Learning to Coordinate Efficiently through Multiagent Soft Q-Learning in the presence of Game-Theoretic Pathologies

Siphelele Danisa

Supervisor: Associate Prof. Jonathan Shock
Dr Arnu Pretorius

Department of Pure and Applied Mathematics
University of Cape Town

This dissertation is submitted for the degree of
Master of Science in Applied Mathematics

September 2022

Declaration

I would like to dedicate this thesis to my family and friends. I would like to also mention specifically my late grandmother and my late sister who passed on having had a tremendous positive impact and influence in my life. I know that they would be very proud of me, and I believe that they are.

Acknowledgements

To start with, I would like to thank my family for being ever patient with me and supportive of my decisions along this journey. Their love and warmth, which I believe has always created a very conducive space for my development, continues to motivate me to do my best. This is something that, I think, is fundamentally the birth place of all the other attributes that have made this work possible.

I would also like to thank my supervisor Associate Professor Jonathan Shock who has been the best supervisor I could ever ask for since 2018. I cannot fully thank him for being so generous with his time and wisdom, and also for believing in me without wavering. I have really learnt so much by working with him, and I am always inspired by the work that he does as well as his work ethic and approach to life. Truly, truly exceptional supervision and mentorship and I am very, very fortunate !

My friends have also made this work possible by keeping my head above the water throughout this journey. I am exceedingly grateful for all the support that they continue to extend to me, and I am lucky to have such an amazing social circle– full of people that I love, and whom I find beyond inspiring. I want to give great thanks to Joshua Hayes, Thato Mgoboli, Sinelethu Hashibi, Aslesha Pokhrel, Manasi Mohan and Gareth McCullagh for their support and assistance with putting the final document together.

I would also like to thank the Mathematics Department for being so supportive to me during my time as a student in the department. I am very grateful to Associate Professor David Erwin for being so generous with his time and for helping me discover myself in Mathematics. Our constant meetings to discuss fundamental corners of Mathematics and the stories he would tell really, really inspired me and made me a better Mathematician in many ways. I am also very grateful to Associate Professor Jeff Murugan for all of his support, also starting at the very beginning of my undergraduate degree. Another person who really believed in me, and from whom I learnt a great deal.

Last but not least, I am grateful to have received the Shuttleworth scholarship which supported my studies during this degree (to Professor Peter Dunsby for all his work towards developing this scholarship, and to the committee for all their work). I would also like to thank InstaDeep Ltd for the internship opportunity (with the Cape Town office) which piqued my interest into multiagent reinforcement learning and, hence, inspired this journey, and Dr Arnu Pretorius for being an amazing mentor and role model during the internship.

They say it takes a community to raise a child, and I see that being something that is applicable to my journey. There are so many inspirational people I've met along the way whose names I have not mentioned, but that does not mean I have forgotten. These are also people from whom I have learnt a lot, people who strongly believe in me and who have shown me immense support. None of this would have been possible without all the big and small efforts carving my trajectory very uniquely. I remain thankful to all of them.

Research Abstract

In this work we investigate the convergence of multiagent soft Q-learning in continuous games where learning is most likely to be affected by relative overgeneralisation. While this will occur more often in multiagent independent learner problems, it is present in joint-learner problems when information is not used efficiently in the learning process. We first investigate the effect of different samplers and modern strategies of training and evaluating energy-based models on learning to get a sense of whether the pitfall is due to sampling inefficiencies or underlying assumptions of the multiagent soft Q-learning extension (*MASQL*). We use the word sampler to refer to mechanisms that allow one to get samples from a given (target) distribution. After having understood this pitfall better, we develop opponent modelling approaches with mutual information regularisation. We find that while the former (the use of efficient samplers) is not as helpful as one would wish, the latter (opponent modelling with mutual information regularisation) offers new insights into the required mechanism to solve our problem. The domain in which we work is called the Max of Two Quadratics differential game where two agents need to coordinate in a non-convex landscape, and where learning is impacted by the mentioned pathology, relative overgeneralisation. We close this research investigation by offering a principled prescription on how to best extend single-agent energy-based approaches to multiple agents, which is a novel direction. The code for this thesis can be found at :
`https://github.com/siphelele danisa/thesis-investigations`.

Table of contents

List of figures

Chapter 1

Introduction

Artificial intelligence may be defined as the theory and development of systems that are able to perform tasks that normally require human intelligence [8]. It is hard to pin point the origins of artificial intelligence but it has certainly been around for well over 70 years [9]. The ideas of artificial intelligence, coupled with our understanding of the brain, gave rise to artificial neural networks as we know them today [10]. Neural networks are trained to perform various tasks using algorithms within the broad field known as machine learning. Machine learning itself can be defined as the use and development of systems that are able to learn and adapt through the use of algorithms and statistical models to analyse and draw inferences from patterns in the data [8]. This field is often viewed as being made up of three main streams–supervised learning, unsupervised learning as well as reinforcement learning [11][12]. Supervised learning involves the mechanisms wherein learning is done on data that has labels (for example, classification), while unsupervised learning is about detecting patterns and abstracting important features from data (for example, clustering). Of interest to us is the third stream, reinforcement learning. Reinforcement learning offers a decision making framework wherein an agent learns by interacting with its environment. The agent takes actions, which are followed by state transitions and receipt of reward signals, and, through this interaction, it is able to learn to perform a wide variety of tasks from household robotics to adaptive controllers in petroleum refineries [12].

The single-agent reinforcement learning framework which allows a single agent to learn can also be extended to multiple agents (to what is known as multiagent reinforcement learning), which is the direction of study that is of interest to us within this thesis. Multiagent reinforcement learning is a rich field wherein behaviours that cannot be imagined in single agents can be exhibited. This is due to the fact that there are multiple agents, in the environment, which can interact with each other in non-trivial ways. For example, Tan [13] investigated aspects of these complex behaviours by studying competitive and cooperative agents in a game of predator-and-prey, and Harries et al.[14] explore the use of reinforcement learning for functional software testing. The formerly mentioned game has agents of two types–predator/hunter and preys– that roam around a grid. The predators/hunters are tasked with capturing the preys. This mentioned work was published in the 1993, and at this time the field didn't have a solid theoretical grounding. The mathematical connection to Markov games (to be defined later) was later established by Littman [15] in 1994. Since then, many advancements have been made in the field. For example, Millidge et al. [16] tell us that we can view reinforcement learning as iterative inference (methods here directly optimise the parameters of the approximate posterior), or amortised inference (where learning is done through a parameterised function which maps states to the parameters of the approximate posterior). We see from this work that both approaches are well established. To get a sense of how far the field has advanced since early 2000s, one may look into Shoham et al. [17] (written in 2003) who offer the perspective that at the time things were very underdeveloped by looking at prominent papers, critiquing them and, at the same time, devising fundamental questions surrounding algorithms' design and perspective that they urged researchers to consider. Examples of surveys that do an amazing job to showcase the developments to date in, respectively, the general field (from training schemes to emergent behaviours) and the vast literature concerned with cooperative games (training schemes and algorithms for the cooperative settings, which are of primary interest to us) include Gronauer et al. [18] and Oroojlooy et al. [19].

The research undertaken in this thesis was inspired by Wen et al. [5] who attempted to extend a reinforcement learning method called soft Q-learning (to be defined formally later) that was introduced by Haarnoja et al. [20] to (cooperative) contexts with multiple agents. In doing this, the authors looked into a reasonably simple continuous game (called the differential game/max of two quadratics) where agents usually struggle to succeed due to game-theoretic technicalities that lead to poor gradient computation. In this game there are two agents whose task is to find a pair of numbers (a, b) that correspond to the global maximum on a given surface. Each agent controls one component on this tuple, and they have to jointly find the optimality. The agents in experiments were shown to achieve reasonable success in this task, but they were only successful 72% of the time. This didn't seem to be consistent with our understanding of what soft Q-learning does, however. Single-agent soft Q-learning learns general and robust policies, which is proven in Fischer et al. [21] and Eysenbach et al. [3], and this breakdown in the extension to multiagent settings seemed surprising. The questions we had to ask ourselves therefore were as follows : Assuming that the policies we are learning are robust, then what causes this inconsistency in performance? Why are we only getting convergence 72% of the time? If we are not learning general policies, then where are we making the simplifying step that causes the breakdown, and can we do better (even at the expense of more stringent requirements)? Our goal was to understand the literature surrounding this problem, the nature of the pathology leading to this breakdown, and to extend the ideas in this space.

In what follows, we considered two lines of investigation in this study. The first was concerned with the way we learn the general policies. In the learning process for reinforcement learning, agents must take actions, and these actions are followed by receiving reward signals and state transitions. As mentioned previously, the policies that are learnt with the soft Q-learning machinery are general (in the sense that they are universal approximators of distributions), and this makes them hard to sample from. The initial thought therefore was that we could be sampling inefficiently from these policies when we want to evaluate the actions for the individual agents. We assumed that the policies we were learning were general, and so looking into this direction made sense, since this was the only (loosely speaking) non-routine step as far as the literature is concerned. Our first sub-question was as follows : Assuming that the policies learnt by the agents are general, is the way we sample from these general policies the cause of the problem (inconsistent convergence)? The corresponding hypothesis was that under the stated assumption, improving the sampling should solve the problem and provide better results. We then considered different samplers that we found relevant for our study. We evaluated these as is needed for our case, and compared the results. The results told us that this was not the main problem, so we proceeded to the next investigation.

The second aspect that we investigated, seeing that the above was not effective, was the underlying assumptions of these models. The first step was observing that the assumption previously made about the generality of the policies that are learned was not, in fact, true. This was also confirmed by recovering implicit assumptions on the derivation of gradients, in the original source, which are certainly not compatible with cooperative games. What these assumptions translate to is an update rule that does not scale the gradients in the correct magnitude and direction. There is work that attempts to remedy this approach by doing some form of opponent modelling [6]. This approach leads to an objective that maximises the reward as well as the entropy of the joint policy, which we find to be inefficient. Inspired by this, we took a slightly different direction with our investigation. The key argument we make, for our approach, is that the perspective to maximise entropy doesn't really sit well with multiple agents since the environment is not stationary from the scope of each agent. This is because the joint policy is dependent on the joint action instead of individual actions, and this influences the state transitions of the game. It is thus hard for each agent to conclude anything about the optimality of its own actions as far as the global reward is concerned. In the context of this non-stationarity problem, it is hard to see how exactly maximising the entropy aids learning. We therefore asked ourselves if the relevant quantity is actually mutual information. Mutual information, instead of focusing on the entropy of one random variable, focuses on how informative a random variable is with respect to another. The main motivation for this is that in any case where two entities are interacting, in reality, coordination comes naturally and more easily if both entities can share information efficiently, which is (intuitively) what mutual information regularisation gives us. Here we hypothesised that including a mutual information term as a regulariser would improve performance, which was confirmed by the experiments.

We expected our method to do reasonably well because we know that regularisation can be used to establish coordination [22]. The cited work argues that instead of learning more than what is required (for example, extra networks that are not needed when the agent is evaluated), we should make use of experiences to regularise the individual policies, which in turn leads to better exploration. This is confirmed by methods that we will see in our study.

We hereby summarise the research questions and hypotheses which drove the work within this thesis.

Research Question : If we know that the policies we learn should be general enough to model arbitrary distributions, then what is the cause of inconsistent convergence in multiagent soft Q-learning?

Sub-research Question 1 : Assuming that the policies learnt by the agents are general, is the way we sample from these general policies the cause of the problem?

(Corresponding) hypothesis 1 : Assuming that the policies learnt by the agents are general, it must be the case that the way in which we sample from them is inefficient and can be improved by better sampling strategies.

Sub-research Question 2 : If the sampling strategies do not improve convergence, then are we simply not learning the policies well (i.e are we not doing the correct optimisation)?

(Corresponding) Hypothesis 2 : If the sampling strategies do not improve convergence, then we not doing the correct optimisation for the learning of policies.

In what follows, we start with a literature review which covers reinforcement learning, regularisation in reinforcement learning, and which then extends the concepts covered to multiple agents in cooperative environments. Following this, we introduce relative overgeneralisation as a concept and show how it manifests in an example of interest called the Max of Two Quadratics. Furthermore, we discuss failure to learn focusing more on the sampling process. Herein we introduce mechanisms that are used for efficient sampling. Lastly, we introduce extensions of multiagent soft Q-learning whose designs are in line with our hypotheses and their respective experimental investigations. This chapter is then followed by the methodology, which contains details about our experiments and contributions, and acts as a doorway to the results and discussion sections. Lastly, we sum everything up.

Chapter 2

Literature Review

In this chapter, we will follow the development of the literature that is required to design the methodology, understand the results, and analyse their significance, as well as place the present work in the context of the field at large. The problem that we are trying to solve (which we discuss in depth in section 2.6 on relative overgeneralisation in team games) involves two agents whose task is to find the point of highest reward on a reward surface with two quadratic peaks at different parts along the surface of the action space. This makes the surface non-convex. The agents struggle to find this because learning algorithms usually compute parameter updates with gradients that are based on averages, which turns out to be insufficient [23]. Since this is a continuous space environment it is not easy to avoid this pitfall, and very well constructed algorithms are needed to solve the problem better. The algorithms that should perform better are those that can enforce a level of coordination between the agents, which gives them a better chance of finding the optimal solution than otherwise. Methods that have become popular for continuous games in reinforcement learning involve energy-based models, for example the so-called soft Q-learning and soft Actor-Critic. The former has been applied to this problem, and variants of it have been established towards improving the performance. It is thus our main focus for this work.

This chapter will take the following structure for the sections :

- An overview of single-agent reinforcement learning.
- Extending ideas from this setting to multiple agents, thereby introducing multiagent reinforcement learning.

- A discussion on information theoretic mutual information and entropy.
- A discussion on communication and coordination.
- Reintroducing reinforcement learning as inference on a graphical model.
- Introducing our environment of interest and relative overgeneralisation.
- An overview of samplers for probability density estimation.
- Introducing regularisation methods in multiagent algorithms for solving our problem of interest.

2.1 Single-Agent Reinforcement Learning

Reinforcement learning agents learn to perform tasks by interacting with the environment which provides feedback about how well the task is being performed. This interaction is usually modelled as a Markov decision process (MDP).

Definition 1 (Markov Decision Process). *A Markov decision process is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(S)$ denotes the probability transition function from a state $s \in \mathcal{S}$ to $s' \in \mathcal{S}$ given an action $a \in \mathcal{A}$, $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function that determines the immediate reward for the agent given a state transition, and γ is a discount factor that weighs the value of instantaneous rewards to future rewards. Notice that we use R to denote the function itself, and r to denote the values that the function takes in what follows.*

The interplay between the agent and the environment in light of the above definition is captured on Fig. 2.1. Here, the agent starts at a time, t, in a state $s_t \in \mathcal{S}$, and takes an action $a_t \in \mathcal{A}$ which is then followed by a transition to state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$. Furthermore, the agent obtains a reward $r_t = R(s_t, a_t, s_{t+1})$ associated with the transition. The goal is to the find a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ such that $a_t \sim \pi(\cdot|s_t)$ and the discounted accumulated reward is maximised. In maximising the reward, we look at value functions that we define as follows.

Definition 2 (Value Functions). *We define state value functions and state-action value functions under a policy π as, respectively,*

$$V_\pi(s) = \mathbb{E}\Big(\sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \Big| a_t \sim \pi(\cdot|s_t), s_0 = s \Big),$$

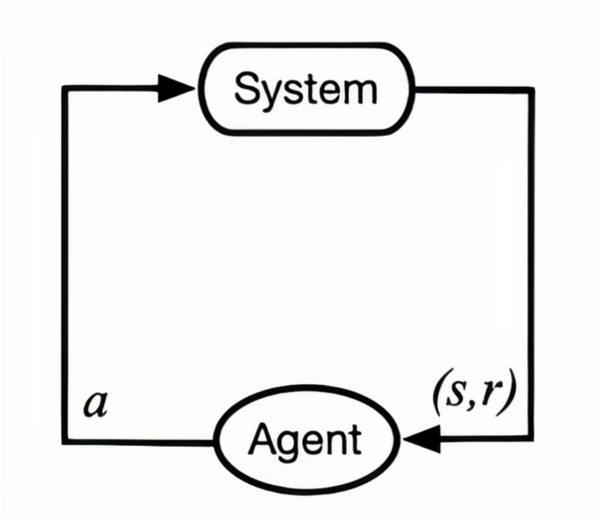

Fig. 2.1 An MDP illustration [1]. The figure depicts an agent interacting with the environment (/system). The agent takes actions, a, to which the system responds with a state transition to state s, and a reward r corresponding to the value attained by the reward function, R, given the tuple (s,a).

and

$$Q_\pi(s,a) = \mathbb{E}\Big(\sum_{t\geq 0}\gamma^t R(s_t,a_t,s_{t+1})\Big| a_t \sim \pi(\cdot|s_t), a_0 = a, s_0 = s\Big).$$

The former defines the expected future returns given a starting state, while the latter gives us the expected returns given a starting state and a particular action as the agent navigates through the MDP.

One of the key algorithms in classical reinforcement learning is Q-learning introduced by Watkins et al. [24]. Q-learning is an off-policy algorithm, in the sense that it learns the policy that approximates optimal value functions while following an independent policy called the behaviour policy. This is in contrast to following the policy it is learning (on-policy learning), as seen in SARSA by Rummery et al. [25]. Beyond the idea that this method is off-policy, the update rule which characterises it is given by

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\Big[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)\Big].$$

The algorithm is given by Algorithm 1, and it is worth noting that this method has been very successfully extended to the case of function approximation (as is often necessitated when there is a large number of states) leading to the so-called Deep

Q-Learning [26]. This work truly set the stage for Q-learning in the context of deep learning (i.e., learning with deep neural networks).

Algorithm 1: Q-Learning

```
The first step is to initialise the rate $\alpha \in (0,1]$ for updates, and an $\epsilon > 0$ for
  epsilon-greedy exploration;
Next, initialise the Q function with the value of terminal states being set to 0;
for each episode do
    Start with an initialisation of the state, $S$;
    for each step of episode do
        Sample action $A$ given the state using a policy derivated from the Q
          function (this can be $\epsilon$-greedy);
        Step into the environment using the action and observe the reward $r$, as
          well as the next state $S'$;
        Do the Q-learning update :
          $Q(S,A) \leftarrow Q(S,A) + \alpha[r + \gamma \max_a Q(S',a) - Q(S,A)]$;
        Update the state variable;
    end
end
```

Let's discuss a special case of this setting where the state information is not too important, but where we don't exactly have a trivial situation. We will approach this from the angle of the so-called k-armed bandit problem.

2.1.1 The k-armed Bandit Problem

Suppose that an agent is faced with the task of repeatedly choosing k different options/actions. After each selection, suppose that we give the agent a reward chosen from a stationary distribution, which depends only on the actions. Let's assume that we want the agent to learn to maximise the expected total reward over some period of time. This situation defines the so-called k-armed bandit problem. This is similar to a person playing a game on a slot machine with k-levers–to further concretise this notion, a slot machine with only one lever defines a 1-armed bandit problem. Sutton et al. [12] proceed to lay down the mathematical framework in Chapter 2 of the text.

The important aspect (in the sense of being of relevance of us) of these problems is that they are independent of states. This is to say that the process that gives the agent rewards only depends on the actions, instead of states. Hence, the value functions are also independent of state. A policy in this instance is a rule that assigns arbitrary state variables to the optimal configuration values (values of the levers that achieve maximum reward). We will see that our problem of interest generalises this setting, in the sense that the reward function is solely dependent on the actions yet it remains non-trivial as will be discussed in Section 2.6. In particular, we will observe that while the state variable is irrelevant (by definition of the reward) we can reformulate our context in a way that allows us to talk about actions, states and the values of actions on states or values of states, as we do in classical reinforcement learning.

The single-agent regime has been studied extensively [12]. Below we give brief summaries of the extensions of the above content to the case of multiple agents. We will see that this is quite similar to the single-agent reinforcement learning formulation up to some technicalities. This exposition is given following Zhang et al. [1].

2.2 The Single-Agent Reinforcement Learning Extensions

The first way to extend single-agent reinforcement learning is rather intuitive. In this case, the multiagent system is modelled as what is known as a Markov game. We start our discussion with this formulation.

Markov Games

Definition 3 (Markov Game). *A Markov game is defined as a tuple* $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i \in N}, \mathcal{P}, \{R^i\}_{i \in N}, \gamma)$. *In this context* $\mathcal{N} = \{1, \cdots, N\}$ *is the set of agent labels,* $\mathcal{S}$ *is the state space, and* $\mathcal{A}^i$ *is the action space for agent* $i \in \mathcal{N}$. *Letting* $\mathcal{A} = \prod \mathcal{A}^i$ *(the direct product of action spaces), then* $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ *as before and similarly* $R^i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ *is a reward function for agent i.*

Remark : Notice that the transitions are determined by the collection of actions that the agents take, and the reward similarly. Thus, for the action space, we have a tuple of length greater than one that plays a role in determining these quantities. This clearly extends the single-agent case where we only needed to look at actions corresponding to the choices available to a single agent. The pioneering work for this approach, to the best of our knowledge, is by Littman, M. [15].

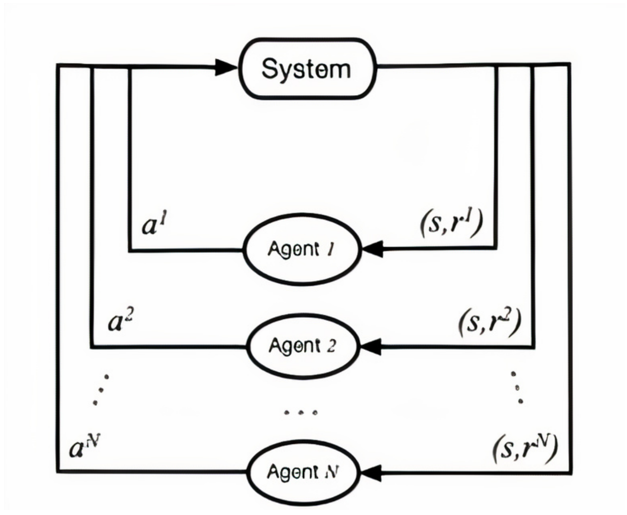

Fig. 2.2 A Markov game illustration [1]. The figure depicts N agents interacting with the environment (/system). Each agent takes an action, a^i, and (based on the received N-tuple) the system responds with a state transition to state s, and a reward r^i corresponding to the value attained by the reward function, R^i, given the tuple (s,a). Notice that the state transition is the same for all agents, but the reward values received need not be the same.

The working principle here is that at time, t, each agent i takes an action a_t^i with the system state at s_t. The system then transitions to a state s_{t+1} and each agent gets a corresponding reward $r_t^i = R^i(s_t, a_t, s_{t+1})$. This is illustrated by Figure 2.2.

The objective, for each agent, is to find a policy $\pi^i : \mathcal{S} \to \Delta(A^i)$ such that $a_t^i \sim \pi^i(\cdot|s_t)$ and that this optimises the long-term reward for that agent. We analogously define the value functions.

Definition 4 (Multiagent Value Functions)**.** *The corresponding value function for each agent, i, becomes a function of the joint policy defined as* $\pi(a|s) := \prod\{\pi^i\}_{i\in\mathcal{N}}(a^i|s)$ *such that, if* $-i$ *represents the indices of all agents except agent i, then*

$$V^i_{\pi^i,\pi^{-i}}(s) = \mathbb{E}\Big(\sum_{t\geq 0}\gamma^t R^i(s_t,a_t,s_{t+1})\Big| a_t^i \sim \pi^i(\cdot|s_t), s_0 = s\Big).$$

Moreover,

$$Q^i_{\pi^i,\pi^{-i}}(s,a) = \mathbb{E}\Big(\sum_{t\geq 0}\gamma^t R^i(s_t,a_t,s_{t+1})\Big| a_t^i \sim \pi^i(\cdot|s_t), a_0^i = a, s_0 = s\Big).$$

Unlike the MDP setting, here the optimal solution does not depend on a single agent. A commonly accepted solution to this kind of game is the so-called Nash equilibrium (NE) which is a strategy in which no agent has incentive to deviate from their policy. Formally, we have the following definition.

Definition 5 (Nash Equilibrium). *Given a Markov game $(\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in N}, \mathcal{P}, \{R^i\}_{i\in N}, \gamma)$, a Nash equilibrium is a joint policy $\pi^* = (\pi^{1,*}, \cdots, \pi^{N,*})$ such that for any $s \in \mathcal{S}$ and $i \in \mathcal{N}$ we have that $V^i_{\pi^{i,*}, \pi^{-i,*}}(s) \geq V^i_{\pi^i, \pi^{-i}}(s)$, for any π^i.*

What we have seen in this section is that Markov games extend single-agent reinforcement learning in a very intuitive and straightforward sense. The major drawback is that they do not clearly model reality as one would expect. For instance, in the real world, for any two interacting entities there is often uncertainty associated with making decisions due to partial observability, or similar circumstances. The structure needed for this is not present in the formulation we have explored, and more work is needed to establish this. It is thus difficult to do work on contexts with uncertainty with Markov games [27] [28]. There is a much more general formulation that extends Markov games, which is also well-studied. This is the context of Extensive-form games [29] [30]. We give a brief discussion of this formulation in what follows. For the purposes of our work, the distinction between these extensions is not crucial, however this exposition is necessary to understand the broad applicability of the work that we are doing, and therefore the implications of it.

Extensive-form Games

The context of extensive-form games is one that is more general compared to what we have seen so far, as mentioned above, and it is of interest because it enables the handling of imperfect information in multiagent decision making. Formally, it is defined as follows [1].

Definition 6. *An extensive-form game is defined by $(\mathcal{N} \cup \{c\}, \mathcal{H}, \mathcal{Z}, \mathcal{A}, \{R^i\}_{i\in\mathcal{N}}, \tau, \pi^c, \mathcal{S})$ where*

- *$\mathcal{N}$ is the number of agents.*
- *The variable c denotes a special agent called chance or nature, which has a fixed stochastic policy that specifies the randomness of the environment.*
- *Again, $\mathcal{A}$ is the set of all possible actions that agents can take.*

- *In addition to this, however, $\mathcal{H}$ is the set of all possible histories, where each history is a sequence of actions taken from the beginning of the game. Let $\mathcal{A}(h) = \{a|ha \in \mathcal{H}\}$ denote the set of actions available after a non-terminal history h, where ha is a sequence h followed by action a. Suppose that an agent takes $a \in \mathcal{A}(h)$ given history $h \in \mathcal{H}$, this then leads to a new history $ha \in \mathcal{H}$.*
- *Amongst all histories $\mathcal{Z} \subset \mathcal{H}$ is a subset of terminal histories that represents the completion of the game.*
- *A utility is assigned to each agent $i \in \mathcal{N}$ at a terminal history, dictated by the function $R^i : \mathcal{Z} \to \mathbb{R}$.*
- *Moreover, $\tau : \mathcal{H} \to \mathcal{N} \cup \{c\}$ is the identification function that specifies which agent takes the action at each history. If $\tau(h) = c$, the chance agent takes an action a according to its policy π^c, i.e., $a \sim \pi^c(\cdot|h)$.*
- *Furthermore, $\mathcal{S}$ is the partition of $\mathcal{H}$ such that for any $s \in \mathcal{S}$ and any $h, h' \in s$, we have $\tau(h) = \tau(h')$ and $\mathcal{A}(h) = \mathcal{A}(h')$. We call the elements $s \in \mathcal{S}$ information sets. In other words, histories h and h' in the same partition are indistinguishable to the agent that is about to take action, namely $\tau(h)$. The elements in $\mathcal{S}$ are referred to as information states.*

Remark : Information sets are defined as those histories that are impossible to distinguish at a given state. This, in addition to the chance agent, is what makes this framework much more resemblent of the uncertainty that exists in the real work. We say that a game has imperfect information whenever there is an information set of length cardinality greater than 1.

An illustration of this, similar to the ones we have seen above, can be seen on Figure 2.3.

In this case, we can discuss the same notions of equilibrium using the so-called ϵ-Nash equilibrium, defined as follows.

