Multi-agent Learning with Bayesian Inference

Phu Sakulwongtana[1]

BSc Computer Science

Jun Wang, Philip Treleaven

Submission date: 18 May 2020

Abstract

In this thesis, we aim to investigate a family of algorithms called maximum entropy multi-agent reinforcement learning (MEMARL), which is based on a well-known single-agent reinforcement learning framework called control as an inference that utilizes variational inference to reinforcement learning problems. By framing multi-agent problems as probabilistic inference, one can include concepts such as recursive reasoning [80, 79] into a multi-agent reinforcement learning algorithms, while being decentralized training and decentralized execution. However, there are some drawbacks with this approach, notably, its inability to train agents in a game which has a conflict of interests. In this work, we try to solve this drawback by investigating the algorithm thoroughly, which leads to valuable insight. Furthermore, we will also argue in support of MEMARL framework by showing that many complex forms of agent can be systematically derived/reinterpret using techniques in this framework. Ultimately, one might view this thesis as a *cookbook* for deriving multi-agent reinforcement learning, which we hope to be useful in future researches.

Finally, looking at the bigger picture, we would like to show one of the ways multi-agent reinforcement algorithms could be derived by providing tools and patterns that anyone can deploy to examine/create a new algorithm. And, by using the framework that is based on a popular single-agent framework, we can apply some of the works done in single-agent literature into the multi-agent domain with relative ease.

Contribution to Science

Our contributions are listed as follows:

- Examine the existing algorithms within MEMARL framework, what make them successful, and what their flaws are. We will also consider another algorithm called Balancing Q-learning [22], which is closely related to algorithms in this framework but lacks proper probabilistic interpretation. By re-deriving results, we can discover the flaws and thus able to mitigate some of the problems.
- Applying the framework to various multi-agent problems that have different characteristics, notably delayed communication and representing belief in an unknown state. We will provide reinterpretations of some of the algorithms in the literature, proving the strength and versatility of the framework.

Contents

1 Introduction **2**

1.1 Research Motivation and Objective 2
1.2 Thesis Content 7
1.3 Thesis Structure 7

2 Background **9**

2.1 Probabilistic Method in Machine Learning 9
2.2 Single Agent Reinforcement Learning 12
2.3 Deep Reinforcement Learning 15
2.4 Control as Inference Framework 19
2.5 Introduction to Multi-Agent Problems 24
2.6 Deep Multi-Agent Reinforcement Learning 26
2.7 Other Related Algorithms 28

3 Unified View of Probabilistic Multi-agent Reinforcement Learning **29**

3.1 Introduction to MAPI framework 29
3.2 Derivation of ROMMEO 31
3.3 Interpretation of ROMMEO 35
3.4 Solving Balancing-Q learning 36
3.5 Probabilistic Balancing-Q 42
3.6 Conclusion 46

4 Hierarchy and Communication in Multi-Agent Reinforcement Learning **47**

4.1 Single Agent Soft-Hierarchical Reinforcement Learning 47
4.2 Multi-Agent Delayed Soft-Option Communication 49
4.3 Implementation 52
4.4 Conclusion 55

5 Probabilistic perspective on Public Belief MDP **56**

5.1 Public Belief MDP 56
5.2 Variational Sequential Monte Carlo and Deep Variational Reinforcement Learning . 57
5.3 Probabilistic Public Agent 59
5.4 Implementation 63
5.5 Conclusion 64

6 Conclusion and Future Work **65**
6.1 Summary 65
6.2 Future Works 65

A Chapter 2: Appendix **73**
A.1 Probabilistic Machine Learning 73
A.2 Single Agent Reinforcement Learning 74
A.3 Control as Inference Framework 75

B Chapter 3: Appendix **78**
B.1 ROMMEO 78
B.2 Balancing-Q 79
B.3 Experimentation 81

C Chapter 4: Appendix **83**
C.1 Solution to Single agent Soft-Hierarchical Reinforcement Learning 83
C.2 Solution to Multi-agent Soft-Hierarchical Reinforcement Learning 85

Chapter 1

Introduction

In this chapter, we investigate the problems posted and trying to present the overview of this thesis in more details. We will start by giving a brief overview of control as inference framework/maximum entropy multi-agent reinforcement learning (MEMARL) and provides some examples of the algorithms in the family. Then, we show some of the extensions of this framework to multi-agent problem, and the problem that is arised from doing so[1]. Finally, we describe the whole layout of this thesis by going through the content of each chapter, in detail.

1.1 Research Motivation and Objective

1.1.1 Control as Probabilistic Inference

We shall start with a probabilistic framework for single-agent reinforcement learning problems, before we even jump into the issue of multi-agent problems. Control as probabilistic inference framework [42, 82], just like other probabilistic inference problems, is trying to infer certain distribution, which in this case is $P(a|s, \mathcal{O})$, where, we have, a that represents action, s that represents state and $\mathcal{O}$ that represents the *optimality* of the action and the state of the agent. The optimality random variable is an observable variable defined by

$$P(\mathcal{O}|s, a) \propto \exp(\beta r(s, a)) \tag{1.1}$$

while the agent's state s and action a are treated as latent/unseen variable. Please, pay attention to a small variable β as it will play a crucial role in the future as it is the variable that determines the "rationality" of the agent[2]. However, inference such a distribution can be intractable. We will have to approximate the distribution $P(a|s, \mathcal{O})$ by another parameterized distribution $q_\theta(a|s)$, which we call it a policy/variational distribution. For more details on probabilistic inference, please see chapter 2. We indeed have turned a probabilistic inference problem into an optimization problem,

[1] We will talk about what these algorithms do, but we refer mainly to chapter 2 and other chapters for a more thorough treatment.

[2] Rationality is defined based on the agent's ability to gain more rewards.

where, our optimization objective is similar to a typical single-agent reinforcement learning problem (where we maximize the sum of reward[1]) with additional regularization on the agent policy:

$$\max_{q} \ \mathbb{E}_{q(\tau)} \left[\sum_{t=1}^{T} \beta r(s_t, a_t) - \log \frac{q(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)} \right] \tag{1.2}$$

where $q(\tau)$ is the probability of a state action trajectory that is induced by the agent acting in an environment. The regularization term is called Kullback-Leibler divergence (KL-divergence) [39, 67], which measure the "distance"[2] between 2 probability distributions. The optimization problem from the probabilistic inference framework leads us to the problem of maximizing total reward while maintaining $q(a_t|s_t)$ to be closer to our "prior" on the agent's policy. If the prior distribution is uniform, then regularization in the optimization problem will turn into entropy regularization, hence the name maximum entropy reinforcement learning (MERL). Entropy is associated with the "randomness" of the probability distribution, meaning that agent will be encouraged to be as random as possible while being able to gain higher rewards. This behaviour helps the agent to explore the environment. Furthermore, it is easier when implementing using an entropy regularization, since we don't have to do any engineer on the prior, which is the reason why people usually refer to the framework as "maximum entropy" as they use this special form of regularization more and partly due to historical reason. However, KL-divergence will be useful for us in multi-agent setting. To reduce the confusion, we will call the framework *control as probabilistic inference* but sometimes will use the term MERL, interchangeably.

Before we move to practical algorithms, let's provide a simple explanation of β. When β is closer to zero, the agent will ignore the reward while paying more attention to the regularization of the policy $q(a_t|s_t)$ (forcing it to be closer to the prior). By making the policy closer to the prior, we reduce the rationality of the agent because in the extreme case, where we set β to be zero, the optimal agent according to our objective is the prior policy - a baseline. We can select β in such a way that provides balance between the complexity of the agent and reward gained. This objective can be a representation of "bounded rationality". Finally, this process can be interpreted in many ways via Occam's razor[3] and Information theory [58, 20].

There are two practical algorithms that we can choose to solve this optimization problem: either to optimize this objective directly via policy gradient method[4] *or* finding the closed-form solution. The policy gradient method can be simple, as we can achieve via auto-differentiation and we,

[1] For details on non-regularized reward maximization problem, please see section 2.2 in chapter 2.

[2] KL-divergence is non-symmetric therefore isn't *technically* a distance.

[3] A principle based on Bayesian probability theory stated that "Accept the simplest explanation that fits the data" [48].

[4] See section 2.3.3 in chapter 2 for more details on the method.

therefore, will not go into further details. Now, for the second option, the closed-form solution is:[1]:

$$\begin{aligned} \pi(a_t|s_t) &= \frac{P_{\text{prior}}(a_t|s_t)\exp\left(Q(s_t,a_t)\right)}{\exp\left(V(s_t)\right)} \\ \text{where } V(s_t) &= \log \int P_{\text{prior}}(a_t|s_t)\exp\left(Q(s_t,a_t)\right)\ \mathrm{d}a_t \\ Q(s_t,a_t) &= \beta r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}}\left[V(s_{t+1})\right] \end{aligned} \tag{1.3}$$

Since $Q(s_t, a_t)$ can be represented by a complex function approximator, such as neural network, then finding $V(s_t)$ would be intractable in continuous space. Some algorithms is developed to tackled this intractability by approximate the intractable terms such as the agent's policy. The examples of such algorithm are mainly soft Q-learning [23] and soft Actor-Critic [24, 25]. In conclusion, we can turn the inference problem into more simple optimization problem via approximation of intractable quantity. The approximation procedure would be a recurrent theme of not only this work but also the majority of the development of machine learning algorithms.

1.1.2 MEMARL framework and Its problem

We can divide maximum entropy multi-agent reinforcement learning (MEMARL) algorithms into 2 sub-types:

1. Multi-agent learning as probabilistic inference (MAPI), which includes Probabilistic Fictitious Play [62], Probabilistic Recursive Reasoning (PR2) [80] , Regularized Opponent Model with Maximum Entropy Object (ROMMEO) [75] and Generalized Recursive Reasoning (GR2) [79]

2. Multi-agent learning with KL-constraint (MAKL), which only includes Balancing Q-learning [22]

Optimization objective for them are all in the similar form, which consists of reward and KL-regularization. For instance, the objective for PR2 is

$$\mathbb{E}\left[\sum_{t=0}^{T}\gamma^t\left(\beta r(s_t,a_t,a_t^{-i}) - \log\frac{\pi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)} - \log\frac{\rho(a_t^{-i}|s_t,a_t^i)}{P_{\text{prior}}(a_t^{-i}|s_t,a_t^i)}\right)\right] \tag{1.4}$$

while the objective for Balancing Q-learning is

$$\mathbb{E}\left[\sum_{t=0}^{T}\gamma^t\left(r(s_t,a_t,a_t^{-i}) - \frac{1}{\beta^i}\log\frac{\pi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)} - \frac{1}{\beta^{-i}}\log\frac{\rho(a_t^{-i}|s_t)}{P_{\text{prior}}(a_t^{-i}|s_t)}\right)\right] \tag{1.5}$$

The difference between MAPI and MAKL framework is how they arrived at the final algorithm. On one hand, MAPI derivation is based on the principled control as probabilistic inference framework[2] by trying to find the posterior of $P(a^i, a^{-i}|s, \mathcal{O}_t^i = 1, \mathcal{O}_t^{-i} = 1)$ where

$$P(\mathcal{O}_t^i = 1|\mathcal{O}_t^{-i} = 1, s_t, a_t, a_t^{-i}) \propto \exp\left(\beta r(s, a^i, a^{-i})\right) \tag{1.6}$$

[1]We will provide more details of the derivation and further interpretation in section 2.4.1 in chapter 2. The derivation itself is very important and will be used throughout the text.

[2]We will provide derivations of some algorithms in section 3.1 in chapter 3

where the difference between each algorithm in the family is how the agents' policy is factorized, in ROMMEO, we assume the agent is best responding to opponents' action (left side of equation 1.7). In contrast, in PR2, the opponent's action is based on the agent's action, which we can see as agent learning to *manipulate* its opponent (right side of equation 1.7).

$$P(a^i, a^{-i}|s) = \pi(a^i|s, a^{-i})\rho(a^{-i}|s) \qquad P(a^i, a^{-i}|s) = \pi(a^i|s)\rho(a^{-i}|s, a^i) \tag{1.7}$$

On the other hand, MAKL takes inspiration from the *result* (not the derivation) of control as inference framework, and a related concept of bounded rationality as they explicitly construct Balancing Q-learning from KL-constraint without the use of inference techniques [22].

