to King of hearts. The pairing bid hearts despite their collective strength also in diamonds since hearts commands a higher per trick score if the contract is met.

NORTH
2♠ 4♠ 7♠ J♠ A♠ K♥ K♦ 3♣ 4♣ 5♣ 6♣ 8♣ A♣
SOUTH
6♠ 9♠ K♠ 7♥ 3♦ 5♦ 6♦ 10♦ Q♦ A♦ 10♣ J♣ K♣

BIDS: 1♣ 1♢ 1♠ 2♣ 3NT 4♠ PASS
REWARD: 0.305

(c) Between them the pairing have a strong hand in diamonds clubs and spades with a lack of hearts (15 + 13 = 28 HCP Total). Bidding opens with 1♣ as North aims to show their strength in clubs. South then conveys their strength in diamonds with a bid of 1♢. Bidding continues as North demonstrates some strength in spades with a bid of 1♠. Later, a bid by North of 3NT demonstrates that North believes the pair have strength across the board as reflected by the bids from all suits except hearts. Bidding ends however, with a bid of 4♠ from South which is then accepted through a PASS bid from North. South makes the bid of 4♠ despite only having 3 spades in their hand. The earlier bid of 1♠ from North informs South that North has some strength in spades and hence South believes that the pair may be able to attain 4♠.

NORTH
3♠ 9♠ J♠ Q♠ 6♥ 10♥ Q♥ 2♦ 3♦ 7♦ A♦ 2♣ 5♣
SOUTH
7♠ 10♠ 4♥ J♥ A♥ 4♦ 8♦ Q♦ K♦ 4♣ 6♣ K♣ A♣

BIDS: PASS 1♣ 1♢ 2♢ 3NT PASS
REWARD: 0.211

(d) North opens bidding with PASS signalling they they have a relatively weak hand (8 HCP Total). South then responds with 1♢ signalling that diamond is the stronger of their suits. South also has strength in diamonds and ultimately the pairing is able to reason about their collective hands and bid 3NT; a bid that would not be reached without forming beliefs due to the relative weakness of North's hand.

Figure 5.6: Bidding Examples by PBL Agents

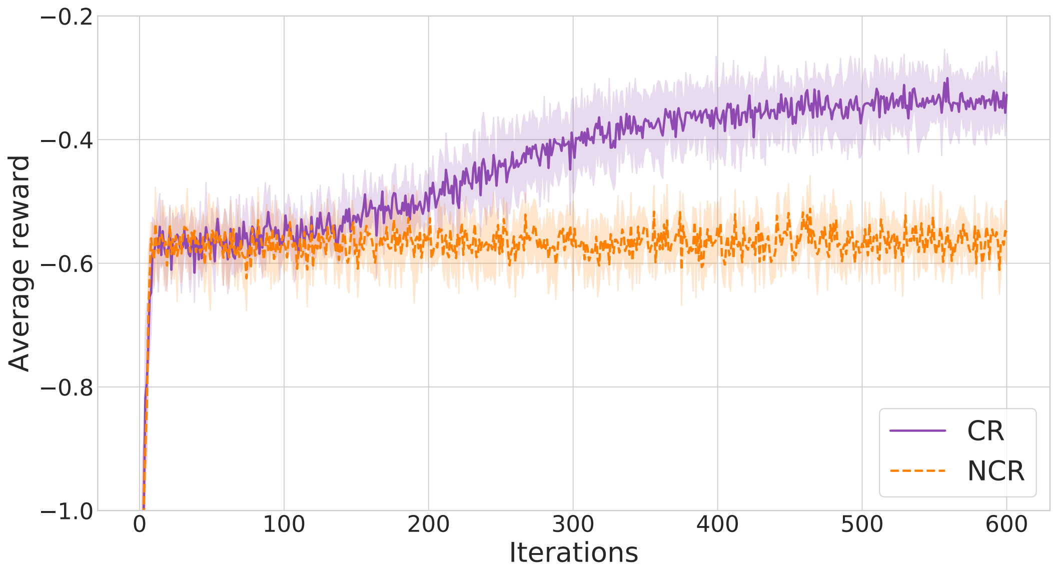


Figure 5.7: Learning curves for Silent Guide. Guide agent trained with communication reward (CR) significantly outperforms the one trained with no communication reward (NCR).

unless it is highly confident about its hand and therefore bids at level 1 whose risk is at the minimum. In this work, we are interested in solving problems where private information needs to be inferred from observed actions for better performance in a game. Therefore, this strategy is less meaningful and the opportunity cost of bidding *Pass* could be high when a player could have won the game with high reward. To reflect this opportunity cost in training, we set the reward for Double *Pass* as the negative of the maximum attainable reward given the players' hands: $r_{dp}(s) = -\max(r_e(s))$. Therefore, a player will be penalised heavily by bidding *Pass* if it could have obtained a high reward otherwise and awarded slightly if they could never win in the current episode. It is worthy of note that an initial *Pass* bid can convey information; however, if it is followed by a second *Pass* bid the game ends and hence no further information is imparted from a second *Pass*.

5.3.3 Silent Guide

We modify a multi-agent particle environment (Lowe et al., 2017) to test the effectiveness of our novel auxiliary reward in a distributed setting. This environment also allows us to explore the potential for implicit communication to arise through machine theory of mind. In the environment there are two agents and three landmarks.

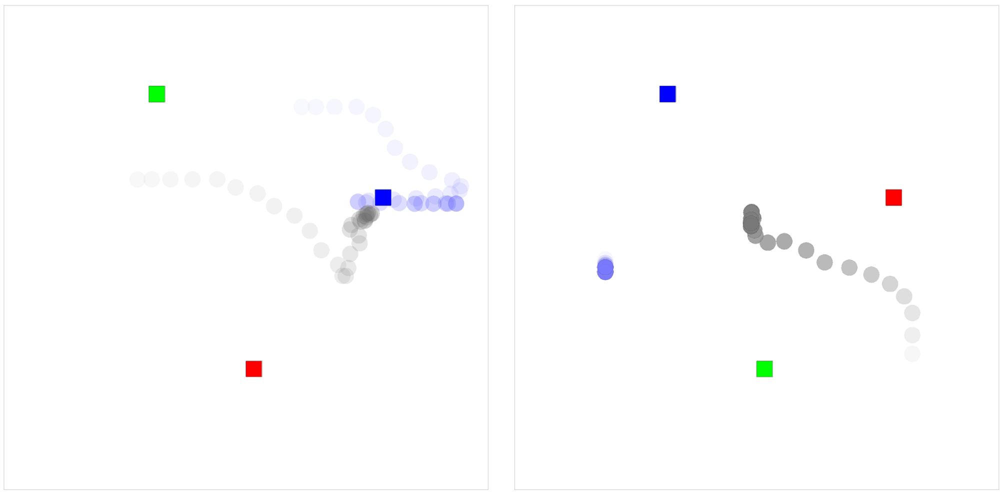

(a) CR **(b)** NCR

Figure 5.8: a) A trajectory of Listener (grey circle) and Guide (blue circle) with CR. Landmarks are positioned randomly and the Goal landmark (blue square) is randomly chosen at the start of each episode. b) A trajectory of Listener and Guide with NCR. Trajectories are presented with agents becoming progressively darker over time.

We name the agents Guide and Listener respectively. Guide can observe Listener's goal landmark which is distinguished by its colour. Listener does not observe its goal. However, Listener is able to infer the meaning behind Guide's actions. The two agents receive the same reward which is the negative distance between Listener and its goal. Therefore, to maximise the cumulative reward, Guide needs to tell Listener the goal landmark colour. However, as the "Silent Guide" name suggests, Guide has no explicit communication channel and can only communicate to Listener through its actions.

In the distributed setting, we train separate belief modules for Guide and Listener respectively. The two belief modules are both trained to predict a naive agent's goal landmark colour given its history within the current episode but using different data sets. We train both Guide and Listener policies from scratch. Listener's policy takes Listener's velocity, relative distance to three landmarks and the prediction of the belief module as input. It is trained to maximise the environment reward it receives. Guide's policy takes its velocity, relative distance to landmarks and Listener's goal as input. To encourage communication by actions, we train Guide policy with the auxiliary reward proposed in our work. We compare our method against a naive Guide policy which is trained without the communication reward. The results are shown in Figure 5.7. Guide when trained with communication reward (CR) learns to inform Listener of its goal by approaching to the goal it observes. Listener learns to follow. However, in NCR setting, Listener learns to ignore Guide's uninformative actions and moves to the centre of three landmarks. While Guide and Listener are equipped with belief models trained from different data sets, Guide manages to use its own belief model to establish the mental state of Listener and learns to communicate through actions judged by this constructed mental state of Listener. We also observe that a trained Guide agent can work with a naive RL listener (best reward -0.252) which has no belief model but can observe PBL guide agent's action. The success of Guide with CR shows the potential for machine theory of mind. We obtain the learning curves by repeating the training process five times and take the shared average environment reward.

5.4 Conclusion

In this chapter, we focus on implicit communication through actions. This draws a distinction of our work from previous works which either focus on explicit communication or unilateral communication. We propose an algorithm combining agent modelling and communication for collaborative imperfect-information games. Our PBL algorithm iterates between training a policy and a belief module. We propose a novel auxiliary reward for encouraging implicit communication between agents which effectively measures how much closer the opponent's belief about a player's private information becomes after observing the player's action. We empirically demonstrate that our methods can achieve near optimal performance in a matrix problem and scale to complex problems such as contract bridge bidding. We conduct an initial investigation of the further development of machine theory of mind. Specifically, we enable an agent to use its own belief model to attribute mental states to others and act accordingly. We test this framework and achieve some initial success in a multi-agent particle environment under distributed training. There are a lot of interesting avenues for future work such as exploration of the robustness of collaboration to differences in agents' belief models.

Chapter 6

Safe Exploitation in Competitive Games

In previous chapters we have looked into challenges of *non-stationarity* and *partial observation* in cooperative games. As discussed in the Chapter 2, these challenges mainly arise from the lack of knowledge of the opponents. Therefore, based on the characteristics of the specific problems, we propose ROMMEO and PBL, two opponent modelling based solutions to the problems respectively.

In this chapter, we will focus on *non-stationarity* and *unclear learning objective* in competitive games. An opponent's learning process can be generally modelled as transitions among a mixture of unknown number of policies. This motivates the usage of a Dirichlet process mixture model. As we can only collect trajectories produced by the adaptive opponents online, we propose to learn our model in a streaming fashion. Given the predicted opponent policy from our model, we provide a general framework for training an agent to safely exploit the non-stationary opponent where safe exploitation means exploiting the current opponent with a low probability of being exploited by other opponents in future. We empirically demonstrate the ability of our approach to safely exploit a non-stationary opponent in Kuhn Poker, a simplified Poker game. Furthermore, once trained, our model can produce strong counter strategies to unseen opponents without any further training in new tournaments.

6.1 Preliminaries

We consider the decentralised training and decentralised execution (DTDE) setting in zero-sum games where we have access to interaction trajectories τ between our agent and the adaptive opponent. To solve *non-stationarity* and *unclear learning objective* in this setting, we approach this problem by formulating the interaction between the training agent and its opponents as a two games at different levels. At the lower level, the game is essentially an actual environment where agents compete with others, for example, a poker game. However, every time before agents playing the lower level game, they need to play a meta-level game. In this game, they need to decide their playing policies for the lower level. One agent's each playing policy can be selected from his/her policies pool or it could be a weighted average over all his/her policies.

6.1.1 Two-Level Games

We assume that interactions of players in lower level environment can be modelled by a stochastic game (Shapley, 1953) which is introduced in Chapter 2. In this work, we specifically denote a trajectory as a set of the opponent's end-game histories collected from d episodes $\tau = \{h^{-i}_{1,T}, h^{-i}_{2,T}, \ldots, h^{-i}_{d,T}\}$, where T is the terminal time step in an episode. This implies that the opponent's policy does not change over d episodes. However, this is not true in our setting or in general. Therefore, we take d as a hyper-parameter and tune it to obtain satisfactory empirical results.

We formulate the meta-level game as a two-player normal-form game (NFG) between our agent and its opponents as a whole with notation adapted to our presentation. We denote a 2-player NFG by a tuple $(\Pi, U, \mathcal{N})$ where Π^i is player i's set of policies and $i \in \mathcal{N}$ where $\mathcal{N} = \{1,2\}$. For ease of notation, we take player 1 as the training agent and player 2 as its opponent. We use $\Pi = \prod_{i \in \mathcal{N}} \Pi^i$ to denote the set of joint policies (strategy profiles). $U(\pi) : \Pi \rightarrow \Re^n$ is a payoff table of utilities for each joint policy π played by all players. $u^i(\pi)$ denotes the utility value for player i and joint policy π. A player can choose a policy π^i from Π^i or sample from a mixture (meta-strategy) over them $\sigma^i \in \Delta\left(\Pi^i\right)$ where Δ is a probability simplex. In the terminology of game theory, σ^i is a mixed strategy and each policy π^i is a pure strategy.