Definition 7 (ϵ - Nash Equilibrium). *An ϵ-Nash equilibrium of an extensive-form game represented by $(\mathcal{N} \cup \{c\}, \mathcal{H}, \mathcal{Z}, \mathcal{A}, \{R^i\}_{i\in\mathcal{N}}, \tau, \pi^c, \mathcal{S})$ is a joint policy $\pi^* = (\pi^{1,*}, \cdots, \pi^{N,*})$ such that for any $i \in \mathcal{N}$, $R^i(\pi^{i,*}, \pi^{-i,*}) \geq R^i(\pi^i, \pi^{-i,*}) - \epsilon$ for any policy π of agent i. In this context, π^{-i} denotes the joint policy of agents in $\mathcal{N}\backslash\{i\}$ where agent j adopts policy π^j for all $j \in \mathcal{N}\backslash\{i\}$. If $\epsilon = 0$, then π^* constitutes a Nash equilibrium.*

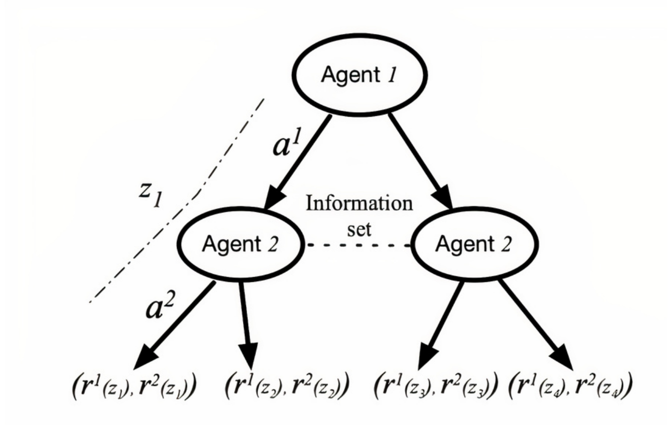

Fig. 2.3 An Extensive-form game illustration [1]. The players take actions a^i alternately and receive rewards values $r^i(z_k)$ at the end of the game, where z_k denotes a terminal history. With imperfect information, player 2 is unsure about where she is in the game. This means that there are multiple trajectories that could have led to where she is, and there is no way of telling between them (in order to make a good decision about pay-offs).

Having established this framework, it can be shown that Markov games are a restricted case of extensive-form games. We shall not go into this, however. It is readily apparent from the ideas regarding the chance agent as well as the information states that this setting can accommodate randomness as well as uncertainty/imperfect information.

We would now like to remark that there have been attempts to extend Q-learning to multiple agents. In fact, this work was done very closely to the development of reinforcement by Littman [15] (in 1994). Littman's work focused on two competitive agents, which means that the author did not have to deal with the challenges that we are exploring here. Moreover, the extension was straightforward in that the state value function was defined through mini-max arguments, the Q-function defined using this state value function, and the update equation for the Q-function as written above except for the part where we have two actions–instead of one. An important aspect of this work is that the agents were trained separately, and they were merely evaluated against each other. The training for each agent was a process of learning defense against worst-case scenarios, hence the mini-max nature of the state value function. This is in contrast to modern approaches where the agents are usually trained simultaneously. To make this more precise, this work assumes that there are two agents with corresponding actions and action spaces $a \in A$ and $o \in O$. They then define

$$V(s) = \max_{\pi \in A} \min_{o \in O} \sum_{a \in A} Q(s,a,o)\pi_a,$$

which is the value of the state s, and from this define

$$Q(s,a,o) = R(s,a,o) + \gamma \sum_{s'} T(s,a,o,s')V(s').$$

The key update on the algorithm is given by

$$Q(s,a,o) \leftarrow (1-\alpha)Q(s,a,o) + \alpha(r + \gamma V(s')),$$

and the algorithm itself follows the steps of algorithm 1. We give this explicitly in Algorithm 2.

Algorithm 2: 2 agent Q-Learning

```
First initialise the step size $\alpha \in (0,1]$ for updates, and then choose some $\epsilon > 0$ for
  exploration-exploitation;
Initialise the joint Q function arbitrarily, except for terminal states, which get
  value 0;
foreach episode do
    Proceed by first initialising the state, S;
    foreach step of episode do
        Choose actions for the agent (A) and its opponent (O) given the state
          using a policy based on the value function (this can be $\epsilon$-greedy);
        Step in the environment using the joint action and then observe the
          reward $r$, as well as the state change $S'$;
        Do the Q-learning update for this setting
          $Q(S,A,O) \leftarrow (1-\alpha)Q(S,A,O) + \alpha[r + \gamma V(s')$;
        Update the state variable;
    end
end
```

Example Applications

The motivation for studying multiagent reinforcement learning is that these methods can be applied within a surprisingly wide variety of contexts. For instance, Liu et al. [31] propose the use of reinforcement learning as a tool to automate feature selection. To give context, feature selection is a pre-processing step for machine learning methods wherein the goal is to determine the features that will be most useful for a downstream task–for example prediction. Done well, the benefits of this procedure include dimensional reduction for the problem, better results and more explainable models. The problem is highly non-trivial, however. Similarly, Kim et al. [32] present a procedure for exploration, for the broader machine learning context. This method focuses on building embedding representations of states and actions, which can then be used to guide exploration. This is done via a mutual information (to be defined in detail in section 2.3) theoretic lower bound optimisation.

While the examples above seem abstract (in that they are not ordinary everyday applications), there are many concrete examples for the usage of multiagent reinforcement learning. For example Hsu et al. [33] uses the idea of mutual information in a seemingly competitive setting of 'target tracking'. While the modelling here is similar to what we are pursuing, it differs since the approach demonstrated adds more complexity to the policies and value functions than is desirable for our goal of truly understanding the mechanism that develops coordination in our context. Further applications include Shamsoshoara et al. [34] who develop a distributed structural procedure for spectrum sharing in the context of drone technology; and Yao et al. [35] who considered, in the same spirit, a collaborative multiagent reinforcement learning anti-jamming method for wireless networks–a single-agent case is given in Elleuch et al. [36] for comparison. Even more concrete applications include work done towards traffic control [37], drug target prediction [38], liquidation strategy analysis [39], and logistic networks [40], to name but a few.

Interestingly, there is a body of work that seeks to relate the way in which we view multiagent settings with society. This can be observed from these works by the usage of language, as well as the motivation for the methods. For example He et al. [41] present an approach which look into the discovery of skills in multiagent systems with cooperative agenda. Their optimisation encodes skills as latent variables and maximises the mutual information between these variables as well as the combined states of all agents. By so doing, the authors provide insights into how coordination comes about in the context of multiagent particle environments–where particle agents can move in various ways, communicate where needed, as well as interact with each other or landmarks in the environment. Lazaridou et al. [42] argue that the ability to cooperate through language is a defining feature of humans. This work then makes a case for the importance of understanding how language comes about (in the sense of the conditions that enable this) as well as how it evolves. The benefits of this are presented as two-fold : 1. This could give us insights into human evolution. 2. This would allow us to create more flexible networks for solving problems in the AI space.

Coordination, mentioned above, is of immense interest to us. Coordination is related to communication, but they are not the same thing. After the following section, we explore this relationship from the lens of existing literature. The following section introduces entropy and mutual information, as well as ideas of regularisation using these quantities. We will see that these are useful for this discussion about communication and coordination.

2.3 Entropy and Mutual Information

We would like to lay down some information theoretic foundations for upcoming discussions here. We have briefly introduced the context of reinforcement learning, and in what follows we introduce the ideas of entropy and mutual information, which we will see come up when we discuss entropy regularised objectives as well as coordination through regularisation. We follow the discussion in [43].

First we recall that given a probability space, a random variable is a map that takes elements on the sample space to the finite real numbers, $\mathbb{R}$, with probability one. Learned-Miller described the entropy of a random variable loosely as a quantity that characterises the unpredictability of the random variable. This is the same intuition that one gets from statistical mechanics [44] wherein given the microstates of a system Ω, the entropy is defined as the quantity

$$S = \log \Omega.$$

The intuition here is that a larger system with more microstates that are equally likely will have a higher entropy, which means that the more microstates there are, the harder it becomes to predict the true state of the system, and vice versa.

We would like to define information theoretic entropy more formally (in a mathematical sense) using probability theory. We shall focus on discrete variables here, but the theory is extensible to continuous variables. We let X be a random variable that takes values in some set Ω. Then the entropy of X is defined as

$$H(X) = -\sum_{x \in \Omega} P(x) \log P(x),$$

where P is the probability measure. The continuous analogue of this is known as differential entropy and, for a probability density $f(x)$ with support X on the real line, it looks like

$$h[f] = \mathbb{E}_X(-\log(f(x))).$$

An example here is a fair coin toss in which case

$$H(X) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{2}\log\frac{1}{2},$$

since $\Omega = \{T, H\}$ (T for tails, and H for Heads), and given that in a fair coin toss $P(X = T) = 1/2 = P(X = H)$.

Another quantity of interest is mutual information, which we shall see much later in a practical setting. Learned-Miller defines this quantity as that which measures the relationship between two simultaneously sampled random variables. It measures the expected amount of information communicated across the variables. He phrases this intuition in the form of a question, and says one might ask : "How much does one random variable tell me about the other?"

Formally, we let X and Y be two random variables whose joint distribution is $P(X, Y)$. We define the mutual information of X and Y by

$$I(X;Y) = \sum_{x\in\Omega_x}\sum_{y\in\Omega_y} P(x,y)\log\frac{P(x,y)}{P(x)P(y)},$$

where X takes values in Ω_x and Y similarly in Ω_y.

The analogue for continuous variables can be written as

$$I(X;Y) = \mathbb{E}_{(X,Y)}\Big(\log\frac{P_{(X,Y)}(x,y)}{P_X(x)P_Y(y)}\Big).$$

Figure 2.4 showcases the relationship between entropy and mutual information visually.

One should note that mutual information is a positive quantity. Furthermore, that

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X).$$

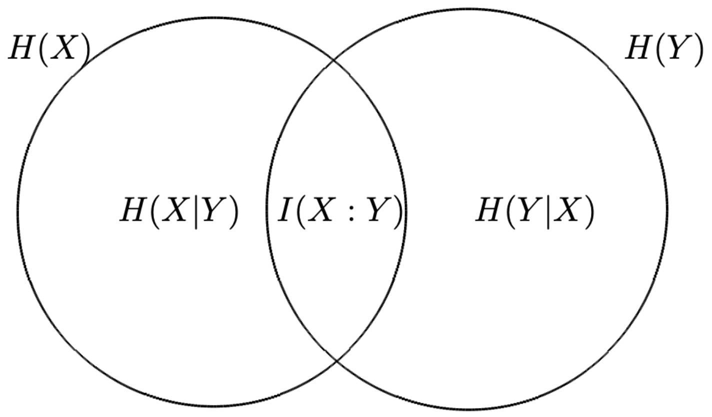

Fig. 2.4 Entropy and Mutual Information [2]. This figure depicts the relationship between entropy, conditional entropy and mutual information given any two random variables X and Y. The mutual information is at the intersection of the individual entropies, and union of the entropies is referred to as the joint-entropy of the random variables. We also observe conditional entropies.

In reinforcement learning, agents are incentivised with rewards to learn, as we saw previously. This is what is often referred to as classical reinforcement learning. The ideas we have discussed here are seen in reinforcement learning via regularisation, which we discuss briefly below. Regularisation is the addition of an extra term in an optimisation process which injects preferences into the model. For example, given an arbitrary objective function L, a neural network/model with weights w, and $\alpha \in [0,1]$ we can define a modified loss function

$$L_\alpha = L + \alpha||w||.$$

Here we are still optimising the original loss function, however this regularisation forces the model to consider small weights. The degree of this preference is controlled using α. In the case of entropy or mutual information, we define the modified objectives as

$$L_\alpha = L + \alpha H(\cdot),$$

and

$$L_\alpha = L + \alpha I(\cdot;\cdot),$$

respectively. Speaking more concretely, classical reinforcement learning seeks to find

$$\pi^* = \text{argmax}_\pi \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}(r(s_t,a_t)).$$

However, we can modify this to become

$$\pi^* = \text{argmax}_\pi \sum_t \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}(r(s_t,a_t)) + \alpha H(\pi(\cdot|s_t)),$$

which is the so-called maximum entropy reinforcement learning objective [20], and we will see precisely how this comes about in Section 2.5. Mutual information based regularisation follows in the same fashion with the difference being that the regularising term is related to mutual information. It is also worth mentioning that, in what follows, when we talk about lower bounds of mutual information, we are talking about all functions f such that $f(\cdot;\cdot) \leq I(\cdot;\cdot)$. The quality of each bound depends on how well it approximates mutual information, given the same variables, and lower bounds are important as they tend to offer trade-offs between accuracy and computational speed [45]. To give an example, again, Kim et al. [46] propose an objective

$$J(\pi) = \mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t\Big(r(s_t,a_t) + \alpha\sum_{(i,j)} I(\pi^i;\pi^j)\Big)\right],$$

where (i,j) are agent pairs. They propose to optimise this via a derived lower bound

$$J(\pi^i) = \mathbb{E}_\pi\left[\sum_{t\geq 0}\gamma^t\Big(r(s_t,a_t) + \alpha H(\pi^i(\cdot|s_t)) + \frac{\beta}{N}\sum_{i\neq j}\left[\log q(a_t^i|a_t^j,s_t) + q(a_t^j|a_t^i,s_t)\right]\Big)\right],$$

where $q(a_t^i|a_t^j,s_t)$ approximates $p(a_t^i|a_t^j,s_t)$ in the dynamics of the Markov game. We see that the first two terms are the usual reward and entropy pair. The last term accounts for the approximation of mutual information, and they show where this comes from in the paper.

To give a glimpse into what will follow, we ask the question : What can we do with these ideas? To start, we know that entropy is helpful for exploration from Liu et al. [47] who show that exploration can be improved through off-policy learning whose behaviour policies are driven by entropy regularisation. Ma et al. [48] study interactions between agents as well as how to best leverage them. They propose an algorithm to understand these agent interactions from the perspective of policy and value functions. The policies they consider are trained with an entropy regularisation term. In similar vein, there is work surrounding strategy building. Strousse et al. [49] and Jiang et al. [50] show how to learn effective strategies in multiagent systems that are either cooperative or competitive where there is no access to a model of the environment or encoded expectations of how the agents should interact with each other. Their approach is rooted in the idea of allowing the agent the freedom of revealing its intentions or not, using a mutual information-theoretic regularising term. We will see more applications that are much closer to our research direction in the next section.

2.4 Notions of Communication and Coordination

In what follows, we will look into communication, coordination as well as the relationship between them. We first note that communication can be used to achieve coordination, so we discuss this relationship. It will be clear from this literature survey why we would like to consider methods that do not explicitly demand communicative structures. Following this, we will move on to discuss the implicit ways of achieving coordination–through regularisation. Communication may be defined as an exchanging of information while coordination may be defined as a harmonious functioning of different parts of a system for effective results. Ideas around communication in multiagent reinforcement learning have been around for at least 29 years [13]. Earlier work in this space was much less organised, and we provide an overview of the ideas in this paragraph before specialising in those that follow. One of the earlier papers by Tan [13] (as previously mentioned) performed studies towards answering questions about the success of agents who can communicate compared to those who cannot in cooperative games. This paper, with its concept of sensation, was one of the first to hint towards communication. The sensations in a game with two hunters, H_1 and H_2, were defined as signals that H_1 will receive should H_2 be in proximity to a prey to facilitate the coordination in capturing the prey. In earlier work there was belief that it is sufficient for agents to be 'aware' of each other to establish coordination [51], however in many applications this is not true. For example in Wei et al. [20] the agents are aware of each other (in the same sense as Claus et al. [51]) but they still cannot solve the differential game problem. This inspired researchers to look at coordination which is facilitated by communication. In this setting, as an example, Szer et al. [52] present an algorithm for cooperative multiagent settings. The key focus on this work is the attainment of correct coordination. Their approach leads to a communication protocol which allows the agents to coordinate better. Forms of communication include signaling [53], intent propagation [54], imagination [55], and we can even add a layer to these strategies towards vote/consensus-based learning [56]. The relationship between coordination and communication is indeed subtle, for instance Havrylov et al. [57] study two agents engaging in a referential game and, fundamentally, develop a way of communicating as needed to succeed in the given game, which makes one think that some coordination is necessary to establish communication. So the relationship is not one-way, as the above might have suggested.

Let's now talk about the structures required for communicating. The mechanics of communication have been established in many ways, as the above has hinted. Often the easiest way is to let the agents share some network that is being used in learning then observe emergent communication behaviours in cooperative tasks [58]. A wide variety of networks can be used here, and an example is the use of graph neural networks [59]. We would like to mention that this approach is not limited to graph neural networks, and we mention Shen et al. [59] as an example out of many such methods that use a variety of neural networks. We argue that this approach seems contrived since we would like our agents to have some form of independence from each other. In this direction of decoupling aspects of the learning process, Eccles et al. [60] attempt to ease the need to centralise training in order to establish communication where this problem arises. In short, the authors introduce inductive biases for signalling and listening, which turn out to be sufficient for the problems with which the paper is concerned. This is in a similar spirit to Hu et al. [61] and Wang et al. [62] who optimise communication under limited bandwidth settings through signalling mechanisms. These are standard practice mechanisms that are used to establish communication.

It should be clear from the examples above that communication can be leveraged for good when we would like to coordinate. However, it is hard to communicate–in the sense that it is expensive to do so–in a decentralised setting (due to limited bandwidth, noisy channels [63], etc.). On the other hand, centralised structures are not natural for real world problems. Another reason for the strong dislike for centralised structures is demonstrated in van der Heiden et al. [64] (very similar to Li et al. [65]) who argue that multiagent reinforcement learning agents trained in a centralised way can be brittle because they can overfit to their training partners. This produces agents that adopt policies that act under the expectation that the other agents in the environment will act a certain way rather than react to their actions. They propose a social empowerment framework to ease the problem, which is achieved through a mutual information theoretic approach. This is similar to

Jaques et al. [66] who do something similar but through the lens of causality. Mutual information based optimisation as a means to disentangle learning dynamics in multiagent reinforcement learning has been studied, for example, by Krupnik et al. [67], and we will explore more of this idea in what follows as well as its implications. Notice that in what follows we will loosely refer to 'other agents' as opponents, even though we are working in the cooperative setting. We take this terminology from Wen et al. [6] and Tian et al. [7].

What we hope to put forward, moving on, is a discussion of how to harness the ideas of mutual information within the multiagent setting, and to hint that small differences between the ideas lead to big differences in modelling power. For example, as we will do as well, Kim et al. [46] use mutual information for coordination but optimise a lower bound, and we will see that optimising lower bounds can lead to instability. More than this the mutual information optimisation is done with an assumption that the policies of different agents are independent. This is similar to Wang et al. (2020) [68], Wang et al. (2019) [69] and Zheng et al. [70] who optimise lower bounds to solve problems in multiagent reinforcement learning. We will see that lower bounds are not sufficient, and beyond that the assumption that policies are independent is not helpful in our setting. A lot of the work that uses mutual information theoretic ideas in multiagent reinforcement learning leans on assumptions that point to either lower bounds, or policy decompositions that are not reasonable. This is important because when we start talking about the work that we studied, the pool of relevant works becomes very small.

2.5 Entropy Regularisation in Single-agent Reinforcement Learning

2.5.1 Background : Function Approximation in Reinforcement Learning

Very often, we need to shift our focus to reinforcement learning through function approximation instead of using tables and arrays to build value functions. In this case, optimisation procedures are required to find the optimalities of these functions. Once such procedure is the so-called gradient descent. For a given function $J : \mathbb{R}^n \to \mathbb{R}^m$ dependent on parameters $\theta \in \mathbb{R}^n$, gradient descent proceeds to find optima through updates

$$\theta_{k+1} \leftarrow \theta_k + \alpha \nabla J(\theta).$$

For reinforcement learning, we can take the state value function as the objective to optimise. In a sense, if we do this, we are saying that we want to only visit states that will yield the optimal reward. With this, we want to demonstrate what the form of this update might look like using an example. Sutton et al. [12] show that in this case, we get

$$\nabla J(\theta) = \nabla V_\pi(s) = \nabla \Big[\sum_a \pi(a|s) Q_\pi(s,a) \Big],$$

for all states s, which follows from the definitions of the value functions. They also show that

$$\nabla \Big[\sum_a \pi(a|s) Q_\pi(s,a) \Big] = \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|s) Q_\pi(s,a),$$

where $Pr(s \to x, k, \pi)$ specifies transition probabilities from state $s \to x$ in k steps under the given policy, where we have basically expanded the details of the equality. From here, we can then re-write this as

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) Q_\pi(s,a) = \mathbb{E}_\pi \Big[\sum_a Q_\pi(S_t, a) \nabla \pi(a|S_t, \theta) \Big],$$

where μ denotes the distribution of the states.

We can stop here and take satisfaction in this form. However, we want to show a well-known update, called REINFORCE, and also introduce what the so called actor-critic methods are doing. In order to do this, we need the following sequence of observations

$$\begin{aligned}
\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) Q_\pi(s,a) &= \mathbb{E}_\pi \Big[\sum_a Q_\pi(S_t, a) \nabla \pi(a|S_t, \theta) \Big] \\
&= \mathbb{E}_\pi \Big[\sum_a \pi(a|S_t, \theta) Q_\pi(S_t, a) \frac{\nabla \pi(a|S_t, \theta)}{\pi(a|S_t, \theta)} \Big] \\
&= \mathbb{E}_\pi \Big[\sum_a Q_\pi(S_t, A_t) \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \Big] \\
&= \mathbb{E}_\pi \Big[\sum_a G_t \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)} \Big],
\end{aligned} \tag{2.1}$$

where $G_t \equiv \sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1})$. The second step essentially multiplies and divides by $\pi(a|S_t, \theta)$, while the third step samples A_t from $\pi(a|S_t, \theta)$. Lastly, we just use the short form for the returns to express the expectation in a clean form. Now, this leads to the so-called REINFORCE update

$$\theta_{k+1} \leftarrow \theta_k + \alpha G_t \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)}.$$

This is important because we see that algorithms can then be defined based on the form of their value functions, as well as how they optimise the objective function. Moreover, we want to introduce the so-called actor-critic methods. Notice that we can consider an update

$$\theta_{k+1} \leftarrow \theta_k + \alpha(G_t - b(S_t)) \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)},$$

for any function $b(s)$ that is independent of the actions, and this leaves the expected value of the updates unchanged. We can choose this function to be the state value function, which is what happens in actor-critic algorithms. Here we have to be a bit careful with the returns for the case n-step methods. For one-step methods, the update becomes

$$\theta_{k+1} \leftarrow \theta_k + \alpha(G_{t:t+1} - v(S_t)) \frac{\nabla \pi(A_t|S_t, \theta)}{\pi(A_t|S_t, \theta)},$$

where

$$G_{t:t+1} = R_{t+1} + \gamma v(S_{t+1}).$$

For $n > 1$ one has to consider what is known as generalised returns, but the idea is that we need to define carefully what the expected returns should be. What we have gotten to here is an update rule for actor-critic methods. The essence, then, of these methods is that there is a learned policy and value function, and this value function is used for bootstrapping, as opposed to only being used as a baseline whose estimate is being updated. When we talk about actor-critic methods moving forward, these are the fundamental ideas that we will have in mind.

2.5.2 Reinforcement Learning as Inference

Reinforcement learning can be regularised with an entropy term that encourages achieving a task in a more robust sense – by this we mean that with reasonable deformations in the environment, the agent can still perform the main task as it will have learnt a variety of ways to achieve it [3]. Eysebach et al.[3] contains the following insightful graphic, Figure 2.5, which demonstrates robustness.

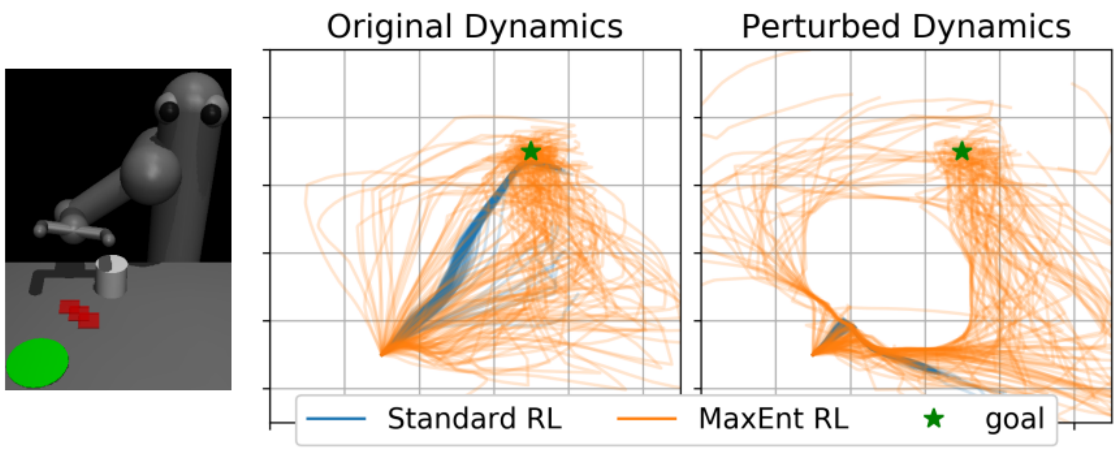

Fig. 2.5 A demonstration of robustness by Eysebach et al.[3]. On the left we have a figure with a robot performing a maniputation task. The left-most plot shows trajectories learned by maximum entropy reinforcement learning and standard reinforcement learning while the right most shows the agent's trajectories given perturbations on the environment.

They used standard reinforcement learning and maximum entropy reinforcement learning to learn a manipulation task (left) without obstacles, but added obstacles (red squares) during evaluation. They then plot the policies learnt without the obstacle (center) and with (right). We see that the maximum entropy regularised policy is still useful even with the environmental disturbances. In this section we discuss why entropy regularisation is such a natural thing to consider. In order to do this, we cast the reinforcement learning problem as an inference problem on a probabilistic graphical model. This treatment follows Levine [71] closely. The treatment establishes reinforcement learning as amortised inference, in the sense of Millidge et al. [16] who give classifications of reinforcement learning based on how they go about the optimisation. Amortised inference procedures learn a parameterised function which maps directly from states to the parameters of the approximate posterior. This is in constrast to iterative methods that directly optimise the parameters of the approximate posterior, a process which is carried out for each data-point. The more theoretical framework is discussed in Fellows et al. [72].

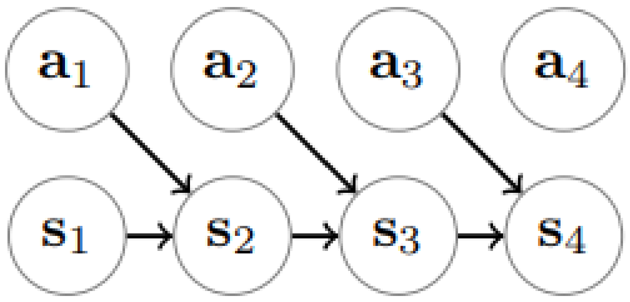

Fig. 2.6 Graphical model with states and actions. Here we have process $X = \{X_t\}$ that takes states s_t, and the transitions between these states are dependent on the action variables a_t.

We consider the relationship in Figure 2.6. This corresponds to a model with factors that take the form $p(s_{t+1}|s_t,a_t)$. This corresponds to Figure 2.1, in the sense that have a given state and the agent can take an action that will take it to a next state with a given probability. In order to encode for optimality, we consider the graphical expression in Figure 2.7. The variables, $\mathcal{O}_t$ are optimality variables which take values in $\mathbb{Z}_2$, where $\mathcal{O}_t = 1$ means that the action taken at that time step under consideration is optimal, and the 0 value denotes suboptimal action .