However, the problem that Balancing Q-learning is trying to solve, and its solution can't be mapped to the current toolbox that MAPI employed, which is when two agents aren't fully cooperative of each others. We will not associated MAKL with how the joint actions[1] are factorized because there is no point comparing an apple to a red pear[2]. So, how can Balancing Q-learning achieve both cooperative and competitive setting? The answer lies in the βs. If both of then have opposite sign, then we have a competitive setting, and if both have the same sign, then we have a cooperative setting. Note that setting negative β in the single-agent case would make the agent trying to avoid any actions that would maximize the reward. An intuitive explanation is that β acts as a mask that changes the agents' perception of the reward. The problem with current MAPI is that we can't force each agent[3] to perceives reward without a common mask or β.

In summary, MAPI provides us with a principled approach to develop multi-agent reinforcement algorithms. However, it lacks expressiveness to solve the competitive game. On the other hand, MAKL gives us a baked formulation that allows us to works on a competitive game, which can be a difficulty when we want to extend the current MEMARL framework. We will need to find a way to develop an algorithm that is similar to one derived from MAKL, but is based on MAPI approach, in order to expand the horizon of multi-agent algorithms development that is based on probabilistic inference, which is one of our goals of this thesis.

1.1.3 Plausible extensions

Now, we have seen the problem regarding the control as probabilistic inference algorithm in multi-agent setting. In this section, we are going to provides the reason *why* we should resolve the problem. We argue that the control as probabilistic inference has shown to be very fruitful in reinterpreting some of the reinforcement learning algorithms. In this thesis, we consider two crucial extensions to multi-agent reinforcement learning, which are hierarchical reinforcement learning (option based) with communication and representing/inferring hidden state in partial observable Markov decision process (POMDP) setting. We choose to tackle both problems first because we consider these abilities to communicate and inferring hidden information to be important in multi-agent setting.

[1] *Spolier* we can map Balancing Q-learning to the probabilistic framework, and it is a PR2-like factorization. This finding would be the main result of chapter 3, which will be in section 3.4.1 and the insight would drive the result of at the end of the chapter.

[2] Of course, this is a reference to "comparing apple to orange" but even if these 2 fruits have the same colour, it doesn't mean we should compare then !

[3] We left the notion of opponent model for the sake of simplification, but the argument still holds

Hierarchical Control with Delayed Option Communication: We want agents to be able to control with temporal abstraction [74, 1]. The setting we are having consists of a master/higher-level agent[1] that emits a new option o_t when the termination policy decided to terminate the option beforehand o_{t-1} by setting $b_t = 1$. The option will condition lower-agent, which associate with real environment action i.e $\pi(a_t|s_t, o_t)$. We can interpret o_t as the mode of action that the agent can execute. This framework has been reinterpreted to probabilistic inference problem in [32] however, it lacks closed formed solution that would allow us to apply soft Actor-Critic (or soft Q-learning) like algorithm. As a minor contribution, we try to solve the probabilistic option framework in single-agent case[2] yielding the closed-form solution that resemblance typical option framework. After tackling single-agent case, we move on to multi-agent case, where we can derive closed-form solution to the probabilistic version of the problem, which leads, ultimately to, a "soft" version of delayed option communication framework proposed in [26].

Representing Belief: In a partially observable setting (either in a single-agent case or a multi-agent case), the agent has to form a belief over the states based on history local observations or information that is received. In the single-agent case, we normally use a recurrent neural network (RNN) as the tools to aggregated this local information and form a belief [27]. However, the algorithm is not "probabilistic" in the sense that there isn't any uncertainty associated with the state representation, since RNN only compress the history into a single variable. There have been developments in representing a belief over state in single-agent domain, and one of the algorithms is called *Deep variational reinforcement learning* (DVRL) [33] that utilizes of variational sequential Monte Carlo (VSMC) [40, 49, 54], an extension of sequential Monte Carlo [12] by parameterizing the transition probability and emission probability that operate on Hidden Markov Model. DVRL allows an agent to jointly learn the model of the environment and maximize its reward. There has been a variance of DVRL, which provides some insight into how similar algorithms can be derived via control as probabilistic inference framework [68]. In a nutshell, VSMC maintains a set of particles, where each of them is a tuple of 2 elements: state representation s_t and its associated weight w_t[3], which is updated every time the agent receives its local information. There has been works done in employing sequential Monte Carlo method in multi-agent case [11] via Interactive-Partial observable Markov decision process (I-POMDP) formulation [21]. The biggest drawback with I-POMDP is that modelling nested belief can be computationally expensive. Given a cooperative setting, we can, alternatively, consider a public belief MDP framework proposed in [57], which has been successfully scaling up via approximation and neural network in [17, 29]. The public belief MDP reduces multi-agent problem into single agent problem and remove nested belief consideration in I-POMDP, which carves the way for us to apply control as probabilistic inference framework to a multi-agent problem.

[1] We will use these terms interchangeably

[2] There has been an attempt in solving it [46]. Still, there are some mistakes with the derivation, as mentioned by the authors. We will point out the error and provide the solution

[3] We have simplified the description, but the arguments still hold. In DVRL, the authors assign particle as a three elements tuple: (z_t, h_t, w_t) where z_t is latent state representation, h_t is the hidden representation of RNN, and w_t is the weight of each particle, which acts as uncertainty quantifier.

1.2 Thesis Content

In chapter 3, we tackles the MAMERL problem, where we showed that all of the algorithms in the MAPI framework doesn't work on the competitive setting. Given the algorithm called Balancing Q-learning, which can work on both cooperative and competitive setting, we reinterpret it via the lens of MAPI framework. This has resulted in probabilistic Balancing Q-learning, which is an algorithm that is similar to Balancing Q-learning while being derived from MAPI framework. By constructing such an algorithm, we have unified MAMERL framework, so that all the algorithms in this framework are based on variational inference technique.

In chapter 4, we solve single-agent soft-option learning problem by finding a solution of the objective, initially, proposed in [32], which leads us to the development of soft actor-critic like, training algorithm, instead of solving the problem via policy gradient. We move on to define an objective for delayed communication and options framework in a multi-agent setting. We show the full derivation of the solution that would optimize the objective we have described. The final optimal answer is, then, turned into an algorithm that utilizes soft Actor-Critic like algorithm allowing off-policy training procedure.

Finally, in chapter 5, we aimed to extend the current MAPI algorithms so that the agents have the belief on the hidden state. Naturally, this would require us to perform a nested belief update. However, via public-belief MDP, we reduce the problem into a POMDP problem, where we can apply control as inference framework. We constructed two components, which are public agent, that performs a belief update, and a low-level agent, that acts on its local information and instruction from the public agent. Finally, we conclude this chapter with the implementation of both components.

1.3 Thesis Structure

This outline in this thesis is in the following list. Chapter 3 onward are the extensions we make given our understanding of MEMARL framework explored in chapter 3.

- **Chapter 2 (Background)**: We will go through the tools necessary to develop the algorithms and formulate of the problem in multi-agent reinforcement learning (MARL)[1]. This chapter includes
  - Variational inference techniques [35], Expectation Maximization (EM), and Stein Variational Gradient Descent (SVGD) [45], which will be used in development and implementation of most of the algorithms.
  - Single agent reinforcement learning: MDP and POMDP formulation, value iteration, and its contraction proof + improvement theorem[2]
    - Basic Deep reinforcement learning, including

[1] We will include an introduction to chapter specific algorithm (for example the explanation of Baysian Action Decoder [17] and DVRL as probabilistic graphical model [68] are in chapter 5) to that chapter. This would reduce the space of the background and make the information easy to search. The intention of this chapter is to the readers to understand the current landscape of MARL algorithms.

[2] We would like to restate those proof because we will encounter similar proof throughout the thesis

        - Deep Q-learning [52] (and its variances such as double Q-learning [76]).
        - Policy Gradient [73] (and its variance such as advantage actor critic (A2C) [51][1], Trust Region Policy Optimization (TRPO) [64] and Proximal Policy Optimization (PPO) [65]).
        - Deep Deterministic Policy Gradient (DDPG) [43] and Twin Delayed DDPG (TD3) [19].
    * Control as inferences frameworks, soft Q-learning [23], and soft Actor-critic [24, 25]
  - Multi-agent reinforcement learning (MARL) algorithms:
    * We will go through stochastic game [66]. This would include the notion of Best Response and Nash Equilibrium [56].
    * We also going to give overviews of value learning based multi-agent reinforcement learning algorithms, which is related to some of the algorithms in the latter chapters. This includes: Nash Q-learning [30] and Friend-or-Foe Q-learning [44].
    * Finally, we will survey importance deep multi-agent reinforcement learning algorithms, including: Multi-Agent DDPG [47], Counterfactual Multi-Agent Policy Gradients (COMA-PG) [16], Value Decomposition Network [71], QMIX [61], and Multi-Agent Variational Exploration (MAVEN) [50].

- **Chapter 3 (Unified view)**: In this chapter, we will consider the full survey of MEMARL framework and try to bring every algorithm into a single control as probabilistic inference framework by examining how each algorithm is derived and observe the standard pattern. This would resulted in a probabilistic version of Balancing Q-learning.

- **Chapter 4 (Hierarchy and Delayed communication MARL)**: We will turn toward augmenting agent with additional components, so that it is able to reason with temporal abstraction. This extension leads to delayed option communication framework. As a minor contribution, we also provide a soft-option learning algorithm for single-agent reinforcement learning based on soft Actor-Critic algorithms.

- **Chapter 5 (Reinterpretation Public Belief MDP)**: Finally, we will propose a reinterpretation of public belief MDP via control as inference framework. This would lead to VSMC like algorithm, which allows the agents to reason in a counterfactual manner without the need for recursive reasoning.

- **Chapter 6 (Conclusion and Future Work)**: We will reviews our progress so far and conclude with future works and research direction.

In the appendixes, we will provide a full derivation of some of the algorithms. We believe that some of the techniques can be useful, thus worth to stated in a complete form. Appendixes will be divided by the content of each chapter.

[1]The citation mainly develop asynchronous update hence the name Asynchronous A2C or A3C. The synchronous version is presented in `https://openai.com/blog/baselines-acktr-a2c/`

Chapter 2

Background

The main aim of this chapter is to make the readers familiar with the background required to understand the development of the algorithms proposed in this thesis. We tried to keep this chapter as small as possible. As noted before, any specific algorithms that will only be discussed in a particular chapter will not be described here; this will also help to find related information more accessible. At the end of this chapter, in section 2.7, we will list the algorithms that will be introduced in the other chapters. Some of the examples here are implemented within mini reinforcement learning library what we have developed.

2.1 Probabilistic Method in Machine Learning

2.1.1 Introduction To Baysian Probability

Most of the focus in this section will be to provide techniques that approximate the solution to Bayes' rule. Suppose we want to infer a parameter θ given training data X and training label Y in supervised learning:

$$P(\theta|X,Y) = \frac{P(Y|X,\theta)P(\theta)}{P(Y|X)} = \frac{P(Y|X,\theta)P(\theta)}{\int P(Y|X,\theta)P(\theta)\ \mathrm{d}\theta} \tag{2.1}$$

For the prediction task, given the input data point x^* that we would like to predict y^*, we can do marginalization of all parameters i.e

$$\int P(y^*|x^*,X,Y,\theta)\ \mathrm{d}\theta = \int P(y^*|x^*,\theta)P(\theta|X,Y)\ \mathrm{d}\theta \tag{2.2}$$

The biggest problem in Bayesian inference is when the integral is intractable, especially in the denominator of the second equality in equation 2.1. Note that in equation 2.2, we can approximate it via Monte-Carlo sample assuming we can sample $P(\theta|X,Y)$. In this thesis, we will focus on variational inference techniques [35]. However, there are many other techniques such as conjugate prior, which we will not discuss them here.