Each player in the game is assumed to maximise their utility. The most well-known steady-state concept of a game is the Nash equilibrium (NE). NE is a strategy profile π such that no player has an incentive to deviate from its current strategy given the strategies of the other players. Namely, each player's strategy is a best response to others' $\mathcal{BR}(\pi^{-i}) = \arg\max_{\pi^i} u^i(\pi^i, \pi^{-i}) \quad \forall i \in \mathcal{N}$. We call a set of policies ε-best responses to a joint opponents' policy π^{-i}, when there exists an $\varepsilon > 0$, such that $\mathcal{BR}_\varepsilon(\pi^{-i}) = \{\pi^i : u^i(\pi^i, \pi^{-i}) \geq u^i(\mathcal{BR}(\pi^{-i}), \pi^{-i}) - \varepsilon\}$. An ε-Nash equilibrium is a strategy profile that satisfies: $u^i(\pi) \geq \max_{\pi^{i\prime}} u^i(\pi^{i\prime}, \pi^{-i}) - \varepsilon \quad \forall i \in \mathcal{N}$.

6.1.2 Exploitability and Exploitation

To evaluate our learned policy π^1, we use two metrics. An agent's policy's π^1 exploitation of an opponent's policy π^2 is the extra gain obtained by the agent compared to its NE value v^1:

$$\omega(\pi^1, \pi^2) = u^1(\pi^1, \pi^2) - v^1.$$

This measures how much the policy π^1 exploits the weakness of the opponent's policy π^2. However, in general, there is no guarantee that the learned policy π^1 has no weakness. Therefore, we also define the exploitability of a policy π^1 which measures the loss incurred when the agent faces the best opponent policy $\pi^2 = \mathcal{BR}(\pi^1)$ compared to the agent's Nash equilibrium value v^1:

$$\begin{aligned} \varepsilon(\pi^1) &= v^1 - u^1\left(\pi^1, \mathcal{BR}(\pi^1)\right) \\ &= \max_{\pi^{1\prime}} \min_{\pi^2} u^1(\pi^{1\prime}, \pi^2) - \min_{\pi^2} u^1(\pi^1, \pi^2). \end{aligned} \tag{6.1}$$

From Equation 6.1 we can see that the exploitability of a policy is non-negative and represents the distance of policy π^1 to an equilibrium.

6.1.3 Restricted Nash Response (RNR)

Johanson et al. (2008) consider a modified zero-sum game where an opponent has a restricted strategy space $\Pi^2_{p,\pi_{\text{fix}}}$ such that it plays a fixed policy π_{fix} with probability p and plays any possible policy from the original strategy space Π^2 with probability $1-p$. Given (p,π_{fix}), they define a restricted Nash equilibrium as a strategy profile (π^{1*},π^{2*}) such that $\pi^{1*} \in \mathcal{BR}(\pi^{2*})$ and $\pi^{2*} \in \mathcal{BR}_{p,\pi_{\text{fix}}}(\pi^{1*})$, where: $\mathcal{BR}_{p,\pi_{\text{fix}}}(\pi^{1*}) = \arg\max_{\pi^2 \in \Pi^2_{p,\pi_{\text{fix}}}} u^2(\pi^1,\pi^2)$. It is shown that π^{1*} is the best response to π_{fix} among strategies which have equal or lower exploitabilities than π^{1*}, i.e.: $\pi^{1*} = \mathcal{BR}_{\varepsilon}(\pi_{\text{fix}})$, where $\varepsilon = \varepsilon(\pi^{1*})$. Therefore, π^{1*} is called a p-restricted Nash response (RNR) to π_{fix}.

An RNR solution for a normal-form game considers a modified game where the opponent plays a fixed strategy π_{fix} with probability p and any strategy from its original strategy space Π^2 with probability $1-p$. We implement a linear programming solver for RNR and present our formulation as follows:

We use π^1 and π^2 to denote an agent's and its opponent's strategies respectively. Further, let U^{Π} denote the utility table for the agent. Since we consider the zero-sum game, the utility table for the opponent is simply trans$(-U^{\Pi})$ where trans denotes the matrix transpose operation. As the opponent is restricted to play a fixed strategy π_{fix} with with probability at least p, the overall opponent policy satisfies:

$$p\pi_{\text{fix}}(a) \leq \pi^2(a) \leq p\pi_{\text{fix}}(a) + 1 - p \quad \forall\, a \in \Pi^2.$$

We define a vector y_a of size $|\Pi|^2 \times 1$ whose element indices correspond to opponent actions, and we have:

$$y_a(j) = \begin{cases} p\pi_{\text{fix}}(a), & j \neq a \\ p\pi_{\text{fix}}(a) + 1 - p, & j = a. \end{cases}$$

Therefore, for π^1_{RNR}, we solve the following linear programming problem:

$$\begin{aligned} &\max u \\ \text{s.t. } &u \leq {\pi^1}^T U^\Pi y_a \quad \forall\, a \in \Pi^2, \\ &\pi^1(b) \geq 0 \quad \forall\, b \in \Pi^1, \\ &\sum_{\Pi^1} \pi^1(b) = 1. \end{aligned}$$

Similarly, for π^2_{RNR}, we solve the following linear programming problem:

$$\begin{aligned} &\min v \\ \text{s.t. } &v \geq {x_b}^T U^\Pi \pi^2 \quad \forall\, b \in \Pi^1, \\ &\pi^2(a) \leq p\pi_{\text{fix}}(a) + 1 - p \quad \forall\, a \in \Pi^2, \\ &\pi^2(a) \geq p\ \pi_{\text{fix}}(a) \quad \forall\, a \in \Pi^2, \\ &\sum_{\Pi^2} \pi^2(a) = 1, \end{aligned}$$

where we define a vector x_b of size $|\Pi^1| \times 1$ whose each element index corresponds to an agent's action, and we have:

$$x_b(j) = \begin{cases} 0 & j \neq b \\ 1 & j = b. \end{cases}$$

6.2 Dirichlet Process Mixture Opponent Modelling

This section presents our non-parametric Bayesian method for modelling a non-stationary opponent. We consider an opponent's learning process as consecutive transitions from one policy to another such that one opponent can theoretically adopt infinitely many policies during its life-time. Therefore, we propose to use a Dirichlet process (DP) mixture to model the learning process as it has the ability to model an infinite number of clusters (policies in this case) while inferring the current number of policies from the data collected thus far. As our agent interacts with the opponent online, we learn a model with a sequential maximum-a-posteriori approach.

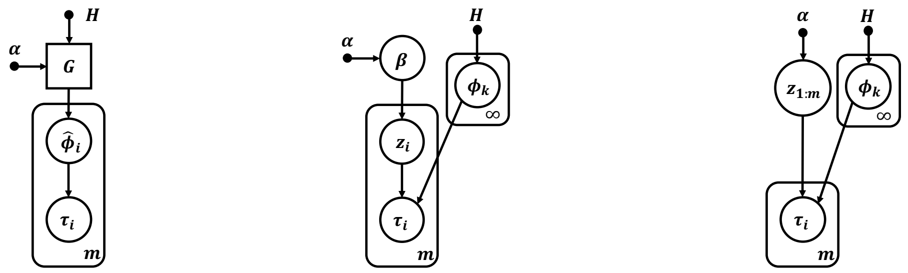
We model an opponent policy as a parameterized function π^2_ϕ and denote the parameter space as Φ. To avoid cluttered notation in this section, we use ϕ to represent the modelled opponent's policy. $\mathbf{DP}(\alpha H)$ is a stochastic process with a concentration parameter α and a base distribution H over Φ. A random draw $G \sim \mathbf{DP}(\alpha H)$ is itself a distribution over Φ, satisfying:

$$(G(A_1), ..., G(A_r)) \sim \mathbf{Dirichlet}\big(\alpha H(A_1), ..., \alpha H(A_r)\big)$$

for every finite measurable partition $A_1, ..., A_r$ of Φ. The full graphical model for opponent modelling is shown in Figure 6.1a. It illustrates a generative process where at step m, the opponent first samples a policy $\hat{\phi}_m \sim G$ and then rolls-out this policy to collect a trajectory τ_m.

To facilitate Bayesian inference, two representations of DP are considered. The stick-breaking representation in Figure 6.1b reveals the discrete nature of G. $G \sim \mathbf{DP}(\alpha H)$ can be constructed as $G = \sum_{k=1}^{\infty} \beta_k \delta_{\phi_k}$ where $\boldsymbol{\beta} \sim \mathbf{GEM}(\alpha)$ is an infinite-dimensional random variable sampled from the Griffiths-Engen-McCloskey (GEM) distribution and $\{\phi_k\}_{k=1}^{\infty}$ are i.i.d. sampled policies from H. At step m, the opponent samples a policy index $z_m \sim \mathbf{Categorical}(\boldsymbol{\beta})$ and rolls-out the policy ϕ_{z_m}. Inference with the stick-breaking representation is required in order to handle the infinite dimensional $\boldsymbol{\beta}$. Therefore, the truncation method (Blei et al., 2006) is commonly used to limit the model capacity to a K mixture and infer the actual number of policies by collapsing redundant ones. This requires tracking all K policies simultaneously and does not adapt well to online settings.

The Chinese restaurant process (CRP) representation in Figure 6.1c can be obtained by integrating out $\boldsymbol{\beta}$. This introduces temporal dependencies between the



(a) DP **(b)** Stick Breaking **(c)** CRP

Figure 6.1: Dirichlet process mixture model

policies, which can be expressed by the conditional distribution:

$$p\big(z_{m+1} = k|z_{1:m}\big) = \begin{cases} \dfrac{\alpha}{m+\alpha}, & k = K_m + 1 \\ \\ \dfrac{|k|_m}{m+\alpha}, & 1 \le k \le K_m \end{cases} \tag{6.2}$$

where $|k|_m = \sum_{i=1}^{m} \mathcal{I}(z_i = k)$ is the total number of trajectories from the k-th policy and K_m is the number of realised policies up until step m. Inference with the CRP representation does not need to handle the infinite dimensional $\boldsymbol{\beta}$. Furthermore, at step m, we only need to track at most m policies ($K_m \le m$) while all policies beyond K_m are independent from the collected trajectories $\tau_{1:m}$, and thus can be discarded from the model. In addition, the temporal dependencies between policies introduced by CRP can be used to develop an online learning algorithm.

Given sampled trajectories $\tau_{1:m}$, the target of our opponent model is to assign each trajectory to a policy and update existing policies with assigned trajectories. This can be achieved by seeking maximum-a-posteriori (MAP) estimations of $z_{1:m}$ and $\phi_{1:K_m}$. To deal with streaming trajectories, the MAP algorithm should operate in an online fashion. Therefore, given the CRP representation, we decompose the posterior into the product of the posterior from the last step, the current priors and the likelihood, which leads to a recursive form:

$$\begin{aligned} p(z_{1:m}, \phi_{1:K_m}|\tau_{1:m}) &\propto \left(\prod_{k=1}^{K_{m-1}+1} p(\phi_k)\right) p(z_1)p(\tau_1|\phi_{z_1}) \left(\prod_{i=2}^{m} p(z_i|z_{1:i-1})p(\tau_i|\phi_{z_i})\right), \\ &\propto p(z_{1:m-1}, \phi_{1:K_{m-1}}|\tau_{1:m-1})p(\phi_{K_m})p(z_m|z_{1:m-1})p(\tau_m|\phi_{z_m}), \end{aligned} \tag{6.3}$$

where $p(\phi_k) = H$ is the base distribution of the DP.