If the distribution over this variable is chosen to be

$$p(\mathcal{O}_t = 1|s_t,a_t) = \exp(r(s_t,a_t)),$$

where the rewards $r(s_t,a_t) \leq 0$, then we find the following, intuitive, posterior distribution over actions, which conditions on $\mathcal{O}_t$ for all $t \in \{1,\cdots,T\}$:

$$\begin{aligned} p(\tau|\mathcal{O}_{1:T}) \propto p(\tau,\mathcal{O}_{1:T}) &= p(s_1)\prod_{t=1}^{T} p(\mathcal{O}_t = 1|s_t,a_t)p(s_{t+1}|s_t,a_t) \\ &= p(s_1)\prod_{t=1}^{T} \exp(r(s_t,a_t))p(s_{t+1}|s_t,a_t) \\ &= \left[p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t,a_t)\right] \exp\left(\sum_{t=1}^{T} r(s_t,a_t)\right). \end{aligned}$$

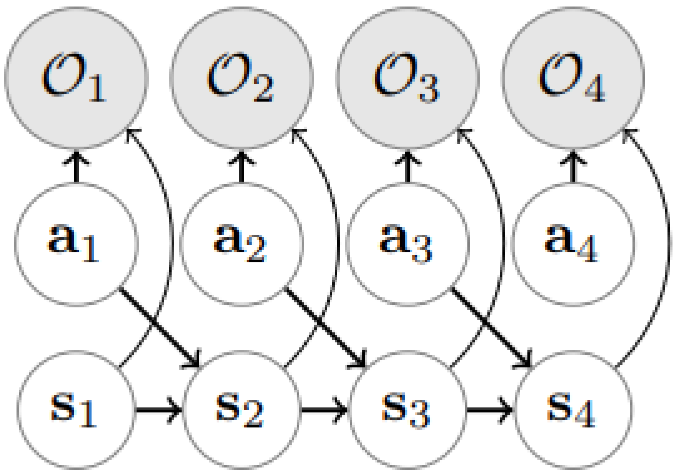

Fig. 2.7 Graphical model with optimality variables. Here we have process $X = \{X_t\}$ that takes states s_t, and the transitions between these states are dependent on the action variables a_t. Moreover, we have variables $\mathcal{O}_t$ which tell us whether the actions chosen yield optimal transitions.

In the case of deterministic dynamics, one finds that

$$p(\tau|\mathcal{O}_{1:T}) \propto 1_{[p(\tau)\neq 0]} \exp\Big(\sum_{t=1}^{T} r(s_t, a_t)\Big).$$

When we speak of the optimality variable moving forward, we shall assume interest in the case where $\mathcal{O}_t = 1$, and in equations we shall omit this specification for simplicity. The optimal policy, in the graphical model setting, can be recovered in the following way. We compute backward messages $\beta_t(s_t, a_t) = p(\mathcal{O}_{t:T}|s_t, a_t)$, which can be interpreted as the probability of optimality of a trajectory from time step t to T where that trajectory starts at s_t with action a_t. We define

$$\beta_t(s_t) \equiv p(\mathcal{O}_{t:T}|s_t) = \int_{\mathcal{A}} p(\mathcal{O}_{t:T}|s_t, a_t) p(a_t|s_t) da_t = \int_{\mathcal{A}} \beta_t(s_t, a_t) p(a_t|s_t) da_t,$$

where this quantity specifies the probability of optimality of a trajectory given a starting state instead. The factor $p(a_t|s_t)$ is an action prior, and can be set to being a uniform distribution based constant over the action set as it is not conditioned on $\mathcal{O}_{1:T}$. Using a recursive backward message passing algorithm to compute $\beta_t(s_t, a_t)$ from time step $t = T$ to $t = 1$ one finds that for the base case $p(\mathcal{O}_T|s_T, a_T) \propto \exp(r(s_T, a_T))$, and the recursive case is $\beta_t(s_t, a_t) = \int_{\mathcal{S}} \beta_{t+1}(s_{t+1}) p(s_{t+1}|s_t, a_t) p(\mathcal{O}_t|s_t, a_t) ds_{t+1}$. Here message passing algorithms are types of iterative decoding algorithms that factorise a global function of many variables into product of simpler local functions, whose arguments are the subset of variables. The backwardness refers to the fact that we start at time $t = T$ to $t = 1$. From here, one finds the optimal policy as

$$p(a_t|s_t, \mathcal{O}_{t:T}) \propto \frac{\beta_t(s_t, a_t)}{\beta_t(s_t)}.$$

We can then ask the question: what objective are we optimising when doing the above? In the deterministic case, we can adopt an approximate inference procedure where we want to find an approximation $\pi(a_t|s_t)$ such that the trajectory distribution $\hat{p}(\tau) \propto 1_{[p(\tau)\neq 0]} \prod_{t=1}^{T} \pi(a_t|s_t)$, matches the above example of the deterministic case. We can do this by minimising the KL-divergence.

$$\begin{aligned}
D_{KL}(\hat{p}(\tau)||p(\tau)) &= -\mathbb{E}_{\tau\sim\hat{p}(\tau)}[\log p(\tau) - \log \hat{p}(\tau)] \\
&= -\mathbb{E}_{\tau\sim\hat{p}(\tau)}\Big[\log p(s_1) + \sum_{t=1}^{T}(\log p(s_{t+1}|s_t, a_t) + r(s_t, a_t)) \\
&\quad - \log p(s_1) - \sum_{t=1}^{T}(\log p(s_{t+1}|s_t, a_t) + \log \pi(a_t|s_t))\Big] \\
&= -\mathbb{E}_{\tau\sim\hat{p}(\tau)}\Big[\sum_{t=1}^{T} r(s_t, a_t) - \log \pi(a_t|s_t) \\
&= -\sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim\hat{p}(s_t,a_t)}[r(s_t, a_t) - \log \pi(a_t|s_t)] \\
&= -\sum_{t=1}^{T} \big(\mathbb{E}_{(s_t,a_t)\sim\hat{p}(s_t,a_t)}[r(s_t, a_t)] + \mathbb{E}_{s_t\sim\hat{p}(s_t)}[\mathcal{H}(\pi(a_t|s_t))]\big).
\end{aligned}$$

In the case of stochastic dynamics, one then observes that the optimised distribution is $\hat{p}(\tau) = p(s_1|\mathcal{O}_{1:T}) \prod_{t=1}^{T} p(s_{t+1}|s_t, a_t, \mathcal{O}_{1:T}) p(a_t|s_t, \mathcal{O}_{1:T})$ where the initial state distribution is conditioned on optimality. The KL-divergence is then

$$D_{KL}(\hat{p}(\tau)||p(\tau)) = -\mathbb{E}_{\tau\sim\hat{p}(\tau)}\Big[\log p(s_1) + \sum_{t=1}^{T} r(s_t, a_t) + \log p(s_{t+1}|s_t, a_t)\Big] - \mathcal{H}(\hat{p}(\tau)).$$

What we have shown here is that we can formulate reinforcement learning as inference on a graphical model. We did this in a very natural way wherein we merely wanted to devise dynamics that are qualified as optimal, and all we had to do was apply probability theory. The point is that maximum entropy reinforcement learning is not a contrived notion. In practice, it is usually preferable to include a coefficient α that multiplies the entropy term in order to control the trade-off between maximising entropy and maximising the main reward.

While the approach presented is very convenient and powerful it has been shown that it has its weaknesses [73]. This work shines a light on problems that arise generally when we blindly take reinforcement learning as inference over a graphical model, and it then asks the question: "Under what conditions, can we make this assumption (that reinforcement learning is equivalent to inference over a graphical model) for best utility?" The best utility case being that the problem doesn't inherently become computationally intractable, which forces us to use approximative procedures.

Moving forward, this regularising coefficient will be included in our equations. The question we ask next concerns the value functions and policies that arise in this setting. We want to write them reasonably explicitly, and see how we can lift this context to involve multiple agents.

2.5.3 From Maximum Entropy to Energy-Based Policies

In the previous section, we introduced the concept of entropy regularisation in reinforcement learning. In this section we will have a more in depth discussion on this context, focusing specifically on results from [20]. The following theorem tells us that given value functions defined using the maximum entropy objective, we have a policy that takes on a very specific (energy) form. Interestingly, in addition to these energy-based policies we get the so-called soft Q-function and soft V-functions, which correspond in effect to the state-action value function and state value functions (respectively) that we saw previously.

Theorem 1. *Let the soft Q-function be defined by*

$$Q^*_{soft}(s_t, a_t) = r_t + \mathbb{E}_{(s_{t+1},\cdots)\sim\rho_\pi}\left[\sum_{l=1}^{\infty}\gamma^l(r_{t+l} + \alpha\mathcal{H}(\pi^*_{MaxEnt}(\cdot|s_{t+l})))\right],$$

and soft value function by

$$V^*_{soft}(s_t) = \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q^*_{soft}(s_t, a')\right) da'.$$

Then the optimal policy for the maximum entropy objective is given by

$$\pi^*_{MaxEnt}(a_t|s_t) = \exp(\frac{1}{\alpha}(Q^*(s_t, a_t) - V^*_{soft}(s_t))).$$

Proof. (Appendix A.1, [20]) □

Remark : The soft V-function is said to be soft because it is only when $Q(a_t, s_t)$ is large that $V(s_t) \approx \max_{a_t} Q(s_t, a_t)$, otherwise it is said to attain a soft-max. The soft Q-function is said to be soft in a similar sense.

The next two theorems give us assurances that we can still look at things in a Bellman sense as we do in classical reinforcement learning. We see that, given the value functions stated above, we obtain notions of convergence that are very similar to what we know from the classical setting.

Theorem 2. *The soft Q-function given above satisfies the soft Bellman optimality equation*

$$Q^*_{soft}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s}[V^*_{soft}(s_{t+1})],$$

where the optimal soft value function is also as given previously.

Proof. (Appendix A.2, [20]) □

Theorem 3. *Let $Q_{soft}(\cdot, \cdot)$ and $V_{soft}(\cdot)$ be bounded and assume that $\int_{\mathcal{A}} \exp(\frac{1}{\alpha} Q_{soft}(\cdot, a'))da' < \infty$ and that $Q^*_{soft} < \infty$. Then the fixed-point iteration*

$$Q_{soft}(s_t, a_t) \leftarrow r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s}[V_{soft}(s_{t+1})], \forall s_t, a_t$$

$$V_{soft}(s_t) \leftarrow \alpha \log \int_{\mathcal{A}} \exp\left(\frac{1}{\alpha} Q_{soft}(s_t, a')\right) da', \forall s_t$$

converges to Q^*_{soft} and V^*_{soft}, respectively.

Proof. (Appendix A.2, [20]). □

Practically, however, what one wants to do is convert the above operation into a stochastic optimisation problem expressing the soft value function (for which we can assume that the corresponding neural network is parametrised by parameters θ) as an expectation via importance sampling. Moving forward, we will often say that a function is parametrised by a set of parameters as a short-hand for the statement that the neural network we use to approximate that function is parametrised using those parameters.

The idea behind importance sampling is that if we want to compute an expectation $\mathbb{E}(f(x)) = \int f(x)p(x)dx$, for an arbitrary function $f(x)$ and density $p(x)$, we can instead compute $\mathbb{E}(f(x)) = \int \frac{f(x)p(x)}{q(x)} q(x)dx$, where $q(x)$ is some density. Notice that the second expectation is taken with respect to the distribution of $q(x)$, and this distribution could be easier to work with than $p(x)$. We want to choose $q(x)$ such that $q(x) = 0$ implies $f(x)p(x) = 0$. We call $q(x)$ the importance distribution and $p(x)$ the target distribution. Using this very same idea, we can express the state value function as

$$V^{\theta}_{soft}(s_t) = \alpha \log \mathbb{E}_{q_{a'}} \left[\frac{\exp(\frac{1}{\alpha} Q^{\theta}_{soft}(s_t, a'))}{q_{a'}(a')} \right],$$

with $q_{a'}$ being a possibly arbitrary distribution over actions.

Observing that for arbitrary functions g_1 and g_2 on domain X we have that $g_1(x) = g_2(x) \forall x \in X \iff \mathbb{E}_{x \sim q}[\frac{1}{2}(g_1(x) - g_2(x))^2]$. Hence, the soft Q-iteration is obtained via an equivalence by optimising

$$J_Q(\theta) = \mathbb{E}_{s_t \sim q_{s_t}, a_t \sim q_{a_t}} \left[\frac{1}{2} \left(\hat{Q}^{\bar{\theta}}_{soft}(s_t, a_t) - Q^{\theta}_{soft}(s_t, a_t) \right)^2 \right],$$

where the q_{s_t}, q_{a_t} are positive over the states and actions, respectively, and

$$\hat{Q}^{\bar{\theta}}_{soft}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s}[V^{\bar{\theta}}_{soft}(s_{t+1})]$$

is the target Q-value with the $t+1$ value estimate given by the above estimate of the value function.

Remark: Observe that the form of policy given is quite general, and sampling from such a policy can be difficult. One overcomes this by using a method called Stein Variational Gradient Descent (*SVGD*), which will be described in detail later, to

minimise
$$D_{KL} = \Big(\pi_\theta(\cdot|s_t)||\exp(\frac{1}{\alpha}Q^*_{soft}(\cdot,s_t) - V^*_{soft}(s_t)\Big).$$

We leave the discussion of this procedure for later, in a specially dedicated section, since it is of special importance for this study. In order to generalise this to multiple agents one follows the algorithm below [5].

Algorithm 3: Soft Q-Learning for Multiple Agents [5]

```
Initialise: The joint Q function, agent policies, the regularisation coefficient $\alpha$,
  and annealing settings.
for episodes={0,.., M} do
    Update the Q function using soft Q-learning.
    for agent ={1,..., N} do
        Update joint policy for each agent through SVGD.
    end
    if episode $\geq t$ then
        anneal the regularisation coefficient $\alpha$
    else
        pass
    end
end
```

While this has covered entropy regularisation, mutual information based regularisation seems to help in single-agent reinforcement learning too, for example Zhao et al. [74] who consider an objective that links states that are tied to the goals with states that the agent has control over. The link is established through mutual information-theoretic optimisation, and this inspires the agent to leverage whatever control it has over the environment for best use. This is one example out of many more.

Moving forward, we see from the algorithm produced above that once one has obtained results for the single-agent case, extending them to multiple agents is reasonably straightforward, but there are subtleties as we will see in our studies. The pseudo algorithms usually seem sensible and robust, but how these are translated into code matters a lot. For example, it is also possible to do optimisation in this context of energy-based policies through the so-called soft-actor-critic approach (SAC) [75] [76]. Upon inspection one can see that the ideas in these papers are not themselves new, and have been around for at least 9 years [77]. It is, however, the way in which they are executed with modern machinery (in the form of tricks, and optimisation methods) that has made them powerful, which is the point that we want to make. We will not take this explicit direction, however. Next, we want to discuss the environment in which we shall be doing our work as well as challenges that are encountered there-in.

2.6 Relative Overgeneralisation in Team Games

When agents fail to coordinate owing to the learned joint policy being sub-optimal, this leads to value function estimates of what would be optimal actions being incorrect (underestimates), and is known as action shadowing.

Palmer et al. [23] define relative overgeneralisation as a type of action shadowing which occurs when the expected reward from a sub-optimal Nash Equilibrium ends up being higher than the optimal equilibrium. In the two agent case this means that if one player is playing optimally, then the other one can deviate from optimality and the expected joint returns will still be higher on average for the sub-optimal equilibrium, by our definition of a Nash Equilibrium. While the origins of this name are unclear, one may think of this effect as one which points towards phenomena where agents are prone to wrongfully overestimate the significance of an optima over competing alternatives.

Following the work by Palmer et al. and Matignon et al. [78], an equilibrium policy $\bar{\pi}$ is shadowed by a policy $\hat{\pi}$ in a state s whenever there exists some agent i whose deviation from the equilibrium $\bar{\pi}$ incurs a greater cost than the worst cost possible for deviating from $\hat{\pi}$.

Next, we define a Pareto-optimal strategy/equilibrium as a joint policy $\hat{\pi}$ where no agent can deviate without this having adverse effects on the returns of at least one agent in the game. To say that π is Pareto-dominated by $\hat{\pi}$ is equivalent to saying that for all states the value functions corresponding to π are greater than or equal to those of $\hat{\pi}$, and, in addition, there exists at least one state where the inequality is strict[78][23]. A Pareto optimal strategy then is one that is not dominated by any other. Now we can define relative overgeneralisation as an occurrence where agents converge to a non-Pareto optimal Nash Equilibrium due to shadowing.

Let us consider the following examples demonstrating these concepts, starting with Table 2.1. In this very easy context, agents that use an average-based learning scheme will reason that since $\sum(A,j) < \sum(C,j)$ and $\sum(B,j) < \sum(C,j)$ then C should be preferred for any of the other agent's actions, where the indices are over columns

		Player 2		
		A	B	C
	A	(11,11)	$(-30,-30)$	(0,0)
Player 1	B	$(-30,-30)$	(7,7)	(6,6)
	C	(0,0)	(0,0)	(5,5)

Table 2.1 This table shows an example of an iterative game where the state is always the same but the agents need to iteratively take playing turns. Both players choose from three actions A, B and C, and the table shows the pay-offs associated with these decisions in the form $(r_{player-1}, r_{player-2})$. Each agent tries to maximise its pay-offs in this game.

and rows. Then both agents will move towards the shadow equilibrium (C,C). If an alternate move is played with some small probability, then the expectation is that player 1 will move from C to B and so will player 2. However they will not move any further as they will have reached a Pareto-dominated sub-optimal Nash equilibrium. One can similarly derive a partially stochastic example (see [23]).

Secondly, the diagram in Figure 2.8 is another example from [5] that demonstrates a simple example of where this pathology can be observed in continuous coordination games. Since this is the context we use to discuss all ideas moving forward, and that we use for the testing, we will devote a susbsection to it.

2.6.1 Differential Game : Max of Two Quadratics

Let's start by looking at Figure 2.8, which shows an environment that we have mentioned is of immense interest to us. This figure shows what is known as a differential game, or more precisely The Max of two Quadratics. It is an environment with two quadratic modes (peaks) that also correspond to the reward function [79]. Two agents are tasked with the duty of finding the mode with the highest pay-off between the two, with one agent controlling the y-position and the other controlling the x-position. There are a couple of things to note about this environment. Firstly, the agents have a continuous action space of $[-10,10]$, and each agent's reward is defined as

$$R^1(a^1,a^2,s) = R^2(a^1,a^2,s) = max(f_1,f_2),$$

where

$$f_1 = 0.8\Big(-\Big(\frac{a^1+5}{3}\Big)^2 - \Big(\frac{a^2+5}{3}\Big)^2\Big),$$

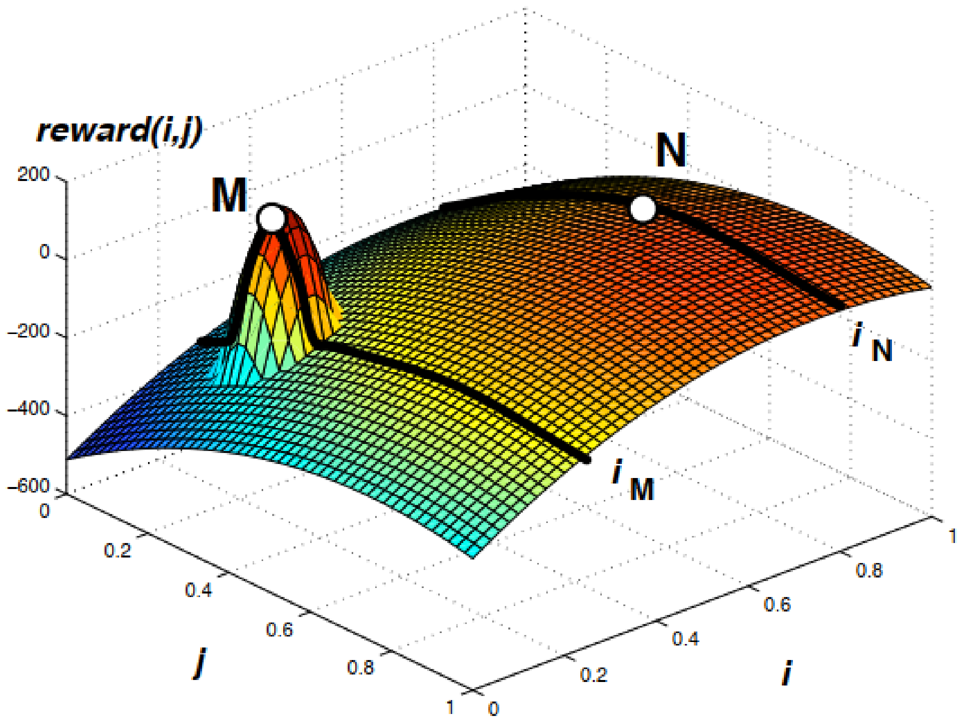

Fig. 2.8 Max of two quadratics/The differential game. Here we see two maxima, M (global) and N (local). This surface defines the reward space for the agents, which is also what the agents see as they navigate. The agents start at (0,0), and they need to find the global maximum.

and

$$f_2 = \left(-(a^1 - 5)^2 - (a^2 - 5)^2 \right),$$

for any state s, using the same notation for actions and rewards' labelling as before.

The reward is independent of state here, and so the state structure that the agents receive can be arbitrary, and is set to a zero vector in practice. This surface gives us a local maximum that has value 0 at $(-5,-5)$ and a global maximum that attains value 10 at $(5,5)$. The fact that the agents share the reward function value is what defines this context as being cooperative. This necessitates cooperation between the two agents for the reward function to attain maximum value.

Recall that in Subsection 2.1.1, we saw the k-armed bandit problem. We can think of the current problem as an extension of this problem in two directions :

- There are two agents instead of one that need to work collaboratively to maximise the total expected reward.
- We are working with $k = 2$ levers, but each agent only controls one.
- Lastly, the range of values that each lever can take is a continuous closed and bounded subset of the reals $[-10,10]$.

In a more concrete sense, one can imagine that there are two people, each with a lever which can be placed anywhere from $[-10,10]$. They don't have an idea of the reward surface but they (jointly) get a reward based on where each of them has placed it. This connection is made to establish the fact that we are still within the reinforcement learning paradigm, but working with continuous spaces and multiple agents as opposed to the classical single agent, discrete space problem.

Recall that we mentioned in Section 2.1.1 that we have a correspondence between our context and a context where states are relevant. In particular, suppose that we have an agent in a one dimensional setting $[a,b], a, b \in \mathbb{R}$ $(a < b)$ where the agent chooses actions, a, and gets rewards $R(a,s) = R(a) \in [c,d]$, for some $c, d \in \mathbb{R}$ $(c < d)$. In this case we can imagine the states corresponding to the value of $R(a)$. Hence, we imagine the agent as a probabilistic process in the codomain $[c,d]$, where the state is in one-to-one correspondence with the action value. In this sense, it still makes sense to talk about states even though the reward function itself is independent of the state variable. In the same vein, we can think about the reward surface for our differential game problem as being a determinant of the state of the agents, i.e., we can say that the agents take actions a^1 and a^2, which lands them at a point (a^1,a^2), as a state. In this sense, the state is then the height of the reward function given an input of the domain (a^1,a^2).

In this setting, the agents will similarly tend to learn to move towards the suboptimal equilibrium, N, due to the average payoff there being higher in comparison to that corresponding to the optimal equilibrium, M, by construction. We are seeing here that these simple settings pose a reasonable problem for agents that are learning to coordinate, and we want to investigate the key component that allows agents to conquer this latter task. In what follows, we want to have a discussion about some methods of sampling from general policies. We will study these first then move on to regularisation mechanisms, which will shed insight into what is sufficient for solving the given problem (given the mentioned challenges).

2.7 Samplers and Sampling

Recall that one of our research questions (sub-research question 1) asked if the unsatisfying performance observed on the multiagent soft Q-learning method could be due to sampling inefficiently from our energy-based policies. Previously, we also mentioned that the method used to sample from the given general policy is the so-called Stein Variational Gradient Descent (*SVGD*). In what follows, we shall discuss this method as well as alternatives to it. Since these methods perform well in the literature in terms of accuracy and efficiency, it seems reasonable to think that if the source of the problem is, indeed, sampling inefficiency then we will be able to see good performance for one of them.

We start by introducing the necessary context of reproducing kernel Hilbert spaces, and then we delve into discussing the methods. Reproducing kernel Hilbert spaces are important to us because the basic theory that we need to talk about efficient samplers is based on elements of these spaces. In supervised learning, given a data set X, we can choose a map $\phi : X \to F$ where F is said to be a feature space–usually a Hilbert space. The function ϕ is called a feature map. Given each labelled example (x,y) where y is the label, we can consider $(\phi(x),y)$, and train our model on these transformed pairs. Using feature maps this way can make learning much easier, since the feature maps can be constructed so that they map the data in a meaningful way [80]. However, working with feature spaces can be expensive, and ideas around reproducing kernel Hilbert spaces attempt to mediate this challenge by using kernels instead of explicit feature maps. The theory associated with the efficient samplers of interest exploits these spaces for the computational efficiency that they enable.

2.7.1 Reproducing Kernel Hilbert Spaces

We start by introducing kernel Hilbert spaces following [81]. Firstly, we let X be a nonempty set. A symmetric function $k : X \times X \to \mathbb{R}$ is called a positive-definite kernel on X if $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$ for any $x_i \in X$, $c_i \in \mathbb{R}$, with $n \in \mathbb{N}$ and the indices going from 1 to n.

It can be shown that if we are given a kernel as above then there is a Hilbert space H and a map $\phi : X \to H$ such that $k(x,y) = \langle \phi(x), \phi(y) \rangle_H$. In fact, we can go the other way around, and this is an equivalence of statements. This Hilbert space produced here is exactly the so-called reproducing kernel Hilbert space.

It's not easy to see how one can recover such a space, H, and how to define the map ϕ. In order to do this, one considers ϕ as a map from $\mathbb{R}^X$, then defines $\phi(x) = k_x = k(x, \cdot)$. One can now consider the set of all these images of elements of x, then define a space G which is the space of linear combinations of these images. This is a vector space of functions. At this point, one's interest is in defining the scalar product. Given two functions $g = \sum_i \alpha_i k(x_i, \cdot)$ and $f = \sum_j \beta_j k(y_i, \cdot)$ we define the scalar product as $\langle f, g \rangle = \sum_{i,j} \alpha_i \beta_j k(x_i, y_j)$. Lastly, we consider the closure of this space under Cauchy limits. So then the reproducing kernel Hilbert space H is the topological closure of G.

The reproducing property is simply the fact that given a function f as above, then $\langle f, k(x, \cdot) \rangle = f(x)$. In fact, any function, k, from $X \times X \to \mathbb{R}$, and a Hilbert space H with the additional property that $k(x, \cdot) \in H$, in addition to this reproducing property inside the Hilbert space, is a kernel, and the space H is a reproducing kernel Hilbert space.

An example of this is the following:

Consider $k : \mathbb{R}^3 \to \mathbb{R}$ such that

$$k(x,y) = \begin{pmatrix} x_1 \\ x_2 \\ x_1 x_2 \end{pmatrix}^T \begin{pmatrix} y_1 \\ y_2 \\ y_1 y_2 \end{pmatrix},$$

where $x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, and y is similarly defined.

We can realise the reproducing kernel Hilbert space here in the following sense: First we consider $k(\cdot, y) = \begin{pmatrix} y_1 \\ y_2 \\ y_1 y_2 \end{pmatrix} = \phi(x)$ which means that $\phi : \mathbb{R}^2 \to \mathbb{R}^3$.

Secondly we observe that if we consider $f(x) = ax_1 + bx_2 + cx_1 x_2 \equiv \begin{pmatrix} a \\ b \\ c \end{pmatrix}^T \phi(x) = f(\cdot)^T \phi(x)$ then f is a mapping from $\mathbb{R}^2$ to $\mathbb{R}$, and so we see from here that in as much as $\phi : \mathbb{R}^2 \to \mathbb{R}^3$ it induces a map from $\mathbb{R}^2 \to \mathbb{R}$.

We then let H be the space of functions from the two-dimensional plane to the reals. The point above gives us that $k(\cdot, x) \in H$ for all $x \in \mathbb{R}^2$.

We can see that from the second bullet point above that $f(x) = f(\cdot)^T\phi(x) = \langle f, k(\cdot, x)\rangle_H$ which is precisely the reproducing property.