2.1.2 Variational inference techniques

We would like to approximate the posterior $P(\theta|X,Y)$ with parametric distribution $q_\phi(\theta)$, which can be done by minimizing Kullback-Leibler divergence (KL-divergence)[1] between them:

$$D_{\mathrm{KL}}\left(q_\phi(\theta)\middle\|P(\theta|X,Y)\right) = \int q_\phi(\theta)\log\left(\frac{q_\phi(\theta)}{P(\theta|X,Y)}\right)\,\mathrm{d}\theta \tag{2.3}$$

Minimizing the KL-divergence is equivalent to *maximized* the following value, which we will call it Evidence lower bound (ELBO) defined by

$$\begin{aligned}\arg\max_\theta \mathbb{E}_{q_\phi(\theta)}\left[P(Y|X,\theta)\right] - D_{\mathrm{KL}}\left(q_\phi(\theta)\middle\|P(\theta)\right) &= \arg\max_\theta \int q_\phi(\theta)\log\left(\frac{P(Y,\theta|X)}{q_\phi(\theta)}\right)\,\mathrm{d}\theta \\ &= \arg\min_\theta D_{\mathrm{KL}}\left(q_\phi(\theta)\middle\|P(\theta|X,Y)\right)\end{aligned} \tag{2.4}$$

Given this, we can calculate the objective without relying on $P(Y|X)$, the intractable terms. The full derivation is in appendix A.1.1 for completeness. Furthermore, we can derive ELBO based on Jensen's inequality, which is shown in A.1.2[2].

2.1.3 Expectation Maximization (EM) Algorithm

Now, after we have explored the use of variational inference technique, we will turn to EM algorithm. We will follow on the example provided in [14]. Suppose, we have N data points $X = \{x^{(n)}\}_{n=1}^N$ that we assume it is each generated by a hidden variable $z = \left\{z^{(n)}\right\}_{n=1}^N$. We want to learn the "process" that generates such a data point, i.e. the variable θ, such that $P_\theta(X|z)$. Finding θ is done by maximizing the log-likelihood of the observed data point

$$l_{\mathrm{unsup}}(\theta) = \log\left(P_\theta(X)\right) = \log\left(\int P_\theta(X,z)\,\mathrm{d}z\right) \tag{2.5}$$

With this, we can also introduce the variational distribution over the posterior hidden variable (since we don't know what $P(z|X)$ is) $q_\phi(z)$. We can decompose the marginal log-likelihood into:

$$\begin{aligned}l_{\mathrm{unsup}}(\theta) &= \int q_\phi(z)\log\left(\frac{P(X,z)}{P(z|X)}\right)\,\mathrm{d}z \\ &= \underbrace{\int q_\phi(z)\log\left(\frac{P_\theta(X,z)}{q_\phi(z)}\right)\,\mathrm{d}z}_{①} + \underbrace{\int q_\phi(z)\log\left(\frac{q_\phi(z)}{P_\theta(z|X)}\right)\,\mathrm{d}z}_{②}\end{aligned} \tag{2.6}$$

We can show the equality above is true in appendix A.1.3. Now, if we look carefully we can see that part ② is KL-divergence between $q_\phi(z)$ and $P_\theta(z|X)$. This decomposition leads to the exact version of EM algorithm as follows: at step k, we first make the gap between, part ① and log-likelihood,

[1] Note that given 2 difference distributions q and p, $D_{\mathrm{KL}}(q\|p) \neq D_{\mathrm{KL}}(p\|q)$.

[2] We think that minimizing the KL-divergence makes more sense compared to Jensen's inequality. However, we still have to mention it because some of the key papers utilized this technique.

tight by setting the exact posterior to the variational distribution $q_\phi(z)$

$$q_{\phi^{(k+1)}}(z) \leftarrow P_{\theta^{(k)}}(z|X) \tag{2.7}$$

We call this *Expectation-Step* or *E-step*, then we can optimizing θ by maximizing the part (2) of the log-likelihood:

$$\theta^{(k+1)} \leftarrow \arg\max_{\theta^{(k)}} \int P_{\theta^{(k)}}(z|X) \log\left(\frac{P_{\theta^{(k)}}(X,z)}{P_{\theta^{(k)}}(z|X)}\right) \mathrm{d}z \tag{2.8}$$

We call this *Maximization-Step* or *M-Step*. However, it is clear that in M-step we are maximizing ELBO with respect to θ, while in E-step we are maximizing ELBO with respect to ϕ[1]. Now, with this "coincident", we can perform variational EM[2] based on the following ELBO:

$$\mathcal{L}(\theta,\phi) = \int q_\phi(z) \log\left(\frac{P_\theta(X,z)}{q_\phi(z)}\right) \mathrm{d}z = \mathbb{E}_{q_\phi(z)}\left[\log P_\theta(X|z)\right] - D_{\mathrm{KL}}\left(q_\phi(z)\middle\|P(z)\right) \tag{2.9}$$

Now, the training scheme for variational EM is as follows (at step k):

$$\text{E-Step:}\quad \phi^{(k+1)} \leftarrow \arg\max_{\phi^{(k)}} \mathcal{L}(\theta^{(k)},\phi^{(k)}) \qquad \text{M-Step:}\quad \theta^{(k+1)} \leftarrow \arg\max_{\theta^{(k)}} \mathcal{L}(\theta^{(k)},\phi^{(k+1)}) \tag{2.10}$$

This is deployed in famous variational auto-encoder [37].

2.1.4 Stein Variational Gradient Descent (SVGD) [45]

SVDG is used in many of the algorithms that we are presenting (for example, soft Q-learning [23] and PR2 [80]). According to the authors, it is a "general purpose variational inference algorithm" [45], which can also be used to "sample" the data point from complex distribution (optimized variational distribution). SVGD is extremely useful since we want to train our neural network to represent a particular distribution.

Suppose that we want to sample the distribution[3] P by a set of n particles $\{\xi_i^{(0)}\}_{i=1}^n$. Then we can update the set of such particles such that the final set of n particles $\{\xi_i^{(T)}\}_{i=1}^n$ represent sampled point from the distribution P. The algorithm does the following update rule (for iteration at step t):

$$x_i^{(t+1)} \leftarrow x_i^{(t)} + \alpha\left(\frac{1}{n}\sum_{j=1}^{n}\left[\kappa(x_j^{(t)},x_i^{(t)})\nabla_{x^{(t)}{}_j}\log P(x_j^{(t)}) + \nabla_{x^{(t)}{}_j}\kappa(x_j^{(t)},x_i^{(t)})\right]\right) \tag{2.11}$$

where α is the updating step and κ is the kernel function, which typically Gaussian. We can use other optimization methods, such as ADAM [36], to optimize the points. The intuition behind this is that the update rule follows the functional gradient of KL-divergence between the variational distribution ($q_{[T]}$ which is the density of $z = T(\xi)$) and the true distribution P (Theorem 3.3). See

[1] Recall from section 2.1.2 that minimizing KL is same as maximizing ELBO
[2] We are using variational distribution to approximate the posterior.
[3] which is usually unnormalized

the original paper for more details.

2.2 Single Agent Reinforcement Learning

In this section, we will go through the classical single agent reinforcement learning as a basis to the next section, which is about deep reinforcement learning - a scale and approximated version of algorithms discuss here. The contents are re-arranged from [72] and [69].

2.2.1 Markov Decision Process

Markov Decision Process (MDP) formulates the task that reinforcement learning trying to solve[1].

Definition 1 *Markov Decision Process (MDP) is a tuple :* $\langle S, A, R, T, p_0, \gamma \rangle$ *where*

- *State space:* $S = \{s_0, s_1, \cdots, s_{|S|}\}$. *Usually we have* $s_i \in \mathbb{R}^{d_s}$
- *Action space:* $A = \{a_0, a_1, \cdots a_{|A|}\}$
- *Reward function:* $\mathcal{R} : S \times A \to \mathbb{R}$
- *Transition function:* $T : S \times A \times S \to [0, 1]$, *where* $s^{(t+1)} \sim T(s^{(t+1)}|s^{(t)}, a^{(t)})$
- *Initial state distribution:* $p_0 : S \to [0, 1]$, *where* $s^{(0)} \sim p_0$
- *Discounted factor:* $\gamma \in (0, 1]$

We define player's policy $\pi : S \to \Delta(A)$ *where* $\Delta(A)$ *is probability simplex over* A, *and* $a^{(t)} \sim \pi(a^{(t)}|s^{(t)})$ *denote player choose action* a_t *at state* s_t *at time* t. *The goal of any reinforcement learning algorithms is to maximizes the sum reward with or without full description/experience of MDP.*

We denote the value function of the player's policy $V^\pi(s)$ as the expected future cumulative reward given the state the agent is in:

$$V^\pi(s) = \mathbb{E}_{a_t \sim \pi, s_t \sim T}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s\right] \tag{2.12}$$

And, the action-value function is the expected future reward when the agent performs an action a at state s:

$$Q^\pi(s, a) = \mathbb{E}_{a_t \sim \pi, s_t \sim T}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 = s, a_0 = a\right] \tag{2.13}$$

The advantage function of an agent is simply the differences between value function and action value function i.e $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$. Furthermore, We establish the connection between both $V(s)$ and $Q(s, a)$ functions as

$$Q^\pi(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)}\left[V^\pi(s')\right] \qquad V^\pi(s) = \mathbb{E}_{a \sim \pi(a|s)}\left[Q^\pi(s, a)\right] \tag{2.14}$$

[1] It is a mathematical description of "game".

We denote the optimal value function and optimal action value function as $V^*(s)$ and $Q^*(s,a)$ when $V^{\pi^*}(s)$ and $Q^{\pi^*}(s,a)$, respectively, where $\pi^* \in \arg\max_\pi V^\pi(s)$ for any state s. Now, we want to establish the connection between these 2 optimal functions, which is:

$$Q^*(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)} \left[V^*(s') \right] \qquad V^*(s) = \max_{a \in A} Q^*(s,a) \tag{2.15}$$

We can now slightly change the objective of reinforcement learning to "How to find such an optimal policy".

2.2.2 Policy Iteration

We will now introduce the first algorithm that will solve the problem of reinforcement learning, called policy iteration. The algorithm is based on a simple observation that the agent will always be improving or remain the same if we greedily choose the action that would maximize expected future rewards given Q^π:

$$\pi'(a|s) = \begin{cases} 1 & \text{if } a = \arg\max\limits_{a \in A} Q^\pi(s,a) \\ 0 & \text{otherwise} \end{cases} \tag{2.16}$$

We call this step, a policy improvement step. One can show that $\forall s \in S : V^\pi(s) \leq V^{\pi'}(s)$, and we are able to reach an optimal value function if we keep improving the policy. The proof, for completeness, will be presented in appendix A.2.1. What we have left is to find an algorithm that gives us value function $V^\pi(s)$ for any policy π, which we will call this step *policy evaluation*. We can expand the equality in equation 2.14 as:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(a|s)} \left[r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)} \left[V^\pi(s') \right] \right] \tag{2.17}$$

We call this equation expected Bellman equation along with the following expected Bellman operator $\mathcal{B}^\pi : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ defined as:

$$\mathcal{B}^\pi V(s) = \mathbb{E}_{a \sim \pi(a|s)} \left[r(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{B}(s'|s,a)} \left[V(s') \right] \right] \tag{2.18}$$

To solve the policy evaluation problem, we want to find $V(s)$ such that $\mathcal{B}^\pi V(s) = V(s)$ however, to solve this via matrix inversion would be too expensive. However, turn out this expected Bellman operator is a contraction mapping on ∞-norm i.e

$$\|\mathcal{B}^\pi V_1(s) - \mathcal{B}^\pi V_2(s)\|_\infty \leq \alpha \|V_1(s) - V_2(s)\|_\infty$$

where $\|V(s)\|_\infty = \max_{s \in S} V(s)$. For some $0 \leq \alpha < 1$ (See appendix A.2.2 for proof), then by Banach fixed-point theorem

Theorem 2 *(Banach fixed-point theorem [53]) Given the complete (every Cauchy sequence converges to a point in that space) metric space $(X, d(\cdot,\cdot))$ and the contraction mapping $\mathcal{B} : X \to X$, the sequence $(\mathcal{B}^{(n)}x)_{n=1}^\infty$ converges to a fixed point x^* where $\mathcal{B}x^* = x^*$, for all points $x \in X$.*

repeatedly apply the Bellman operator will lead us to a solution of V^π. In conclusion, Policy

iteration is an algorithm that solves the MDP by involving the alternation between the following two steps.