The opponent model either assigns the current trajectory τ_m to a previous policy ϕ_k or creates a new policy $\phi_{K_{m-1}+1}$ to model τ_m. The choice is made according to the MAP trajectory assignment $z_m^* = \arg\max_{z_m} p(z_m, z_{1:m-1}^*|\tau_{1:m})$:

$$p(z_m = k, z_{1:m-1}^*|\tau_{1:m}) \propto \begin{cases} \int_{\phi_k} \alpha p(\phi_k)p(\tau_m|\phi_k)\, d\phi_k, \; k = K_{m-1}+1 \\ p(z_m = k|z_{1:m-1}^*)p(\tau_m|\phi_k^{m-1}) = |k|_{m-1}^* p(\tau_m|\phi_k^{m-1}), \text{ otherwise} \end{cases} \tag{6.4}$$

where $|k|_{m-1}^* = \sum_{i=1}^{m-1} \mathcal{I}(z_i^* = k)$. Here, the hard assignment z_m^* for τ_m is based on previous assignments $z_{1:m-1}^*$ and policies ϕ_k^{m-1}, which is equivalent to applying assumed density filtering (ADF) (Tank et al., 2015) to approximate the true posterior in Equation 6.3 with a Delta distribution $\delta(z_{1:m}^*)$. The hard assignment prevents creating a new policy at each step if τ_m is assigned to an existing policy, which significantly reduces the memory usage. Furthermore, the MAP estimations for all existing policies, except $\phi_{z_m^*}$, remain unchanged, which dramatically accelerates the algorithm. We then optimise the policy $\phi_{z_m^*}$ by maximising the likelihood of all trajectories assigned to it:

$$\phi_{z_m^*}^m = \arg\max_{\phi_{z_m^*}} \log p(\phi_{z_m^*}) \prod_{z_i^*=z_m^*} p(\tau_i|\phi_{z_m^*}). \tag{6.5}$$

Where finding the global optimum is not tractable in non-conjugate cases, we take gradient steps to update $\phi_{z_m^*}^n$ as

$$\phi_{z_m^*}^n = \phi_{z_m^*}^{m-1} + \lambda \nabla_{\phi_{z_m^*}} \log p(\phi_{z_m^*}) \prod_{z_i^*=z_m^*} p(\tau_i|\phi_{z_m^*}).$$

Our streaming MAP algorithm for opponent modelling fits into the general expectation-maximisation (EM) framework. In the E-step, we seek a Delta distribution $q(z_{1:m}) = \delta(z^*_{1:m})$ to approximate the posterior $p(z_{1:m}|\tau_{1:m})$. Therefore, $z^*_{1:m}$ are the MAP trajectory assignments. Following Equation (6.3), $p(z_{1:m}|\tau_{1:m})$ can be obtained by integrating out $\phi_{1:K_m}$:

$$\begin{aligned} p(z_{1:m}|\tau_{1:m}) &= \int_{\phi_{1:K_m}} p(z_{1:m}, \phi_{1:K_m}|\tau_{1:m})\, d\phi_{1:K_m} \\ &\propto \int_{\phi_{1:K_m}} p(z_{1:m-1}, \phi_{1:K_{m-1}}|\tau_{1:m-1}) p(\phi_{K_m}) p(z_m|z_{1:m-1}) p(\tau_m|\phi_{z_m})\, d\phi_{1:K_m}. \end{aligned} \tag{6.6}$$

To solve the E-step in an online fashion, we apply assumed density filtering (Tank et al., 2015) and approximate $p(z_{1:m-1}, \phi_{1:K_{m-1}}|\tau_{1:m-1})$ in Equation (6.6) with $q(z_{1:m-1})q(\phi_{1:K_{m-1}}) = \delta(z^*_{1:m-1})\delta(\phi^{m-1}_{1:K_{m-1}})$. Therefore, $q(z_{1:m})$ is computed recursively by reusing $q(z_{1:m-1})$ and the policies $\phi_{1:K_{m-1}}$ are fixed to their latest value $\phi^{m-1}_{1:K_{m-1}}$. Since a new policy may be created, we also need to integrate over $\phi_{K_{m-1}+1}$ to incorporate the possibility that $K_m = K_{m-1}+1$. In the M-step, we optimise the policies $\phi_{1:K_m}$ to maximise $\mathbb{E}_{q(z_{1:m})}[\log p(\tau_{1:m}, z_{1:m}|\phi_{1:K_m})]$ with an extra regularisation term, $\log p(\phi_{1:K_m})$, introduced by the policy prior. The whole procedure is summarised in Algorithm 5. For the initial step, the trivial solution is given by: $z^*_1 = 1, \phi^1_1 = \arg\max_{\phi_1} p(\phi_1)p(\tau_1|\phi_1)$.

The original CRP in Equation 6.2 encapsulates a prior that the distribution of the next policy mimics the empirical policy distribution from the history. This prior is not consistent with our knowledge of the policy evolution process since a new opponent policy is commonly updated from the previous one. Therefore, we adopt a sticky variant in Equation 6.7 to incorporate the belief that the opponent tends to persist in the latest policy (Fox et al., 2008; Xu et al., 2020).

$$p(z_m = k|z_{1:m-1}) = \begin{cases} \dfrac{\alpha}{m-1+\alpha+\kappa}, & k = K_{m-1}+1 \\ \\ \dfrac{|k|_{m-1} + \kappa\hat{\delta}(K_{m-1}, k)}{m-1+\alpha+\kappa}, & 1 \le k \le K_{m-1} \end{cases} \tag{6.7}$$

Algorithm 5 Streaming MAP for Opponent Modelling

```
  **Initial Step:**
  Solve $z_1^*, \phi_{z_1}^1 = \arg\max_{z_1^*, \phi_{z_1}} p(\phi_{z_1}) p(z_1) p(\tau_1 | \phi_{z_1})$.
  **for** $m = \{2, 3, \dots\}$ **do**
    **Update** $z_{1:m}^*$ **(E-Step)**:
    Maximum-a-posterior (MAP) trajectory assignments $z_{1:m}^*$:
    $$\begin{aligned} z_{1:m}^* &= \arg\max_{z_{1:m}} p(z_{1:m} | \tau_{1:m}) \\ &\approx \arg\max_{z_{1:m}} \int_{\phi_{1:K_{m-1}+1}} q(z_{1:m-1}) q(\phi_{1:K_{m-1}}) p(\phi_{K_{m-1}+1}) p(z_m | z_{1:m-1}) p(\tau_m | \phi_{z_m}) d\phi_{1:K_{m-1}+1} \end{aligned}$$
    $z_{1:m-1}^*$ remains unchanged and
    $$z_m^* = \arg\max_{z_m} \int_{\phi_{1:K_{m-1}+1}} q(\phi_{1:K_{m-1}}) p(\phi_{K_{m-1}+1}) p(z_m | z_{1:m-1}^*) p(\tau_m | \phi_{z_m}) d\phi_{1:K_{m-1}+1}$$
    which corresponds to Equation (6.4). Then $q(z_{1:m}) = \delta(z_{1:m}^*)$ and $K_m$ is set according to $z_m^*$.
    **Update** $\phi_{1:K_m}^m$ **(M-Step)**:
    $$\begin{aligned} \phi_{1:K_m}^m &= \arg\max_{\phi_{1:K_m}} \mathbb{E}_{p(z_{1:m} | \tau_{1:m})} [\log p(\tau_{1:m}, z_{1:m} | \phi_{1:K_m})] + \log p(\phi_{1:K_m}) \\ &\approx \arg\max_{\phi_{1:K_m}} \int_{z_{1:m}} q(z_{1:m}) \log \prod_{i=1}^{m} p(\tau_i | \phi_{z_i}) dz_{1:m} + \log \prod_{k=1}^{K_m} p(\phi_k) \\ &= \arg\max_{\phi_{1:K_m}} \log \prod_{k=1}^{K_m} p(\phi_k) \prod_{i=1}^{m} p(\tau_i | \phi_{z_i^*}) \end{aligned}$$
    For $k \neq z_m^*$, $\phi_k^m = \phi_k^{m-1}$ and $\phi_{z_m^*}^m = \arg\max_{\phi_{z_m^*}} \log p(\phi_{z_m^*}) \prod_{z_i^* = z_m^*} p(\tau_i | \phi_{z_m^*})$ which corresponds to Equation (6.5).
  **end for**
```

where $\kappa \geq 0$ is a 'stickiness' parameter and $\hat{\delta}$ is the Kronecker delta function.

Following Equation 6.4, the probability of creating a new policy for τ_n is given by:

$$p(z_m^* = k) \propto \int_{\phi_k} \alpha p(\phi_k) p(\tau_m | \phi_k) \, d\phi_k, \tag{6.8}$$

where $k = K_{m-1} + 1$. We use a Monte Carlo method to estimate Equation 6.8 by sampling new policies from $p(\phi_k)$. However, sampling new policies from a data-independent prior $p(\phi_k)$ is likely to yield a low trajectory likelihood $p(\tau_m|\phi_k)$, which prevents the new policy creation. Therefore, we update the sampled policies to increase the likelihood $p(\tau_m|\phi_k)$ by taking a few gradient steps before estimating the integration in Equation 6.8.

According to Equation 6.7, the CRP prior encourages the opponent model to create redundant policies at the early stage when the number of trajectories n is small and α dominates. Redundant policies could hurt the algorithm's performance as it incurs extra cost in terms of computation and memory. Trajectories from the same ground truth policy could be assigned to different ϕ_ks and these assignments never revisited. Therefore, an error correction mechanism has to be introduced. Here, we adopt a symmetric distance metric between two policies and develop a policy merge procedure based on the metric. Given a set of states $\mathcal{S}$, we define $d(\phi_k, \phi_j) = \mathbb{E}_{s \sim \textbf{Uniform}(\mathcal{S})} \left[\mathcal{JS}\Big(\phi_k(\cdot|s) || \phi_j(\cdot|s)\Big)\right]$, where $\mathcal{JS}(\cdot||\cdot)$ is the Jensen–Shannon divergence and $\phi_k(\cdot|s)$ is the action distribution given state s under the policy ϕ_k. When the distance between two policies is below a pre-defined threshold η, the merge procedure simply re-assigns all trajectories of ϕ_k to ϕ_j.

With the opponent model developed in this section, at step m, we can construct an opponent policy set $\tilde{\Pi}^2 = \{\phi_k^m\}_{k=1}^{K_m}$ and a distribution $\tilde{\sigma}^2$ over $\tilde{\Pi}^2$. The distribution $\tilde{\sigma}^2(\phi = \phi_k) \propto |k|_m + \kappa\hat{\delta}(K_m, k)$ is essentially the empirical distribution of $z_{1:m}^*$ altered by the stickiness factor κ.

6.3 Exploit Policy-Space Opponent Model

In this section, we present how to learn a safe best response to this meta strategy, given a predicted distribution $\tilde{\sigma}^2_\phi$[1] over opponent's policies. The advantages of our approach of focusing on policy space are two-fold: first, we do not need to assume the access to the opponent's learning characteristics such as its training algorithm, its neural network's architecture or its update frequency; we only require past trajectories. Additionally, the distribution of the types of opponent policy $\tilde{\sigma}^2$ gives us an approximate stable overview of the current opponent's playing behaviour compared to the opponent's current policy whose updates greatly depend on the opponent's learning characteristics and randomness from playing (e.g. exploration behaviour) and training (e.g. stochastic gradient descent). Therefore, learning a response to this meta-strategy $\tilde{\sigma}^2$ will rely on less prior knowledge about the opponent's learning characteristics and is more robust to noise.

However, there is no guarantee that our learned meta strategy σ^1 has no (or at least low) exploitability. It has been shown that overfitting to an opponent strategy $\tilde{\sigma}^2$ often renders the resulting learned strategy brittle (Johanson et al., 2008; Lanctot et al., 2017; Wu et al., 2021). Such a brittle strategy performs badly when playing against different opponent strategies $\tilde{\sigma}^{2\prime}$. Therefore, a more desirable goal is to learn a safe best response to an opponent meta-strategy $\tilde{\sigma}^2$. RNR solutions consider cases where the game to solve is fixed and known and the opponent's policy is stationary. However, when we consider non-stationary opponent exploitation on policy space, the size of the meta-game to solve increases with the number of interactions between the training agent and the adaptive opponent. Furthermore, each player is free to learn and update its policy at any time point during the process.

To address the above issues, we combine DO with RNR to solve a meta-game built from EGTA where the opponent's policies are predicted by the opponent model from Section 6.2. Pseudo-code explaining our approach is presented in Algorithm 6. We maintain a utility table $U^{\tilde{\Pi}}$ wherein rows represent learned policies for the training agent and columns represent a modelled policy of the opponent respectively. An

[1]To simplify our notation, we will ignore the subscript ϕ henceforth.