This can be done even with an infinite dimensional space of features with some mild restrictions. Reproducing kernel Hilbert spaces are useful because, amongst other things, they give us a way to do computations that involve feature maps without having to go through the trouble of explicitly defining the feature maps or having to work in spaces that have a greater dimension than the data space.

On a theoretical level once again, we can actually characterise reproducing Hilbert spaces as those spaces (of functions from X to $\mathbb{R}$) whose evaluation functionals are continuous. Functions in these spaces are point-wise well-defined. Furthermore, given a kernel the space that one can produce as done above is unique up to isometries, and given a reproducing kernel Hilbert space the kernel is unique.

2.7.2 Stein Variational Gradient Descent

This quick overview of Stein Variational Gradient Descent (*SVGD*) follows [82] and [83]. This method attempts to estimate a probability density with a finite number of points (which we shall also refer to as particles). This is done by first initialising the n particles arbitrarily, then updating these estimates using the Stein discrepancy, defined in what follows, which measures closeness between the target distribution and the distribution from which the particles are sampled.

Formally, let $p(x)$ be a positive density function $\mathcal{X} \subset \mathbb{R}^d$ that we want to approximate with a set of particles $\{x_i\}_{i=1}^n$. The initialisation of these particles is done with a simple distribution (i.e., easy to sample from) q_0, and, given a step size ϵ as well as a velocity field $\phi(x)$, this is iteratively updated as follows

$$x_i \leftarrow x_i + \epsilon\phi(x_i), \forall i = 1, \cdots, n.$$

The velocity field (or perturbation direction) is chosen to maximally decrease the KL divergence between the distribution of the updated particles and the target. Mathematically, denoting the density of the updated particle $x' = x + \epsilon\phi(x)$ as $q_{[\epsilon\phi]}$ when the density of the original particle x is q, and with $\mathcal{F}$ being the set of perturbation directions to optimise over, we would like

$$\phi = \text{argmax}_{\phi\in\mathcal{F}}\left\{ -\frac{d}{d\epsilon}KL(q_{[\epsilon\phi]}||p)|_{\epsilon=0}\right\}.$$

The set of perturbation directions is taken to be the unit ball of a vector-valued reproducing kernel Hilbert space, which is essentially a product space of regular reproducing kernel Hilbert spaces.

It turns out that the above objective is a linear functional of ϕ, and that

$$-\frac{d}{d\epsilon}KL(q_{[\epsilon\phi]}||p)|_{\epsilon=0} = \mathbb{E}_{x\sim q}[\mathcal{T}_p\phi(x)],$$

where $\mathcal{T}_p\phi(x) \equiv \nabla_x \log p(x)^T\phi(x) + \nabla_x \cdot \phi(x)$.

The latter is a linear operator acting on the perturbation direction, and is known as the Stein operator–in connection with Stein's identity [84]. Stein's identity says that

$$\mathbb{E}_{x\sim q}[\mathcal{T}_p\phi(x)] = 0 \iff p = q.$$

The optimisation problem then is reduced to minimising

$$\mathbb{D}(q||p) \equiv \max_{\phi\in\mathcal{F}} \{\mathbb{E}_{x\sim q}[\mathcal{T}_q\phi(x)]\}.$$

The quantity on the left hand side is defined as the kernelised Stein discrepancy between p and q where the $\phi(x)$ are chosen from the space of functions $k(\cdot, x')$.

The solution to the optimisation problem has a closed form

$$\phi^*(x') \propto \mathbb{E}_{x\sim q}[\mathcal{T}_p k(x, x')].$$

The algorithm for this method starts with an initialisation of particles, then does updates based on the Stein discrepancy for a set number of times l. This is detailed in Algorithm 4.

Algorithm 4: Stein Variational Gradient Descent (SVGD)

```
Initialise: a target density $p(x)$ and a set of initial sample points (initial estimates)
  $\{x_i^0\}_{i=1}^n$.
for iteration $l$ do
    $x_i^{l+1} \leftarrow x_i^l + \epsilon_l\hat{\phi}^*(x_i^l)$ where $\hat{\phi}^* = \frac{1}{n}\sum_{j=1}^n \left[k(x_j^l, x)\nabla_{x_j^l}\log p(x_j^l) + \nabla_{x_j^l}k(x_j^l, x)\right]$
      where $\epsilon_l$ is the step size at the $l$-th iteration.
end
```

2.7.3 Neural Stein Sampler

In the discussion above we saw a quantity that we named the kernelised Stein discrepancy (*KSD*). Here we use neural networks to do density estimation as we have just seen it. The neural network is trained to generate data from the target distribution, using the Stein discrepancy, once again. Now, following [4] : Consider a reproducing kernel Hilbert space (RKHS) of dimension d, say, denoted $\mathcal{H}^d$ where this is associated with a kernel $k(\cdot,\cdot)$, and let $S_q(x) = \nabla_x \log q(x)$. In this setting, it can be shown that if the function space on which the optimisation above is done is the unit ball of the RKHS, the *KSD* has a closed form that we will denote as

$$KSD(p,q) = \mathbb{E}_{x,x' \sim p} u_q(x,x'),$$

where

$$u_q(x,x') = S_q(x)^T k(x,x') S_q(x') + S_q(x)^T \nabla_x k(x,x') + \nabla_x k(x,x')^T S_q(x') + tr(\nabla_{x,x'} k(x,x')).$$

This follows from Liu et al. [85] who derive it through a lemma which says that

$$KSD(p,q) = \mathbb{E}_{x,x' \sim p}[(S_q(x) - S_p(x))^T k(x,x')(S_q(x) - S_p(x))],$$

which can be algebraically manipulated to the given form.

The corresponding optimal discriminative function $\mathbf{f}^*$ such that $||\mathbf{f}^*||_{\mathcal{H}^d} = 1$ and

$$\mathbf{f}^* \propto \mathbb{E}_{x \sim p}[S_q(x) k(x,\cdot) + \nabla_x k(x,\cdot)],$$

which becomes relevant if one wants to do sampling in a generative-adversarial network style [86][4], when the data has a high intrinsic dimension, for instance. We shall not do that here, however, as there is a more straightforward approach through forward passes.

The empirical *KSD* measures the goodness-of-fit of samples $X = \{x_1, \cdots, x_n\}$ to a density $q(x)$. The corresponding minimum variance unbiased estimator can be written as

$$\widehat{KSD} = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i}^{n} [u_q(x_i, x_j)].$$

The context here is the following. We let $q(x)$ denote an unnormalised target density with support $\mathcal{X} \subset \mathbb{R}^d$ and $p_z(z)$ be a reference distribution that generates noise $z \in \mathbb{R}^{d_0}$. We let G_θ be a sampler parametrised by θ. If $p_\theta(x)$ is the underlying density of samples $x = G_\theta(z)$, then we want to learn network parameters θ such that $p_\theta(x)$ approximates the target density well. The algorithm is given below.

Algorithm 5: Neural Stein Sampler (*KSD*) [4]

```
Initialise: an unnormalised target density $q$, generator $G_\theta$, a reference
  distribution $p_z$, number of iterations $N$, learning rate $\alpha$, and mini-batch size $n$.
for iteration $t = 1, \cdots, N$ do
    Generate i.i.d noise samples $z_1, \cdots, z_n \sim p_z$;
    Obtain fake samples $G\theta(z_1), \cdots, G_\theta(z_n)$;
    Compute the empirical discrepancy $\widehat{KSD}(p_\theta, q)$;
    Compute the gradients $\nabla_\theta \widehat{KSD}(p_\theta, q)$;
    Update $\theta \leftarrow \theta - \alpha \nabla_\theta \widehat{KSD}(p_\theta, q)$;
end
```

Moving forward, we shall refer to the sampler method with the acronym *KSD*, and it will be clear when we talk about the sampler as opposed to the mathematical form of the Stein discrepancy.

2.7.4 Learning the Stein Discrepancy

In this section we explore yet another method similar to *SVGD*. Recall that Stein's identity says that

$$\mathbb{E}_{x \sim q}[\mathcal{T}_p \phi(x)] = 0 \iff p = q,$$

where

$$\mathcal{T}_p \phi(x) \equiv \nabla_x \log p(x)^T \phi(x) + \nabla_x \cdot \phi(x).$$

In our *SVGD* framework, we find that the optimal perturbation direction is

$$\phi^*(x') \propto \mathbb{E}_{x \sim q}[\mathcal{T}_p k(x, x')].$$

In higher dimensions, however, kernel methods tend to be difficult to use, and learning Stein's discrepancy is a way to circumvent the difficulties [87]. In particular, one wants to use a critic f_ϕ (replacing the kernel) whose parameters maximise

$$L_\lambda(f_\phi, p, q) = \mathbb{E}_{p(x)}[\nabla_x \log q(x)^T f_\phi(x) + Tr(\nabla_x f_\phi(x))] - \lambda \mathbb{E}_{p(x)}[f_\phi(x)^T f_\phi(x)],$$

with the last term being a regulariser that limits the optimisation problem to functions whose squared norm has finite expectation under the distribution of the data. Notice that what we have on the first term is very similar to the Stein discrepancy. The difference essentially is that we have replaced the kernel by an arbitrary function approximator, f_ϕ, which we make L_2 integrable through the regularising term.

The model itself is then trained to minimise the following objective

$$LSD(f_\phi, p, q) = \mathbb{E}_{p(x)}[\nabla_x \log q(x)^T f_\phi(x) + Tr(\nabla_x f_\phi(x))],$$

called *LSD* which stands for Learned Stein Discrepancy. This is an alternative to minimising the ideal

$$L(\theta) = \sup_\phi LSD(f_\phi, p, q_\theta) - \lambda \mathbb{E}_{p(x)}[f_\phi(x)^T f_\phi(x)],$$

which would be infeasible since the supremum needs to be computed over a large space of functions. This resembles the min-max framework of generative adversarial networks [86]. The formal algorithm is as follows:

Algorithm 6: Learning the Stein Discrepancy [87]

```
Initialise: Critic with network $f_\phi$, a model network $q_\theta$, and the target distribution
  via a density $x = \{x_i\}_{i=1}^n \sim p(x)$, the regularisation parameter $\lambda$
for iteration $t = 1, \cdots, T$ do
    for iteration $t = 1, \cdots, T$ do
        Start by sampling a mini-batch from a buffer $x'$
        Update the $\phi$ parameters using the phi-gradient on the objective $L_\lambda$.
    end
    Next sample another mini-batch from the buffer, $x'$, once again
    Now update the parameters $\theta$ with the theta-gradients for the objective
      $LSD(f_\phi, x', q_\theta)$.
end
```

For our purposes, we shall focus on the model aspect of things. This means that we shall retain the updates for the Q function, i.e optimising

$$J_Q(\theta) = \mathbb{E}_{s_t \sim q_{s_t}, a_t \sim q_{a_t}} \left[\frac{1}{2} \left(\hat{Q}^{\bar{\theta}}_{soft}(s_t, a_t) - Q^{\theta}_{soft}(s_t, a_t) \right)^2 \right],$$

where the q_{s_t}, q_{a_t} are positive over the states and actions, respectively, and

$$\hat{Q}^{\bar{\theta}}_{soft}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1} \sim p_s}[V^{\bar{\theta}}_{soft}(s_{t+1})]$$

is the target Q-value with the $t+1$ value estimate given by the above estimate of the value function. The reason is that the updates implicitly minimise the discrepancy and arguably do so efficiently already given that we are working on the data space. As with the *KSD* method, we shall refer to this sampler as simply *LSD*.

2.8 Regularisation in Multiagent Reinforcement Learning

2.8.1 Probabilistic Recursive Reasoning

In Probabilistic Recursive Reasoning for Multiagent Reinforcement Learning (Wen et al. 2019) [6], the authors introduce an agent-reasoning framework that they call (*PR2AC*) for the multiagent case. Their recursive reasoning hypothesis states that it is beneficial for each agent to, as best as possible, account for how it believes its opponents will respond to its own actions. Notice that this idea of belief modelling is not new, but again the way in which modern, powerful machinery is used makes all the difference. For example, one may want to look at Makino et al. [88] where agents learn policies that incorporate aspects of beliefs (of their coplayers). There is also a level of sophistication with the new approach of Wen et al. however as it not only models beliefs but goes a level deeper. As the authors put it, the agent reasons, "I believe that you believe that I believe ...". While the paper covers only *PR2AC*, Wen Y. [89] covers the comprehensive theory and extensions thereof, the latter of which considers level-k recursive reasoning.

The authors propose that we decompose the joint policy using a Bayes-type rule

$$\pi(a^i, a^{-i}|s) = \pi(a^i|s)\pi(a^{-i}|s, a^i),$$

where the former term, $\pi(a^i|s)$, denotes the agent i's prediction of the appropriate action given the state, and the latter, $\pi(a^{-i}|s, a^i)$, denotes the agent's prediction of how the other players in the game will react to their action of choice. We call this a Bayes-type decomposition since Bayes' rule says that given a probability measure P, and events A, B and C : $P(A, B) = P(A|B)P(B)$. In particular, we can condition C which then yields $P(A,B|C) = P(A|B,C)P(B|C)$, and this looks like the decomposition we have here. To make the learning more in accord with what the real world is, the authors propose that the agent should learn the latter, and so they denote the same decomposition as

$$\pi(a^i, a^{-i}|s) = \pi^i(a^i|s)\rho^{-i}(a^{-i}|s, a^i),$$

from agent i´s perspective, where ρ is a network that is parametrised by ϕ, and we write this as $\rho^{-i}_{\phi^{-i}}(a^{-i}|s, a^i)$ whereas π is parametrised by θ. We can also write the above decomposition from the opponents´ perspective, which yields

$$\pi(a^i, a^{-i}|s) = \pi^{-i}(a^{-i}|s)\rho^i(a^i|s, a^{-i}),$$

and we get that if we devise proper training mechanisms, then both of these approximations should converge to the joint policy as it has been defined.

The next two theorems offer similar insights into what we had in the single-agent case. They are both equivalent statements about the optimisation of the θ-network. The first one is more theoretical concerning the form of the gradients, and the second one gives us a mechanism to estimate these gradients in practice.

Theorem 4. *In a stochastic game, under the recursive framework defined, the update for the multiagent recursive reasoning policy gradient method can be written as :*

$$\nabla_{\theta^i}\eta^i = \mathbb{E}_{s\sim p, a^i\sim\pi^i}\left[\nabla_{\theta^i}\log\pi^i_{\theta^i}(a^i|s)\int_{a^{-i}}\pi^{-i}_{\theta^{-i}}(a^{-i}|s, a^i)Q^i(s, a^i, a^{-i})da^{-i}\right].$$

Proof. See section 4.2 [6] □

Theorem 5. *In a stochastic game, under the recursive framework defined, the update for the multiagent recursive reasoning policy gradient method can be written as :*

$$\nabla_{\theta^i}\eta^i = \mathbb{E}_{s\sim p, a^i\sim\pi^i}\left[\nabla_{\theta^i}\log\pi^i_{\theta^i}(a^i|s)\left[\mathbb{E}_{a^{-i}\sim\rho^{-i}_{\phi^{-i}}}\left[\frac{\pi^{-i}_{\theta^{-i}}(a^{-i}|s, a^i)}{\rho^{-i}_{\phi^{-i}}(a^{-i}|s, a^i)}Q^i(s, a^i, a^{-i})\right]\right]\right].$$

Proof. See section 4.2 [6]. □

In a similar fashion, it can be shown that we can infer $\rho^{-i}_{\phi^{-i}}(a^{-i}|s,a^i)$ via variational inference. Following Levin [71] one finds that

$$D_{KL}(\hat{p}(\tau)||p(\tau)) = -\sum_{t=1}^{T} \mathbb{E}_{\tau\sim\hat{p}}\Big[r^i(s,a^i,a^{-i}) + H(\pi^i_\theta(a^i|s)\rho^{-i}_{\phi^{-i}}(a^{-i}|s,a^i))\Big].$$

This leads to the following theorem.

Theorem 6. *The optimal Q-function for agent i that satisfies the minimisation of the above equation is formulated as*

$$Q^i_{\pi_\theta}(s,a^i) = \log \int_{a^{-i}} \exp(Q^i_{\pi_\theta}(s,a^i,a^{-i}))da^{-i},$$

and the corresponding optimal opponent conditional policy reads

$$\rho^{-i}_{\phi^{-i}}(a^{-i}|s,a^i) = \frac{1}{Z}\exp(Q^i_{\pi_\theta}(s,a^i,a^{-i}) - Q^i_{\pi_\theta}(s,a^i)).$$

This theorem gives us a sense of what the policies we would like to learn look like, which we can use to derive convergence guarantees. This theory leads to two algorithms. The first one is strongly reminiscent of Q-learning as we know it from classical literature but for continuous games and for multiagent settings (Algorithm 7) while the second one is based on the same theory but uses actor-critic machinery (Algorithm 8). We give the algorithms, respectively, as Algorithm 7 and Algorithm 8. Observe that the former does its Q-updates in a way that is consistent with what we deemed Q-learning type updates earlier. The difference here is that now we are considering a case where the agents are not trained separately as Littman [15] did in our previous demonstration. Here, each agent is aware of others, and has to learn to compute the best response of the opponents. This variant works well with a small state space since it can be tabularised. On the other hand, Algorithm 8 employs functional approximation tools with the so-called experience replay to handle higher dimensional state spaces. The actor is updated using policy gradients–modifications of what we saw previously. We see that *SVGD* is used to update the response estimates.

Moving forward, we now look at another implementation that has proven itself as a robust alternative for solving the problem than the others we have seen so far, and we will see that it is closer to what we are trying to implement in terms of a practical

Algorithm 7: MAPR2-Q [6]

```
Start by initialising each agents' Q-function (arbitrarily), and then fix a learning
  rate,$\alpha$, and the usual discount factor.
while not having converged do
    for iteration $t = 1, \cdots, T$ do
        For the present state (for the agent, $i$), sample action $a^i$ based on
          $\rho^{-i}_{\phi^{-i}}(a^{-i}|s,a^i)$ by marginalising over the possible opponent actions.
        Step on the environment with the joint action and observe the received
          reward value $r^i$ and the subsequent state $s$.
        Now, at this point do
            $$Q^i(s,a^i,a^{-i}) \leftarrow (1-\alpha)Q^i(s,a^i,a^{-i}) + \alpha(r^i + \gamma V^i(s)),$$
          and
            $$Q^i(s,a^i) \leftarrow (1-\alpha)Q^i(s,a^i) + \alpha(r^i + \gamma V^i(s)),$$
          where $V^i(s)$ takes the best action $a^i$ after having marginalised
          $Q^i(s,a^i,a^{-i})$ over the opponent actions.
    end
end
```

structure. We will also discuss the differences at a later stage but go over the basic ideas here. We will also give results of implementations as well as reason about why it works better than the other alternative in our discussion section.

2.8.2 Regularised Opponent Modelling with Maximum Entropy Objectives

Tian et al. [7] focus on cooperative multiagent reinforcement learning. The algorithm produced in the paper is called Regularised Opponent Modelling with Maximum Entropy Objectives (*ROMMEO*). We know that in the cooperative setting, the maximum reward is attained if all players work together meaningfully. Letting $\mathcal{O}$ denote the optimality variables in the multiagent context. Agent learning can be guided towards cooperation by allowing each agent to believe that its opponents are playing within the agent's view of optimality. Here, each agent (i, say) can then reason that the probability that its own actions are optimal takes form $P(\mathcal{O}^i = 1|\mathcal{O}^{-i} = 1)$. We can thus define the maximum objective to be optimised as the logarithm of this given probability.

Algorithm 8: Multiagent Probabilistic Recursive Reasoning Q-Learning (*PR2AC*) [6]

```
Initialise the needed parameters of networks $\theta^i$(policy), $\phi^i$ (for opponent model),
  $w^i$ (Q-function weights) for each agent i, and a process N for action exploration.
Next initialise target variables $w^{i'} \leftarrow w^i$ (Q-function), and $\theta^{i'} \leftarrow \theta^i$ (for the policy)
Initialise an empty buffer for each agent, $i$, under naming $D^i$
for iteration $t = 1, \cdots, T$ do
    Initialise the exploration process.
    for each step $t$ do
        For a state $s$, for agent $i$, choose $a^i = \mu^i_{\theta^i}(s) + N_t$ (an action)
        Step on the environment with the joint action then observe the reward,
          next state, and update the buffer for each agent.
        for agent id, $i$ do
            Query the replay buffer for a collection $\{(s, a^i_j, a^{-i}_j, r^i_j, s'_j)\}^N_{j=0}$
            Sample $a^{i'}_j = \mu^{i'}_{\theta^i}$ for all $s'_j$
            Get opponent model samples $\{a^{-i'}_{k,j}\}^M_{k=0} \sim \rho^{-i}_{\phi^{-i}}(\cdot|s'_j, a^{i'}_j)$ for all action
              values $a^{i'}_j$ and state values $s'_j$
            Set
                $$y^i_j = r^i_j + \gamma \frac{1}{M} \sum_{k=0}^{M} Q^i_{\mu^{i'}}(s', a^{i'}, a^{-i'}_{k,j})$$
              Update the critic by minimising
                $$L(w^i) = \frac{1}{N} \sum_{k=0}^{N} (y_j - Q^i_{\mu^{i'}}(s', a^{i'}, a^{-i'}_{k,j}))^2$$
              Next, perform updates for the actor
                $$\nabla_{\theta^i} \eta^i \approx \frac{1}{N} \sum_{j=0}^{N} \nabla_{\theta^i} \mu^i(s_j) \nabla a^i \frac{1}{M} \sum_{k=0}^{M} Q^i_{\mu^{i'}}(s', a^{i'}, a^{-i'}_{k,j})$$
              At this stage compute $\Delta\rho^{-i}_{\phi^{-i}}$ using empirical estimations
                $$\Delta\rho^{-i}_{\phi^{-i}}(\cdot|s, a^i) = \mathbb{E}_{a^{-i}_t \sim \rho^{-i}_{\phi^{-i}}} \Big[ k(a^{-i}_t, \rho^{-i}_{\phi^{-i}}(\cdot; s_t, a^i_t)) \nabla_{\tilde{a}^{-i}} Q^i(s_t, a^i_t, \tilde{a}^{-i}_t)|_{\tilde{a}^{-i}=a^{-i}_t}$$
                $$+ \nabla_{\tilde{a}^{-i}} k(\tilde{a}^{-i}_t, \rho^{-i}_{\phi^{-i}}(\cdot; s_t, a^i_t))|_{\tilde{a}^{-i}=a^{-i}_t} \Big],$$
              where $k$ denotes kernel function as we have introduced them.
            Get the gradient estimates $\hat{\nabla}_{\phi^{-i}} J_{\rho^{-i}}$ then update $\phi^{-i}$.
        end
        Lastly, update the target parameters across all agents.
    end
end
```

This leads to the following lower bound :

$$\log P(\mathcal{O}^i_{1:T}=1|\mathcal{O}^{-i}_{1:T}=1) \geq \sum_t \mathbb{E}_{s_t}[\mathbb{E}_{a^i_t\sim\pi, a^{-i}\sim\rho}[r^i(s_t,a^i_t,a^{-i}_t) + H(\pi(a^i_t|s_t,a^{-i}_t,\mathcal{O}^i_t=1,\mathcal{O}^{-i}_t=1))]$$

$$-\mathbb{E}_{a^{-i}_t\sim\rho}[D_{KL}(\rho(a^{-i}_t|s_t,\mathcal{O}^{-i}_t=1)||P(a^{-i}_t|s_t,\mathcal{O}^{-i}_t=1))]],$$

which essentially takes the standard reinforcement learning objective and subtracts a non-negative term : $\mathbb{E}_{a^{-i}_t\sim\rho}[D_{KL}(\rho(a^{-i}_t|s_t,\mathcal{O}^{-i}_t=1)||P(a^{-i}_t|s_t,\mathcal{O}^{-i}_t=1))$. The inequality follows from precisely the fact that we are making this subtraction. It is not clear what the motivation behind this lower bound is from Tian et al. [7], as they do not define an objective from which they then deduce this logically. We know what it does, in the sense that the KL-divergence regularisation will do distribution matching between the distributions being passed as arguments. However, its origins or motivations are a mystery.

We rewrite this more compactly as

$$\log P(\mathcal{O}^i_{1:T}=1|\mathcal{O}^{-i}_{1:T}=1) \geq \sum_t \mathbb{E}_{s_t}[\mathbb{E}_{a^i_t\sim\pi, a^{-i}\sim\rho}[r^i(s_t,a^i_t,a^{-i}_t) + H(\pi(a^i_t|s_t,a^{-i}_t))]$$

$$-\mathbb{E}_{a^{-i}_t\sim\rho}[D_{KL}(\rho(a^{-i}_t|s_t)||P(a^{-i}_t|s_t))]],$$

where on the right-hand side we are also conditioning on optimality, but this is omitted in notation for compactness. The context is similar to what we had before where the agent is estimating its own policy while also estimating the behaviour of others through ρ. Here P denotes the prior, which is set to the observed empirical distribution of opponents' actions given states. We parametrise the networks as before.

We give the following result that demonstrates essential features of the algorithm that are of interest for comparisons and coding.

Theorem 7. *We define the soft action value function of agent i as*

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

and soft state value function as

$$V^*(s) = \log\sum_{a^{-i}} P(a^{-i}|s)\Big(\sum_{a^i}\exp(\frac{1}{\alpha}Q^*(s,a^i,a^{-i}))\Big)^{\alpha}.$$

The optimal conditional policy and opponent model for the variational problem are respectively

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

and

$$\rho^*(a^{-i}|s) = \frac{P(a^{-i}|s)\Big(\sum_{a^i}\exp(\frac{1}{\alpha}Q^*(s,a^i,a^{-i}))\Big)^{\alpha}}{\exp(V^*(s))}.$$

Remarks :

- The value function can be shown to satisfy Bellman-type equations, in the sense described in Section 2.5.3.
- There is proven convergence for symmetric games with only one global optimum, but not a lot of work has been done beyond this.

The approach to the rest of the algorithm follows actor-critic methodology, and we shall give the objective functions defined and from this gradients can be extracted by analytic computation (if needed), and these are given explicitly in the paper. After this, we shall proceed to state the algorithm for this method in a compact sense, but (again) the explicit version can be found on the paper itself.

First, we parametrise the networks representing the Q-function, conditional policy and opponent model by $Q_w(s,a^i,a^{-i})$, $\pi_\theta(a_t^i|s_t,a_t^{-i})$ and $\rho_\phi(a_t^{-i}|s_t)$. Then we let $a_t^i = f_\theta(\epsilon_t^i;s_t,\hat{a}_t^{-i})$, where $\hat{a}_t^{-i} = g_\phi(\epsilon_t^{-i};s_t)$.

We have the following objective functions. These are derived by looking into the KL-divergence definition, then applying the so-called reparametrisation trick.

$$J_Q(w) = \mathbb{E}_{(s_t,a_t^i,a_t^{-i})\sim\mathcal{D}}[\frac{1}{2}Q_w(s_t,a_t^i,a_t^{-i}) - r(s_t,a_t^i,a_t^{-i}) - \gamma\mathbb{E}_{s_{t+1}\sim p_s}[\bar{V}(s_{t+1})]]^2,$$

where

$$\bar{V}(s_{t+1}) = Q_w(s_{t+1},a_{t+1}^i,\hat{a}_{t+1}^{-i}) - \log\rho_\phi(\hat{a}_{t+1}^{-i}|s_{t+1}) - \alpha\log\pi_\theta(a_{t+1}^i|s_{t+1},\hat{a}_{t+1}^{-i})$$
$$+\log P(\hat{a}_{t+1}^{-i}|s_{t+1}).$$

Observe that this looks like the typical update for the Q-function except for the fact that the form of the state value function has been modified since we use the lower bound given above. Furthermore, for policy optimisation, we can write

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}, \epsilon_t^i \sim N, \hat{a}^{-i} \sim \rho}(\alpha \log \pi_\theta(f_\theta(\epsilon_t^i; s_t, \hat{a}_t^{-i})|s_t) - Q_w(s_t, f(\epsilon_t^i; s_t, \hat{a}_t^{-i}), \hat{a}_t^{-i}), \tag{2.2}$$

This equation ensures that the policy is optimised considering the entropy term (first term) and the rewards computed through the second term.

$$\begin{aligned} J_\rho(\phi) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}, \epsilon_t^{-i} \sim N}[\log \rho_\phi(g_\phi(\epsilon_t^{-i}; s_t)|s_t) - \log P(\hat{a}_t^{-i}|s_t) - Q(s_t, a_t, g_\phi(\epsilon_t^{-i}; s_t)) \\ + \alpha \log \pi_\theta(a_t^i | s_t, g_\phi(\epsilon_t^{-i}; s_t))]. \end{aligned} \tag{2.3}$$

Lastly, this equation optimises entropies of multiple distributions namely, and respectively, the distribution of best responses, a learned distribution for the prior, and the policy entropy. This, in addition to the reward term computed using the Q-function once again. These correspond to all the updates we have to make. We merely need to collect data, and constantly update the parameters using the above equations. We give the simplified algorithm on the next page.