1. **Policy Evaluation**: Starting with randomized value function and repeatedly applying Bellman operator defined in equation 2.18 until converge to the true value function $V^{\pi}(s)$.

2. **Policy Improvement**: Update the policy by choosing the best action from action-value function (we can calculate this by the equation 2.14) following the equation 2.16.

Since the value function always increases in every state, this algorithm is guaranteed to reach the optimal value function and policy.

2.2.3 Value Iteration

Now, it is computational ineffective to evaluate the policy every time to update the policy. Can we update the value function using single Bellman update to reach the optimal value function? The answer is *Yes*. We define optimal value Bellman update operator as:

$$\mathcal{B}_V^* V(s) = \max_{a \in A} \Big[r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)} \left[V(s') \right] \Big] \tag{2.19}$$

Note that it is related to optimal Bellman equation represented in equation 2.15, where V^* is a fixed point. In this case, we update the value function toward the action that yields the highest value given only one update. This operator is, indeed, a contraction mapping, thus repeatedly updating the value function by optimal value Bellman operator will provide us with optimal value function. The proof will be similar to the proof done in the section above. Finally, given the optimal value function, one can calculate the optimal action-value function and the optimal policy.

2.2.4 Q Iteration

Now, instead of using value function as in value iteration, it is always more desirable for us to estimate optimal action-value function $Q^*(s,a)$ directly. The optimal action-value function has an advantage, in which we can find the best action directly without any extra calculation. One can define the optimal action-value Bellman operator as

$$\mathcal{B}_Q^* Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim T(s'|s,a)} \left[\max_{a \in A} Q(s',a) \right] \tag{2.20}$$

this operator is indeed a contraction mapping with Q^* as its fixed point. The proof is similar compare to other algorithms. However, there is more to this. Since there is no need to consider the transition function explicitly, we defined the following update rule given iteration k and (s_t, a_t) experience:

$$Q^{(k+1)}(s_t,a_t) \leftarrow (1-\alpha_t) Q^{(k)}(s_t,a_t) + \alpha_t \Big[r(s_t,a_t) + \gamma \max_{a'} Q^{(k)}(s_{t+1},a') \Big] \tag{2.21}$$

where $0 < \alpha < 1$. Following the theorem 1 of [34], which is also a basis of proofing Q-Iteration states that:

Theorem 3 *[34] The process* $\Delta_{t+1} = (1 - \alpha_t(x))\Delta_t + \alpha_t(x)F_n(x)$ *will converge to zero if and only if The state space is finite,* $\sum_t \alpha_t(x) = \infty$, $\sum_t \alpha_t(x)^2 < \infty$, $\left\|\mathbb{E}\left[F_n(x)\middle|P_n\right]\right\| \leq \beta\|\Delta_n\|$ *where* $\beta \in (0,1)$ *and* $\mathrm{Var}\left[F_n(x)\middle|P_n\right] \leq C\left(1 + \|\Delta_n\|\right)^2$*, where* P_n *is a collection of histories.*

By subtracting optimal Q function at both ends i.e $\Delta_t = Q^{(k)}(s_t, a_t) - Q^*(s_t, a_t)$ and $F_n(x) = r(s_t, a_t) + \gamma \max_{a'} Q^{(k)}(s_{t+1}, a') - Q^*(s_t, a_t)$, we can proof that the update in equation will converge to the optimal Q-value. Given Theorem 3, we can now change any contraction mapping into a stochastic update rule and still maintains the convergence property. We will use this throughout the text.

2.2.5 Partially observable Markov decision process

Before we move to deep reinforcement learning, we shall consider another problem that requires special attention, which will be useful in chapter 5. Partial observable MDP (POMDP) is defined as

Definition 4 *We define a POMDP as a tuple* $\langle S, A, O, \mathcal{T}, p_0, U, R, \gamma\rangle$ *where*

- *State space:* $S = \{s_0, s_1, \cdots, s_{|S|}\}$
- *Action space:* $A = \{a_0, a_1, \cdots a_{|A|}\}$
- *Observation space:* $O = \{o_0, o_1, \cdots, o_{|O|}\}$
- *Transition function:* $T : S \times A \times S \to [0, 1]$
- *Initial state distribution:* $p_0 : S \to [0, 1]$
- *Noisy or partially occluded observation:* $o_{t+1} \sim U(o_{t+1}|s_{t+1}, a_t)$
- *Reward distribution function:* $r_{t+1} \sim R(r_{t+1}|s_{t+1}, a_t)$
- *Discount factor* $\gamma \in \mathbb{R}$

The policy conditions its action based on histories of observation $\pi(a_t|o_{\leq t})$ *where* $o_{\leq t}$ *is the history of observation before time* $t + 1$*.*

2.3 Deep Reinforcement Learning

Now, it is always the case that the total number of states is too big for us to represent the exact value function or action-value function. We will now present the way that we can approximate these values, together with another method for control, notably policy gradient and the combination of both paradigms: Actor-Critic algorithm.

2.3.1 Deep Q-learning [52]

Deep Q-learning is based on traditional Q-iteration, whereby instead of using stochastic iteration (see section 2.2.4), the authors try to approximate optimal Q-function using neural network $Q_\theta(s, a)$

by training it to minimize mean square error between its prediction and the target prediction i.e

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')_{i=1}^{n} \sim B(\mathcal{D})} \left[\frac{1}{n} \sum_{i=1}^{n} \left(Q_\theta(s_i, a_i) - r(s_i, a_i) - \gamma \max_{a'} Q_{\theta^-}(s'_i, a'_i) \right)^2 \right] \tag{2.22}$$

The training relies on experience replay $\mathcal{D}$, which stores the past experiences in the tuple, and target action-value function $Q_{\theta'}(s, a)$ added to increase stability in training. The following process updates the target action-value function at time step t

$$\theta^-_{t+1} \leftarrow \rho\theta^-_t + (1 - \rho)\theta_t \tag{2.23}$$

which is a common way to update a target function. $\mathcal{B}$ is a mini-batch sampler. Furthermore, due to the size of state space, the authors use ε-greedy policy to help the agent collecting the experience (better exploration) so that it gets more diverse set of training data. The policy is defined as

$$\pi(a|s) = \begin{cases} \arg\max_{a'} Q(s, a') & \text{with probability } \varepsilon \\ a' \sim \text{Uniform}(a_0, a_1, \ldots, a_{|A|}) & \text{otherwise} \end{cases} \tag{2.24}$$

and ε decays overtime. The authors showed that given the following training scheme (with some minor tricks), the agent could achieve super-human performance on Arcade Learning Environment [3]. Note that there are some games that the agent doesn't learn at all. The infamous Montezuma's revenge is one instance of this. Because the agent has to learn to reason over long time steps to solve the game, Q-learning isn't sufficient enough for reasoning over the long time step. This problem can be mitigated via hierarchical reinforcement learning [38, 10, 77] or intrinsic motivation [60].

Other techniques that improve the training stability or increase its performance are, notable, Double-Q learning [76], which tackles the over-optimism prediction by using the action output of *current iteration* of agent estimate the value of target network in the target i.e

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')_{i=1}^{n} \sim B(\mathcal{D})} \left[\frac{1}{n} \sum_{i=1}^{n} \left(Q_\theta(s_i, a_i) - r(s_i, a_i) - Q_{\theta^-}(s'_i, \arg\max_{a'} Q_\theta(s'_i, a')) \right)^2 \right] \tag{2.25}$$

Furthermore, Prioritized DQN [63], which allows the experience replay to sample higher quality experience tuple, Dueling DQN [78], which increase the capacity of the estimation of action-value function by combining approximate value function and approximate advantage function, Distributional DQN [2], which estimates the distribution over the action-value function [59], Noisy Net [18], which injects the noise into the output of neural network layers, aiming to improves explorations, and combination of them all - rainbow DQN [28].

2.3.2 Policy Gradient [73]

Now, instead of approximating value function, why don't we directly optimized $\pi_\theta(a|s)$ given the sum of expected reward as objective i.e

$$\pi^*_\theta(a|s) = \arg\max_\theta \eta(\pi_\theta) = \arg\max_\theta \mathbb{E}_{a_t, s_t \sim \pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \tag{2.26}$$

The strategy is to find the gradient of the sum of expected reward, which we can state that it is equal to

$$\frac{\partial}{\partial\theta}\mathbb{E}_{a_t,s_t\sim\pi_\theta}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\right]=\mathbb{E}_{a_t,s_t\sim\pi_\theta}\left[R_t\frac{\partial}{\partial\theta}\log\pi_\theta(a_t|s_t)\right] \tag{2.27}$$

where $Q^\pi(s_t,a_t)=\mathbb{E}\left[R_t|s_0=s_t,a_0=a_t\right]$. Now, we can perform stochastic approximation via Monte Carlo sampling, while we can train a neural network using stochastic gradient ascent. We call this basic algorithm REINFORCE [73]. However, this algorithm is notoriously hard to train due to the high variance of the estimator. We can reduce the variance by minus the Q-value estimation by a baseline $B(s)$ value that is independent of θ and a. In usual case, we use value function estimator B_t[1], which leads us to use of advantage function $A^\pi(s,a)$ estimator[2]. Furthermore, we can add an entropy regularizer to help the policy learns via maximum entropy principles, see section 1.1.1 for a quick introduction.

2.3.3 Actor-Critic

Now, instead of using a Monte Carlo estimation of the reward, why don't we try to estimate this quality using function approximator like a neural network? By using a neural network approximation, we now move toward Actor-Critic algorithm; hence, the name "Advantage Actor-Critic Algorithms". [51][3]. The usual implementation is as follows: for the actor/policy network, the implementation is similar. But for critic/advantage function estimator, we only need to train value function $V_\theta(s_t)$ and estimate the advantage function:

$$A(s_t,a_t)\approx\underbrace{\sum_{i=0}^{n}\gamma^i r_{t+i}+\gamma^{n+1}V_\theta(s_{n+1})}_{\approx Q_\theta(s_t,a_t)}-V_\theta(s_t) \tag{2.28}$$

The longer the n times steps, the more accurate (and more variance) the estimator going to become. We can use this value to train the agent. Now, we can train the value function by minimizing the following objective:

$$\mathcal{L}(\theta)=\mathbb{E}_{(r_0,\cdots,r_{n+1}),s_{n+1}\sim\mathcal{D}}\left[\frac{1}{2}\left(V_\theta(s_0)-\sum_{i=0}^{n}\gamma^i r_i-\gamma^{n+1}V_\theta(s_{n+1})\right)^2\right] \tag{2.29}$$

which is a mean-square error between predictive value and bootstrapped[4] value. Now, we have laid the essential foundation to Actor-Critic (AC) method let's consider two variances of successful extension to AC, which are TRPO [64] and PPO [65]. Let's start with TRPO, which is based on the following theorem presented in [64]:

[1] where $V^\pi(s_t)=\mathbb{E}\left[B_t|s_0=s_t\right]$

[2] This would make sense intuitively because advantage function only estimates how good each action is given the current state, which is the only measurement we want for policy gradient algorithm.