Algorithm 6 Exploit Policy-Space Opponent Model (EPSOM)

```
Input: Hyper-parameter $p, H, E$, an adaptive opponent $-i$
Output: Policy $\pi^i_{1,\dots,E}$ and meta-policy $\sigma^i$
Initialise learning agent $i$'s policy $\pi^i_0$
Initialise a memory buffer $\mathbf{B}$
Initialise opponent meta-policy $\sigma^{-i}(\cdot) = 1$
for epoch $e$ in $\{1,2,\dots,E\}$ do
    for episode $h \in \{1,2,\dots,H\}$ do
        Play an episode against the opponent with strategy $\sigma^1_{RNR}$
        Collect the trajectory $\tau_{e,h}$ and save them into $\mathbf{B}$
    end for$\tilde{\sigma}^2, \tilde{\Pi}^2 = \text{opponent\_modelling}(\mathbf{B})$
    $\bar{p} = \frac{1}{|\tilde{\Pi}^2|}\sum_j p^j \tilde{\sigma}^2(j)$
    compute missing entries in $U^{\tilde{\Pi}}$ from $\tilde{\Pi} = \Pi^1 \times \tilde{\Pi}^2$ by simulations
    $\_, \sigma^2_{RNR} = \text{RNR\_solver}(U^{\tilde{\Pi}}, \bar{p}, \tilde{\sigma}^2)$
    for episode $h \in \{1,2,\dots,H\}$ do
        Sample $\tilde{\pi}^2$ from $\sigma^2_{RNR}$
        Train oracle $\pi^1$ over $\rho \sim \left(\pi^1, \tilde{\pi}^2\right)$
    end for$\Pi^1 = \Pi^1 \cup \{\pi^1\}$
    Compute missing entries in $U^{\tilde{\Pi}}$ from $\tilde{\Pi} = \Pi^1 \times \tilde{\Pi}^2$ by simulations
    $\sigma^1_{RNR}, \_ = \text{RNR\_solver}(U^{\tilde{\Pi}}, \bar{p}, \tilde{\sigma}^2)$
end for
```

epoch is defined as a fixed amount of episodes of games where we play against the opponent holding our strategy σ^1 fixed. At each epoch, we run our opponent model to predict the current distribution $\tilde{\sigma}^2$ of the opponent's policies. If a new policy is detected, we will add it into $\tilde{\Pi}^2$. Given $\tilde{\sigma}^2$, we run a p-RNR solver to obtain the opponent's RNR meta-strategies σ^2_{RNR} which is a restricted Nash solution to the current meta-game assuming that the opponent is playing according to $\tilde{\sigma}^2$ with probability at least p. Then we train an (approximate) best-response policy to σ^2_{RNR} and add the new policy into Π^1. We re-run a p-RNR solver to obtain our RNR meta-strategies σ^1_{RNR} which we use to mix the policies in population Π^1 for the next epoch's playing policy.

When a new type j of modelled policy $\pi^{2,j}$ is added by our opponent model, we initialise a p-value $p^j = p_{init}$ to this type. Its p-value is incremented proportionally to the probability that the opponent plays this policy in the following epochs, $\tilde{\sigma}^2(j)$, and clipped at 1. At each epoch, we calculate the average p-value $\bar{p} = \frac{1}{|\tilde{\Pi}^2|}\sum_j p^j \tilde{\sigma}^2(j)$ for solving current RNR strategies. In the extreme case where $\bar{p} = 0$, σ^i_{RNR} is the same as the Nash strategy in the current meta game. At another extreme where $\bar{p} = 1$, $\sigma^2_{RNR} = \tilde{\sigma}^2$ and $\sigma^1_{RNR} = \mathcal{BR}(\tilde{\sigma}^2)$. Therefore, when we have low confidence in $\tilde{\sigma}^2$ ($\bar{p}$ is low), we learn an approximate best response to opponent's current Nash mixture which will enlarge our current empirical gamescape (Balduzzi et al., 2019) and thus help to find strategies with lower exploitability. At the same time, the training agent becomes risk-adverse and the next epoch strategy σ^1_{RNR} becomes a strategy closer to the Nash strategy of the current meta game.

When $\bar{p}$ is high, it means that our opponent model has high confidence that the opponent is playing $\tilde{\sigma}^2$ and an approximate best counter strategy to $\tilde{\sigma}^2$ will be added into our policy population. The training agent becomes profit-driven and σ^1_{RNR} becomes a strategy closer to $\mathcal{BR}(\tilde{\sigma}^2)$ in the next epoch. Therefore, mixing the playing policy by σ^1_{RNR} flexibly switches the agent between risk-adverse and profit-driven depending on the confidence of the opponent model. In contrast with previous RNR solution, EPSOM can always recover a strategy with approximately the lowest exploitability it has seen so far as we maintain a population of policies.

6.4 Experiments

In this section, we empirically investigate whether the proposed method can (1) exploit an unknown non-stationary opponent while still maintaining a strategy with low exploitability, (2) improve its performance by continued training against different opponents and (3) exploit previously unseen opponents without further training. Variants of Poker offer a rich arena for developing artificial intelligence. The games feature stochasticity, partial observability and competitive dynamics with unknown adversaries. In this work, we conduct experiments and evaluation in a simplified Poker game, Kuhn Poker (Kuhn, 1950). This variant of poker is amenable to game theoretic analysis whilst retaining all of the elements of the more challenging larger scale poker games. We use an agent learning to play using the PPO (Schulman et al., 2017) algorithm as our opponent, which suffices to provide a non-stationary setting. Our hyperparameter values are presented in Table 6.1.

Settings	**Value**	**Description**
	EPSOM	
Oracle method	Analytical best response	subroutine of getting oracles
d	Every 64 episodes of a game	Update frequency for EPSOM
Meta-solver	Linear Programming Solver	Meta-solver method
	Dirichlet process mixture opponent model	
p_{init}	0.1	initial p value for a new policy
p_{step}	0.05	p value increment when a new trajectory is assigned to the policy
α	1.0	concentration parameter of the Dirichlet Process
κ	1.0	'stickiness' factor for modified CRP prior
θ	5.0	std of the policy base distribution: $p(\boldsymbol{\phi}_k) = \mathcal{N}(0, \theta^2 I)$
η	0.1	threshold for merging two policies into one
	PPO Opponent	
Learning rate	0.0003	Learning rate for PPO
Optimizer	Adam	Gradient ascent optimizer
NN architecture	$12 \times 64 \times 64 \times 2$	Neural network architecture
Mini batch size	128	Mini batch size for SGD
Update frequency	Every 128 steps	Opponent update after every 128 steps
Update epoch	20	Training epochs in an update
Clip ratio	0.2	PPO clip ratio
γ	0.99	Discount factor
λ	0.97	Lambda-return factor
	Kuhn Poker	
Observation Dimension	12	12 combinations for self hand and game history
Action Space	2	Pass or Bet

Table 6.1: Hyper-parameter settings.

Kuhn Poker (Kuhn, 1950) is a simple, zero-sum two-player imperfect information game. The deck of cards is limited to simply a Jack, a Queen and a King with no notion of suits. Ordering is as usual: Jack $<$ Queen $<$ King. If the game reaches a showdown, the player with the highest card wins. If either player folds at any time they lose the round and their opponent takes the entire pot. The game opens with a round of antes of 1. Then each player is dealt a single card and the remaining card is placed face down. Once the deal is complete it is time for the first round of bidding: Player 1 may check (no bet) or bet 1. If Player 1 bet Player 2 may call the bet or fold. If Player 2 calls there is a showdown for the pot of 4, if they fold Player 1 wins the pot. If Player 1 checked, Player 2 may check or bet 1. If both players check then there is a showdown for the pot of 2. If Player 2 bets, following a check by Player 1, then Player 1 can either fold or call. If player 1 calls there is then a showdown for the pot of 4. The starting player may alternate or be chosen at random for each deal. Kuhn Poker has the second-mover advantage, i.e., the second player to bet (Player 2 above) will win in expectation when both players play the best response to each other. To remove this advantage, we alternate the playing turn between our agent and the opponent after every episode of a game. In this simple game, we do

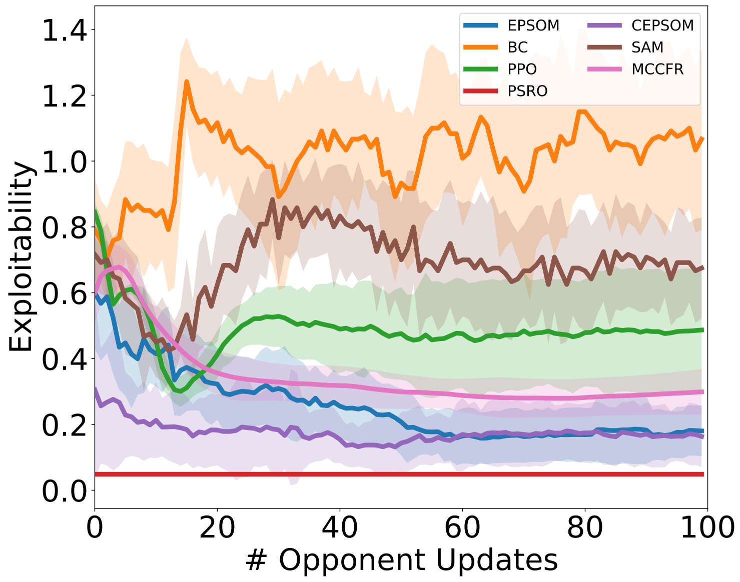


Figure 6.2: Exploitability of different algorithms against a non-stationary opponent implemented by PPO in Kuhn Poker.

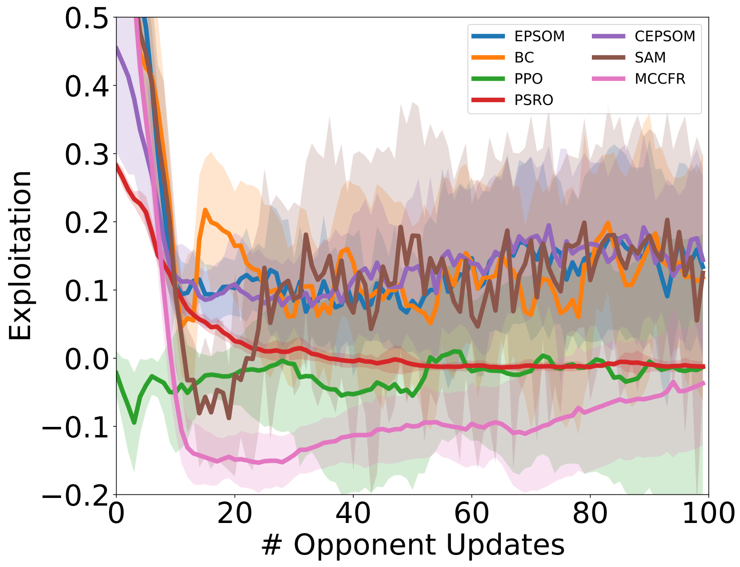


Figure 6.3: Exploitation of different algorithms against a non-stationary opponent implemented by PPO in Kuhn Poker.

not discriminate between the Pass and Fold actions, and thus, each player need only choose from Pass or Bet.

We select 5 representative algorithms as baselines. PSRO is a popular algorithm which guarantees the convergence to an approximate NE. As PSRO is a self-play algorithm, its is trained before playing against any adaptive opponents. Be-

Kuhn Poker						
	PPO		TRPO		A2C	
EPSOM [0.109]	0.037	(0.042)	0.097	(0.061)	0.187	(0.075)
CEPSOM [0.080]	**0.114**	(0.022)	**0.115**	(0.037)	**0.187**	(0.029)
BC [1.333]	−0.562	(0.011)	−0.276	(0.126)	−0.148	(0.171)
PSRO [0.000]	0.050	(0.014)	0.030	(0.006)	0.086	(0.042)
PPO [0.477]	−0.405	(0.014)	−0.358	(0.031)	−0.347	(0.036)
SAM [0.5]	−0.270	(0.003)	−0.135	(0.010)	−0.107	(0.009)
MCCFR [0.28]	−0.154	(0.023)	−0.138	(0.027)	0.115	(0.068)

Table 6.2: Zero-shot learning exploitation results. Trained agents (row players) play against adaptive opponents (column players). Adaptive opponents are allowed to update 100 times and a trained agent's average exploitation are taken over these 100 updates and 20 random seeds. Values in square brackets are each trained agent's exploitability and values in parentheses are stds taken over 20 random seeds.

haviour cloning (BC) models the opponent's policy by taking maximum likelihood estimation of history trajectories stored in a sliding-window buffer and learns an (approximate) best-response policy to it. PPO represents a canonical choice among many SARL algorithms. Switching Agent Model (SAM) (Everett and Roberts, 2018) deals with the non-stationarity problem by switching between different opponent models. We also consider Counterfactual Regret Minimisation (CFR) (Zinkevich et al., 2007a) as another baseline. However, because all other baselines have no direct access to their opponent's policy, to have a relatively fair comparison, we adopt MCCFR (Lanctot et al., 2009), the Monte Carlo sampling version of CFR. Even in this setting, MCCFR still has advantages as it can query its opponent to sample actions for its update which is not allowed in our other experiments. In our work, as agents and their opponents update asynchronously, we always evaluate each algorithm's performance right after the opponent's update for a more robust evaluation. The following results reported with mean and standard deviation (std) are all obtained by repeating the corresponding experiment over 20 random seeds.