Notice that this is similar to Grau et al. [90] who provided further insights into the modelling of soft Q-learning with mutual information regularisation. The authors show strong ties between mutual information regularisation and standard reinforcement learning optimisation with an extra term to optimise the prior distribution over actions. The argument is that this aids learning by correctly assigning probability mass to actions according to importance. This is a single-agent specification, however, which does not deal with the complexities that we shall see–for example with value functions.

2.9 Evaluation of Previous Work

In this section, we would like to explore results that are of relevance to us in the literature that we have discussed so far. We will start with comparison between two of our samplers, *KSD* by Hu et al. [4] and *SVGD* by Liu et al. [82]. Figure 2.9 demonstrates a toy example with multimodal 2D Gaussian mixtures. The red

Algorithm 9: Pseudo algorithm for *ROMMEO* [7]

```
Initialise the network parameters $\theta^i$ for policies, $\phi^i$ for opponent models , $w^i$ for
  Q functions and $\psi^i$ for the prior estimates for all agents, and an exploration
  process N.
Assign $w^{i'} \leftarrow w^i$ for the joint Q-function.
Initialise learning rates, $\alpha$ and $\gamma$ and replay buffer $D^i$
for iteration $t = 1, \cdots, T$ do
    Initialise the exploration process.
    for each step t do
        Given the present state $s_t$, sample the agent action and
          opponents´action(s) by first selecting an action $\hat{a}^{-i} \leftarrow g_{\phi^{-i}}(\epsilon^{-i}; s_t)$ with
          the noise variable sampled as $\epsilon^{-i} \sim N$
        Then following this by assigning $a_t^i \leftarrow f(\epsilon_t^i; s_t, \hat{a}_t^{-i})$
        Update replay buffer with observed quantities
        Update the replay buffer prior via

            $$\psi^i = \text{argmax}\mathbb{E}_{\mathcal{D}^i}[-P(a^{-i}|s) \log P_{\psi'}(a^{-i}|s)]$$

        for agent id i do
            Get a collection $\{(s, a_j^i, a_j^{-i}, r_j^i, s_j')\}_{j=0}^N$ from the replay buffer
            Compute the gradients.
        end
        Update target network parameters for all the agents.
    end
    After a set number of gradient descent steps, reset the target network
      parameters.
end
```

contour shows the target distribution and the green dots are generated sample(s) / particles at iteration 0, 1k, 2k, 5k and 10k respectively. It is to be noted that this mixture of Gaussians is 'uniform' in the sense that we cannot distinguish between the quadratic components. This is important because in our case, as demonstrated by Figure 2.10, we have only two quadratics but they are very much distinguishable, which is what makes learning in this environment non-trivial. Hence, it is not clear how these two methods should compare in this case in the context of the literature. Experiments in Grathwohl et al. [87] focus on perturbations and RBM size, which does not give us results that we can compare to.

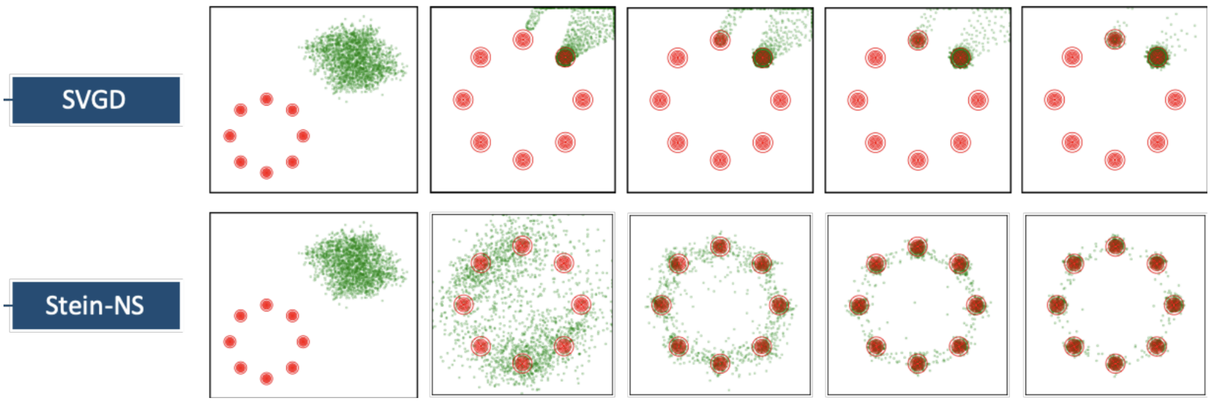

Fig. 2.9 This is a demonstration from Hu et al. [4], which compares *SVGD* with *KSD*. This is a toy example with multimodal 2D Gaussian mixtures. The red contour shows the target distribution and the green dots are generated sample(s) / particles at iteration 0, 1k, 2k, 5k and 10k respectively. Notice that this context in Hu et al. isn't about reinforcement but merely probability density estimation.

Having seen the differential game, we now show what a successful episode of this looks like. Here a successful run is one that discovers the optimal equilibrium. A sample path is shown in Figure 2.10. Below, we show some results on this environment from the literature with which we are concerned. In the literature, the number of steps for training in this problem is set to a maximum of 25 while the number of epochs has varied from 150 in Wei et al. [5] to 350 in Wen et al. [6]. The first 1000 steps were used for exploration then learning begins after with annealing coefficient (to balance reward and entropy maximisation) set to 0.5 and a noise variable set to 0.1. Let's now look at results from the previous work.

Wei et al. [5] show Figure 2.11 which demonstrates the training curves of multiagent soft Q learning (*MASQL*) and the baseline at the time, *MADDPG* (multiagent deep deterministic policy gradient learning). The solid lines demonstrate averages over

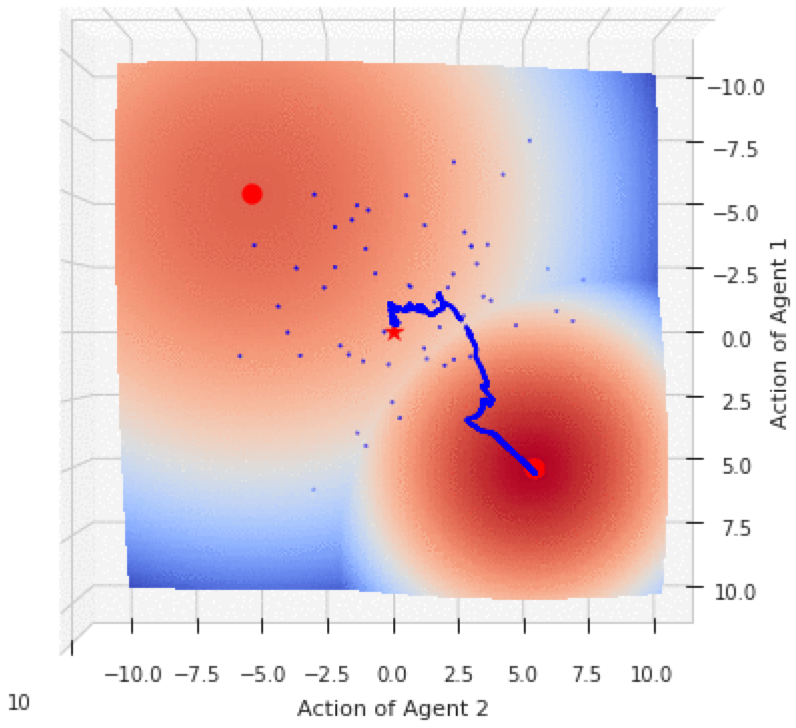

Fig. 2.10 Differential Game Demo. The colour gradient represents height differences, and the dark-blue line represents a learned trajectory by the agents from the initialisation. The dots that are not on the line represent sampled actions during the exploration phase.

50 independent trials. We see that MADDPG did not do well, at all, converging on average to the zero reward optimum. On the other hand, we see that *MASQL* convergences more often, and it is said to converge 72% of the time.

On the other hand, Wen et al. [6] plot training curves for their method probabilistic recursive reasoning (*PR2AC*) given in Figure 2.12. This method is compared to *MADDPG* variants as well as *MASQL*. Here however, there are no details about which runs are plotted here. It is evident that single runs are plotted, and there is no interest in observing how often each of these methods converges, or does not, which is in contrast to the previous example by Wen et al. We see that the other methods (baselines) seem inadequate, and converge to suboptimality for the particular runs. However, in light of the single run focused results it is hard to say more than this here.

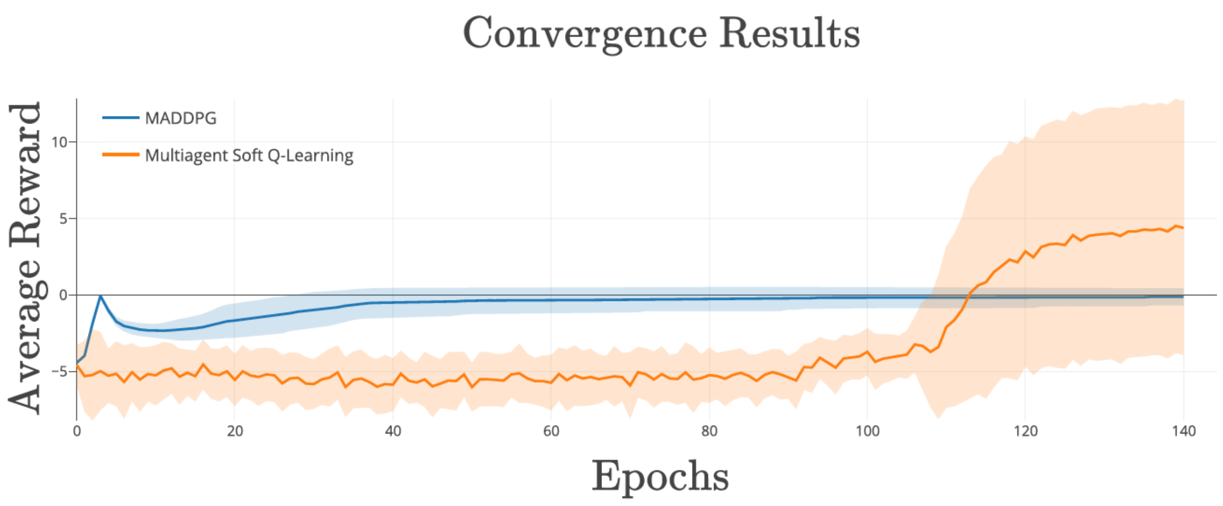

Fig. 2.11 Results for *MASQL* [5]. This figure shows training curves for MADDPG and *MASQL*. We see that *MADDPG* only converged to the suboptimal solution, while *MASQL* attained optimality every so often. This is over 50 independent trials.

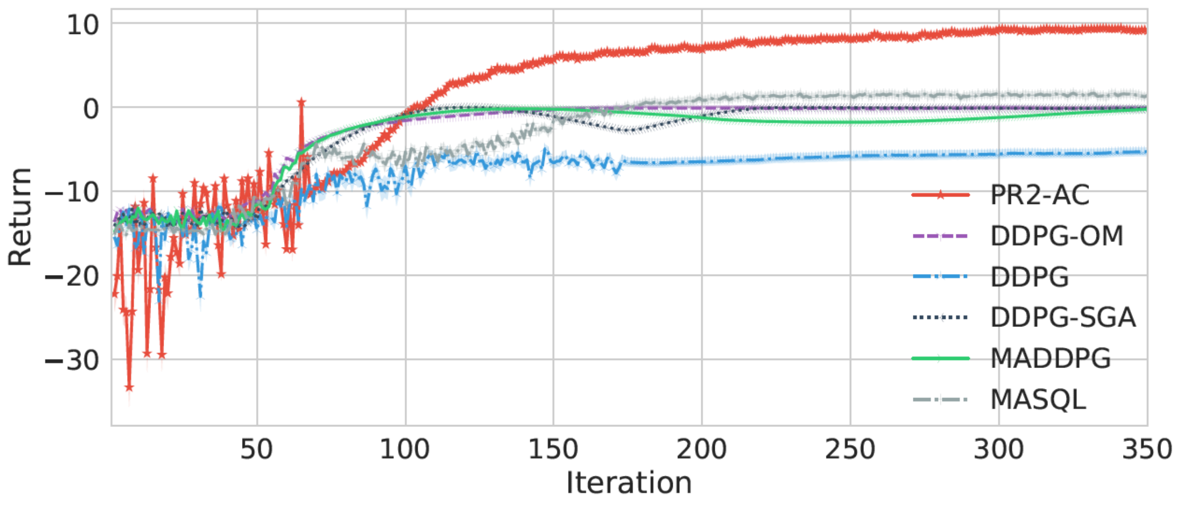

Fig. 2.12 Results for *PR2AC* [6]. We see training curves for *PR2AC*, *MADDPG* variants as well as *MASQL*. We see that on the plotted runs, *PR2AC* does better than all their competitors.

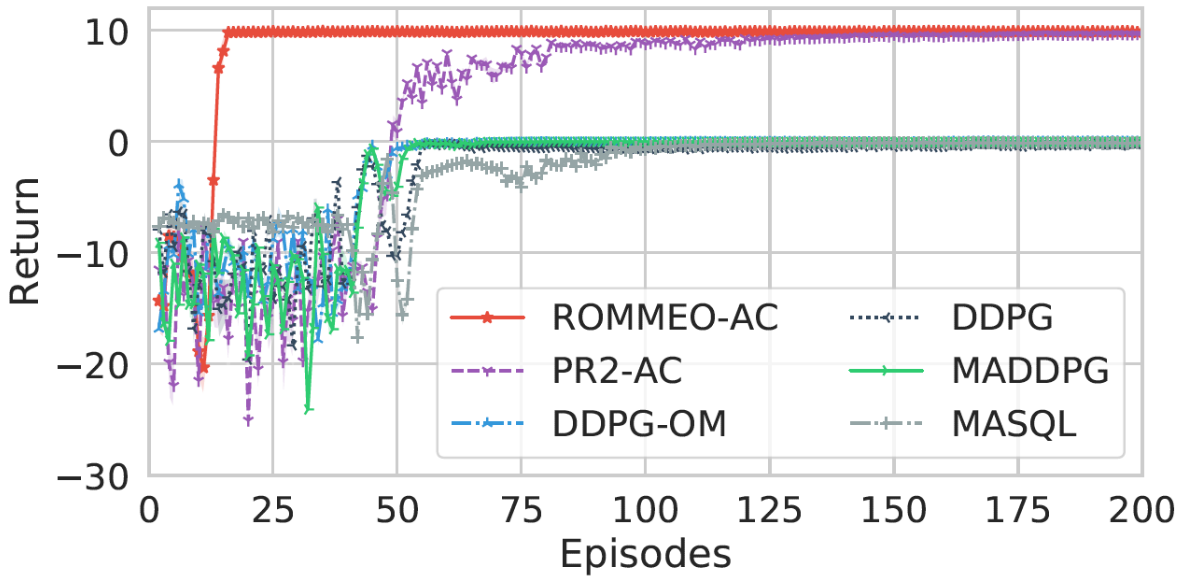

Fig. 2.13 Results from *ROMMEO* [7]. Results for *PR2AC* [6]. We see training curves for *ROMMEO*, *PR2AC, MADDPG* variants as well as *MASQL*. We see that on the plotted runs, *PR2AC* and *ROMMEO* do better than all its competitors.

Lastly, Tian et al. [7] evaluate their method regularised opponent modelling with maximum entropy objective (*ROMMEO*) against *MASQL*, *PR2AC*, and *MADDPG* variants, shown in Figure 2.13. Once again, they plot training results for individual runs. We observe the previous behaviour once again, but this time we observe that the convergence rate of *PR2AC* is slower than *ROMMEO*, and the two methods are the ones that attain the maximal score in their respective runs. The implementation of *ROMMEO* is done via actor-critic methods, hence we should be calling it *ROMMEOAC* to highlight this fact, but we will abbreviate this to *ROMMEO*.

What we see here is that the policy decomposition based methods with regularisation are somewhat effective, in the sense that they can achieve the maximum score when compared to others. It is not clear, however, how these plotted trials were selected. It is also not clear how these methods perform on average, which where are details that are explicitly shown in Wei et al., who introduced *MASQL*. We will take very good care to look at not just convergence for our studies, but also the statistics of convergence behaviour, in the mentioned sense.

In this chapter we introduced reinforcement learning, which we extended to multiple agents. After this, we discussed ideas about mutual information and entropy, whose usage we saw with regards to communication and coordination, and that we also saw when we reframed reinforcement learning as inference. This paved way to entropy regularised multiagent reinforcement learning methods, which we mentioned are of interest to us. Along the way, we also saw density estimation methods that we used to sample from what we referred to as general policies. We lastly introduced the differential game, which is a game comprised of two agents that need to coordinate in order to find a global maximum, and we saw that learning in this context is hard due to the agents being prone to relatively overstate the importance of the local minimum (relative overgeneralisation). What we would like to do moving forward is introduce our methodology for solving the problem, give results, and then end with a discussion. Our methodology takes inspiration from the policy decomposition based alternatives, and we extend these methods by regularising differently than what has been done–using mutual information. We will see that this greatly improves convergence and stability, numerically, and we will also give a rigorous discussion on why this helps.

Chapter 3

Methodology

The goal for this chapter is to introduce the methodology that we used in trying to answer our research questions. Recall that our set-up considers the differential game wherein we have two agents trying to jointly locate a global maximum, from Subsection 2.6.1. We noted that this is hard because agents are prone to overestimating the importance of the suboptimal (local) maximum. Our goal was then to understand the literature surrounding this problem, the nature of the pathology leading to this breakdown, and to extend the ideas in this space. Our research questions were inspired by this, in the context of multiagent soft Q-learning, and we reiterate them below.

Research Question : If we know that the policies we learn should be general enough to model arbitrary distributions, then what is the cause of inconsistent convergence in multiagent soft Q-learning?

Sub-research Question 1 : Assuming that the policies learnt by the agents are general, is the way we sample from these general policies the cause of the problem?

(Corresponding) hypothesis 1 : Assuming that the policies learnt by the agents are general, it must be the case that the way in which we sample from them is inefficient and can be improved by better performing sampling strategies.

Sub-research Question 2 : If the sampling strategies do not improve convergence, then are we simply not learning the policies well (i.e are we not doing the correct optimisation)?

(Corresponding) Hypothesis 2 : If the sampling strategies do not provide insight, then we not doing the correct optimisation for the learning of policies.

Notice that we are interested in soft Q-learning because we believe that soft Q-learning agents should be capable of learning policies given the literature on soft Q-learning [20], and we are interested in getting to the bottom of whatever could be causing this. So, we are aware that there may be other methods that are more different but that solve the problems, but we are interested in discovering the potential flaw in the multiagent extension of soft Q-learning.

Before we begin we would like to discuss some terminology to ensure that our ideas are coming across clearly. In Section 2.7, we discussed samplers which are mechanisms for extracting sample points from a target distribution that is defined. In Section 2.8 we started talking much less about sampling but more about objectives. What's the difference? The difference is that objectives define functions that we want to maximise. Usually, maximising objectives maximises the reward function by construction. In single-agent reinforcement learning, we want an agent to learn a sequence of actions that maximise this reward, and we refer to a function that outputs either this exact sequence or an approximation thereof as the policy. The analytical form of this policy is defined by the objective. In contrast, when we talk about sampling we have a target model (usually approximate) that we want to query in order to get sample points. These sample points enable us to build sample trajectories in order to evaluate the expectations during the learning process. Evaluating these expectations is important since we want to maximise their value in building our optimal behaviour models/optimal policies. To this end, we provide a table that summarises these elements. We state each of the mentioned samplers and objectives mentioned in the literature review as well as the optimisation procedure that we have developed, and give a brief summary about each. All of this information is provided in Table 3.1.

Method		Brief Description
	SVGD	This is a sampling method that learns a target distribution by updating a set of initial estimates in the direction that minimises the Stein discrepancy, via kernels.

	KSD	This is a sampling method that learns a target distribution by updating a set of initial estimates in the direction that maximises the gradient of the Stein discrepancy, via kernels.
	LSD	This is a sampling method that learns a target distribution by updating a set of initial estimates in the direction that minimises the Stein discrepancy, but doesn't use explicit kernels.
	MASQL	This is a model building method that is based on the naive generalisation of Maximum Entropy Reinforcement Learning to multiple agents. This method assumes that the actions are independent given the state.
	PR2AC	This is a model building method that generalises *MASQL*. In contrast to *MASQL*, this method assumes that the policy takes what we have called a Bayes-type decomposition (as opposed to being independent given the state).
	ROMMEO	This is a model building method that specialises *PR2AC*. In contrast to *PR2AC*, the objective here has KL-divergence for regularisation in light of the learned opponent model.
	MI	This is the method that we developed which builds on *ROMMEO*. In contrast to *ROMMEO*, the objective here links the joint policy to the learned model of opponents and the policy of the specific agent.

Table 3.1 This table aims to disambiguate samplers from the actual mechanisms that determine the models/forms of policies that we work with. We start by listing the three samplers that we worked with, and providing brief description of these, then we move on to the optimisation descriptions (that determine the way we learn the policies/behaviour models, as we mentioned above) and, again, describe these briefly.

In our pursuit, we will start by giving a description of the optimisation procedure that we devised for this problem. This will be followed by a description of the experiments we designed based on the subresearch questions. It should be noted that in both experiments we only consider the situation where both agents in a given trial share the same algorithm for learning as these have been proven to be more effective, in our context, in the literature [5][6][7]. This chapter will be slightly short because the results chapter, Chapter 4, will contain quite a lot about the methodological exploration as well. We found that this design allowed us to better present the current work.

3.1 Proposed Extensions of the Information Framework

What we would like to do in this section is establish an optimisation procedure that performs best given the differential game. Mutual information-based optimisation seems reasonable because when each agent, instead of optimising this, only focuses on its own entropy then this may make it hard for the other agent to reason about how its opponent will react to its actions, which we see in our context through the observed inconsistent convergence [20]. Hence, often coordination doesn't follow from this case and, instead, we get inaccurate gradients for non-convex reward functions. However, focusing on maximising the mutual information allows the agents to explore, as needed, while also paying close attention to how their opponents are doing by the definition of this quantity [43].

We propose that the objective that solves this problem in the context of the previous chapter is

$$J^i = \mathbb{E}[r(s_t, a_t) + I(\pi^i(\cdot|s_t); \rho^{-i}(\cdot|s_t, a_t^i))],$$

Notice that again, the state information is not really important in this case, since the reward function only depends on the actions. This is written from the perspective of agent i, looking at the mutual information term. However, we know from the earlier discussion that mutual information is symmetric. Hence, the perspective from which we look at things does not make a difference.

This was inspired by the papers we discussed in Section 2.4 which sought to understand skills discovery, emergence of coordination and communication, respectively, He et al. [41] and Lazaridou et al. [42], who used mutual information based optimisation in this endeavour in Section 2.2. Still in the former section, we saw that Jacques et al. [66], van der Heiden et al. [64] and Krupnik et al. [67] use mutual information to improve agent learning in Section 2.4. Lastly, we saw Kim et al. [46] who also consider mutual information but optimise via derived lower bounds, which we gave in Section 2.3. Motivated by the results obtained in these studies, we modified the objective and considered mutual information instead.

Notice also that we are considering the decomposition of the policy as is given in Wen et al. [6] and Tian et al. [7] so that we do not have the agnostic agents that we had earlier, which we found were insufficient to solve the game consistently when we looked at the original *MASQL* implementation (refer to Chapter 4, Section 4.1). Numerical results also show however that *PR2AC* on its own is also lacking, which we shall see in Chapter 4, Section 4.2. We demonstrate this in the upcoming results' section. Hence, we make this assumption on that basis.

In order to gain more insights from this, we expand the mutual information term, and we see that this will be exactly

$$I(\pi^i(\cdot|s_t);\rho^{-i}(\cdot|s_t,a_t^i)) = \mathbb{E}[\log\pi^i(\cdot|s_t) - \log\rho^{-i}(\cdot|s_t,a_t^i)],$$

in the context of our optimisations. We get this directly from the definition. Following the approach in Haarnoja et al. [20], we find that what this tells us is that agent i should optimise the objective defined by

$$J_\pi(\cdot) = D_{KL}\left(\pi^i(\cdot|s_t,\hat{a}_t^{-i})||\frac{\exp((1/\alpha)Q(s_t,\cdot,\hat{a}_t^{-i}))\rho^{-i}(\hat{a}_t^{-i}|s_t,\cdot)}{Z}\right),$$

while similarly the other agents are seen by this agent as optimising

$$J_{\phi}(\cdot) = D_{KL}\Big(\rho^{-i}(\cdot|s_t)||\frac{(\exp((1/\alpha)Q(s,a^i,\cdot))\pi^i(a_t^i|s_t,\cdot))}{Z'}\Big).$$

Using the reparametrisation trick [91], we can rewrite everything here as

$$J_Q(w) = \mathbb{E}_{(s_t,a_t^i,a_t^{-i})\sim\mathcal{D}}\Big(\frac{1}{2}[Q_w(s,a^i,a^{-i}) - r(s_t,a_t^i,a_t^{-i}) - \gamma\mathbb{E}_{s_{t+1}\sim p_s}[\bar{V}(s_{t+1})]]^2\Big),$$

where

$$\bar{V}(s_{t+1}) = Q_w(s_{t+1},a_{t+1}^i,\hat{a}_{t+1}^{-i}) - \alpha(\log\rho_{\phi}(\hat{a}_{t+1}^{-i}|s_{t+1}) - \log\pi_{\theta}(a_{t+1}^i|s_{t+1},\hat{a}_{t+1}^{-i})).$$

Furthermore, the objective for agent i's policy optimisation becomes

$$J_{\pi}(\theta) = \mathbb{E}_{s_t\sim\mathcal{D},\epsilon_t^i\sim N,\hat{a}^{-i}\sim\rho}(\alpha(\log\pi_{\theta}(f_{\theta}(\epsilon_t^i;s_t,\hat{a}_t^{-i})|s_t) - \log\rho_{\phi}(\hat{a}_t^{-i}|s_t)) - Q_w(s_t,f_{\theta}(\epsilon_t^i;s_t,\hat{a}_t^{-i}),\hat{a}_t^{-i})), \tag{3.1}$$

and that for its opponents becomes

$$J_{\rho}(\phi) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D},\epsilon_t^{-i}\sim N}(\alpha(\log\rho_{\phi}(g_{\phi}(\epsilon_t^{-i};s_t)|s_t) - \log\pi_{\theta}(a_t^i|s_t,g_{\phi}(\epsilon_t^{-i};s_t))) - Q_w(s_t,a_t^i,g_{\phi}(\epsilon_t^{-i};s_t))). \tag{3.2}$$

We are not doing anything complicated here. What we have done is defined our objectives, then we see from this that the agent and its opponent model should be trained by minimising the stated KL-divergences, which translates by definition to the above forms. This is to say that the reparametrisation part comes in when it comes to estimating actions, but the forms of these objective functions themselves are ultimately informed by the decision to consider KL-divergences as they are given above, inspired by Haarnoja et al. [20]. In a sense then, the algorithm should follow [7] closely except the fact that we have new objective functions, and simpler gradients to compute. We give the algorithm in Algorithm 10.