[3] Also see `https://openai.com/blog/baselines-acktr-a2c/`

[4] Bootstrap is the act that we include $V_\theta(s_{n+1})$ after a certain length of cumulative reward instead of having full trajectory, i.e. using one prediction as part of the target for other prediction

Theorem 5 *Let* $D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}) = \max_s D_{\mathrm{KL}}\left(\pi(\cdot|s) \middle\| \tilde{\pi}(\cdot|s)\right)$ *then the following holds*

$$\eta(\tilde{\pi}) \geq L_{\pi}(\tilde{\pi}) - \frac{4\varepsilon\gamma}{(1-\gamma)^2} D_{\mathrm{KL}}^{\max}(\pi, \tilde{\pi}) \tag{2.30}$$

where $\varepsilon = \max_{s,a} |A_{\pi}(s,a)|$ *and* $L_{\pi}(\tilde{\pi})$ *is defined as:*

$$L_{\pi}(\tilde{\pi}) = \eta(\pi) + \sum_s \rho_{\pi}(s) \sum_a \tilde{\pi}(a|s) A_{\pi}(s,a) \tag{2.31}$$

As the visitation frequency $\rho_{\pi}(s) = \sum_{n=0}^{\infty} \gamma^n P(s_n = s)$

Given $M_i(\pi) = L_{\pi_i}(\pi) - C D_{\mathrm{KL}}^{\max}(\pi_i, \pi)$, we have the inequality $\eta(\pi_{i+1}) \geq M_i(\pi_{i+1})$ and $\eta(\pi_i) = M_i(\pi_i)$, this leads to $\eta(\pi_{i+1}) - \eta(\pi_i) \geq M_i(\pi_{i+1}) - M_i(\pi_i)$. This implies that if we optimize $M_i(\pi)$, then we would optimize $\eta(\pi)$. Now, we can turn $M_i(\pi)$ into constrained optimization problem

$$\begin{aligned} &\max_{\theta} L_{\theta_{\text{old}}}(\theta) \\ &\text{such that } D_{\mathrm{KL}}^{\max}(\theta_{\text{old}}, \theta) \leq \delta \end{aligned} \tag{2.32}$$

The problem is that the KL-divergence might not be tractable. We can approximate this as $D_{\mathrm{KL}}^{\rho}(\theta_1, \theta_2) = \mathbb{E}_{s\sim\rho}\left[D_{\mathrm{KL}}\left(\pi_{\theta_1}(\cdot|s) \middle\| \pi_{\theta_2}(\cdot|s)\right)\right]$. Furthermore, the authors also proposed the importance sampling scheme, the objective becomes

$$\begin{aligned} &\mathbb{E}_{s\sim\rho_{\theta_{\text{old}}}, a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_{\pi_{\theta_{\text{old}}}}(s,a)\right] \\ &\text{such that } D_{\mathrm{KL}}^{\rho_{\theta_{\text{old}}}}(\theta_{\theta_{\text{old}}}, \theta) \leq \delta \end{aligned} \tag{2.33}$$

The trust region is the KL-constraint imposed on policy optimization by not allowing the updated policy to be too far from the current policy. All the proofs and implementations are explored in details in [64]. PPO [65] is the approximate version of TRPO. There are two variances for PPO that are proposed in the paper:

- Clipped Surrogate Objective: Using the following object for the objective

$$\mathbb{E}\left[\min\left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_t, \operatorname{clip}\left(\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}, 1-\varepsilon, 1+\varepsilon\right) A_t\right)\right] \tag{2.34}$$

  where ε is hyper-parameter and usually set to 0.2

- Adaptive KL Penalty Coefficient: We consider the unconstrained version of KL-divergence and update its weight on KL-divergence β based on the current evaluation:

$$\mathbb{E}\left[\frac{\pi_{\theta}(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A_t - \beta D_{\mathrm{KL}}\left(\pi_{\theta_{\text{old}}}(\cdot|s) \middle\| \pi(\cdot|s)\right)\right] \text{ where } \beta \leftarrow \begin{cases} \beta/2 & \text{if } d < d_{\text{target}}/1.5 \\ \beta * 2 & \text{if } d > d_{\text{target}} * 1.5 \\ \beta & \text{otherwise} \end{cases} \tag{2.35}$$

2.3.4 Deep Deterministic Policy Gradient

Now, we will consider the problem in which the agent's action is continuous. The resulting algorithm based on Actor-Critic variance called Deep Deterministic Policy Gradient (DDPG) [43]. We will also consider the extension to Twin Delayed DDPG (TD3) [19] to make the training of DDPG more stable. Starting with DDPG, the core intuition behind it[1] is that we train the agent to maximize critic's prediction of reward, i.e. maximizing the following objective with respect to θ

$$\mathcal{L}(\theta) = Q_\phi(s, \pi_\theta(s)) \tag{2.36}$$

Note that we didn't do any sampling the agent, hence the name *deterministic*. The actor is training with typical mean-square error loss, similar to equation 2.29:

$$\mathcal{L}(\phi) = \mathbb{E}_\mathcal{D}\left[\frac{1}{2}\left(Q_\phi(s,a) - r - \gamma Q_{\phi^-}(s', \pi_\theta(s'))\right)^2\right] \tag{2.37}$$

Now, TD3 extends DDPG algorithms as follows

- Using double-Q learning. By training the 2 critics together with the following target (hence the name "twin")

$$\begin{gathered} r + \gamma \min_{i\in\{1,2\}} Q_{i,\phi^-}(s', a') \\ \text{where } a' = \text{clip}\left(\pi_{\theta^-}(s'), \text{clip}(\varepsilon, -c, c), a_{\text{low}}, a_{\text{high}}\right) \qquad \varepsilon \sim \mathcal{N}(0, \sigma) \end{gathered} \tag{2.38}$$

- We update actor network in lesser frequency than the critic (this includes the target actor, which is updated similar to equation 2.23)

This two techniques will be a trick for other algorithms to improve the performance and stability.

2.4 Control as Inference Framework

2.4.1 Derivations for closed form solution

We will base the derivation on [42]. The derivation of this will be the basis for *all* derivations of the algorithms in this thesis; therefore, we will state full proof within the section. As introduced in section 1.1.1, we would like to find the posterior where the agent is optimal. Let's recall the definition of optimality.

$$P(\mathcal{O}_t = 1 | s_t, a_t) \propto \exp(\beta r(s_t, a_t)) \tag{2.39}$$

Let's consider the graphical model representation for joint distribution, which is in figure 2.1. The joint probability is equal to

$$P(s_{1:T}, a_{1:T}, \mathcal{O}_{1:T} = 1) = P(s_0)\prod_{t=1}^{T} P(s_{t+1}|s_t, a_t) P_{\text{prior}}(a_t|s_t) P(\mathcal{O}_t = 1|s_t, a_t) \tag{2.40}$$

[1] There are more into the theory of how it works. We suggest reading [70]

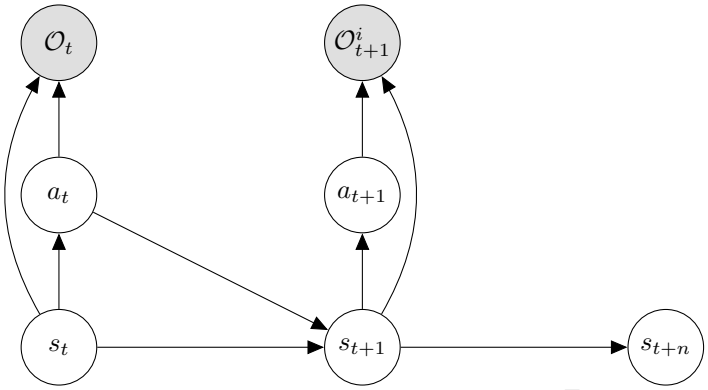


Figure 2.1: The graphical model representation of joint probability that we want to approximate for single agent reinforcement learning.

Now, we want to approximate this joint probability using the following graphical model, which is depicted in figure 2.2. Now, the variational joint probability is equal to[1]

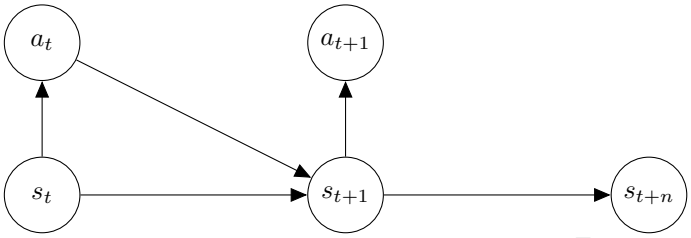


Figure 2.2: The graphical model representation of the variational joint probability that we will use to approximate optimal join probability in represented in equation 2.40.

$$q(s_{1:T}, a_{1:T}) = P(s_0) \prod_{t=1}^{T} P(s_{t+1}|s_t, a_t)\pi(a_t|s_t) \tag{2.41}$$

Let's try to maximizing the negative KL-divergence between 2 probability distribution:

$$\begin{aligned} -D_{\mathrm{KL}}\left(q(s_{1:T}, a_{1:T}) \middle\| P(s_{1:T}, a_{1:T}|\mathcal{O}_{1:T} = 1)\right) &= -\mathbb{E}_{q(s_{1:T}, a_{1:T})}\left[\log \frac{q(s_{1:T}, a_{1:T})}{P(s_{1:T}, a_{1:T}|\mathcal{O}_{1:T} = 1)}\right] \\ &= \mathbb{E}_{q(s_{1:T}, a_{1:T})}\left[\log \frac{P(s_{1:T}, a_{1:T}|\mathcal{O}_{1:T} = 1)}{q(s_{1:T}, a_{1:T})}\right] \\ &= \mathbb{E}_{q(s_{1:T}, a_{1:T})}\left[\sum_{t=1}^{T} \beta r(s_t, a_t) - \log \frac{\pi(a_t|s_t)}{P_{\mathrm{prior}}(a_t|s_t)}\right] \end{aligned} \tag{2.42}$$

[1] see the equation below

The full expansion will be in the appendix A.3.1 with alternative Jensen's inequality derivation in appendix A.3.2 for completeness[1], and please note that $\tau = (s_{1:T}, a_{1:T})$. We can use a policy gradient to help us on optimizing the objective: that is fine. But, let's consider the closed-form solution to this problem. We are trying to solve it from the last time step and then inductively solving the equation to general time step t. Start with the final time step T, which we want to maximize the following objective:

$$\mathbb{E}_q \left[\beta r(s_T, a_T) - \log \frac{\pi(a_T|s_T)}{P_{\text{prior}}(a_T|s_T)} \right] \tag{2.43}$$

We will introduce $V(s_T)$; however, its value will be clear after the derivation:

$$\mathbb{E}_q \left[\beta r(s_T, a_T) - \log \frac{\pi(a_T|s_T)}{P_{\text{prior}}(a_T|s_T)} + V(s_T) - V(s_T) \right] \tag{2.44}$$

Now, expand and rearrange the equation, which leads to:

$$\mathbb{E}_q \left[V(s_T) \right] - \mathbb{E}_q \left[D_{\text{KL}} \left(\pi(a_T|s_T) \middle|\middle| \frac{\exp(\beta r(s_T, a_T)) P_{\text{prior}}(a_T|s_T)}{\exp(V(s_T))} \right) \right] \tag{2.45}$$

If we want to maximize the objective, which is the same as minimizing KL-divergence, we can set the policy to be

$$\begin{aligned} \pi(a_T|s_T) &= \frac{\exp(\beta r(s_T, a_T)) P_{\text{prior}}(a_T|s_T)}{\exp(V(s_T))} \\ \text{where } V(s_T) &= \log \int \exp(\beta r(s_T, a_T)) P_{\text{prior}}(a_T|s_T) \, \mathrm{d}a_T \end{aligned} \tag{2.46}$$

We can see that $V(s_T)$ is a normalizing factor for a probability distribution. Note that it is only the case where the agent is in the form of Boltzmann policy that the value function is the normalizing factor. By setting the policy π to minimize the KL-divergence, the objective becomes $V(s_T)$, which we would call it "backward message" and we pass it down to the time step before, where we maximize the following objective with backward messages $\mathbb{E}_{s_{t+1}}[V(s_{t+1})]$ with discounted factor as:

$$\mathbb{E}_q \left[\underbrace{\beta r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}}[V(s_{t+1})]}_{Q(s_t, a_t)} - \log \frac{\pi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)} + V(s_t) - V(s_t) \right] \tag{2.47}$$

Using the same method, we have the following general time policy t.We, sometimes, call this kind of policy a Boltzman Policy:

$$\begin{aligned} \pi(a_t|s_t) &= \frac{\exp(Q(s_t, a_t)) P_{\text{prior}}(a_t|s_t)}{\exp(V(s_t))} \\ \text{where } V(s_t) &= \log \int \exp(Q(s_t, a_t)) P_{\text{prior}}(a_t|s_t) \, \mathrm{d}a_t \end{aligned} \tag{2.48}$$

while we have Bellman like equation (see equation 2.14 for comparison)

$$Q(s_t, a_t) = \beta r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}}[V(s_{t+1})] \tag{2.49}$$