As shown in Figure 6.2 and 6.3, EPSOM can achieve a safe strategy with relatively low exploitability while still being able to exploit its opponent. Though PSRO plays a strategy with the lowest exploitability (≈ 0) it also has very low exploitation against its opponent. In contrast, BC can exploit its opponent to a similar extent as

EPSOM but it comes with the cost of high exploitability. The PPO algorithm has large variance and performs badly on average in this non-stationary setting. Similar to BC, SAM has high exploitability when it is trained against a PPO opponent. Its exploitation to the PPO agent also varies greatly. MCCFR trained against adaptive PPO can reduce its exploitability to a relatively low level (0.28) but is limited to exploit the PPO opponent, especially considering that we only evaluate how much a baseline can exploit the PPO opponent after every time the PPO is updated. We also test a continual learning version of EPSOM which we name CEPSOM. It is implemented by training an EPSOM agent against 5 different opponents without re-initialisation thereby building up a richer set of modelled opponent policies and a more robust best-response policy population. Its average performance over these opponents is also reported in Figure 6.2 and 6.3. In our experiments, we use an analytical method to calculate a best response to a given policy.

Next, we test these agent's performance against three adaptive opponents implemented by PPO, TRPO (Schulman et al., 2015) and A2C (Mnih et al., 2016) without further training and results are presented in Table 6.2. According to the Table 6.2, relying on an opponent model to predict the current opponent's policy type and flexibly adjusting its playing strategy accordingly, CEPSOM achieves the highest average exploitation against adaptive opponents. EPSOM also obtains positive average exploitation but with lower values, since EPSOM has only ever been trained with one opponent. PSRO plays a safe strategy and performs only slightly better than EPSOM in terms of opponent exploitation when against PPO opponent. BC, PPO and SAM perform badly as they overfit to one opponent, and thus they are exploited by other adaptive opponents. Trained MCCFR also perform badly against strong adaptive opponents (PPO and TRPO) but it obtains positive exploitation against A2C, a weaker adaptive opponent. Note that, in this zero-shot learning tournament, although we do not train EPSOM and CEPSOM, they still need to predict the opponent's policy and solve the meta-game for a RNR solution given the prediction.

In Figure 6.4, we want to visualise and compare an RL agent's learning pro-

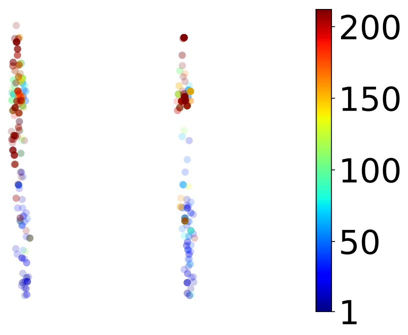


Figure 6.4: Opponent's learning process modelled by CEPSOM: (left) CEPSOM adjusts playing strategy online; (right) CEPSOM plays an approximate Nash strategy.

cess in stationary and non-stationary environments respectively. To this end, we first create a stationary environment by fixing our trained agent CEPSOM to always play a Nash equilibrium strategy. Then from the perspective of the PPO agent, the environment becomes stationary. To create a non-stationary environment, we allow the trained CEPSOM to adjust its meta-strategy online based on the prediction of its opponent model. Creating a non-stationary opponent this way also allows us to understand how CEPSOM's online adaptation affects the opponent's learning process. To visualise the learning process, we rely on our opponent model. During the PPO agent's training, our opponent model will predict its type based on recent trajectories produced by PPO. Then we use the t-SNE method to present these predictions in a 2-D diagram. As we can see from Figure 6.4, when the environment is stationary, PPO can learn and gradually converge towards the top side of the plot (right in Figure 6.4) However, when the environment is non-stationary, because CEPSOM adapts online, the PPO agent struggles to converge and bounce between the top and middle part of the plot (left in Figure 6.4).

6.5 Conclusion

In this work, we propose a framework for training an agent to safely exploit its opponent. Compared to RNR and its variants, our work focuses on non-stationary opponents. We consider the opponent's learning as a series of policy transitions and model such a process by a Dirichlet Process. Safe exploitation means that an agent can exploit an opponent's weakness to maximise our utility while simultaneously maintaining a strategy which has low exploitability. This property is desirable as naively overfitting to one type of opponent could easily lead to exploitation by other opponents. We empirically verify our algorithm's performance on Kuhn Poker, a simplified version of Poker.

Opponent modelling based MARL algorithms typically require extra computation for learning a good opponent model. This cost often scales dramatically with the number of opponents, action space dimensionality and the complexity of the problem. It can be a heavy burden on an agent if it learns an opponent model from scratch online. Therefore, a more realistic way for utilising the power of an opponent model is offline training and online prediction. We build CEPSOM based on this idea where we train one EPSOM agent across different opponents and aggregate knowledge by maintaining a never-reinitialised opponent model and policy population. Our experiment results show that CEPSOM can achieve high exploitation against a new adaptive opponent without further training, outperforming other representative baselines from SARL and MARL. In complex competitive games, a strong player can often encounter sub-optimal opponents and playing a Nash strategy can potentially forego significant profit. EPSOM, alongside many prior works, shows the potential of an opponent modelling based approach for solving this problem, and our preliminary results from CEPSOM demonstrate the possibility of a trained agent beating an as yet unseen adaptive opponent.

EPSOM is limited by its computation and memory complexity. Naively applying EPSOM to more complex problems requires a great amount of resources. To alleviate this problem, we introduce policy merge to remove redundant policies in our opponent model. This approach could be improved by applying game theoretic analysis to our policy populations (agent's self policies and modelled opponent policies). We leave the study of improving EPSOM's scalability to future work.

Chapter 7

Conclusions and Future Work

In this thesis we look into three challenges in multi-agent reinforcement learning (MARL), i.e., the *non-stationarity*, the *partial observation* and the *unclear learning objective*. We observe that these issues share a common cause which is the lack of knowledge of the opponents. Specifically, if we knew the exact actions opponents will take, we could integrate them into the environment's states and the *non-stationarity* problem would disappear. If we knew what our agent's opponents observe, the *partial observation* related to private information would be solved. Finally, if we knew who our opponents would be, our objective would become clear and that is to train an agent which can best respond to the opponents. However, in real problems, we rarely have an oracle to tell us this critical information about our opponents. One approximate solution would be to learn a model which can reason about our opponents, which is commonly known as opponent modelling. Based on this observation, we proposed to solve the aforementioned problems with opponent modelling based approaches. We investigated solutions for the above challenges under different conditions. Based on the characteristics of different settings, we proposed different opponent modelling based methods accordingly.

7.1 Contributions

In Chapter 4, we consider the *non-stationarity* problem in cooperative games. The *non-stationarity* problem arises from opponents' adaptation. Therefore, to solve this problem, we could build an opponent model which can predict what opponents' policies will be after their adaptation at the current time step. This is generally a difficult problem which normally requires a lot of prior knowledge about the opponents[1] such as its learning rate, its learning algorithm, its network's architecture. However, in a cooperative game, we observe that it is reasonable to assume that the opponent is also learning and adapting towards maximising the shared rewards. This assumption provides us an approximate direction that our opponents' policies are updated. Therefore we proposed an algorithm named ROMMEO which is based on the observation. It is derived by formulating the reward-maximisation problem in cooperative games as a Bayesian inference problem. We first derive the evidence lower bound (ELBO) for our objective and propose two methods for optimising the ELBO, one exact (ROMMEO-Q) and one approximate (ROMMEO-AC). We test our methods on the challenging matrix game and differential game and show that they can outperform a series of strong base lines.

In Chapter 5, we look into the *partial observation* problem in cooperative games. As researched in many other prior works, an effective way to resolve partial observation caused by players having private information in cooperative settings is to train players to communicate. However, in contrast with previous works, we focus on the setting where explicit communication is not allowed. Therefore, we proposed an algorithm named PBL which trains agents to communicate implicitly by actions. This algorithm sets a training framework which iterates the training between a policy and belief network. The belief network learns to predict the opponent's hand by the observed bidding history. The policy learns to bid the optimal contract given its hand and the prediction about its opponent's hand from the belief network. As players cannot communicate about their own hands explicitly, the policy also needs to learn to communicate about this information by actions implicitly. Part of our work's novelty comes from the invention of the communication reward. It is an auxiliary reward which encourages the policy to take actions which convey information to the opponents. We empirically demonstrate that our methods can achieve near optimal performance in a matrix problem and scale to complex prob-

[1]We assume our opponent is also a deep reinforcement learning algorithm.

lems such as contract bridge bidding. We con-duct an initial investigation of the further development of machine theory of mind. Specifically, we enable an agent to use its own belief model to attribute mental states to others and act accordingly. We test this framework and achieve some initial success in a multi-agent particle environment under distributed training.

In Chapter 6, we study the *non-stationarity* and *unclear learning objective* in zero-sum games. We propose to consider the opponents' non-stationary learning process as transitions from one policy type to another. Then we can model these transitions by a Dirichlet process such that we could have the theoretically infinite number of policy types. To the best of our knowledge, this is the first work which handles non-stationary opponent modelling by the Dirichlet process. For defining the learning objective, we consider the RNR solution concept proposed in (Johanson et al., 2008) and further studied in (Johanson and Bowling, 2009; Ponsen et al., 2014; Bard et al., 2013). Our contribution to this topic is that we take into account the non-stationarity induced by opponents when we try to optimise for this trade-off. Most prior works only consider how to exploit one stationary opponent while keeping the agent itself with low exploitability. However, in realistic settings, we need to consider how to exploit opponents which change over time while maintaining low exploitability. Changes can come from opponents also having the ability to learn and adapt or opponents that are different in different tournaments or both. Therefore, our work extends previous works to the non-stationarity setting which is more realistic and therefore more significant to the progress of research in this area. Our empirical results show that our agents can learn a safe policy while still exploiting the non-stationary opponents during its training. Once trained and fixed, our agents can still exploit different adaptive opponents from ones seen in the training time by adjusting its meta-strategy according to the prediction of its opponent model.

7.2 Future Work

Our works are well motivated as they target classical challenges in MARL and novel in the sense that we consider the settings or approaches with characteristics which are rarely discussed in prior works. However, two common problems associated with opponent modelling methods are not discussed well in this thesis, which are the scalability to multiple opponents and the expensive computation cost. These theoretical or empirical limitations illuminate our future research works' directions and we will discuss them in detail in the rest of this section.

7.2.1 Scalability to Multiple Opponents

Works introduced in this thesis empirically investigated problems where there is only one opponent, but an agent normally has to interact with multiple opponents concurrently in real problems. Therefore, strong MARL algorithms which can be applied to solve real-world problems are expected to have the scalability to multiple opponents (Yang and Wang, 2020; Zhang et al., 2019). ROMMEO and EPSOM study problems whose formulation is not limited to only one-opponent setting. However, extending these two works to multiple opponents setting is non-trivial. As with many other works, they need to build an opponent model which predicts joint actions of the agent's opponents. However, the dimension of the opponents' joint actions space grows exponentially with the number of opponents (Jonsson and Rovatsos, 2011). This is also known as the combinatorial nature of MARL (Hernandez-Leal et al., 2019). It greatly increases the computation cost, the number of training data and the difficulty of learning an accurate opponent model. A typical approach to this problem is to additionally assume that there exists a factorised structure among agents or introduce additional assumption about opponents' impact on an agent's play (Crosby et al., 2013; Yang et al., 2018). For example, if we assume each of the opponents learn and act independently, then we could model these opponent independently with fully separate models. This strong assumption can reduce the exponential complexity to polynomial complexity, but it is not realistic. Another promising direction is to observe that an agent may only need to attend a very limited number of opponents rather than all opponents. An illustrative example would be autonomous driving where an agent only need to reason about other vehicles near it. Therefore, we could learn an additional attention model (Vaswani et al., 2017) which can help the agent to identify important opponents the agent should attend to.

For approaches focusing on the communication between agents such as PBL, the problem is more complex because they not only suffer from the exponentially growing complexity but also need to consider how to communicate to multiple opponents effectively. For instance, recall we calculate the communication reward in PBL by comparing how much the prediction of the opponent's model becomes closer to an agent's private information before and after the opponent observing the corresponding action. Therefore, if there are multiple opponents in one system, we need to consider which opponent's model to use for calculating an agent's communication reward. A natural solution we could look into for this problem would be to introduce the common knowledge (Schroeder de Witt et al., 2019) mechanism to agents in a system. Then an agent's learning objective for communication skills would be to convey information in the way that an opponent with the common knowledge would be able to understand, which mimics how humans learn to communicate.