Algorithm 10: Pseudo algorithm for *MI*. Our algorithm.

```
Start by initialising the parameters $\theta^i$ for the policy, $\phi^i$ for the opponent models,
  and $w^i$ for Q function weights for all agents, and an exploration process N.
Assign $w^{i'} \leftarrow w^i$ for the joint Q-function.
Initialise learning rates, $\alpha$ and $\gamma$ and replay buffer $D^i$
for iteration $t = 1, \cdots, T$ do
    Initialise the exploration process.
    for a given step $t$ do
        Given a present state $s_t$, choose an agent action and then proceed to find
          opponents' action(s) via the following assignments :
        Start with $\hat{a}^{-i} \leftarrow g_{\phi^{-i}}(\epsilon^{-i}; s_t)$ where $\epsilon^{-i} \sim N$
        Followed by similarly assigning $a_t^i \leftarrow f(\epsilon_t^i; s_t, \hat{a}_t^{-i})$
        Update replay buffer with observed quantities
        for each agent id, $i$ do
            Query the buffer for a collection $\{(s, a_j^i, a_j^{-i}, r_j^i, s_j')\}_{j=0}^N$
            Compute the gradients $\nabla_{w^i} J_Q(w^i), \nabla_{\theta^i} J_\pi(\theta^i), \nabla_{\phi^i} J_\rho(\phi^i)$.
        end
        Update target network parameters for each agent $i$.
            $$w^i \leftarrow w^i - \lambda_Q \nabla_{w^i} J_Q(w^i)$$
            $$\theta^i \leftarrow \theta^i - \lambda_\pi \nabla_{\theta^i} J_\pi(\theta^i)$$
            $$\phi^i \leftarrow \phi^i - \lambda_\phi \nabla_{\phi^i} J_\rho(\phi^i)$$
    end
    Given a fixed number gradient descent steps, do a reset for the target
      network parameters, $\bar{w}$. Compute
            $$\bar{w}^i \leftarrow \beta w^i + (1 - \beta)\bar{w}^i$$
end
```

3.2 Sub-research question 1 : Experimental Design

Looking into the literature informed us about a promising direction to investigate for the sampling experiments. In particular, sampling via reproducing kernel Hilbert spaces has been established as being very efficient above other known techniques [82]. Hence we focused on techniques that leverage the ideas in this space. Moreover, the key feature of methods in this space is that they exploit Stein's discrepancy, which is a measure of the difference between two distributions that we may think of [83]. The key then is that the literature often tries to optimise objectives by sampling in a way that minimises this discrepancy.

There are three methods that we investigated :

- Stein Variational Gradient Descent : The updates here are based on the first derivative of the KL divergence using kernels.
- Kernelised Stein Discrepancy : The updates here are based on the second derivative of the KL divergence using kernels.
- Learning the Stein Discrepancy : The updates are based on the first order derivative of the KL divergence but the kernel is replaced by a critic.

The important thing for our investigation is that we were interested in finding out which method provided convergence in a way that is consistent. What this translates to is learning to converge to the optimal solution every time. The way in which we did this was as follows :

- code up each strategy for sampling,
- do multiple experiments for the given strategy,
- then looking at aggregated expected returns and their associated statistical variables.

In our case, there is knowledge of exactly how the reward function should behave when we are converging, and so we plotted the average reward function over epochs as well as the average maximum episode return over the epochs as well. The used specifications are as covered in [6], and the number of performed separate runs is 50, from which one would expect to see convergence 36 times [20]. We computed the average for the mean path returns as well as the average for the maximum episode returns to make the plots. The number of iterations/epochs was 2000, with the number of steps bounded by 100 as done in the literature [20][6][7]. This number of epochs is larger than what is used in the already cited literature to give a fair chance of convergence for the methods. We provide the said plots each method, and also show the relative levels of the suboptimal and optimal reward levels where this is relevant.

3.3 Sub-research question 2 : Experimental Design

The experiments for this hypothesis were informed by the literature once again. The first thing to note is that the probabilistic recursive reasoning method by Wen et al. [6] essentially only imposes the Bayes-type decomposition of the policy, wherein the agent models the beliefs of other agents using varying degrees of reasoning. We chose level $k = 1$ because :

- This level gives the agent much less extra information than $k > 1$.
- The paper focused on this level for demonstrating the results in the game, and so this is the fair choice for making comparisons with literature.

Secondly, the method on regularised opponent modelling by Tian et al. [7] is related to probabilistic recursive reasoning in the sense of having this Bayes-type decomposition of the policy, but it adds an extra layer in the optimisation state wherein the agent maximises the reward and entropy while also trying to explicitly sharpen its prior beliefs about the other agents through a regularising KL-divergence term.

Again, we get the sense from these three methods that at each stage we are changing one aspect of multi-agent soft Q-learning, or the subsequent method that was demonstrated to work. It is worth stressing that the other two methods mentioned above, unlike the original extension, were not studied well in terms of convergence behaviour, which is what we are trying to consolidate here. The focus there was on evaluation, but it wasn't clear whether the methods were given a fair chance at convergence, which we strongly believe is important for the question that we are asking.

The method we propose takes the kind of modelling done through the regularised opponent modelling channel, and modifies it. We change the way we optimise the functions of interest by doing mutual information regularisation. So, to be more precise, we only add a regularising term on the reward for mutual information, and we adapt the value functions and policies appropriately as given before, in Section

2.8. In essence, again, we are adding minimal building blocks, and observing the behaviour for convergence.

For the experiments, we performed 50 runs for each model (dropping multi-agent soft Q-learning since it is already inefficient) similar to Wen et al. [20]. The number of epochs was 3000 for probabilistic recursive reasoning, and 1750 for the other two methods (the number of steps was bounded 350). This is higher than Wen et al. [6], Tian et al. [7] and Wei et al. [20]. We wanted to give all the methods a fair chance at convergence, because it was important to us whether a method ultimately solves a problem or not. We then plot the maximum path returns and mean path returns as well as associated standard deviations as before, and observe the behaviour. We also analyse the number of runs that converged to optimum, suboptimum, or did not converge.

The design choice for presenting these results sheds additional elements that will allow us to talk about the quality of our results. We plot the standard deviations because they give us a sense of how the mean path returns behave across the trials. Notice that the standard deviation plots do not show the exact distribution of rewards along the training curve (currently assumed to be normal but it need not be). It does, however, demonstrate clearly the bounds of observed behaviour, and allows us to see whether or not convergence is consistent–which is what we care about. We also choose to use the aggregated maximum path returns. The latter allow us to essentially ask the following question : How often could the agents find optimal trajectories across the trials? If the agents could find optimal paths in their runs, could they also learn to converge with as high probability as we would expect? This seems to be a relevant because it tells us about the efficiency of gradient updates, which is relevant for our second sub-research question.

Having reiterated our research question as well as the sub-research questions we derived from this, and stated how we further proceeded to investigate these questions, our next stop is the results' section where we demonstrate the observations we made while testing our experimental designs. This will be followed by a discussion of these results, after which we will draw our conclusions.

Chapter 4

Results

4.1 Results : Sub-research Question 1

Recall that in terms of the dynamics of the game itself, we expect the behaviour detailed on Figure 4.1 from the agents. This is from a successful run that discovers the optimal equilibrium.

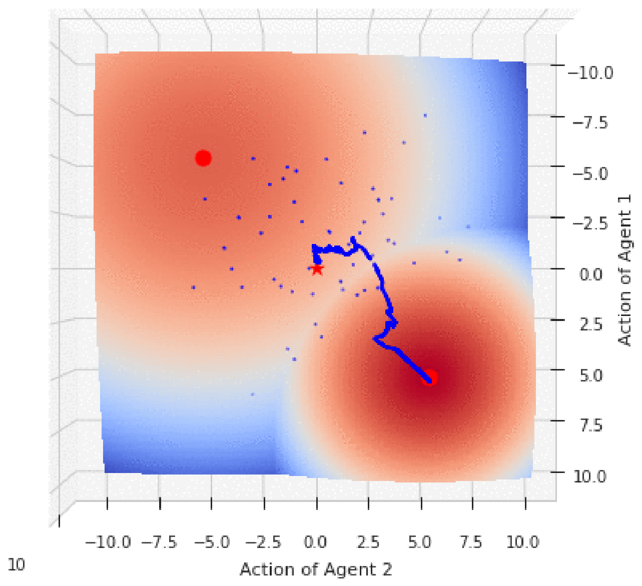

Fig. 4.1 Differential Game Demo. The colour gradient represents height differences, and the dark-blue line represents a learned trajectory by the agents from the initialisation

Below, we provide not visuals of this simulated behaviour (which is sometimes done in the literature), but the associated learning curves of the methods which allow us to comment on not only whether there was convergence–which we know is the case when we achieve a desired score–but also the statistics from independent trials. For this reason, they are the more preferred mechanism for displaying the results.

Now, let's look at aggregated behaviour to understand average performance. In the following (recall from the methodology), we did 50 separate runs, then computed the mean for the plots. We focus on the aggregated mean reward as well as the aggregated maximum reward for our analyses but we also provide standard deviation plots. The number of iterations was 2000 to give a fair chance of convergence, and to also test for stability of the mechanisms. The first method that we look into is *SVGD* whose results are found in Figure 4.2. We plot the averaged rewards over the trials, and the averaged maximum reward attained by some trajectory over the independent trial.

The way by which we obtain this plot is the following. First, we start with aggregated mean and maxima plots for the runs of the methods, i.e. we plot the mean reward using the independent trials, and we also keep the maximum reward attained over the runs (/episodes) of each trial and we take the mean similarly. The first plot for *SVGD* (Liu et al. [82]) demonstrate this in the already mentioned Figure 4.2. What we see here is overlay of the mentioned curve means, with the mean path return in blue and the maximum episode return in green.

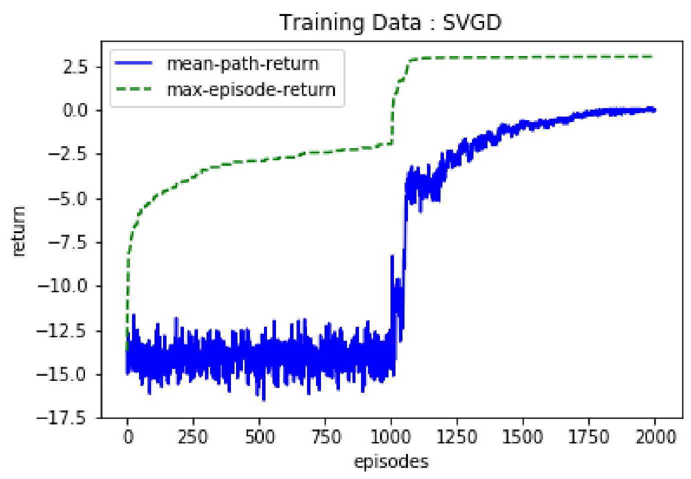

Fig. 4.2 *SVGD* aggregated performance. This figure shows aggregated mean path return curves for 50 runs of *SVGD* (blue line), and it shows the aggregated maximum episode return (green).

This curve shows that the average reward tended towards the zero reward reference, which is the suboptimal equilibrium. This, in our case, means that at programme termination the curves were getting closer and closer to the zero reward value. Looking at the standard deviation associated with the mean path returns, we obtain Figure 4.3.

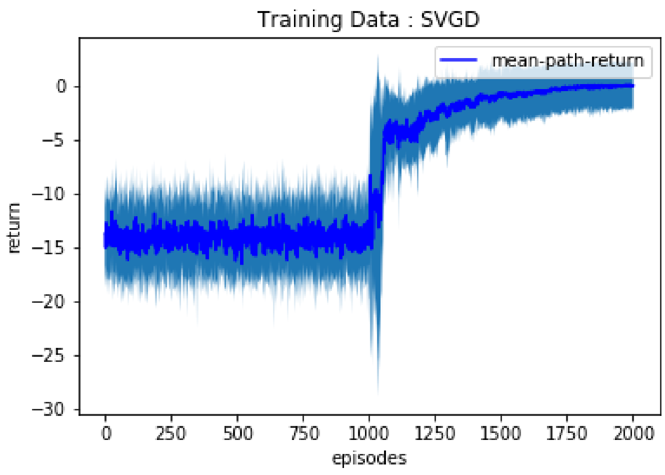

Fig. 4.3 *SVGD* mean path return aggregate and standard deviation. This plot shows the mean path return aggragate over the 50 independent trials (solid blue line) as well as the standard deviation associated with these returns (light blue shading) for *SVGD*.

We see in Figure 4.3 that the convergence of the average around zero (solid blue curve) has an associated standard deviation for each point (demonstrated by the light blue region), which gives us a sense of how different the mean path return outcomes could be for the trials in consideration. The next result we look at, Figure 4.4, is that of the learned Stein discrepancy mechanism.

Here we consider the *LSD*-variant (Grathwohl et al. [87]), where the reward signal is quite noisy, as we can see on Figure 4.4. This method equally did not do too well, as can be seen. In the set of runs intended for this evaluation there was no optimal run. We provide the aggregated mean path returns (blue) as well as maximum episode returns (green).

Here, unfortunately, the results were not as clear as we would have liked. This might have been caused by a couple of things, which we discuss later. We see that the signal was two noisy, and when aggregated we actually get a signal that converges to something slightly less than zero, which is not good. Again, the blue

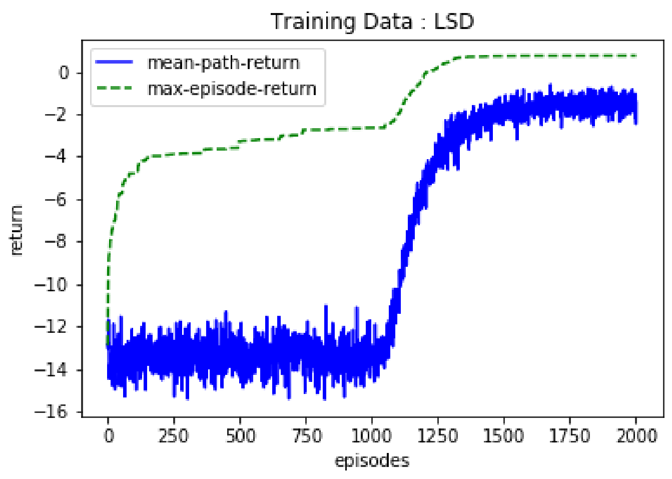

Fig. 4.4 *LSD* aggregated performance. This figure shows aggregated mean path return curves for 50 runs of *LSD* (blue line), and it shows the aggregated maximum episode return (green).

curve represents the mean path returns and the green curve is the maximum episode return.

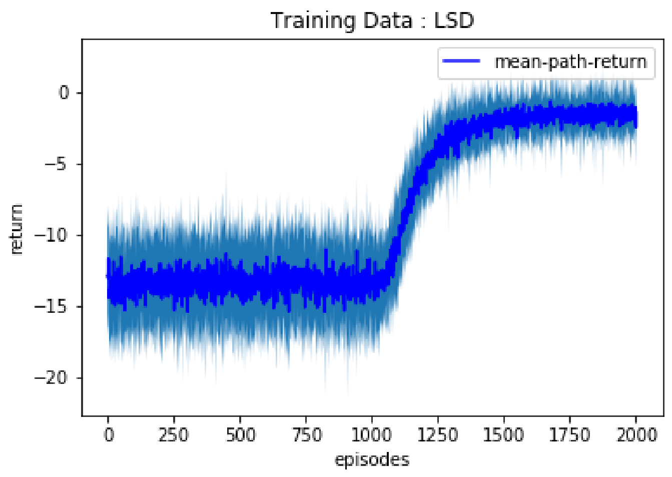

Fig. 4.5 *LSD* mean path return aggregate and standard deviation. This plot shows the mean path return aggragate over the 50 independent trials (solid blue line) as well as the standard deviation associated with these returns (light blue shading) for *LSD*.

Similar, we see from Figure 4.3 that the convergence of the average around zero (solid blue curve) has an associated standard deviation for each point (demonstrated by the light blue region) for the particular method. Here the signal is noisy, as mentioned above but the bounding light blue region seems to fall within similar bounds to what is observed in Figure 4.3. Lastly, the results for *KSD* (Hu et al. [4]) are found in Figure 4.6. We see that the performance is similar to that of *SVGD*. In particular, this method merely makes it to the suboptimal equilibrium.

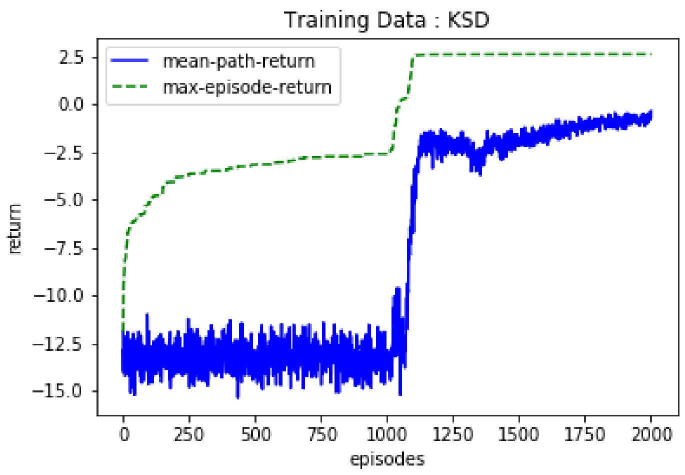

Fig. 4.6 *KSD* aggregated performance. This figure shows aggregated mean path return curves for 50 runs of *KSD* (blue line), and it shows the aggregated maximum episode return (green).

When looking at the standard deviation associated with the curve in Figure 4.6, we observe similar behaviour to Figures 4.3 and 4.5. However, the notable difference here is the smaller bounding region for the mean path curve, meaning that the curves behaved consistently to a greater degree than the other methods we have seen above. This information is encapsulated by Figure 4.7.

For the statistics we look at Table 4.1. This table qualifies the behaviour seen on the plots according to counts on the trials, which is also informative. We see, from here, that *KSD* and *LSD* did not converge optimally, while SVGD converged optimally twice, did not converge twice and converged suboptimally the rest of the time.

Table 4.1 should be taken as an instrument that provides more details by means of counting. So, while we were interested in averages before, here we look at the counts for possible observable behaviours. There are three such behaviours – convergence to the optimal solution, convergence to the suboptimal solution and no convergence at all at programme termination. Since non-convergence can offset the final reward in the average in unpredictable ways, it becomes meaningful to know exactly what the convergence numbers are in their respective bins of optimality and suboptimality.

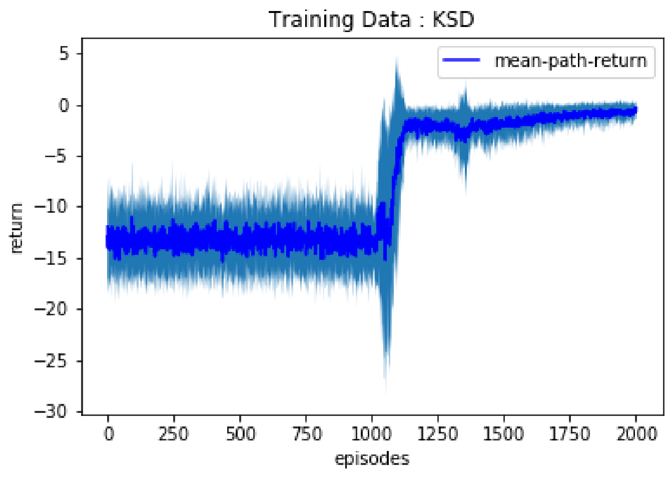

Fig. 4.7 *KSD* mean path return aggregate and standard deviation. This plot shows the mean path return aggragate over the 50 independent trials (solid blue line) as well as the standard deviation associated with these returns (light blue shading) for *KSD*.

	Converged	Converged Suboptimally	No Convergence
KSD	0/50	50/50	0/50
SVGD	2/50	46/50	2/50
LSD	0/50	50/50	0/50

Table 4.1 This table shows the convergence statistics as proportions. In particular, it demonstrates the percentage of trials that converged optimally, converged suboptimally and that did not converge, for each of the three methods we looked into –*LSD*, *KSD*, and *SVGD*.

4.2 Results : Sub-research Question 2

This section explores the experiments regarding the methods that do some form of opponent modelling. We start with pure opponent modelling, wherein the agents are trained using *PR2AC* (Wen et al. [6]). Recall that the only assumption here is that the agent merely assumes that the opponents are able to act in a way that is informed about the actions that the agent might take. This does not happen explicitly, but the agents themselves have to build a model for how they think the opponents are behaving.

We obtain the results in Figure 4.8. Firstly, we see that *PR2AC* grossly underperforms compared to the other models explored previously. This model certainly did converge to the optimum a number of times, but for most part it simply did not even converge to an equilibrium. This is precisely why the value is below zero. When there was convergence, it was optimal but it was not a usual occurrence.

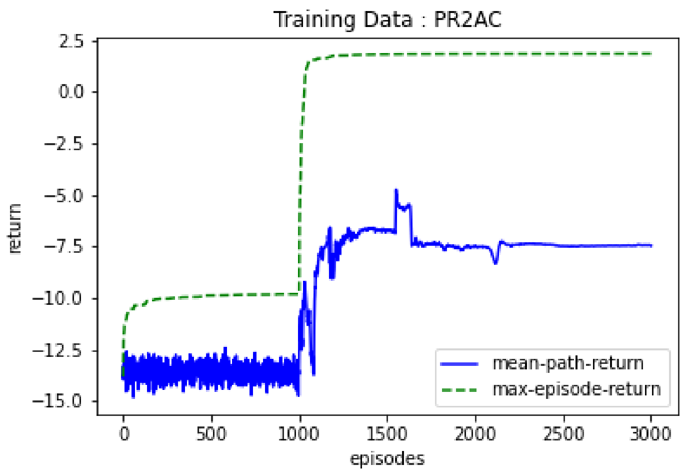

Fig. 4.8 Aggregated performance for *PR2AC*. This figure shows the aggregated training curves of *PR2AC* (blue) and the aggregated maximum episode returns (green) for each of the corresponding 50 trials. The mean path returns denote the average reward over episodes in a run, while the max episode returns denote the maximum reward that has been attained at each episode given a run.

Moving forward from this, we consider *ROMMEO* (Tian et al. [7]). Recall that the difference between this method and *PR2AC* is that the former behaves from an optimality assumption. Each agent assumes that its opponents are acting optimally, and it optimises the objective resulting from this through a lower bound that has an extra KL-regularisation on the prior distribution. This optimality assumption is not explicitly present in the former. We obtain good results here as demonstrated by Figure 4.9.

Lastly, looking into our own method we see quite an improvement from *ROMMEO*. It appears that the gradient updates are always good enough to move the agents towards the maximum at some point (or for some trajectory), and it seems that it is likely the case that when the agents find a maximum trajectory they learn from it and succeed. We see that this is not always true since there is a discrepancy between the maximum episode return and the mean path return in Figure 4.10.

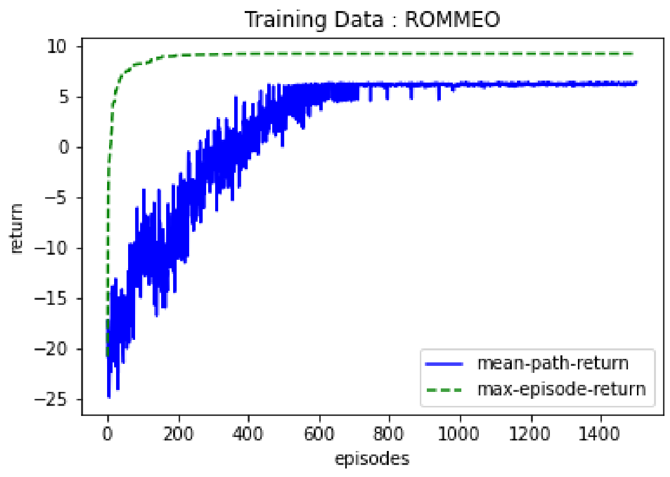

Fig. 4.9 Aggregated performance for *ROMMEO*. This figure shows the aggregated training curves of *ROMMEO* (blue) and the aggregated maximum episode returns (green) for each of the corresponding 50 trials. The mean path returns denote the average reward over episodes in a run, while the max episode returns denote the maximum reward that has been attained at each episode given a run.

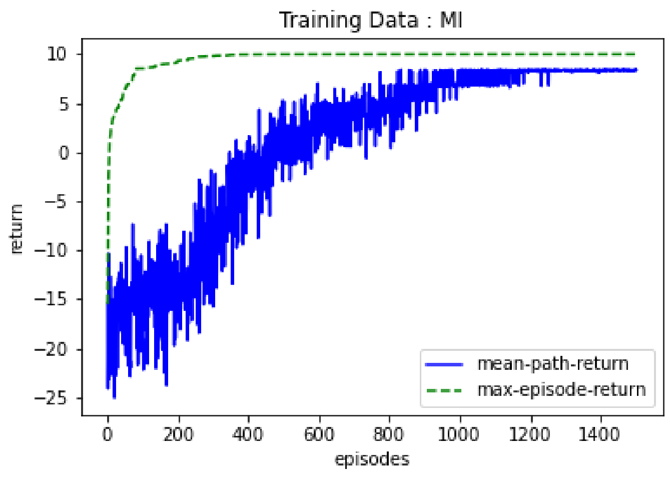

Fig. 4.10 Aggregated performance for *MI*. This figure shows the aggregated training curves of *MI* (blue) and the aggregated maximum episode returns (green) for each of the corresponding 50 trials. The mean path returns denote the average reward over episodes in a run, while the max episode returns denote the maximum reward that has been attained at each episode given a run.

The important thing is, of course, the statistics. We find the number on Table 4.2. This table is quite revealing in the sense that we see that the method proposed only failed minimally on our trial, and in this case it did not even converge. Because we

	Converged	Converged Suboptimally	No Convergence
PR2AC	2/50	33/50	15/50
ROMMEO	39/50	9/50	2/50
MI	48/50	0/50	2/50

Table 4.2 This table shows the convergence statistics. In particular, it counts the number of trials that converged optimally, converged suboptimally and that did not converge, for each of the three methods we looked into – *PR2AC*, *ROMMEO*, and *MI*.

truncated the experiment runs at a fixed time (to allow for these comparisons), it is not clear whether this is due to that or whether the agents were just not going to converge. This is promising, however, especially given the method's good convergence track in the other runs, and we suspect that this doesn't violate claims that we could make about robustness.

At this point, we have seen that *ROMMEO* and *MI* seem to be close in performance with each other (but not equivalent) and better than the previous methods we have considered (MASQL and PR2AC), so we can look closely at their performances in Figure 4.11.

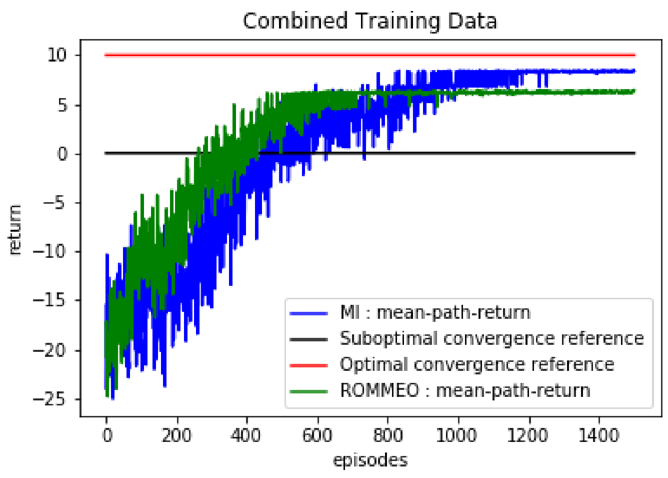

Fig. 4.11 Comparison of *ROMMEO* and *MI* agents. The figure shows the aggregated mean path return curves above for *ROMMEO* (green) and *MI* (blue) agents. The references for maximum reward (orange) and the suboptimal convergence zero reward (black) are plotted for ease of comparison.

The main purpose of this visual is to show that the proposed method performs better than *ROMMEO*, with *ROMMEO* being the best stable alternative in the list of methods from the literature. We put everything together in Figure 4.12 for some overall perspective.