[1]We have shown in section 2.1.2, and this is going to be the last derivation with Jensen's inequality

Now, let's consider the meaning of for $V(s_t)$, [42] consider the "value function"[1] to be a "soft-max"[2] of the action value function. In [41], the authors changes the value of β to balance between traditional deep Q-learning (when $\beta \to \infty$) and expected value based on prior i.e $\mathbb{E}_{a \sim P_{\text{prior}}}[Q(s,a)]$ (when $\beta \to 0$) in order to mitigate the problem of over-estimation of deep Q-learning. Furthermore, in [25], the authors has consider automatic β adjustment via constrained optimization. Let's further consider other form of $V(s_t)$, which we re-arrange the policy function:

$$V(s_t) = \mathbb{E}_{a_t \sim \pi(a_t|s_t)} \left[Q(s_t, a_t) - \log \frac{\pi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)} \right] \tag{2.50}$$

Note that, we consider the policy to be the one represented in equation 2.48[3]. Now, substitute the Q-function into the value function, which we have the recursive definition of value function:

$$V(s_t) = \mathbb{E}_{a_t \sim \pi(a_t|s_t)} \left[\beta r(s_t, a_t) - \frac{\pi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)} + \gamma \mathbb{E}_{s_{t+1}}[V(s_{t+1})] \right] \tag{2.51}$$

Which we can roll out as the sum of regularized reward. This equality, therefore, has established the connection between this value function and reinforcement learning value function. Before we move to the implementation, let's consider some theoretical property of this Bellman equation. First, we can construct the soft Bellman operator as follows:

$$\mathcal{B}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p_s} \left[\log \int \exp(Q(s_{t+1}, a)) P_{\text{prior}}(a|s_{t+1}) \, \mathrm{d}a \right] \tag{2.52}$$

which we can prove that it is a contraction mapping, see appendix A.3.3 for the proof, which is a restatement from [23], while this will be useful later. Furthermore, we can show that the update of the policy leads to an increase in Q-function, which is proven in [24] and will be restated and proof here.

Theorem 6 *Let the updated policy π_{new} be*

$$\pi_{new} = \arg\min_{\pi' \in \Pi} D_{\text{KL}} \left(\pi'(\cdot|s_t) \middle|\middle| \frac{\exp(Q^{\pi_{old}}(s_t, \cdot)) P_{prior}(\cdot|s_t)}{V^{\pi_{old}}(s_t)} \right) \tag{2.53}$$

Then $\forall s_t, a_t : Q^{\pi_{new}}(s_t, a_t) \geq Q^{\pi_{old}}(s_t, a_t)$.

The proof is in appendix A.3.4 for completeness. Both results will be useful in chapter 3 for the analysis of algorithms. Finally, this gives us the basic theoretical finding that is analogous to policy iteration for normal reinforcement learning.

[1] In our case, it acts like a value function, however, for now (before showing a sufficient reason), we will consider it only to be a normalizing factor.

[2] This isn't the same *softmax* as the one used in classification. However, it has the same property as maximize if $\beta \to \infty$, which is proven in [14]

[3] Some of the mistakes made comes from the consideration of expected Q function without regularized terms for value function.

2.4.2 Soft Q-Learning

Now we have the closed-form solution for the optimization problem. For implementation procedure, first, let's consider how can we model the value function, from the definition, we can derive the following Monte-Carlo estimation with importance sampling i.e

$$V_\theta(s_t) = \log \mathbb{E}_{q_{a'}} \left[\frac{\exp(Q(s_t, a_t)) P_{\text{prior}}(a_t|s_t)}{q_{a'}(a')} \right] \tag{2.54}$$

Now, for the action-value function, we have the following prediction problem with minimizing Mean-square error:

$$\mathcal{L}_Q(\theta) = \mathbb{E}_{s_t, a_t} \left[\frac{1}{2} \left(Q_\theta(s_t, a_t) - \hat{Q}_{\theta^-}(s_t, a_t) \right)^2 \right] \tag{2.55}$$

where the target is defined by

$$\hat{Q}_{\theta^-}(s_t, a_t) = \beta r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p_s} \left[V_{\theta^-}(s_{t+1}) \right] \tag{2.56}$$

Now, for the policy, we start by consider a policy to be in a form of $a_t = f_\phi(\xi; s_t)$ where $\xi \sim \mathcal{N}(0, I)$, which we try to minimize the following function:

$$\mathcal{L}(\phi; s_t) = D_{\text{KL}} \left(\pi_\phi(\cdot|s_t) \middle\| \exp\left(Q_\theta(s_t, \cdot) - V_\theta(s_t) \right) P_{\text{prior}}(\cdot) \right) \tag{2.57}$$

We can find the direction $\Delta f_\phi(\cdot; s_t)$ that reduces the KL-divergence via SVGD, which gives us following update terms:

$$\begin{aligned} \Delta f_\phi(\cdot; s_t) = \mathbb{E}_{a_t \sim \pi^\phi} \Big[& \kappa\left(a_t, f_\phi(\cdot; s_t)\right) \nabla_{a'} \left(Q_\theta(s_t, a') + \log P_{\text{prior}}(a') \right) \Big|_{a'=a_t} \\ & + \alpha \nabla_{a'} \kappa\left(a', f_\phi(\cdot; s_t)\right) \Big|_{a'=a_t} \Big] \end{aligned} \tag{2.58}$$

Now, the training direction for the policy is:

$$\frac{\partial \mathcal{L}(\phi; s_t)}{\partial \phi} \propto \mathbb{E}_\xi \left[\Delta f_\phi(\cdot; s_t) \frac{\partial f_\phi(\xi; s_t)}{\partial \phi} \right] \tag{2.59}$$

2.4.3 Soft Actor-Critic

In Soft Actor-Critic case, we learn 3 networks: Q-function $Q_\theta(s_t, a_t)$ and Value-Function $V_\psi(s_t)$ and Policy $\pi_\phi(a_t|s_t)$. Let's start with value function, instead of using Monte-Carlo estimation, we shall use definition from equation 2.50 as the target value, which we can minimize mean-square error:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V_\psi(s_t) - \mathbb{E}_{a_t \sim \pi_\phi} \left[Q_\theta(s_t, a_t) - \log \frac{\pi_\phi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)} \right] \right)^2 \right] \tag{2.60}$$

where $\mathcal{D}$ is just the replay memory. The action-value function training objective is the same as soft Q-learning (see equation 2.55 and 2.56) but,now, we have independent value function instead. For policy, we directly optimize the KL-divergence, which we can ignore the normalizing terms because it doesn't depend on the action, while the policy can be represented as $f_\phi(\xi; s_t)$ similar to soft

Q-learning:

$$\mathcal{L}_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[D_{\text{KL}} \left(\pi_\phi(\cdot|s_t) \middle|\middle| \frac{\exp(Q_\theta(s_t, \cdot)) P_{\text{prior}}(\cdot|s_t)}{Z_\theta(s_t)} \right) \right] \tag{2.61}$$

The authors also use normalizing flow to get the output of the policy to be bounded within a range. We suggested [24] for more details.

2.5 Introduction to Multi-Agent Problems

2.5.1 Problem Formulation and Solution Concepts

Now, we have explored single-agent reinforcement learning. Let's move the multi-agent reinforcement learning. We will start by defining the problem definition in a multi-agent case, which is a stochastic game [66] formation.

Definition 7 *Stochastic Game is a tuple:* $\langle S, N, A_1, \cdots, A_N, \mathcal{T}, p_{01}, \cdots, p_{0N}, R_1, \cdots, R_N, \gamma \rangle$ *where*

- *State space:* $S = \{s_0, s_1, \cdots, s_{|S|}\}$.
- *Number of agents:* $N \in \mathbb{N}$.
- *Action space for player i:* $A_i = \{a_0, a_1, \cdots, a_{|A_i|}\}$.
- *Transition function:* $T : S \times A_1 \times \cdots \times A_N \times S \to [0, 1]$.
- *Initial state distribution for player i:* $p_{0i} : S \to [0, 1]$.
- *Reward distribution function for player i:* $r_{i\ t+1} \sim R_i(r_{i\ t+1}|s_{t+1}, a_{1\ t}, \cdots a_{N\ t})$.
- *Discount Factor:* $\gamma \in \mathbb{R}$.

Now, the goal of agents is to maximize its expected reward

$$\arg\max_{\pi_i} \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r^i(s_t, a_t^i, a_t^{-i}) \right] \tag{2.62}$$

This objective might be too vague as we don't know what kind of opponent we are going to play against, and the same problem also applied to the opponent. Let's simplify the problem by *fixing* the opponent and then updating the agent's policy alone. By doing this, we reduce the problem into a single-agent problem, which we can define this as the agent that best responses to a particular set of opponent i.e

$$\begin{aligned} &\text{BR}^i(\pi_1, \cdots, \pi_{i-1}, \pi_{i+1}, \cdots, \pi_N) \\ &\qquad = \arg\max_{\pi_i} \mathbb{E}_{\pi_1, \cdots, \pi_i, \cdots, \pi_N} \left[\sum_{t=0}^{\infty} \gamma^t r^i(s_t, a_t^1, \cdots, a_t^i, \cdots, a_t^N) \right] \end{aligned} \tag{2.63}$$

Now, Nash equilibrium is the set of polices[1] where every policy doesn't benefit from any deviation, suppose $\pi_1^*, \cdots, \pi_N^*$ are set of policies that are Nash equilibrium then

$$\begin{aligned} \forall i \in [1, N], \forall \pi_i \in \Pi_i : \mathbb{E}_{\pi_1^*, \cdots, \pi_i^*, \cdots \pi_N^*} \left[\sum_{t=0}^{\infty} \gamma^t r^i(s_t, a_t^1, \cdots, a_t^i, \cdots, a_t^N) \right] \\ \geq \mathbb{E}_{\pi_1^*, \cdots, \pi_i, \cdots \pi_N^*} \left[\sum_{t=0}^{\infty} \gamma^t r^i(s_t, a_t^1, \cdots, a_t^i, \cdots, a_t^N) \right] \end{aligned} \tag{2.64}$$

Nash equilibrium exists for all the games [56], which is a very important result. Nash equilibrium is where we should aim, however, solving Nash equilibrium is in complexity class PPAD, which is a subset of NP problems [9] and even approximating it is PPAD-complete [8]. We will leave the problem of define the *optimality* of agent here and will explore it in chapter 3. Finally, we would like to denote the action that are not coming from agent i as a^{-i} i.e $a^{-i} = (a^1, \cdots, a^{i-1}, a^{i+1}, \cdots, a^N)$. Finally, we define cooperative game to be the game that all the agents are optimizing a common reward.

2.5.2 Classical Reinforcement Algorithms

We will consider two basic algorithms for solving multi-agent reinforcement learning, which are both value function based[2]. Starting with Nash Q-learning [30], which we aim to estimate Nash Q-function. It is defined as: Given a set of Nash Equilibrium policies $\pi_1^*, \cdots, \pi_N^*$, we have

$$Q_{\text{Nash}}^i(s_t, a_t^1, \cdots, a_t^N) = r(s_t, a_t^1, \cdots, a_t^i, \cdots, a_t^N) + \gamma \mathbb{E}_{s_{t+1}} \left[V^i(s_{t+1}, \pi_1^*, \cdots, \pi_N^*) \right] \tag{2.65}$$

where V^i is the cumulative discounted reward that follows $\pi_1^*, \cdots, \pi_N^*$. Based on the use of maximum to estimate optimality Q-value[3] we can, instead, use the following operation:

$$\text{Nash}\, Q_t^i(s) = \pi^1(s) \cdots \pi^n(s) Q^i(s, a_1, \cdots, a_n) \tag{2.66}$$

which is the expected payoff of Nash equilibrium solution to matrix game[4] given a list of value functions $Q^1(s', \vec{a}), \cdots, Q^N(s', \vec{a})$ as a payoff function. The authors shown the following operator

$$\mathcal{B}Q^i(s', a_1, \cdots, a_N) = r_i + \gamma \mathbb{E}_{s_{t+1}} \left[\text{Nash}\, Q_t^i(s_{t+1}) \right] \tag{2.67}$$

is a contraction mapping, which means when repeated applying the map will lead to Nash action value function. The problem with Nash-Q learning is how can we select that Nash equilibrium. [44] proposed to use friend-or-foe Q-learning that is based on the identification of the agents whether they are friends (i.e trying to maximize our reward) or foes (i.e trying to minimize our reward),

[1] Or sets of policies as a game can have more than 1 Nash equilibrium
[2] We find the optimal Q-value and then acts according to it.
[3] See equation 2.15 and 2.2.4
[4] There is one state game with immediate reward given once all the actions are executed.

and using this to update the value,

$$\max_{\pi_1,\cdots,\pi_k} \min_{\pi_k,\cdots,\pi_N} \sum_{s'} \pi^i(s') \cdots \pi^n(s') Q^i(s, a_1, \cdots, a_n) \tag{2.68}$$

Friend-or-foe Q-learning has the same convergence property as Nash Q-learning[1], but has a unique solution unlike the Nash equilibrium solution.