7.2.2 Expensive Computation Cost

Methods introduced in this thesis build separate models to predict information about the opponents. The training of the models and inference from the models will both incur extra computation cost. To obtain an opponent model with adequate accuracy in complex problems, a prohibitively large number of training examples are normally required (He et al., 2016). Therefore, to learn an opponent model online while playing against the modelled opponents is challenging. Furthermore, when we have an accurate opponent model, learning an approximate best response to the model will also needs many training samples as we know that deep RL algorithms often suffer from poor sample efficiency (Bard et al., 2013). Hence, we could see that opponent model based approaches often have expensive computation costs and it is normally unrealistic to deploy the algorithm online.

When we consider fully cooperative problems, we might assume that agents trained together will also be tested together so that no extra opponent models and the corresponding adaptation are required online in test time. However, this is not always true in real problems (Hu et al., 2020). For example, rescue teams trained in different organisations may need to coordinate together immediately to conduct space search and rescue tasks when a natural disaster happens. In competitive games, it is much less likely to assume the opponents seen in test time will be the same as ones seen in training time. Therefore, we believe enabling agents to reason and adapt their opponents efficiently online in test time is an important problem for our future work.

A promising solution would be to move the heavy computation of an opponent model based approach offline and only do computationally light inference online. CEPSOM introduced in Chapter 6 is motivated by this intuition. Specifically, before we start a real tournament and play against real opponents, we first build a population of opponent models and a population of the corresponding robust response policies by playing CEPSOM against 'pseudo' opponents created by us. Then in the real tournament, we do not need to train CEPSOM but only solve RNR solutions for the meta-game online given the predictions of our trained opponent models. We effectively trade space complexity for time complexity. Similar idea is also studied in (Bard et al., 2013). However, CEPSOM is only verified empirically on small scale problems. When applied to large scale problems where opponents have widely varied strategies, naively adding opponent models or response policies into a population may soon exhaust our space resources which leads to poor performance. Therefore, introducing measures such as diversity (Nieves et al., 2021) and relative population performance (Balduzzi et al., 2019) to keep the populations diverse and representative is a future direction we will look into.

7.2.3 A General Solution

The *non-stationarity*, *partial observation* and *unclear learning objective* problems share the same cause which is the training agent has limited knowledge of its opponents. Therefore, one would intuitively expect we could solve all of these problems together with one solution. However, in this thesis, we propose three different opponent modelling based algorithms to solve these problems individually or a combination of two. Moreover, these algorithms can only be applied to or have only be verified in specific settings. Specifically, ROMMEO and PBL are only considered in cooperative setting in the thesis. EPSOM, on the other hand, is only applicable to zero-sum games. Therefore, a solution which can solve the three challenges together and is general to different settings is another future work direction for us to explore.

7.3 Why Opponent Modelling?

We have discussed several times the motivation of focusing on opponent modelling based approaches in this thesis. Namely, they can be solutions to the *non-stationarity*, *partial observation* and *unclear learning objective* problems which are all important issues in the MARL community. However, in this ending section, I want to discuss why we need to study opponent modelling in a broader sense.

As stated in the previous section, opponent modelling is often computationally expensive. However, the distribution of computation resources between different entities such as individuals, organisations, companies and countries is imbalanced. Therefore, an agent which has enormous computation power may utilise an opponent model to obtain advantages when it interacts with other agents. A typical example we often see nowadays is price discrimination powered by user profiling. As computation cost also has the economies of scale property, it is reasonable to believe the gap between different entities will continue to enlarge in future. Therefore, understanding the capabilities and limits of opponent modelling based approaches is an important topic. It will help us to regularise the use of opponent models by agents with vast computation resources and prevent them from taking unfair advantages over agents with limited computation resources.

Bibliography

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. Maximum a posteriori policy optimisation. *CoRR*, abs/1806.06920, 2018. URL http://arxiv.org/abs/1806.06920.

Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641*, 2017.

Stefano V. Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2018.01.002. URL https://www.sciencedirect.com/science/article/pii/S0004370218300249.

John R. Anderson, C.Franklin Boyle, Albert T. Corbett, and Matthew W. Lewis. Cognitive modeling and intelligent tutoring. *Artificial Intelligence*, 42(1):7 – 49, 1990. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(90)90093-F. URL http://www.sciencedirect.com/science/article/pii/000437029090093F.

Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. *CoRR*, abs/1705.08439, 2017. URL http://arxiv.org/abs/1705.08439.

Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony

Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. doi: 10.1109/MSP.2017.2743240.

Karl Johan Åström. Optimal control of Markov Processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10:174–205, January 1965. URL `http://www.cs.utexas.edu/~shivaram`.

Michael Baker, Tia Hansen, Richard Joiner, and David Traum. The role of grounding in collaborative learning tasks. *Collaborative learning: Cognitive and computational approaches*, 1999.

Sander C.J. Bakkes, Pieter H.M. Spronck, and Giel van Lankveld. Player behavioural modelling for video games. *Entertainment Computing*, 3(3):71–79, 2012. ISSN 1875-9521. doi: https://doi.org/10.1016/j.entcom.2011.12.001. URL `https://www.sciencedirect.com/science/article/pii/S1875952111000486`. Games and AI.

David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*, pages 434–443. PMLR, 2019.

Nolan Bard, Michael Johanson, Neil Burch, and Michael Bowling. Online implicit agent modelling. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 255–262, 2013.

Samuel Barrett, Peter Stone, Sarit Kraus, and Avi Rosenfeld. Teamwork with limited knowledge of teammates. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, July 2013. URL `http://www.cs.utexas.edu/users/ai-lab/?barrett:aaai13`.

Andrew G. Barto and Richard S. Sutton. Simulation of anticipatory responses in classical conditioning by a neuron-like adaptive element. *Behavioural Brain Research*, 4(3):221–235, 1982. ISSN 0166-4328. doi: https://doi.org/

10.1016/0166-4328(82)90001-8. URL https://www.sciencedirect.com/science/article/pii/0166432882900018.

Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952. ISSN 0027-8424. doi: 10.1073/pnas.38.8.716. URL https://www.pnas.org/content/38/8/716.

Richard Bellman. A markovian decision process. *Indiana Univ. Math. J.*, 6:679–684, 1957. ISSN 0022-2518.

Richard Bellman and Rand Corporation. *Dynamic Programming*. Rand Corporation Research Study. Princeton University Press, 1957. URL https://books.google.co.uk/books?id=Pfgtj_klGsoC.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira 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. *CoRR*, abs/1912.06680, 2019. URL http://arxiv.org/abs/1912.06680.

Dimitri Bertsekas and David Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6): 589–598, 1989. doi: 10.1109/9.24227.

Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron. Opponent modeling in poker. In *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*, AAAI '98/IAAI '98, pages 493–499, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL http://dl.acm.org/citation.cfm?id=295240.295723.

Darse Billings, Neil Burch, Aaron Davidson, Robert Holte, Jonathan Schaeffer, Terence Schauenberg, and Duane Szafron. Approximating game-theoretic optimal strategies for full-scale poker. pages 661–668, 01 2003.

David M Blei, Michael I Jordan, et al. Variational inference for Dirichlet Process mixtures. *Bayesian analysis*, 1(1):121–143, 2006.

Guillaume Bono, Jilles Steeve Dibangoye, Laëtitia Matignon, Florian Pereyron, and Olivier Simonin. Cooperative multi-agent policy gradient. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 459–476, Cham, 2019. Springer International Publishing. ISBN 978-3-030-10925-7.

Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *IJCAI*, San Francisco, CA, USA, 2001. ISBN 1-55860-812-5, 978-1-558-60812-2. URL http://dl.acm.org/citation.cfm?id=1642194.1642231.

George W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, New York, 1951.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018. ISSN 0036-8075. doi: 10.1126/science.aao1733. URL https://science.sciencemag.org/content/359/6374/418.

Noam Brown and Tuomas Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890, 2019. ISSN 0036-8075. doi: 10.1126/science.aay2400. URL https://science.sciencemag.org/content/365/6456/885.

Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 3642039901, 9783642039904.

David Carmel and Shaul Markovitch. Learning models of the opponent's strategy in game playing. In *Proceedings of The AAAI Fall Symposium on Games: Planing and Learning*, pages 140–147, North Carolina, 1993. URL http://www.cs.technion.ac.il/~shaulm/papers/pdf/Carmel-Markovitch-fss1993.pdf.

David Carmel and Shaul Markovitch. Incorporating opponent models into adversary search. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 120–125. AAAI, 1996a.

David Carmel and Shaul Markovitch. Learning and using opponent models in adversary search. Technical Report CIS9609, Technion, 1996b. URL http://www.cs.technion.ac.il/~shaulm/papers/pdf/Carmel-Markovitch-CIS9609.pdf.

Doran Chakraborty and Peter Stone. Cooperating with a markovian ad hoc teammate. In *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '13, pages 1085–1092. International Foundation for Autonomous Agents and Multiagent Systems, 2013. ISBN 978-1-4503-1993-5. URL http://dl.acm.org/citation.cfm?id=2484920.2485091.

Georgios Chalkiadakis and Craig Boutilier. Coordination in multiagent reinforcement learning: A bayesian approach. In *AAMAS*, pages 709–716, New York, NY, USA, 2003. ACM. ISBN 1-58113-683-8. doi: 10.1145/860575.860689. URL http://doi.acm.org/10.1145/860575.860689.

Xi Chen and Xiaotie Deng. Settling the complexity of two-player Nash equilibrium. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 261–272. IEEE, 2006.

Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. AAAI '98/IAAI '98, Menlo Park, CA, USA,

1998. American Association for Artificial Intelligence. ISBN 0-262-51098-7. URL http://dl.acm.org/citation.cfm?id=295240.295800.

Matthew Crosby, Michael Rovatsos, and Ronald PA Petrick. Automated agent decomposition for classical planning. In *Twenty-Third International Conference on Automated Planning and Scheduling*, 2013.

Wojciech Marian Czarnecki, Gauthier Gidel, Brendan Tracey, Karl Tuyls, Shayegan Omidshafiei, David Balduzzi, and Max Jaderberg. Real world games look like spinning tops. *arXiv preprint arXiv:2004.09468*, 2020.

Aaron Davidson. Using artifical neural networks to model opponents in texas hold'em, 1999.

Aaron Davidson, Darse Billings, Jonathan Schaeffer, and Duane Szafron. Improved opponent modeling in poker. pages 493–499. AAAI Press, 2000.

Randall Davis. Report on the workshop on distributed ai. 06 1980.

Brian D. Davison and Haym Hirsh. Predicting sequences of user actions. 1998.

Harmen de Weerd, Rineke Verbrugge, and Bart Verheij. Higher-order theory of mind in the tacit communication game. *Biologically Inspired Cognitive Architectures*, 2015. ISSN 2212-683X. doi: https://doi.org/10.1016/j.bica.2014.11.010. URL http://www.sciencedirect.com/science/article/pii/S2212683X14000735.

Jilles Dibangoye and Olivier Buffet. Learning to act in decentralized partially observable MDPs. 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 1233–1242. PMLR, 10–15 Jul 2018. URL http://proceedings.mlr.press/v80/dibangoye18a.html.

Pierre Dillenbourg. What do you mean by collaborative learning? In P. Dillenbourg, editor, *Collaborative-learning: Cognitive and Computational Approaches.* Oxford: Elsevier, 1999. URL https://telearn.archives-ouvertes.fr/hal-00190240.

Prashant Doshi and Dennis Perez. Generalized point based value iteration for interactive pomdps. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1*, AAAI'08, pages 63–68. AAAI Press, 2008. ISBN 978-1-57735-368-3. URL http://dl.acm.org/citation.cfm?id=1619995.1620007.

Anca D Dragan, Kenton CT Lee, and Siddhartha S Srinivasa. Legibility and predictability of robot motion. In *Proceedings of the 8th ACM/IEEE international conference on Human-robot interaction*, pages 301–308. IEEE Press, 2013.

Tom Eccles, Edward Hughes, János Kramár, Steven Wheelwright, and Joel Z Leibo. Learning reciprocity in complex sequential social dilemmas. *arXiv preprint arXiv:1903.08082*, 2019.

R. Everett and S. Roberts. Learning against non-stationary agents with opponent modelling and deep reinforcement learning. In *2018 AAAI Spring Symposium Series*, 2018.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic Neural Networks for Hierarchical Reinforcement Learning. *arXiv e-prints*, art. arXiv:1704.03012, April 2017.

Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *CoRR*, abs/1709.04326, 2017a.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *CoRR*, abs/1605.06676, 2016.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *CoRR*, abs/1705.08926, 2017b. URL `http://arxiv.org/abs/1705.08926`.

Jakob N. Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian Action Decoder for Deep Multi-Agent Reinforcement Learning. *arXiv e-prints*, art. arXiv:1811.01458, November 2018.

Emily B Fox, Erik B Sudderth, Michael I Jordan, and Alan S Willsky. An HDP-HMM for systems with state persistence. In *Proceedings of the 25th international conference on Machine learning*, pages 312–319, 2008.

Roy Fox, Ari Pakman, and Naftali Tishby. Taming the Noise in Reinforcement Learning via Soft Updates. *arXiv e-prints*, art. arXiv:1512.08562, December 2015.

Drew Fudenberg. *The Theory of Learning in Games*. MIT Press, Cambridge, MA, 1998.

Drew Fudenberg and Jean Tirole. *Game Theory*. MIT Press, Cambridge, MA, 1991.

Thomas Furmston and David Barber. Variational methods for reinforcement learning. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 241–248, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL `http://proceedings.mlr.press/v9/furmston10a.html`.

Ya'akov Gal, Avi Pfeffer, Francesca Marzo, and Barbara J. Grosz. Learning social preferences in games. In *AAAI*, 2004.

Christopher Geib and Robert Goldman. Plan recognition in intrusion detection systems. In *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, volume 1, pages 46–55 vol.1, June 2001. doi: 10.1109/DISCEX.2001.932191.

Naomi Gildert, Alan G. Millard, Andrew Pomfret, and Jon Timmis. The need for combining implicit and explicit communication in cooperative robotic systems. *Frontiers in Robotics and AI*, 5:65, 2018. ISSN 2296-9144. doi: 10.3389/frobt.2018.00065. URL https://www.frontiersin.org/article/10.3389/frobt.2018.00065.

Piotr J. Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *J. Artif. Int. Res.*, 24(1):49–79, July 2005. ISSN 1076-9757. URL http://dl.acm.org/citation.cfm?id=1622519.1622521.

Claudia V. Goldman and Shlomo Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *CoRR*, abs/1107.0047, 2011. URL http://arxiv.org/abs/1107.0047.

Jordi Grau-Moya, Felix Leibfried, and Haitham Bou-Ammar. Balancing Two-Player Stochastic Games with Soft Q-Learning. *arXiv e-prints*, art. arXiv:1802.03216, February 2018.

Sven Gronauer and Klaus Dieopold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 04 2021. doi: 10.1007/s10462-021-09996-w.

Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, 12(3):175–204, July 1986. ISSN 0891-2017. URL http://dl.acm.org/citation.cfm?id=12457.12458.

Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 66–83, Cham, 2017. Springer International Publishing. ISBN 978-3-319-71682-4.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *CoRR*, abs/1702.08165, 2017. URL http://arxiv.org/abs/1702.08165.

Tuomas Haarnoja, Vitchyr Pong, Aurick Zhou, Murtaza Dalal, Pieter Abbeel, and Sergey Levine. Composable Deep Reinforcement Learning for Robotic Manipulation. *arXiv e-prints*, art. arXiv:1803.06773, March 2018a.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv e-prints*, art. arXiv:1801.01290, January 2018b.

Bo Haglund. Search algorithms for a bridge double dummy solver, 2010.

John Harsanyi. Games with incomplete information played by 'bayesian' players, part i. the basic model. *Management Science*, 14(7):486–502, 1967. URL https://EconPapers.repec.org/RePEc:inm:ormnsc:v:14:y:1968:i:7:p:486-502.

John C. Harsanyi. Bargaining in ignorance of the opponent's utility function. *The Journal of Conflict Resolution*, 6(1):29–38, 1962. ISSN 00220027, 15528766. URL http://www.jstor.org/stable/172875.

Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527, 2015. URL http://arxiv.org/abs/1507.06527.

He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé, III. Opponent modeling in deep reinforcement learning. In *ICML '16*, volume 48, pages 1804–1813, 2016.

Fritz Heider and Marianne Simmel. An experimental study of apparent behavior. *The American journal of psychology*, 1944.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, abs/1603.01121, 2016. URL http://arxiv.org/abs/1603.01121.

Pablo Hernandez-Leal and Michael Kaisers. Learning against sequential opponents in repeated stochastic games. In *The 3rd Multi-disciplinary Conference on Reinforcement Learning and Decision Making*, 2017.

Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *CoRR*, abs/1707.09183, 2017. URL `http://arxiv.org/abs/1707.09183`.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

Trong Nghia Hoang and Kian Hsiang Low. Interactive POMDP lite: Towards practical planning to predict and exploit intentions for interacting with self-interested agents. *CoRR*, abs/1304.5159, 2013. URL `http://arxiv.org/abs/1304.5159`.

Ronald A Howard. Dynamic programming and markov processes. 1960.

Hengyuan Hu, Adam Lerer, Alex Peysakhovich, and Jakob Foerster. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pages 4399–4410. PMLR, 2020.

Michael C Hughes and Erik B Sudderth. Memoized online variational inference for dirichlet Process mixture models. Technical report, BROWN UNIV PROVIDENCE RI DEPT OF COMPUTER SCIENCE, 2014.

Hiroyuki Iida, YOSHIYUKI KOTANI, and Jos Uiterwijk. Tutoring strategies in game-tree search (extended abstract). 18:433–435, 12 1996.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent 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 2961–2970, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/iqbal19a.html.

Hemant Ishwaran and Lancelot F James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, Dj Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International Conference on Machine Learning*, pages 3040–3049, 2019.

Michael Johanson and Michael Bowling. Data biased robust counter strategies. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, pages 264–271. PMLR, 2009. URL http://proceedings.mlr.press/v5/johanson09a.html.

Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper/2007/file/6e7b33fdea3adc80ebd648fffb665bb8-Paper.pdf.

Anders Jonsson and Michael Rovatsos. Scaling up multiagent planning: A best-response approach. In *Twenty-First International Conference on Automated Planning and Scheduling*, 2011.

Rudolf Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of basic Engineering*, 82:35–45, 01 1960.

Spiros Kapetanakis and Daniel Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Eighteenth National Conference on Artificial Intelligence*, Menlo Park, CA, USA, 2002. ISBN 0-262-51129-0. URL http://dl.acm.org/citation.cfm?id=777092.777145.

Hilbert J. Kappen'. Path integrals and symmetry breaking for optimal control theory. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(11), 2005. URL http://stacks.iop.org/1742-5468/2005/i=11/a=P11011.

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

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

A. Harry Klopf. *Brain Function and Adaptive Systems: A Heterostatic Theory*. Special reports. Air Force Cambridge Research Laboratories, Air Force Systems Command, United States Air Force, 1972. URL https://books.google.co.uk/books?id=rkmSPwAACAAJ.

Ross A. Knepper, Christoforos I. Mavrogiannis, Julia Proft, and Claire Liang. Implicit communication in a joint action. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '17. ACM, 2017. ISBN 978-1-4503-4336-7. doi: 10.1145/2909824.3020226. URL http://doi.acm.org/10.1145/2909824.3020226.

William J. Knottenbelt, Demetris Spanias, and Agnieszka M. Madurska. A common-opponent stochastic model for predicting the outcome of professional tennis matches. *Computers & Mathematics with Applications*, 64(12):3820–3827, 2012. ISSN 0898-1221. doi: https://doi.org/10.1016/j.camwa.2012.03.005. URL https://www.sciencedirect.com/science/article/pii/S0898122112002106. Theory and Practice of Stochastic Modeling.

Kevin B. Korb, Ann E. Nicholson, and Nathalie Jitnah. Bayesian poker. *CoRR*, abs/1301.6711, 2013. URL `http://arxiv.org/abs/1301.6711`.

Harold W Kuhn. A simplified two-person poker. *Contributions to the Theory of Games*, 1:97–103, 1950.

Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte carlo sampling for regret minimization in extensive games. *Advances in neural information processing systems*, 22:1078–1086, 2009.

Marc Lanctot, Vinícius Flores Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. *CoRR*, abs/1711.00832, 2017. URL `http://arxiv.org/abs/1711.00832`.

Guillaume Laurent, Laëtitia Matignon, and Nadine Fort-Piat. The world of independent learners is not markovian. *KES Journal*, 15:55–64, 03 2011. doi: 10.3233/KES-2010-0206.

Kennard Laviers, Gita Sukthankar, David Aha, and Matthew Molineaux. Improving offensive performance through opponent modeling. 01 2009.

Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *CoRR*, abs/1612.07182, 2016.

American Contract Bridge League. Laws of duplicate bridge, September 2017. URL `http://web2.acbl.org/documentlibrary/play/Laws-of-Duplicate-Bridge.pdf`.

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural Dirichlet Process mixture model for task-free continual learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Séverin Lemaignan and Pierre Dillenbourg. Mutual modelling in robotics: Inspirations for the next steps. *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 303–310, 2015.

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, *NIPS '13*. 2013. URL http://papers.nips.cc/paper/5178-variational-policy-search-via-trajectory-optimization.pdf.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. 17, 04 2015.

Lihong Li, T. Walsh, and M. Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, 09 2015.

Dahua Lin. Online learning of nonparametric mixture models via sequential variational approximation. *Advances in Neural Information Processing Systems*, 26: 395–403, 2013.

Diane J. Litman and James F. Allen. A plan recognition model for clarification subdialogues. In *Proceedings of the 10th International Conference on Computational Linguistics*, COLING '84, pages 302–311, Stroudsburg, PA, USA, 1984. Association for Computational Linguistics. doi: 10.3115/980431.980554. URL https://doi.org/10.3115/980431.980554.

Alan J Lockett, Charles L Chen, and Risto Miikkulainen. Evolving explicit opponent models in game playing. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 2106–2113. ACM, 2007.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR*, abs/1706.02275, 2017. URL http://arxiv.org/abs/1706.02275.

Peter Marbach and John N. Tsitsiklis. Approximate gradient methods in policy-space optimization of markov reward processes. *Journal of Discrete Event Dynamical Systems*, 13:2003, 2003.

Stephen McAleer, John B. Lanier, Roy Fox, and Pierre Baldi. Pipeline PSRO: A scalable approach for finding approximate Nash equilibria in large games. *CoRR*, abs/2006.08555, 2020. URL https://arxiv.org/abs/2006.08555.

Gordon McCalla, Julita Vassileva, Jim Greer, and Susan Bull. Active learner modelling. In Gilles Gauthier, Claude Frasson, and Kurt VanLehn, editors, *Intelligent Tutoring Systems*, pages 53–62, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45108-2.

H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543, 2003.

Michael F. McTear. User modelling for adaptive computer systems: a survey of recent developments. *Artificial Intelligence Review*, 7(3):157–184, Aug 1993. ISSN 1573-7462. doi: 10.1007/BF00849553. URL https://doi.org/10.1007/BF00849553.

Richard Mealing and Jonathan L. Shapiro. Opponent modeling by expectation–maximization and sequence prediction in simplified poker. *IEEE Transactions on Computational Intelligence and AI in Games*, 9(1):11–24, March 2017. ISSN 1943-068X. doi: 10.1109/TCIAIG.2015.2491611.

Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1): 8–30, 1961. doi: 10.1109/JRPROC.1961.287775.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas 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*, 518:529–33, 02 2015. doi: 10.1038/nature14236.

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. URL http://proceedings.mlr.press/v48/mniha16.html.

Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. ISSN 0036-8075. doi: 10.1126/science.aam6960. URL https://science.sciencemag.org/content/356/6337/508.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *CoRR*, abs/1703.04908, 2017.

Oskar Morgenstern and John Von Neumann. *Theory of games and economic behavior*. Princeton university press, 1953.

Paul Muller, Shayegan Omidshafiei, Mark Rowland, Karl Tuyls, Julien Pérolat, Siqi Liu, Daniel Hennes, Luke Marris, Marc Lanctot, Edward Hughes, Zhe Wang, Guy Lever, Nicolas Heess, Thore Graepel, and Rémi Munos. A generalized training approach for multiagent learning. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30,* *2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Bkl5kxrKDr.

Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Bridging the Gap Between Value and Policy Based Reinforcement Learning. *arXiv e-prints*, art. arXiv:1702.08892, February 2017.

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multi-agent systems: A review of challenges, solutions and applications. *CoRR*, abs/1812.11794, 2018. URL http://arxiv.org/abs/1812.11794.