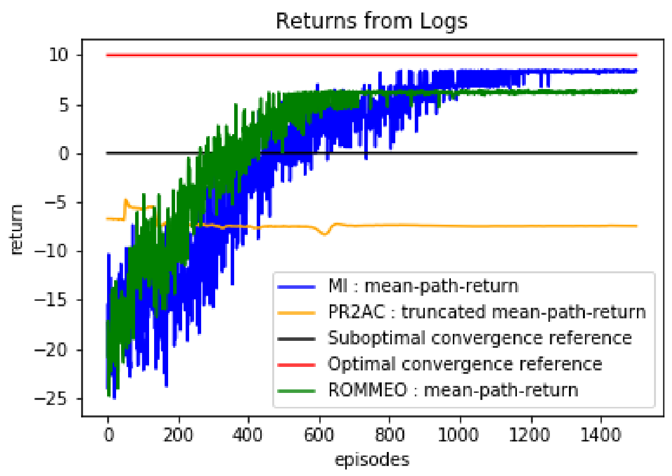

Fig. 4.12 Our Comparison of *PR2AC*, *ROMMEO* and *MI* agents. The figure shows the aggregated mean path return curves above for *PR2AC* (yellow), *ROMMEO* (green) and *MI* (blue) agents. The references for maximum reward (red) and the suboptimal convergence zero reward (black) are plotted for ease of comparison. We only plot the tail of the *PR2AC* vector since this method required more time for convergence, and we wanted to present the methods on the same x-axis scale (otherwise the lengths for the axes would have been different, since *PR2AC* would have more points than the other methods). This means that we are looking at what *PR2AC* eventually converges to here, but the behaviour at start time can be viewed in Figure 4.8.

Nothing is new here, once again, we just put in *PR2AC* for perspective on how badly it behaved.

Notice here that we have only incorporated means, absolute maximum rewards, and we have not focused much on standard deviations. We find that the information captured by standard deviations is something that we can infer from the figures that we have focused on up to this point. However, we include this part for completeness. To this end, we give plots with the mean and standard deviation.

We start by giving a chart that shows the training curve for the mean returns of PR2AC as well as its standard deviation in Figure 4.13. Notice here that the region for the deviation of points along the curve is big compared to what we will see below. A standard deviation as huge (at the end of the episode horizon) means that the method was not consistent in its convergence, but we see that it did converge optimally, but sometimes not so optimally or not at all.

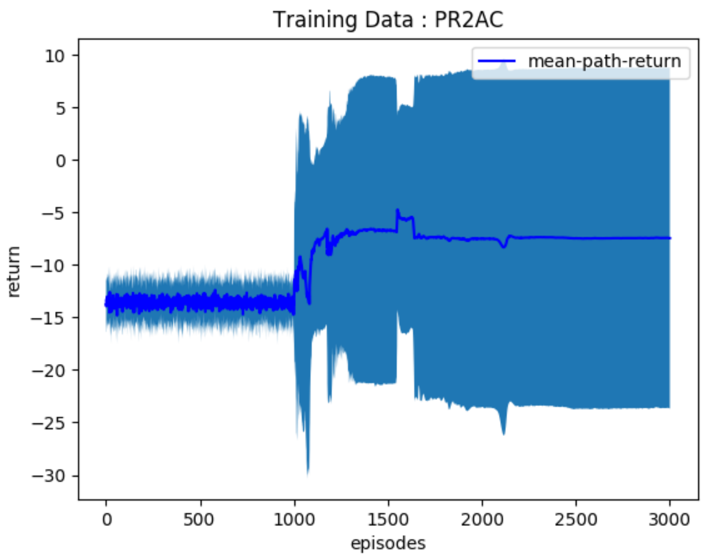

Fig. 4.13 Aggregated performance and standard deviation for PR2AC. This figure shows the aggregated training curve of PR2AC (solid blue line) and the standard deviation (shaded region) corresponding to the training curve for the corresponding 50 trials.

Next we look at the same data for ROMMEO in Figure 4.14. We see here that the standard deviation is better behaved, as one would expect for a more consistently converging method. The region as seen to be slightly breaching into negative values denotes non-convergence, but we see that the average signal lies in the positive quarter of the 2D plane eventually, and this tells us that there is convergence to the optimal solution on some runs–in agreement with the numbers in Table 4.2.

Lastly, we observe that MI gives similar a similar standard of points along the reward curve to ROMMEO, except for the fact that the aggregate line is closer to the maximum edge of the shaded 'deviation region' on the plot. Again, this is what one would expect for a more consistently converging method. The standard deviation region is similar to that of Figure 4.14 except that the region is further constrained, and this (again) tells us that there is convergence to the optimal solution occasionally but for a greater number of times than *ROMMEO* (Figure 4.14)–in agreement with Table 4.2.

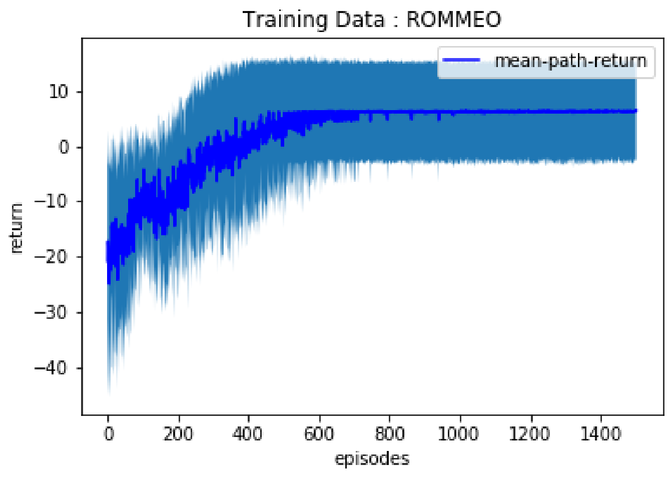

Fig. 4.14 Aggregated performance and standard deviation for ROMMEO. This figure shows the aggregated training curve of ROMMEO (solid blue line) and the standard deviation (shaded region) of the curve for the corresponding 50 trials.

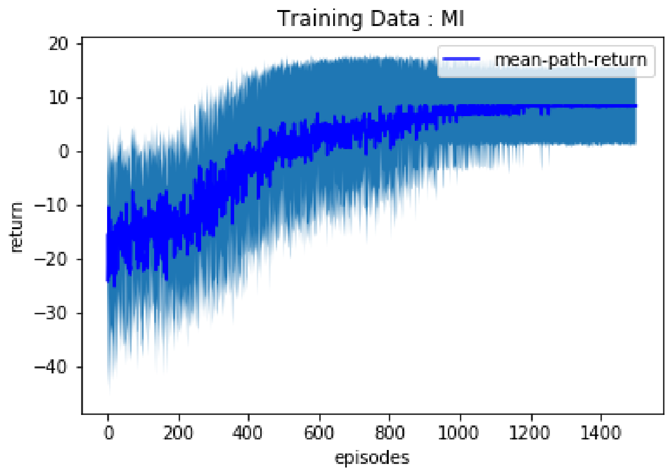

Fig. 4.15 Aggregated performance and standard deviation for MI. This figure shows the aggregated training curve of MI (solid blue line) and the standard deviation of the curve (shaded region) for the corresponding 50 trials.

The last point is that this is mostly information that we observe when we look at Figure(s) 4.9 and 4.10. The huge standard deviation of Figure 4.13 is also captured very well by Figure 4.8. Hence, we shall continue our discussion with the Figure(s) 4.8, 4.9 and 4.10. So, what have we learnt? We have learnt that mutual information is actually quite efficient when it comes to efficiency with coordination, and that it allows us to learn even in contexts that are usually difficult to learn in due to relative overgeneralisation-based pathologies. We will discuss why this is the case in the discussion.

Chapter 5

Discussion

In this section, we consolidate our results with the literature. The starting point, Section 5.1, seeks to address the first research question wherein we asked whether the part of the algorithm that was failing was the sampling mechanism, from the perspective of Section 4.1 and Section 2.9. The second part, Section 5.2, addresses the second questions which was asking whether or not the agents are learning the policies well to start with. In the same vein, we discuss this section looking at the literature we provided and previous results in the literature (Section 2.9) as well as the results we produced in Section 4.2. Each of these will be discussed separately after which we will conclude the work.

5.1 Research Question 1 : Discussion

In the first part of the experiments sections, Section 4.1, for the first hypothesis, we saw results for different samplers implemented on the naive multiagent soft Q-learner. Here we notice that the behaviour for *LSD* (Figure 4.4) and *KSD* (Figure 4.6) was very similar. Both methods converged suboptimally in all the 50 runs that we did, showing very similar behaviour to each other, while *SVGD* in Figure 4.2 converged twice, did not converge twice and converge suboptimally the remaining times. Additionally, we noticed that for *LSD* (Figure 4.4) the signal was noisy.

Let's discuss this further. This behaviour could be due to a couple of reasons.

- The first is that we did not train the critic as detailed on Grathwohl et al. [87], which involves learning a function by an explicit adversarial process. Instead we used a value function that is derived from the learning process of the agent. There is little reason to believe that this should have a significant impact on the results.

- Secondly, it could just be the fact that the method is approximation-based, and need not necessarily trump the other options (*SVGD* and *KSD*) in lower dimensions without it having to do much work (but it should do well in higher dimensions), especially given the first point.

Going back to the other observation above, we mentioned that there is very similar behaviour between *KSD* and *LSD*. Is this to be expected? While *KSD* was shown to perform best in multimodal (quadratic mixture) settings in Hu et al. [4] through Figure 2.9, recall that the context we have for the differential game is different in that the modes are not uniform, as we demonstrated in Section 4 using Figure 4.1 (from our discussion in Section 2.9). This is, in fact, what makes this aspect challenging.

The first point here is that the Stein discrepancy is the key element of optimisation. For *SVGD*, the action updates are done in the direction that minimises the discrepancy. It is also worth noting that if we denote the discrepancy between distributions p and q as $\mathcal{S}(p,q)$ then

$$\nabla D_{KL}(q||p) = -\mathcal{S}(q||p).$$

We noted that, for *LSD*, we are doing the same thing in essence but trying to learn the kernel part of the expanded discrepancy.

So, by optimising in this direction we are essentially using the KL-divergence in our updates. The *KSD* optimises based on the gradient of the discrepancy, which we can now see is the second derivative of the KL-divergence [82]. However, interestingly enough this is the so-called Fisher information matrix. This hence, leads to a second order optimisation. Second order methods, when they converge, converge very fast. This is demonstrated, for example, in Ferreira et al. [92] on a discussion about Kantorovich's theorem on Newton's Method. The standard deviation plots show us the consistency of reward gains through which each method converged, and we see that *KSD* was more consistent from Figure 4.7, which we suspect is owed to these second-order, deterministic properties (as opposed to *SVGD* in Figure 4.3 and *LSD* in Figure 4.5).

This brings us to the following point about equivalence. The plots for *KSD* and *LSD* show similar behaviour, however these methods are not equivalent. Theoretically, the two methods should converge to the same distribution (when there is convergence), and they should perform very similarly in lower dimensions (which we can see is the case) [82]. On the other hand, we see that *SVGD* performs better in some ways (two optimal convergences) but not great in others (two non-convergences). We should note that *SVGD* and *LSD* are quite similar to each up to the part where we have to learn a kernel-like function (the critic), so one would expect similar behaviour from these two which we don't observe here.

Recall that Wei et al. [5] demonstrated that *MASQL* converges 72% of the time in their experiments, which we saw in Section 2.9 on Figure 2.11. Our results, from Section 4.1 in Table 4.1, did not demonstrate the same behaviour in terms of convergence proportions, if we interpret the way the word converge is used in that work to mean 'convergence to the optimal solution' (which we assume is what the Wei et al. was using, instead of 'convergence to some solution'). We found a much lower convergence ratio, which was 2/50, as opposed to 36/50. It also seems like the subsequent work done on this problem by Wen et al. [6] and Tian et al. [7] confirms this since the experiments demonstrated in this work seems to show consistent lack of convergence for *MASQL*. However, these two papers focus on arbitrary runs rather than multiple trials aggregated, so we don't get to understand the behaviour over a number trials. The code that they provide, however, gives us results consistent with our own regarding *MASQL*, and the modifications in consideration.

The key takeaway here is that while improving sampling will aid learning very often, essentially without extra cost, it will not automatically lead to better performance in cases where there are underlying game-theoretic pathologies. In such cases, we need to think about the pathology in consideration, and possibly devise a method that accounts for the kind of 'thinking' that is needed given the pathology, because the problem does not seem to be based on sample quality [83]. The general policies that are possible in multiagent reinforcement learning certainly benefit from the study of samplers, as we have seen that by increasing the order of optimisation we can gain more consistency in convergence. With the advent of scalable methods (such as *LSD*), it looks like there are promising prospects towards the direction of scalable joint policy searching, where there are large state and action spaces, especially in the context of entropy regularised multiagent reinforcement learning [93] [94]. Of course, care needs to be taken here for the optimisation as naive implementations may not be efficient, which we see above. We now talk about the next question that we asked.

5.2 Research Question 2 : Discussion

Let's now look at the methods for investigating the second sub-research question. The second sub-research question was stated as follows : If the sampling strategies do not provide insight, then are we simply not learning the policies well? For this we hypothesised that if the sampling strategies do not improve learning, then the problem has to be that we are not learning the policies well (in terms of the underlying optimisation). The first method is *PR2AC* (Wen et al. [6]), which we see does suboptimally compared to other methods in question from results in Section 4.2, looking at Figure 4.12. The decomposition of the joint policy in this method, we might remember from Subsection 2.8.1, takes a Bayes-type form. Beyond this, nothing else changed from the original *MASQL*. It is clear from these experiments that this is not sufficient for consistent convergence, in the sense that we are interested in. The statistics related to convergence were not discussed in the original paper, but the code for the experiments detailed in the paper was made available. Hence, we are confident that the results we are obtaining are exactly as the authors had them. The focus in the paper, instead, was on demonstrating that the agents can perform well when compared to other methods based on selected single runs.

Recall from Section 2.8.1, Theorem 5, that for this method it was established that

$$\nabla_{\theta^i}\eta^i = \mathbb{E}_{s\sim p, a^i\sim\pi^i}\left[\nabla_{\theta^i}\log\pi^i_{\theta^i}(a^i|s)\left[\mathbb{E}_{a^{-i}\sim\rho^{-i}_{\phi^{-i}}}\left[\frac{\pi^{-i}_{\theta^{-i}}(a^{-i}|s,a^i)}{\rho^{-i}_{\phi^{-i}}(a^{-i}|s,a^i)}Q^i(s,a^i,a^{-i})\right]\right]\right].$$

We know that the direction of the scaled expectation of

$$-\nabla_{\theta^i}\log\pi^i_{\theta^i}(a^i|s)$$

is what we want to update the gradients towards from Subsection 2.5.1. The other important part of this is the scaling of the gradients, which we see is a scaled Q-function. We would hope that this Q-function scaling will behave in ways that are 'aware' of the underlying pathology [5]. However, this is not the case from Section 4.2 as shown in Figure 4.12. We should recall that in this section, we obtained the Figure 4.8, which shows suboptimal performance for this method. Hence, it seems that the gradients are constantly scaled in a way that leads to the same fate as the original extension of soft Q-learning. It is hard to say why this is, theoretically. However, we can make the following remark about the objective function used. We would further expect that this method would not be as robust as one might hope, because the objective itself,

$$D_{KL}(\hat{p}(\tau)||p(\tau)) = -\sum_{t=1}^{T} \mathbb{E}_{\tau\sim\hat{p}}\Big[r^i(s,a^i,a^{-i}) + H(\pi^i_\theta(a^i|s)\rho^{-i}_{\phi^{-i}}(a^{-i}|s,a^i))\Big],$$

seeks to maximise the entropy of the approximate joint function. In the early stages, an agent trying to maximise the joint policy in the multiagent setting–which is non-stationary–will take a lot of time to learn, since (in addition to the difficulty to learn with non-stationarity) entropy optimisation on its own incentivises exploration [47]. That is if the agent has sufficient capacity to learn anyway. We are making this remark drawing from the non-stationarity as well as the way in which we introduced entropy in Section 2.3. This is to say that while the method seems convincing, and while it converges sometimes to the correct solution there are aspects of it that are to be questioned as far as efficiency and stability are concerned. What we see is that this forced entropy regularisation leads to gradient updates that are not necessarily aware of the pathology in question. We use the word aware loosely here to mean that the gradients are not being computed correctly, which has to be because the objective functions being optimised are not suitable for the problem.

Moving on, we consider *ROMMEO* (Tian et al. [7]). Recall that for this algorithm the objective was given as a lower bound of a log-probability as follows

$$\log P(o^i_{1:T}=1|o^{-i}_{1:T}=1) \geq \sum_t \mathbb{E}_{s_t}[\mathbb{E}_{a^i_t\sim\pi,a^{-i}\sim\rho}[R^i(s_t,a^i_t,a^{-i}_t) + H(\pi(a^i_t|s_t,a^{-i}_t))]$$
$$-\mathbb{E}_{a^{-i}_t\sim\rho}[D_{KL}(\rho(a^{-i}_t|s_t)||P(a^{-i}_t|s_t))]],$$

and here the agent tries to maximise its own entropy while minimising the KL divergence between what its opponents are doing as well as its model of the other agents. The form of the log probability itself is not given in Tian et al., however, as discussed in Subection 2.8.2, which would have been informative (in the sense of telling us more about the optimisation being done to achieve good results for *ROMMEO*). The strength of this method (as demonstrated clearly in Figure 4.9, which we saw in Section 4.2) lies in the KL-divergence term, which ensures that the agent links directly to its policy optimisation the need to know what its opponents are doing–as opposed to doing this implicitly as we saw previously in *MASQL* and *PR2AC* in Subsections 2.8.1 and 2.5.3, respectively. While *ROMMEO* seems to work well, one aspect to the method that is worth looking into is its motivation in the context of its complexity. For example, it is not clear why each agent's maximisation of entropy should be helpful in a non-stationary setting with multiple agents. The intuitive effects of this term, which we feel should be non-desirable (as we have already discussed), are seemingly cancelled by the KL-divergence term.

We also saw that the objectives for the policy and Q-function were (from Subsection 2.8.2, equations (2.2) and (2.3)), respectively, the following

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}, \epsilon_t^i \sim N, \hat{a}^{-i} \sim \rho}(\alpha \log \pi_\theta(f(\epsilon_t^i; s_t, \hat{a}_t^{-i})) - Q_w(s_t, f(\epsilon_t^i; s_t, \hat{a}_t^{-i}), \hat{a}_t^{-i}),$$

and

$$J_\rho(\phi) = \mathbb{E}_{(s_t, a_t) \sim \mathcal{D}, \epsilon_t^{-i} \sim N}[\log \rho_\phi(g_\phi(\epsilon_t^{-i}; s_t)|s_t) - \log P(\hat{a}_t^{-i}|s_t) - Q(s_t, a_t, g_\phi(\epsilon_t^{-i}; s_t))$$

$$+\alpha \log \pi_\theta(a_t^i|s_t, g_\phi(\epsilon_t^{-i}; s_t))].$$

The first two terms of objective function, $J_\pi(\theta)$, are (as one would expect) merely doing reward-entropy maximisation. The construction of the latter, $J_\rho(\phi)$, however, is not clear. Again, the question that we ask is:

Q0 If we look closely to the $J_\rho(\phi)$ objective function, then we see that the Q-function, which corresponds to the reward, and the prior are being optimised. However, it is not clear what role the entropy terms play in the multiagent setting.

The way in which these function are presented in the paper are as 'derivatives' (in the sense of being derived) of a process that uses the reparametrisation trick on KL divergence-type objectives, but these objectives themselves are not too well-motivated. One might be skeptical here because the way the form of the policies are given is a theorem. The argument we make here is that what that theorem does is provide argument that a certain function will provide monotonic improvements to the policy. It doesn't say whether this function is the best one can do, or anything of that form–merely that it should work in improving the policy. This is all to say, beyond the diligently derived objective, there are many questions we can ask about this method, which ultimately lead us to making improvements. It turns out (however) in this case, that the objectives, $J_\pi(\theta)$ and $J_\rho(\phi)$, improve the policy quite impressively in the face of the adversity that the agents face in our environment. This is clear from the experiments as we saw in Section 4.2, and as we once again demonstrate in Figure 4.9. From the figure, we see the following :

- The agents converge to optimality 39 out of 50 times (which we also see from the high aggregated rewards as well as the closeness of the max- and mean-returns, which tells us about the efficiency of the gradient updates), then there are runs that converge to suboptimality and runs that do not converge. While we may express slight dissatisfaction about the results and motivations of the method, this is still an immense improvement from previous methods. This is evident from Table 4.2, which we saw in Section 4.2.
- We see that there is increased likelihood from previous methods that if agents can find an optimal trajectory then they can exploit it (learn from it). This is visible from the tight gap between the curves on Figure 4.9. This is related to the efficiency with which we compute our gradients.

So, with all the points of concern discussed and the results above, we have shown that *ROMMEO* is quite competitive. This is because it converges more often than methods previously could (it is consistent in this sense), and more than that we see the gradient efficiency mentioned above. In addition to this, the method is actually quite efficient in terms of convergence. It could converge much more quickly than the other methods that we present, and this can be observed in Figure 4.11 and Figure 4.12 from Section 4.2. When taking the averages, we don't really observe the sharp gradients, which tell us convergence rates, however we still see quite clearly from Figure 4.12 that it converges faster in comparison. To make things concrete the difference between the needed time for convergence in *ROMMEO* compared to *MI* was around 300 runs, all else being fixed. That being said, each run was very fast, as can be observed from the steep gradients in Figure 4.12, and moreover we saw that convergence rate is something that can be improved if we choose a good sampler using the work on sampling methods, as we saw them in Section 2.7.

We see that *ROMMEO* was exactly as advertised in the results of Tian et al. [7]. Recall that in Section 2.9 we saw how the results of each of these methods were spelled out in the original papers, and we discussed these results. Recall once again, that we saw that *ROMMEO* was very competitive compared to both *PR2AC*, which converged to optimality, and *MASQL*, which converged suboptimally. These results are encapsulated in Figure 2.13 which we saw in Section 2.9, where we see *ROMMEO* being very quickly convergent, followed by *PR2AC*, then everything else being suboptimally convergent. Our main concern with the figure was that it was not clear which runs were selected for these experiments.

Lastly, we are focusing on the method being introduced and advocated for in this thesis. Inspired by the fact that we are doing mutual information regularisation, as discussed in Chapter 3, we will refer to this as the *MI* algorithm (and *MI* in plots and tables). Recall that in Section 3.1 we propose that we should consider opponent modelling objectives of the form

$$J_\pi(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}, \epsilon_t^i \sim N, \hat{a}^{-i} \sim \rho}(\alpha(\log \pi_\theta(f(\epsilon_t^i; s_t, \hat{a}_t^{-i})|s_t) - \log \rho_\phi(g_\phi(\epsilon_t^{-i}; s_t)|s_t)) - Q_w(s_t, f(\epsilon_t^i; s_t, \hat{a}_t^{-i}), \hat{a}_t^{-i})),$$

and

$$J_\rho(\phi) = \mathbb{E}_{(s_t,a_t) \sim \mathcal{D}, \epsilon_t^{-i} \sim N}(\alpha(\log \rho_\phi(g_\phi(_t^{-i}; s_t)|s_t) - \log \pi_\theta(f(\epsilon_t^i; s_t, \hat{a}_t^{-i})|s_t)) - Q_w(s_t, a_t, g_\phi(\epsilon_t^{-i}; s_t))).$$

In the specific section, these equations are labelled (3.1) and (3.2).

The first observation here is that in the spirit of the energy-based policies' context, we are optimising the reward along with some entropy terms. The reward function manifests itself as the Q-function, consistent with the single-agent soft Q-learning approach [20]. The appearance of this term in both optimisation objectives gives insight into why this method should work. In particular, the agent models its opponents in a way that is aware that the opponents are trying to also maximise the same reward. This is not unique to this method, however, as all the previously mentioned methods here make this assumption–the previous alternatives being Wei et al. [5], Wen et al. [6] and Tian et al. [7]. The reason why this makes sense is that we are not really interested in the emergence of behaviours, which we mentioned was a wide branch of research in the literature review (Section 2.4). In that section, we looked into work which sought to understand skills discovery, emergence of coordination and communication, respectively, He et al. [41] and Lazaridou et al. [42] We also saw that Jacques et al. [66], van der Heiden et al. [64] and Krupnik et al. [67] use mutual information to improve agent learning in Section 2.4. We are more interested, in contrast, in being able to perform tasks well in the cooperative space, especially with pathologies like the one that impedes learning in our environment.

Secondly, we notice that both objectives, $J_\pi(\theta)$ and $J_\rho(\phi)$ carry terms that can be seen to be differences of entropies. As previously discussed in Section 2.3, these terms are responsible for the mutual information computation. In particular, we mentioned in the methodology (Section 3) that we would like to maximise the mutual information in the sense of works like Kim et al. [46] who also consider a similar form but optimise via derived lower bounds, which we gave initially in Section 2.3. One is likely to ask : 'Are these optimisations reasonable? Should we allow both objectives to have access to both entropy terms?' The answer to these questions is a yes. The key thing to notice here is that the opponent model is an agent's own estimate of how the other agents will behave, and we are not explicitly providing access to this to the agent–instead, it learns it. Hence, having learnt this the agents should be allowed to do with it what they want to.

Moreover, it is important to remember that mutual information is symmetric in its variables, which is to say that for any two random variables X and Y

$$I(X;Y) = H(X) - H(Y|X) = H(Y) - H(X|Y) = I(Y;X).$$

Hence, the objectives above ($J_\pi(\theta)$ and $J_\rho(\phi)$) are essentially maximising the same quantity here, and this is important to strengthen the way we encode cooperative dynamics. Remember, simply allowing the agent to assume that the opponent also has intentions of maximising the reward was not enough for Wen et al. [6]. In this case, we are adding something extra but which the agent believes the opponents share–the desire to maximise mutual information. But why is the above chain of equalities true? The key to understanding this comes from looking at the definition we gave for mutual information, i.e that if X and Y are random variables with joint probability distribution $P(X,Y)$ then

$$I(X;Y) = \sum_{x\in\Omega_x}\sum_{y\in\Omega_y} P(x,y)\log\frac{P(x,y)}{P(x)P(y)}.$$

From this definition, it is clear that we can interchange the variable without consequence. This is also clear from Figure 2.4 in Section 2.3.

The other question that may come to mind is the following. So far in our arguments we have seemingly been making a case that entropy terms may lead to instability, as discussed above, but problems with maximising entropies also came up in Section 2.5.2. Recall that here we saw that while the maximum entropy approach presented is very convenient and powerful it has been shown that it has its weaknesses [73]. This work by O'Donoghue, we said, brings up a key shortcoming in the approach, and clarifies the sense in which reinforcement learning can be coherently cast as an inference problem. The discussion is that in all but the most simple settings, the resulting inference is computationally intractable so that practical algorithms must resort to approximation. The big concern then might come from the fact that, for our method, we just introduced another entropy term, so the question becomes : What is really the mechanism which makes this method functional? To answer this, we observe that there is actually a connection between mutual information and KL divergence. This is derived in equation 5.1 following [95].