2.6 Deep Multi-Agent Reinforcement Learning

There are two types of algorithm used for training for multi-agent problems: centralized training and decentralized training algorithms. Centralized training algorithms are algorithms that require shared components between each agent to train, while the decentralized algorithms don't. Note that most of the algorithms here are centralized training algorithms. On the other hand, we can also follow single-agent training algorithms classification, which separates value-based algorithms (for example DQN [52]) from policy-based algorithms (for example, REINFORCE [73] and A2C [51]).

2.6.1 Policy Based

Counterfactual Multi-Agent Policy Gradients (COMA-PG) [16]

We consider solving cooperative multi-agent problems where the main problem comes from the fact that it is difficult for the training algorithm to assign correct credit to correct agent. COMA-PG [16] solved this problem by proposing the following baseline for a policy gradient algorithm:

$$D^i = r(s, a) - r(s, (a_c^i, a^{-i})) \tag{2.69}$$

where a_c is the *default action* for agent i. The default action can be, simply, an expected action, so now the advantage function becomes:

$$A^i(s, a) = Q(s, a) - \sum_{a_i'} \pi^i(a_i'|s) Q(s, (a_i', a^{-i})) \tag{2.70}$$

We can now use it to train a normal policy gradient algorithm.

Multi-Agent DDPG (MADDPG) [47]

Now, we can consider the policy gradient for any multi-agent problem as, where we can treat a joint actions $(a_1, \ldots, a_N)$ as one action from one agent:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim d(s), a_i \sim \pi_i} \left[\nabla_{\theta_i} \log \pi_i(a_t|s_i) Q_i^{\pi}(s, a_1, \ldots, a_N) \right] \tag{2.71}$$

[1] *Spolier*: Balacing Q-learning is a "soft" version of friend-or-foe Q-learning

Now, the authors extended this to deterministic policy (similar to DDPG, see section 2.3.4 for more details) μ_{θ_i}, which is defined as

$$\nabla_{\theta_i} J(\mu_{\theta_i}) = \mathbb{E}_{s,a\sim\mathcal{D}} \left[\nabla_{\theta_i} \mu_{\theta_i}(a_i|s_i) \nabla_{a_i} Q_i^{\mu}(s, a_1, \ldots, a_N)|_{a_i=\mu(s_i)} \right] \tag{2.72}$$

where $\mathcal{D}$ is experience replay that contains a tuple $(s, s', a_1, \ldots, a_N, r_1, \ldots, r_N)$, and, finally, the critic can be trained by minimizing the following objective

$$\mathcal{L}(\theta_i) = \mathbb{E}_{(s,a,s',r)\sim\mathcal{D}} \left[\left(Q_i^{\mu}(s, a_1, \ldots, a_N) - r_i - \gamma Q_i^{\bar{\mu}}(s', a'_1, \ldots, a'_N)|_{a'_i=\bar{\mu}(s'_i)} \right)^2 \right] \tag{2.73}$$

The algorithm is *centralized* as we need have to have full knowledge of other agent's policy and observation.

2.6.2 Value Function Based

Now we have seen the use of value estimation in MADDPG; let's consider the method of centralized value function estimation. Please note that all the algorithms in this section assume a cooperative setting. The main intuition behind all value function based algorithm is that the action value function Q can be decomposed to each agent's action value function. In value decomposition network (VDN) [71], the authors assume the following factorization of the agent:

$$Q(s(a^1, \ldots, a^N)) \approx \sum_{i=1}^{N} \tilde{Q}_i(s, a^i) \tag{2.74}$$

Note that this works nicely in the case of cooperative setting as in competitive game the value function mixing is likely to be more *complex* interactions, in contrast to, cooperative as they are likely moving toward the same goal. QMIX [61] is an extended version of VDN, by including a mixing network, instead of using the summation. However, there has to be some constrain on the function. Notably,

$$\arg\max_a Q_{\text{tot}}(s, a) \;=\; \begin{pmatrix} \arg\max_a Q_1(s, a) \\ \arg\max_a Q_2(s, a) \\ \vdots \\ \arg\max_a Q_N(s, a) \end{pmatrix} \tag{2.75}$$

The VDN satisfies the condition but there is a more generalized notion of this, which is monotonicity.

$$\forall n \in [N] : \frac{\partial Q_{\text{tot}}}{\partial Q_n} \geq 0 \tag{2.76}$$

A monotonic function can be represented relatively well [13] by making the function positive (≥ 0), which we can use ReLU function [55] to satisfies this condition. The training is a standard Q-value objective. Finally, as shown in [50], QMIX can suffer from learning sub-optimal action-value function with a poor exploration. To prevent this from happening, MAVEN [50] learns multiple modes of value functions given latent variable z, with the following process

- The value function is conditioned on shared latent variables z which is controlled by hierar-

chical policy $z = g_\theta(x, s_0)$ where x is value that is sampled from simple distribution $P(x)$

- For a given z, each joint action-value function is monotonic approximation of optimal action value function $g^a_\eta(o^a_{1:t}, a_{1:t}, u^a_{1:t-1})$ together with $g_\phi(z, a)$ that is a neural network that modify the utility for particular mode of exploration. Now all value function are mixed using g_ψ function

We will train the Q-network as following

$$\mathcal{L}_Q(\phi, \eta, \psi) = \mathbb{E}_{\pi_a}\left[Q(s_t, a_t; z) - \left[r(s_t, a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; z)\right]\right] \tag{2.77}$$

while the hierarchical policy is trained cumulative rewards $R(\tau, z|\phi, \eta, \psi) = \sum_t r_t$

$$\mathcal{L}_{RL}(\theta) = \int R(\tau_A|z) P_\theta(z|s_0) \;\; \mathrm{d}z \; \mathrm{d}s_0 \tag{2.78}$$

To prevent collapse of the latent variable z, the mutual information objective for a trajectory $\tau = \{(u_t, s_t)\}$ is introduced. Now, the mutual information loss is lower bounded by

$$\begin{aligned} \mathcal{L}_{ML} &= \mathcal{H}(\sigma(\tau)) - \mathcal{H}(\sigma(\tau)|z) = \mathcal{H}(z) - \mathcal{H}(z|\sigma(z)) \\ &\geq \underbrace{\mathcal{H}(z) + \mathbb{E}_{\sigma(z),z}\left[\log q_v(z|\sigma(\tau))\right]}_{\mathcal{L}_v(\phi,\eta,\psi,v)} \end{aligned} \tag{2.79}$$

where we need σ, an operator which returns Boltzman policy given the action-value function, while the trajectory is encoded using RNN. This regularization leads to an auxiliary reward as $r^z_{\text{aux}}(z) = \log(q_v(z|\sigma(z))) - \log(p(z))$. The final objective is

$$\max_{v,\phi,\eta,\psi,\theta} \mathcal{L}_{RL}(\theta) + \lambda_{MI}\mathcal{L}_v(\phi, \eta, \psi, v) - \lambda_Q \mathcal{L}_Q(\phi, \eta, \psi) \tag{2.80}$$

2.7 Other Related Algorithms

This is the list of the algorithms that will be introduced in the following chapters:

- Chapter 3: Regularized Opponent Model with Maximum Entropy Objective (ROMMEO) [75], Probabilistic Recursive Reasoning (PR2) [80], and Balancing Q-learning [22].
- Chapter 4: Soft Hierarchical reinforcement learning [32, 46] and Delayed communication or Dynamic policy termination Q-learning [26].
- Chapter 5: Variational Sequential Monte Carlo, Deep Variational Reinforcement Learning [33, 68] and Sequential Variational Soft Q-Learning (SVQN) [31]. They are the basis for representing belief, which is a basis for public belief MDP [57] and Bayesian Action Decoder (BAD) [17].

Chapter 3

Unified View of Probabilistic Multi-agent Reinforcement Learning

Now, we will survey the maximum entropy multi-agent reinforcement learning (MEMARL) framework, where we examine into algorithms within control as probabilistic inference framework (MAPI). By, re-deriving one of the algorithms, ROMMEO [75], we understand how the algorithms in MAPI generally operates. The rederivation and re-interpretation of ROMMEO will lead us to the alternative derivation of Balance Q-learning [22] and a probabilistic interpretation of this algorithm.

3.1 Introduction to MAPI framework

3.1.1 What is MAPI ?

All of the algorithms in MAPI frameworks are decentralized MARL algorithms. They are based on an opponent model but not a kind of opponent model in fictitious play [4], where the agent assumes its opponent to be stationary as it tries to predict the opponent's move and best response to that opponent model. MAPI agents, instead, train its opponent model given its opponent reward. Since we know the agent's opponent model, we can reduce the multi-agent problem into a single-agent problem. During the training phase, we will assume that the action given by the real opponent comes from our assumed opponent model, and we train our agent accordingly. The difference between ROMMEO [75] and PR2 [80] happens when we define which component we optimize first. In ROMMEO's case, we train the agent's policy first then train the opponent model, and PR2 vice versa. Thus, we can view PR2's opponent model $\pi(a^{-i}|s, a^i)$ as an opponent's policy doing ROMMEO like training on its opponent (the agent) model $\pi(a^i|s)$.

A training algorithm that incorporates the learning of the opponent model with the learning of the agent's policy can be beneficial in a cooperative case since both agent's policy and the real opponent (not agent's opponent model) try to optimize the same objective. However, this is also a weakness because if we have an opponent with drastically difference goal (for example in the case of

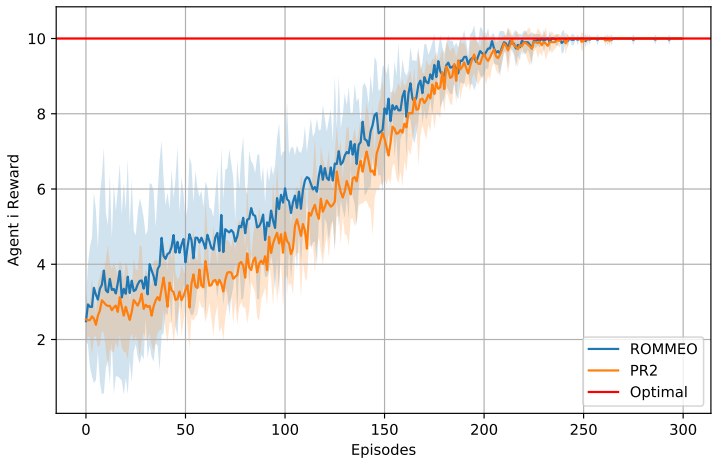


Figure 3.1: The training progress of ROMMEO and PR2 on the cooperative game where the payoff is defined in equation 3.1.