Nicolas Perez Nieves, Yaodong Yang, Oliver Slumbers, David Henry Mguni, and Jun Wang. Modelling behavioural diversity for learning in open-ended games. *CoRR*, abs/2103.07927, 2021. URL https://arxiv.org/abs/2103.07927.

Brendan O'Donoghue, Ian Osband, Rémi Munos, and Volodymyr Mnih. The uncertainty bellman equation and exploration. *CoRR*, abs/1709.05380, 2017. URL http://arxiv.org/abs/1709.05380.

Jean Hyaejin Oh, Felipe Meneguzzi, Katia Sycara, and Norman S Kerman. An agent architecture for prognostic reasoning assistance. In *In Proc. 22nd Int. Joint Conf. on Artificial Intelligence, (IJCAI2011), Barcelona, Spain, pages 2513-2518*, July 2011.

Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *CoRR*, abs/1703.06182, 2017. URL http://arxiv.org/abs/1703.06182.

Shayegan Omidshafiei, Christos H. Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M. Czarnecki, Marc Lanctot, Julien Pérolat, and Rémi Munos. α-rank: Multi-agent evaluation by evolution. *CoRR*, abs/1903.01373, 2019. URL http://arxiv.org/abs/1903.01373.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Ronald Parr, Christopher Painter-Wakefield, Lihong Li, and Michael Littman. Analyzing feature generation for value-function approximation. volume 227, pages 737–744, 06 2007. doi: 10.1145/1273496.1273589.

Marc J. V. Ponsen, Steven de Jong, and Marc Lanctot. Computing approximate Nash equilibria and robust best-responses using sampling. *CoRR*, abs/1401.4591, 2014. URL http://arxiv.org/abs/1401.4591.

David Premack and Guy Woodruff. Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, 1, 1978. doi: 10.1017/S0140525X00076512.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994. ISBN 0471619779.

Neil C. Rabinowitz, Frank Perbet, H. Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick. Machine theory of mind. *CoRR*, abs/1802.07740, 2018.

Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. *CoRR*, abs/1802.09640, 2018.

Amir Rasouli, Iuliia Kotseruba, and John K. Tsotsos. Agreeing to cross: How drivers and pedestrians communicate. In *Intelligent Vehicles Symposium (IV), 2017 IEEE*. IEEE, 2017.

Bharanee Rathnasabapathy, Prashant Doshi, and Piotr Gmytrasiewicz. Exact solutions of interactive pomdps using behavioral equivalence. In *Proceedings* *of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '06, pages 1025–1032, New York, NY, USA, 2006. ACM. ISBN 1-59593-303-4. doi: 10.1145/1160633.1160816. URL http://doi.acm.org/10.1145/1160633.1160816.

Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference (extended abstract). IJCAI '13, pages 3052–3056, 2013. ISBN 978-1-57735-633-2. URL http://dl.acm.org/citation.cfm?id=2540128.2540576.

Maayan Roth, Reid Simmons, and Manuela Veloso. What to communicate? execution-time decision in multi-agent pomdps. In *DARS '06*, pages 177–186. Springer, 2006.

Michael Rovatsos, Gerhard Weiß, and Marco Wolf. Multiagent learning for open systems: A study in opponent classification. In Eduardo Alonso, Daniel Kudenko, and Dimitar Kazakov, editors, *Adaptive Agents and Multi-Agent Systems*, pages 66–87, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg. ISBN 978-3-540-44826-6.

Jonathan Rubin and Ian Watson. Computer poker: A review. *Artificial Intelligence*, 175(5):958–987, 2011. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2010.12.005. URL https://www.sciencedirect.com/science/article/pii/S0004370211000191. Special Review Issue.

Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959. doi: 10.1147/rd.33.0210.

Tuomas Sandholm. Perspectives on multiagent learning. *Artificial Intelligence*, 171(7):382–391, 2007. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2007.02.004. URL https://www.sciencedirect.com/science/article/pii/S0004370207000525. Foundations of Multi-Agent Learning.

Michael Schillo, Petra Funk, and Michael Rovatsos. Using trust for detecting deceitful agents in artificial societies. *Applied Artificial Intelligence*, 14(8):825–848, 2000. doi: 10.1080/08839510050127579. URL https://doi.org/10.1080/08839510050127579.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2014.09.003. URL https://www.sciencedirect.com/science/article/pii/S0893608014002135.

Christian Schroeder de Witt, Jakob Foerster, Gregory Farquhar, Philip Torr, Wendelin Boehmer, and Shimon Whiteson. Multi-agent common knowledge reinforcement learning. *Advances in Neural Information Processing Systems*, 32: 9927–9939, 2019.

John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

John Schulman, Xi Chen, and Pieter Abbeel. Equivalence Between Policy Gradients and Soft Q-Learning. *arXiv e-prints*, art. arXiv:1704.06440, April 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Andrew Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577: 1–5, 01 2020. doi: 10.1038/s41586-019-1923-7.

Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent reinforcement learning: a critical survey. 06 2003.

David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016. doi: 10.1038/nature16961.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 10 2017. doi: 10.1038/nature24270.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362:1140–1144, 12 2018. doi: 10.1126/science.aar6404.

Satinder Singh, Tommi Jaakkola, and Michael Jordan. Reinforcement learning with soft state aggregation. *Adv. Neural Inf. Process. Syst.*, 7, 11 1999.

Max Smith, Thomas Anthony, and Michael Wellman. Iterative empirical game solving via single policy best response. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=R4aWTjmrEKM.

Ekhlas Sonu and Prashant Doshi. Scalable solutions of interactive pomdps using generalized and bounded policy iteration. *Autonomous Agents and Multi-Agent*

Systems, 29(3):455–494, May 2015. doi: 10.1007/s10458-014-9261-5. URL https://doi.org/10.1007/s10458-014-9261-5.

Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and D. Chris Rayner. Bayes' bluff: Opponent modelling in poker. *CoRR*, abs/1207.1411, 2012. URL http://arxiv.org/abs/1207.1411.

D J Strouse, Max Kleiman-Weiner, Josh Tenenbaum, Matt Botvinick, and David Schwab. Learning to share and hide intentions using information regularization. NIPS'18, 2018. URL http://dl.acm.org/citation.cfm?id=3327546.3327688.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. *Advances in neural information processing systems*, 29:2244–2252, 2016.

Richard Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 08 1988. doi: 10.1007/BF00115009.

Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. Touretzky, M. C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1996. URL https://proceedings.neurips.cc/paper/1995/file/8f1d43620bc6bb580df6e80b0dc05c48-Paper.pdf.

Richard S. Sutton and Andrew G. Barto. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88(2):135–170, 1981. doi: 10.1037/0033-295X.88.2.135.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPs*, volume 99, pages 1057–1063. Citeseer, 1999.

Alex Tank, Nicholas Foti, and Emily Fox. Streaming variational inference for Bayesian nonparametric mixture models. In *Artificial Intelligence and Statistics*, pages 968–976. PMLR, 2015.

Yee Whye Teh. *Dirichlet Process*, pages 280–287. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_219. URL https://doi.org/10.1007/978-0-387-30164-8_219.

Evangelos Theodorou, Jonas Buchli, and Stefan Schaal. A generalized path integral control approach to reinforcement learning. *J. Mach. Learn. Res.*, 11, dec 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1953033.

Edward L. (Edward Lee) Thorndike. *Animal intelligence; experimental studies*. New York,The Macmillan Company, 1911. URL https://www.biodiversitylibrary.org/item/115711. https://www.biodiversitylibrary.org/bibliography/55072.

Zheng Tian, Ying Wen, Zhichen Gong, Faiz Punakkath, Shihao Zou, and Jun Wang. A regularized opponent model with maximum entropy objective. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, page 602–608. AAAI Press, 2019. ISBN 9780999241141.

Zheng Tian, Shihao Zou, Ian Davies, Tim Warr, Lisheng Wu, Haitham Bou Ammar, and Jun Wang. Learning to communicate implicitly by actions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:7261–7268, 04 2020. doi: 10.1609/aaai.v34i05.6217.

Zheng Tian, Hang Ren, Yaodong Yang, Yuchen Sun, Xiaohang Tang, Ian Davies, Ziqi Han, and Jun Wang. Learning to safely exploit a non-stationary opponent. *submitted to AAAI 2022 for review*, 05 2021.

Emanuel Todorov. Policy gradients in linearly-solvable mdps. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *NIPS*. 2010. URL http://papers.nips.cc/paper/4013-policy-gradients-in-linearly-solvable-mdps.pdf.

Marc Toussaint. Robot trajectory optimization using approximate inference. ICML '09, pages 1049–1056, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553508. URL http://doi.acm.org/10.1145/1553374.1553508.

Marc Toussaint and Amos Storkey. Probabilistic inference for solving discrete and continuous state markov decision processes. ICML '06, pages 945–952, New York, NY, USA, 2006. ACM. ISBN 1-59593-383-2. doi: 10.1145/1143844.1143963. URL http://doi.acm.org/10.1145/1143844.1143963.

Sun Tzu. *The Art of War*. Dover Military History, Weapons, Armor. Dover Publications, 2002. ISBN 9780486425573. URL https://books.google.co.uk/books?id=UTGnopblxt8C.

Jan van den Brand. A deterministic linear program solver in current matrix multiplication time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 259–278. SIAM, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John P. Agapiou, Julian Schrittwieser, John Quan, Stephen Gaffney, Stig Petersen, Karen Simonyan, Tom Schaul, Hado van Hasselt, David Silver, Timothy P. Lillicrap, Kevin Calderone, Paul Keet, Anthony Brunasso, David Lawrence, Anders Ekermo, Jacob Repp, and Rodney Tsing. Starcraft II: A new challenge for reinforcement learning. *CoRR*, abs/1708.04782, 2017. URL http://arxiv.org/abs/1708.04782.

Oriol Vinyals, Igor Babuschkin, Wojciech Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John Agapiou, Max Jaderberg, and David Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575, 11 2019. doi: 10.1038/s41586-019-1724-z.

Chong Wang and David Blei. Truncation-free stochastic variational inference for Bayesian nonparametric models. *Advances in neural information processing systems*, 25:422–430, 2012.

Chris Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992. ISSN 0885-6125.

Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, May 1989. URL `http://www.cs.rhul.ac.uk/~chrisw/new_thesis.pdf`.

Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. Multiagent soft q-learning. *AAAI*, 2018.

Gerhard Weiss. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, volume 1, pages –648. 07 2000. ISBN 0262731312.

Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rkl6As0cF7`.

Ying Wen, Yaodong Yang, and Jun Wang. Modelling bounded rationality in multi-agent interactions by generalized recursive reasoning. In *IJCAI*, pages 414–421, 2020. URL `https://doi.org/10.24963/ijcai.2020/58`.

Paul J. Werbos. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transac-*

tions on Systems, Man, and Cybernetics, 17(1):7–20, 1987. doi: 10.1109/TSMC.1987.289329.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Aaron Wilson, Alan Fern, and Prasad Tadepalli. Bayesian policy search for multi-agent role discovery. AAAI'10, pages 624–629. AAAI Press, 2010. URL http://dl.acm.org/citation.cfm?id=2898607.2898708.

Zhe Wu, Kai Li, Enmin Zhao, Hang Xu, Meng Zhang, Haobo Fu, Bo An, and Junliang Xing. L2E: Learning to exploit your opponent, 2021.

Mengdi Xu, Wenhao Ding, Jiacheng Zhu, Zuxin Liu, Baiming Chen, and Ding Zhao. Task-agnostic online reinforcement learning with an infinite mixture of Gaussian Processes. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *CoRR*, abs/2011.00583, 2020. URL https://arxiv.org/abs/2011.00583.

Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. 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 5571–5580. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/yang18d.html.

Yaodong Yang, Rasul Tutunov, Phu Sakulwongtana, and Haitham Bou Ammar. α^2-rank: Practically scaling α-rank through stochastic optimisation. In *Proceedings* *of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, page 1575–1583, Richland, SC, 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450375184.

Chih-Kuan Yeh and Hsuan-Tien Lin. Automatic bridge bidding using deep reinforcement learning. *CoRR*, abs/1607.03290, 2016. URL `http://arxiv.org/abs/1607.03290`.

Kaiqing Zhang, Zhuoran Yang, and Tamer Basar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *CoRR*, abs/1911.10635, 2019. URL `http://arxiv.org/abs/1911.10635`.

Yan Zheng, Zhaopeng Meng, Jianye Hao, Zongzhang Zhang, Tianpei Yang, and Changjie Fan. A deep Bayesian policy reuse approach against non-stationary agents. In *Advances in Neural Information Processing Systems*, 2018.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI '08*.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. *Advances in neural information processing systems*, 20:1729–1736, 2007a.

Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. *Advances in neural information processing systems*, 20:1729–1736, 2007b.