What we notice here is that what the mutual information term does is try to reconcile the assumed product decomposition with observations of how the other agents are behaving. So, for example, the term $p(x,y)$ is a joint distribution that will reflect the actual behaviour of the agents, and $p(x)$ and $p(y)$ are the parts that the agent and its opponents try to learn. One can map this directly to our discussion earlier concerning *PR2AC* (which we mentioned is carried through the methods that followed). We said that the assumed that we could learn an agent's policy and its opponents' model in a way that leads to

$$\pi(a^i,a^{-i},s) = \pi^i(a^i|s)\rho^{-i}(a^{-i}|s,a^i),$$

from agent i´s perspective, or equivalently

$$\pi(a^i,a^{-i},s) = \pi^{-i}(a^{-i}|s)\rho^i(a^i|s,a^{-i}),$$

from the opponents' perspective. It would seem like previous methods could not consistently achieve this, and the mutual information term is merely enforcing this.

$$\begin{aligned}
I(X;Y) &= H(X) - H(X|Y) \\
&= \sum_x p(x)\log\frac{1}{p(x)} - \sum_y p(y)\log H(X|Y=y) \\
&= \sum_{x,y} p(x,y)\log\frac{1}{p(x)} - \sum_{x,y} p(x,y)\log\frac{1}{p(x|y)} \\
&= \sum_{x,y} p(x,y)\log\frac{p(x|y)}{p(x)} \\
&= \sum_{x,y} p(x,y)\log\frac{p(y)p(x|y)}{p(x)p(y)} \\
&= \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} \\
&= D_{kL}(p(x,y)||p(x)p(y)).
\end{aligned} \tag{5.1}$$

Is this reasonable, or too constraining with respect to the literature? The truth is that we are not really doing something extreme here. Recall that the objective lower-bound for *ROMMEO* had a term

$$\mathbb{E}_{a_t^{-i}\sim\rho}[D_{KL}(\rho(a_t^{-i}|s_t)||P(a_t^{-i}|s_t))],$$

which sought to capture the dynamics in the actual environment, and reconcile them with the agent's beliefs/model for the opponents. However, it seems like this is not sufficient to achieve stability. Looking at everything we have so far, it seems that the biggest problem could be that the learned parts do not always converge to the joint distribution in product as we would hope. In particular, we see that *ROMMEO* is maximising a reward term with conditional entropies, which is not enough from the results we presented above in Table 4.2 from Section 4.2.

This connection to the KL divergence offers a radical shift in perspective, however. The worries about entropy slowing down learning, are suddenly swept away by recognising that we are actually doing optimisation of the form that we are used to–i.e., just regularising with KL divergence. Regularisation, as we know, is just a mechanism to offer extra guidance to a model [96]. It allows us to inject preferences into the model, which is exactly what we are doing here. Now, is this satisfactory? Wouldn't it be much better to strive towards something that can do this learning without what seems to be too much of a helping hand? This latter is definitely something that is of immense interest. However, seeing that the literature body inspiring this work itself needed to make extra assumptions to achieve progress informs us that there are adjustments to be made (added assumptions). Whether or not these are the best we can do (in a sense necessary and sufficient) is a huge question that we shall not attempt to delve into.

We can, however, find comfort in the fact that the method seems to work exceedingly well. This is demonstrated in Table 4.2 and Figure 4.12 (Subsection 4.2). In Figure 4.12 we see that the average reward across multiple runs is quite high. Moreover, we get the sense that whenever the method could find an optimal trajectory it exploited it without a doubt (look at the closeness between the max- and mean -episode-returns). This is quite remarkable, given that previously the other methods could not do this. We confirm this by looking at Table 4.2 as given, where we see that this method actually converged to the optimal solution, and one run when this did not happen. We suspect that this is because the number of steps is bounded, which means that it is important for the agents to not wander around too much. However, we cannot be certain about this. With all this said, we also see on Figure 4.12. That the method performs exceedingly well compared to the previous alternatives. We expected our method to work because we know that regularisation helps establish communication when the policies are defined reasonably (Siu et al. [22]). Siu et al.'s work argues that instead of learning redundant structures which are removed during agent execution, we should leverage shared experiences of the agents to regularise the individual policies in order to promote structured exploration, which is what this regularisation does.

Recall that our second hypothesis was as follows. (Corresponding) Hypothesis 2 : If the sampling strategies do not improve convergence, then we are not doing the correct optimisation for the learning of policies. This work has beyond doubt confirmed this in two ways.

- Firstly, in a weak sense, we see that it is this step that may have led to massive improvement for the *ROMMEO* algorithm.
- Secondly, in a more convincing sense, we note that putting extra constraints to help with learning the joint policy produces remarkable results.

This is all to say that the results confirm this hypothesis. We now move on to the conclusion, which summarises our work and the progress that we made. We also outline in the following chapter what the key takeaways are as well as the limitations of our work.

Chapter 6

Conclusion

Recall that this entire journey was motivated by Wei et al. [5] who sought to extend soft Q-learning to multiple agents. In this paper there was an inconsistency with theory that lead to us being curious about what the cause of the inconsistency might be. Essentially the problem was that for the given extension, in the context of the differential game [79], optimal convergence occurred less than 75% of the time. This seemed to be in contradiction with the theory which says that energy-based policies are general, and they should be able to learn arbitrary distributions well. We wanted to know why this was happening, and/or what we could improve in the implementation to get good convergence behaviour (consistent optimal convergence).

Our research question was : If we know that the policies we learn should be general enough to model arbitrary distributions, then what is the cause of inconsistent convergence in multiagent soft Q-learning? In order to make progress with this question, we started by hypothesising that if indeed the agents are learning optimal policies then the only part where they could have been struggling in the learning process was when we tried to sample from these general policies. We made this assumption since it seemed like the method described in Wei et al. [5] was, indeed, learning the policies well. We found, in the first part of the results (Section 4.1), that this was not true. The study that we did for this research question is incomplete as it stands, but it certainly offers some direction into an interesting and useful research track. When it comes to things we would have liked to do (in this corner of the investigation) the list is as follows:

- 1. An experiment-based study on the usage of different kernels. This study used a Gaussian Kernel with a dynamic width similar to Wen et al. [6] and Tian et al. [7].
- 2. Continuing the study on methods of training energy-based models, in particular using Variational Entropy Regularised Approximate maximum likelihood by Grathwohl, W. et al. [97] as well as on samplers outside the Stein class.
- A study on the samplers themselves as well as possible generalisations of the methods, as a much bigger theoretical task.

We make note that the methods within the Stein class are on equivalent standing with everything we have explored. Hence, we do not expect to see convergence improvements in terms of optimality but we may get some variations in convergence rates. On a similar note, it may have been informative to do an in-depth statistical analysis with the data collected, but our approach got us to a point where we could evaluate our hypothesis, and hence was sufficient for our purposes.

In conclusion, we notice that our first hypothesis has not been verified by the data. Recall that this hypothesis was stated as :

- Assuming that the policies learnt by the agents are general, it must be the case that the way in which we sample from them is inefficient and can be improved by better sampling strategies.

We then got the sense that our assumption about the generality of the policies being learnt by the agents (in practice) is not true. That is, the assumed decomposition of the joint policy in the algorithm by Wei et al. [5] is not quite right. Hence, it looked like modified opponent-modelling-type approaches, such as *PR2AC* (Wen et al. [6]) and *ROMMEO* (Tian et al. [7]), are necessary to make a significant difference in the learning from the original method used for multiagent soft Q-learning, and we explored this discussion in Section 5.2. We moved to this point having statistically ruled out the need to do an investigation into what exactly the difference was in the runs that were successful in the experiments related to this hypothesis, since such a study would be meaningful if there was a reasonable number of convergent runs–which is not the case here. That is to say we steered away from pursuing the path that seeks to answer the question: What was different in the starting behaviour that led to a win? We were aware that identifying the essence of what is required for functionality would provide insights towards a reliable prescriptive mechanism for extending single-agent algorithms to multiple agents–for cooperative settings.

Peeks into the data in this regard, however, would have demanded a lot of work to the algorithm itself, whose productivity trajectory was uncertain. The evidence would have clearly lead to the updates essentially not being accurate (since, if these are accurate, then we are guaranteed convergence)–thereby leading to the kind of failure that we saw in Section 4.1.

The previous study then lead to the second hypothesis, which said that if the samplers do not improve learning then it has to be the case that we are not learning good policies to start with. We made this hypothesis because the sampling step was the only non-standard step (comparing to the broad literature at the time). Hence, it made sense to think that if that step was working well, then we must be failing to learn the policies. This hypothesis was confirmed by experiments in Section 4.2, and we actually gained insight into what the missing element was. We saw in the previous discussion section (Section 5.2) that we needed to make a connection between the components of the assumed policy decomposition and the joint policy itself. In this sense, we answered the question that we posed in the research question. To put this answer concretely : The flaw with Wei et al.'s *MASQL* was that the agents seemed to either learn independently [5], or they learned policies in a way that was aware of other agents but where these learned policies were not properly tied to the joint policy. It was assumed that this will happen automatically [6] [7]. Hence, the required mechanism was a way to facilitate this connection to the joint policy, which we found can be achieved by KL regularisation between the observed joint behaviour and the product of the learned components.

This is a step towards prescriptions for extending single-agent algorithms to multiple agents in cooperative games. We have shown the key aspects of what is required for building strong extensions of algorithms, and we summarise them below.

- We can use energy-based policies in cooperative games, which we know are general enough to learn any distribution.
- We have a vast pool of methods for sampling from general distributions, and they are very efficient. Hence, we should leverage these when needed (for instance, in high dimensional settings).
- In addition to naively defining an objective that focuses on reward, we need to ensure that each agent is able to model the other agents clearly. One can do this by assuming the Bayes-type decomposition of the policy. Moreover, it is imperative to ensure that the parts of the decomposition obtained in training

are connected to the joint policy of interest. This is done by adding a mutual information term, or equivalently a KL-divergence term.

In doing all the work in the literature for this thesis, a guide like this is something that was not present in any of the papers. Each paper offered a method that seemed to work, but there were no prescriptive guides, in this sense. The relevance of this work then, in a nutshell, is that it outlines a common flaw in naive extensions of methods to multiple agents in the sense that we have described.

With all that being said we acknowledge the following limitations, which can also be phrased as questions for further studies :

- This study only focuses on the simplest environment that exhibits the specific pathology, but also opens doors for future work that could focus on generalising the ideas to other environments.
- It is unclear whether the mutual information term is necessary and sufficient, or if we can achieve the same results by minimising something that is, loosely speaking, simpler.
- There is a huge connection between multiagent reinforcement learning and game-theory, and game-theory has seen a lot of similar pathologies. It is not immediately clear whether or not this method could be robust against a reasonable set of those pathologies. In this case the pathology destroyed gradient computation and efficiency, but another pathology that we could look into relates to state exploration (Wen [89])– in particular, it is sometimes hard to find a balance between optimising the reward and exploring states. This is especially hard but crucial if there are many states. In this case, the problem would not be gradient computation, but rather having to deal with the exploration-exploitation balancing efficiently.
- It is also unclear what kind of inference we could be doing in this case, in the sense of Gupta et al. [98] who attempted to establish a probabilistic framework for multiagent reinforcement. For example, Sergey's survey [71], presents a graphical model from which we see the emergence of an optimisation procedure that seeks to minimise the KL divergence of two given distributions, which results in the maximum entropy reinforcement learning objective. Given an objective, however, it is not easy to trace back whether this comes from some form of inference over a graphical model in the same sense.

- It should be possible to verify the claims in the work by potentially studying simulated behaviour on the surface, and then using that to make a point about what it means to have efficient gradient updates/for agents to do 'good optimisation', which we did not get to in this work.

The work we have done here has shown us that energy-based policies in multiagent reinforcement learning are very useful. We have learnt that the way we extend these models, from single-agent reinforcement learning, is important. In our approach, we carefully pulled apart the naive extension that seemed to underperform, and we studied this in order to devise a better extension. We also shed light into why our extension works, which allowed us to see the gaps that the literature had. If there is one thing that someone who's reading this should take away, it is the importance of caution when extending single-agent algorithms to multiple agents. Here we provide a template of how to go about extending mechanisms, and how to study extensions that fail. We think that multiagent reinforcement learning needs more principled approaches to algorithms' development. We certainly have a lot of inspiration to take from reinforcement learning, but we need to put in effort into understanding ways of extending single-agent algorithms effectively. This will allow us to better leverage the developments that have been made on the single-agent case.

References

[1] K. Zhang, Z. Yang, and T. Basar, “Multi-agent reinforcement learning: A selective overview of theories and algorithms,” *CoRR*, vol. abs/1911.10635, 2019.

[2] M. Vu, “Entropy and mutual information (ee194-16 – network information theory),” 2015.

[3] B. Eysenbach and S. Levine, “Maximum entropy RL (provably) solves some robust RL problems,” *CoRR*, vol. abs/2103.06257, 2021.

[4] T. Hu, Z. Chen, H. Sun, J. Bai, M. Ye, and G. Cheng, “Stein neural sampler,” 2021.

[5] E. Wei, D. Wicke, D. Freelan, and S. Luke, “Multiagent soft q-learning,” *CoRR*, vol. abs/1804.09817, 2018.

[6] Y. Wen, Y. Yang, R. Luo, J. Wang, and W. Pan, “Probabilistic recursive reasoning for multi-agent reinforcement learning,” *CoRR*, vol. abs/1901.09207, 2019.

[7] Z. Tian, Y. Wen, Z. Gong, F. Punakkath, S. Zou, and J. Wang, “A regularized opponent model with maximum entropy objective,” *CoRR*, vol. abs/1905.08087, 2019.

[8] I. A. Joiner, “Chapter 1 - artificial intelligence: Ai is nearby,” in *Emerging Library Technologies* (I. A. Joiner, ed.), Chandos Information Professional Series, pp. 1–22, Chandos Publishing, 2018.

[9] A. M. Turing, “Computing machinery and intelligence,” *Mind*, vol. 59, no. October, pp. 433–60, 1950.

[10] B. Widrow and M. E. Hoff, “Adaptive switching circuits,” in *1960 IRE WESCON Convention Record, Part 4*, (New York), pp. 96–104, IRE, 1960.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[12] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book, 2018.

[13] M. Tan, “Multi-agent reinforcement learning: Independent vs. cooperative agents,” in *In Proceedings of the Tenth International Conference on Machine Learning*, pp. 330–337, Morgan Kaufmann, 1993.

[14] L. Harries, R. S. Clarke, T. Chapman, S. V. P. L. N. Nallamalli, L. Özgür, S. Jain, A. Leung, S. Lim, A. Dietrich, J. M. Hernández-Lobato, T. Ellis, C. Zhang, and K. Ciosek, "DRIFT: deep reinforcement learning for functional software testing," *CoRR*, vol. abs/2007.08220, 2020.

[15] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, (San Francisco, CA, USA), p. 157–163, Morgan Kaufmann Publishers Inc., 1994.

[16] B. Millidge, A. Tschantz, A. K. Seth, and C. L. Buckley, "Reinforcement learning as iterative and amortised inference," *CoRR*, vol. abs/2006.10524, 2020.

[17] Y. Shoham, R. Powers, and T. Grenager, "Multi-agent reinforcement learning: a critical survey," 06 2003.

[18] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artificial Intelligence Review*, pp. 1–49, 2021.

[19] A. Oroojlooyjadid and D. Hajinezhad, "A review of cooperative multi-agent deep reinforcement learning," *CoRR*, vol. abs/1908.03963, 2019.

[20] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," *CoRR*, vol. abs/1702.08165, 2017.

[21] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," pp. 14–36, 01 2012.

[22] C. Siu, J. Traish, and R. Y. D. Xu, "Regularize! don't mix: Multi-agent reinforcement learning without explicit centralized structures," *CoRR*, vol. abs/2109.09038, 2021.

[23] G. Palmer, R. Savani, and K. Tuyls, "Negative update intervals in deep multi-agent reinforcement learning," *CoRR*, vol. abs/1809.05096, 2018.

[24] C. J. C. H. Watkins and P. Dayan, "Technical note q-learning.," *Mach. Learn.*, vol. 8, pp. 279–292, 1992.

[25] G. Rummery and M. Niranjan, "On-line q-learning using connectionist systems," *Technical Report CUED/F-INFENG/TR 166*, 11 1994.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

[27] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, Incorporated, 1st ed., 2016.

[28] D. S. Bernstein, S. Zilberstein, and N. Immerman, "The complexity of decentralized control of markov decision processes," in *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, UAI'00, (San Francisco, CA, USA), p. 32–37, Morgan Kaufmann Publishers Inc., 2000.

[29] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. USA: Cambridge University Press, 2008.

[30] Y. Bachrach, T. Lattimore, M. Garnelo, J. Perolat, D. Balduzzi, T. Anthony, S. Singh, and T. Graepel, "Multiagent reinforcement learning in games with an iterated dominance solution," 2020.

[31] K. Liu, Y. Fu, P. Wang, L. Wu, R. Bo, and X. Li, "Automating feature subspace exploration via multi-agent reinforcement learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '19, (New York, NY, USA), p. 207–215, Association for Computing Machinery, 2019.

[32] H. Kim, J. Kim, Y. Jeong, S. Levine, and H. O. Song, "EMI: exploration with mutual information maximizing state and action embeddings," *CoRR*, vol. abs/1810.01176, 2018.

[33] C. D. Hsu, H. Jeong, G. J. Pappas, and P. Chaudhari, "Scalable reinforcement learning policies for multi-agent control," *CoRR*, vol. abs/2011.08055, 2020.

[34] A. Shamsoshoara, M. Khaledi, F. Afghah, A. Razi, and J. D. Ashdown, "Distributed cooperative spectrum sharing in UAV networks using multi-agent reinforcement learning," *CoRR*, vol. abs/1811.05053, 2018.

[35] F. Yao and L. Jia, "A collaborative multi-agent reinforcement learning anti-jamming algorithm in wireless networks," *IEEE Wireless Communications Letters*, vol. 8, no. 4, pp. 1024–1027, 2019.

[36] I. Elleuch, A. Pourranjbar, and G. Kaddoum, "A novel distributed multi-agent reinforcement learning algorithm against jamming attacks," *IEEE Communications Letters*, vol. 25, no. 10, pp. 3204–3208, 2021.

[37] A. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Autonomous Agents and Multi-Agent Systems*, vol. 18, pp. 342–375, 06 2009.

[38] T. M. Nguyen, T. P. Quinn, T. Nguyen, and T. Tran, "Counterfactual explanation with multi-agent reinforcement learning for drug target prediction," *CoRR*, vol. abs/2103.12983, 2021.

[39] W. Bao and X. yang Liu, "Multi-agent deep reinforcement learning for liquidation strategy analysis," 2019.

[40] X. Li, J. Zhang, J. Bian, Y. Tong, and T. Liu, "A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network," *CoRR*, vol. abs/1903.00714, 2019.

[41] S. He, J. Shao, and X. Ji, "Skill discovery of coordination in multi-agent reinforcement learning," *CoRR*, vol. abs/2006.04021, 2020.

[42] A. Lazaridou and M. Baroni, "Emergent multi-agent communication in the deep learning era," *CoRR*, vol. abs/2006.02419, 2020.

[43] E. G. Learned-Miller, "Entropy and mutual information," 2013.

[44] L. LANDAU and E. LIFSHITZ, "Chapter i - the fundamental principles of statistical physics," in *Statistical Physics (Third Edition)* (L. LANDAU and E. LIFSHITZ, eds.), pp. 1–33, Oxford: Butterworth-Heinemann, third edition ed., 1980.

[45] B. Poole, S. Ozair, A. van den Oord, A. A. Alemi, and G. Tucker, "On variational lower bounds of mutual information," 2018.

[46] W. Kim, W. Jung, M. Cho, and Y. Sung, "A maximum mutual information framework for multi-agent reinforcement learning," *CoRR*, vol. abs/2006.02732, 2020.

[47] I. Liu, U. Jain, R. A. Yeh, and A. G. Schwing, "Cooperative exploration for multi-agent deep reinforcement learning," *CoRR*, vol. abs/2107.11444, 2021.

[48] X. Ma, Y. Yang, C. Li, Y. Lu, Q. Zhao, and Y. Jun, "Modeling the interaction between agents in cooperative multi-agent reinforcement learning," *CoRR*, vol. abs/2102.06042, 2021.

[49] D. Strouse, M. Kleiman-Weiner, J. Tenenbaum, M. Botvinick, and D. J. Schwab, "Learning to share and hide intentions using information regularization," *CoRR*, vol. abs/1808.02093, 2018.

[50] J. Jiang and Z. Lu, "The emergence of individuality in multi-agent reinforcement learning," *CoRR*, vol. abs/2006.05842, 2020.

[51] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," in *AAAI/IAAI*, pp. 746–752, 1998.

[52] D. Szer and F. Charpillet, "Coordination through mutual notification in cooperative multiagent reinforcement learning," pp. 1254–1255, 01 2004.

[53] L. Chen, H. Guo, H. Zhang, F. Fang, Y. Zhu, M. Zhou, W. Zhang, Q. Wang, and Y. Yu, "Signal instructed coordination in team competition," *CoRR*, vol. abs/1909.04224, 2019.

[54] C. Qu, H. Li, C. Liu, J. Xiong, J. Zhang, W. Chu, Y. Qi, and L. Song, "Intention propagation for multi-agent reinforcement learning," *CoRR*, vol. abs/2004.08883, 2020.

[55] W. Kim, J. Park, and Y. Sung, "Communication in multi-agent reinforcement learning: Intention sharing," in *International Conference on Learning Representations*, 2021.

[56] Y. Xu, Z. Deng, M. Wang, W. Xu, A. M. So, and S. Cui, "Voting-based multi-agent reinforcement learning," *CoRR*, vol. abs/1907.01385, 2019.

[57] S. Havrylov and I. Titov, "Emergence of language with multi-agent games: Learning to communicate with sequences of symbols," *CoRR*, vol. abs/1705.11192, 2017.

[58] S. Gupta, R. Hazra, and A. Dukkipati, "Networked multi-agent reinforcement learning with emergent communication," *CoRR*, vol. abs/2004.02780, 2020.

[59] S. Shen, Y. Fu, H. Su, H. Pan, P. Qiao, Y. Dou, and C. Wang, "Graphcomm: A graph neural network based method for multi-agent reinforcement learning," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3510–3514, 2021.

[60] T. Eccles, Y. Bachrach, G. Lever, A. Lazaridou, and T. Graepel, "Biases for emergent communication in multi-agent reinforcement learning," *CoRR*, vol. abs/1912.05676, 2019.

[61] G. Hu, Y. Zhu, D. Zhao, M. Zhao, and J. Hao, "Event-triggered multi-agent reinforcement learning with communication under limited-bandwidth constraint," *CoRR*, vol. abs/2010.04978, 2020.

[62] R. Wang, X. He, R. Yu, W. Qiu, B. An, and Z. Rabinovich, "Learning efficient multi-agent communication: An information bottleneck approach," *CoRR*, vol. abs/1911.06992, 2019.

[63] A. Mostaani and O. Simeone, "On learning how to communicate over noisy channels for collaborative tasks," *CoRR*, vol. abs/1810.01155, 2018.

[64] T. van der Heiden, C. Salge, E. Gavves, and H. van Hoof, "Robust multi-agent reinforcement learning with social empowerment for coordination and communication," *CoRR*, vol. abs/2012.08255, 2020.

[65] C. Li, C. Wu, T. Wang, J. Yang, Q. Zhao, and C. Zhang, "Celebrating diversity in shared multi-agent reinforcement learning," *CoRR*, vol. abs/2106.02195, 2021.

[66] N. Jaques, A. Lazaridou, E. Hughes, Ç. Gülçehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas, "Intrinsic social motivation via causal influence in multi-agent RL," *CoRR*, vol. abs/1810.08647, 2018.

[67] O. Krupnik, I. Mordatch, and A. Tamar, "Multi agent reinforcement learning with multi-step generative models," *CoRR*, vol. abs/1901.10251, 2019.

[68] T. Wang, H. Dong, V. R. Lesser, and C. Zhang, "ROMA: multi-agent reinforcement learning with emergent roles," *CoRR*, vol. abs/2003.08039, 2020.

[69] T. Wang, J. Wang, Y. Wu, and C. Zhang, "Influence-based multi-agent exploration," *CoRR*, vol. abs/1910.05512, 2019.

[70] L. Zheng, J. Chen, J. Wang, J. He, Y. Hu, Y. Chen, C. Fan, Y. Gao, and C. Zhang, "Episodic multi-agent reinforcement learning with curiosity-driven exploration," *CoRR*, vol. abs/2111.11032, 2021.

[71] S. Levine, "Reinforcement learning and control as probabilistic inference: Tutorial and review," *CoRR*, vol. abs/1805.00909, 2018.

[72] M. Fellows, A. Mahajan, T. G. J. Rudner, and S. Whiteson, "VIREL: A variational inference framework for reinforcement learning," *CoRR*, vol. abs/1811.01132, 2018.

[73] B. O'Donoghue, I. Osband, and C. Ionescu, "Making sense of reinforcement learning and probabilistic inference," *CoRR*, vol. abs/2001.00805, 2020.

[74] R. Zhao, V. Tresp, and W. Xu, "Mutual information-based state-control for intrinsically motivated reinforcement learning," *CoRR*, vol. abs/2002.01963, 2020.

[75] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *CoRR*, vol. abs/1812.05905, 2018.

[76] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018.

[77] N. Heess, D. Silver, and Y. W. Teh, "Actor-critic reinforcement learning with energy-based policies," in *Proceedings of the Tenth European Workshop on Reinforcement Learning* (M. P. Deisenroth, C. Szepesvári, and J. Peters, eds.), vol. 24 of *Proceedings of Machine Learning Research*, (Edinburgh, Scotland), pp. 45–58, PMLR, 30 Jun–01 Jul 2013.

[78] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems," *The Knowledge Engineering Review*, vol. 27, pp. 1 – 31, 2012.

[79] L. Panait, K. Tuyls, and S. Luke, "Theoretical advantages of lenient learners: An evolutionary game theoretic perspective," *J. Mach. Learn. Res.*, vol. 9, p. 423–457, jun 2008.

[80] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning - From Theory to Algorithms.* Cambridge University Press, 2014.

[81] A. Gretton, "Introduction to rkhs, and some simple kernel algorithms.," 2019.

[82] Q. Liu and D. Wang, "Stein variational gradient descent: A general purpose bayesian inference algorithm," 2019.

[83] J. Gorham and L. Mackey, "Measuring sample quality with kernels," 2020.

[84] C. M. Stein, "Estimation of the Mean of a Multivariate Normal Distribution," *The Annals of Statistics*, vol. 9, no. 6, pp. 1135 – 1151, 1981.

[85] Q. Liu, J. D. Lee, and M. Jordan, "A kernelized stein discrepancy for goodness-of-fit tests," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 276–284, JMLR.org, 2016.

[86] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[87] W. Grathwohl, K.-C. Wang, J.-H. Jacobsen, D. Duvenaud, and R. Zemel, "Learning the stein discrepancy for training and evaluating energy-based models without sampling," 2020.

[88] T. Makino and K. Aihara, “Multi-agent reinforcement learning algorithm to handle beliefs of other agents’ policies and embedded beliefs,” in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS ’06, (New York, NY, USA), p. 789–791, Association for Computing Machinery, 2006.

[89] Y. Wen, “Modeling mutual influence in multi-agent reinforcement learning,” 08 2020.

[90] J. Grau-Moya, F. Leibfried, and P. Vrancx, “Soft q-learning with mutual-information regularization,” in *International Conference on Learning Representations*, 2019.

[91] D. Kingma and M. Welling, “Auto-encoding variational bayes,” *ICLR*, 12 2013.

[92] O. P. Ferreira and B. F. Svaiter, “Kantorovich’s theorem on newton’s method,” *arXiv: Numerical Analysis*, 2012.

[93] Y. Tian, Q. Gong, and T. Jiang, “Joint policy search for multi-agent collaboration with imperfect information,” *ArXiv*, vol. abs/2008.06495, 2020.

[94] T. Gabel and M. A. Riedmiller, “Joint equilibrium policy search for multi-agent scheduling problems,” in *MATES*, 2008.

[95] D. Witmer, “Notes for 15-859: Information theory and applications in tcs– lecture 3: Kl-divergence and connections,” 2013.

[96] J. Kukacka, V. Golkov, and D. Cremers, “Regularization for deep learning: A taxonomy,” *ArXiv*, vol. abs/1710.10686, 2017.

[97] W. S. Grathwohl, J. J. Kelly, M. Hashemi, M. Norouzi, K. Swersky, and D. Duvenaud, “No {mcmc} for me: Amortized sampling for fast and stable training of energy-based models,” in *International Conference on Learning Representations*, 2021.

[98] S. Gupta and A. Dukkipati, “Probabilistic view of multi-agent reinforcement learning: A unified approach,” 2020.