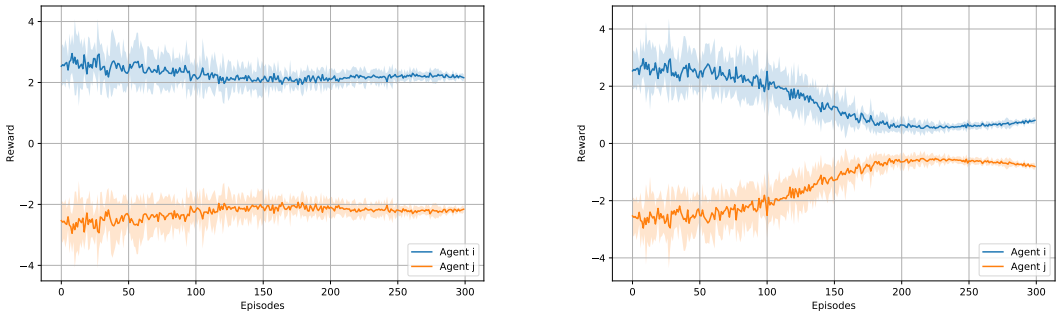
a competitive game), the agent would still update its opponent model to act as if it is maximizing the agent's reward. The problem will be more apparent once we have understood the result on the impact of βs in theorem 8. We propose a solution that decouples the agent's training from the opponent's model. We have created an experiment showing how PR2 and ROMMEO fail to understand its opponent's action. We shall start with the payoff function, which is

$$R = \begin{bmatrix} 2 & 3 & 3 \\ 1 & 4 & 0 \\ 1 & 0 & 10 \end{bmatrix} \tag{3.1}$$

If agent i selects a as its action and if agent j selects b as its action, then the reward of agent i, r^i, is R_{ab} while the reward for agent j, r^j is $-R_{ab}$. The Nash equilibrium is when agent i selects action 1 and agent j selects action 1 as their action, represented as $(1, 1)$, which would gives agent i reward 2 and -2 for agent j. Furthermore, in the cooperative case, where $r^j = R_{ab}$ there are 2 Nash equilibriums, which are $(2, 2)$ and $(3, 3)$, where Nash equilibrium at $(3, 3)$ is better. This game is similar to a climbing game proposed in [6] that is used in [75]. While having similar property to the climbing game, the game we proposed has a pure Nash equilibrium at it zero-sum counter part.

Let's start with training ROMMEO [75] and PR2 [80] on the cooperative setting. We calculate the action-value function $Q(a_t^j, a_t^j)$ using the stochastic update rule as following a stochastic updating rule presented in theorem 3. We also employ a temperature scheduling because during training due to reward differences the agent's policy converges prematurely. The details of the experiment will be discussed in appendix B.3.1. The experiment results are collected by average over five experiments, each with different seeds. The baseline results are presented in figure 3.1. The reward of agent j would be the same. We can see that ROMMEO does slightly better than PR2 as expected from the results presented in [75]. We can see that due to the temperature schedule, the variances of both curves for the first 150 episodes are high, which is also a sign of exploration. Now, let's consider agent i training when dealing with adversarial agent j to study the impact of

the variable β^i and β^j of agent i. We will follow the Balancing Q-learning algorithm, in which we have the following results shown in figure 3.2. First, in figure 3.2a, both agents converge to a Nash equilibrium when using Balancing Q-learning, which isn't the case for PR2. By misrepresentation of its opponent, agent i trained under PR2 algorithm will choose action 3 (since in cooperative case this is the best action) giving a lower reward of almost 0 leading to suboptimal performance. On the other hand, agent i that is trained under Balancing-Q will choose action 1 yielding the reward of 2, and reaches Nash equilibrium.



(a) Balancing Q-learning with adversarial opponent (b) PR2 with misrepresentation of the opponent

Figure 3.2: A comparison between PR2 and Balancing Q-learning with adversarial opponent. PR2 can only represent cooperative opponent. Given the same set of seed the final reward are differ.

3.1.2 What is Optimal ?

We usually assume the optimality of the agent *relative* to our knowledge of the opponent. If we are best responding to an optimal agent then we are likely to be optimal too[1]. By jointly optimizing the agent with its opponent model, we can participate in what our opponent *could be* given our assumptions and objective. This assumption is crucial, which will allow us to extend the framework so that it covers MAKL framework. We want to note that the assumption of an opponent's optimality is used in [75]. However, it lacks the insight that the opponent's model doesn't have to follow the same objective as the agent, which we can show in the derivation of ROMMEO. In conclusion, MAPI is an enhanced probabilistic version of the fictitious play, where the opponent model is a fully trained policy. At the same time, the optimality of each agent should be independent while being based on each other, which is an inevitable contradiction but we will show that this is possible given special circumstances of Balancing Q-learning.

3.2 Derivation of ROMMEO

Now, we shall consider the derivation of ROMMEO using the technique introduced in [42], which is shown briefly in section 2.4.1. The working is partially covered in [81, 79] however a full derivation is required for the understanding of the algorithm. We will base our derivation for ROMMEO [75],

[1]This is true in Nash equilibrium case by the definition.

while the same technique can be applied to PR2 [80]. ROMMEO assumes that the joint probability between 2 agents, which can be factorized as

$$\pi(a, a^{-i}|s) = \pi(a|a^{-i}, s)\rho(a^{-i}|s) \tag{3.2}$$

factorizing in other ways yields PR2. We will start by drawing the graphical model of joint probabilities that we want to approximate, and is depicted in Figure 3.3. Before we move on, please note

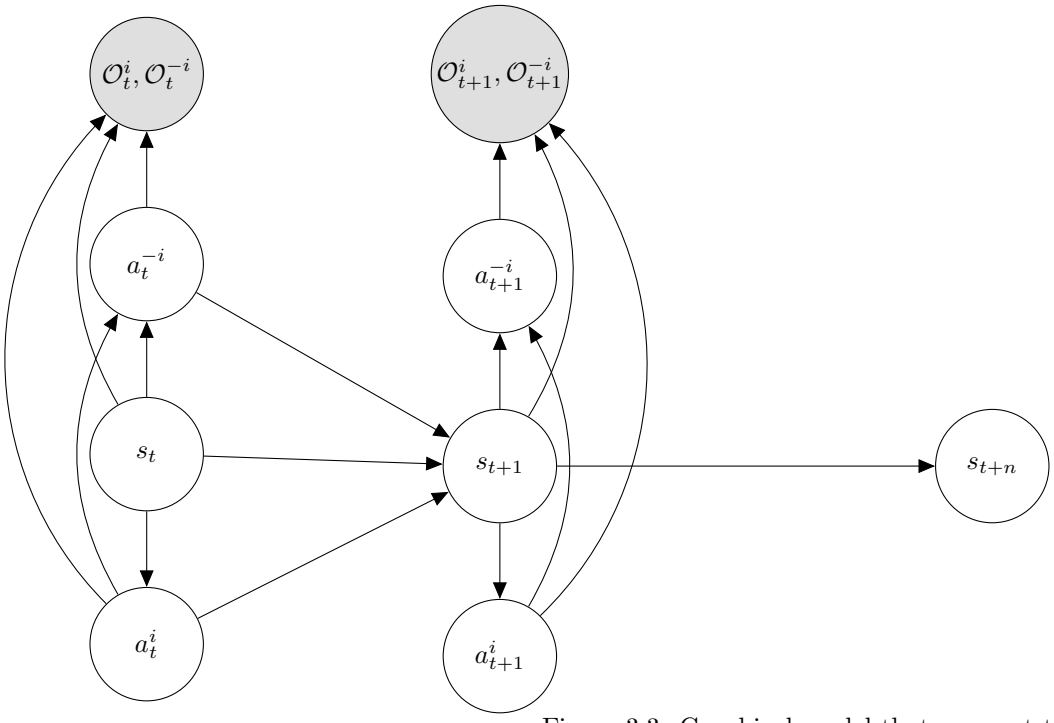


Figure 3.3: Graphical model that we want to approximate. This is based on the joint probability provided in ROMMEO paper, with slight modification. The authors assume that the opponent we are playing against is optimal

that this is all within an agent calculation, while the opponent's policy will have the same process. Given this, we can show that the prior joint probability is equal to the following:

$$\begin{aligned} &P(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T}, \mathcal{O}^{-i}_{1:T} = 1, \mathcal{O}^i_{1:T} = 1) \\ &\quad = P(s_0) \prod_{t=0}^{T} P_{\text{prior}}(a_t^i|s_t, a_t^{-i}) P_{\text{prior}}(a_t^{-i}|s_t) P(s_{t+1}|s_t, a_t^i, a_t^{-i}) P(\mathcal{O}_t^{-i} = 1, \mathcal{O}_t^i = 1|s_t, a_t^i, a_t^{-i}) \end{aligned} \tag{3.3}$$

where the optimality variable is defined as

$$\begin{aligned} &P(\mathcal{O}_t^{-i} = 1, \mathcal{O}_t^i = 1|s_t, a_t^i, a_t^{-i}) \propto \exp\left(\beta r(s_t, a_t^i, a_t^{-i})\right) \\ &P(\mathcal{O}_t^{-i} = 1, \mathcal{O}_t^i = 1|s_t, a_t^i, a_t^{-i}) = P(\mathcal{O}_t^{-i} = 1|\mathcal{O}_t^i = 1, s_t, a_t^i, a_t^{-i}) P(\mathcal{O}_t^i = 1|\mathcal{O}_t^{-i} = 1, s_t, a_t^i, a_t^{-i}) \end{aligned} \tag{3.4}$$

This definition is different from the one proposed in [75]. However, we would like to point out that the opponent model has been optimized *after* the agent's policy, which will show in a later section. Therefore, during the optimization of the opponent model, it has assumed that the agent's policy is optimal, hence the factorization of the optimality. Furthermore, the factorization of the optimality random variables above is a mere representation because the distinction between agent's and its opponent model lies during the calculation of the solution and not during the inference. Now for the variational joint probabilities that we want to optimize on the graphical model is depicted in Figure 3.4. The joint probability of variational distribution can be calculated as following

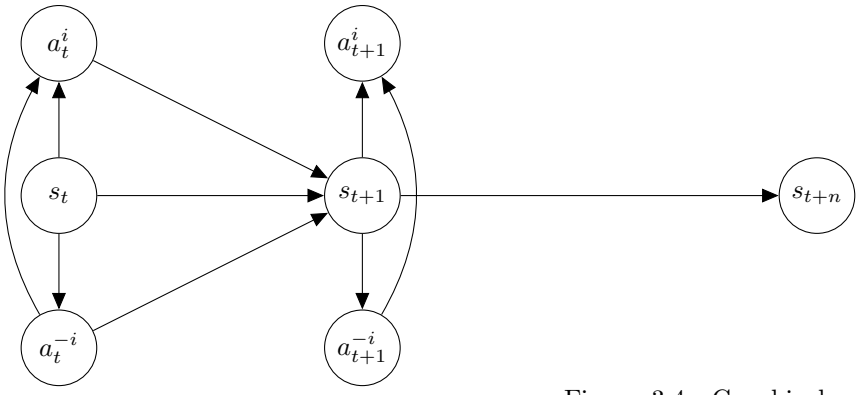


Figure 3.4: Graphical model that we are going to optimize our policies on. This has almost the same structure as the version that we want to approximate. However, we don't care about the optimality of the agent itself, as we want to approximate its policy (denote as π) and the its opponent model (denote as ρ)

$$q(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T}) = P(s_0) \prod_{t=0}^{T} \pi_\theta(a^i_t|s_t, a^{-i}_t)\rho_\phi(a^{-i}_t|s_t)P(s_{t+1}|s_t, a^i_t, a^{-i}_t)P(\mathcal{O}^{-i}_t = 1) \tag{3.5}$$

We would like to solve the following optimization problem

$$\underset{\pi,\phi\in\Pi\times\Phi}{\arg\min}\ D_{\mathrm{KL}}\left(q(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T}) \middle\| P(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T}|\mathcal{O}^i_{1:T} = 1\mathcal{O}^{-i}_{1:T} = 1)\right) \tag{3.6}$$

This leads to optimizing the following ELBO. The derivation is shown in the appendix B.1.1, which translates the problem into a maximization problem of the following objective

$$\underset{\pi,\phi\in\Pi\times\Phi}{\arg\max}\ \mathbb{E}_q\left[\sum_{t=0}^{T}\gamma^t\left(\beta r(s_t, a^i_t, a^{-i}_t) - \log\frac{\pi_\theta(a^i_t|s_t, a^{-i}_t)}{P_{\text{prior}}(a^i_t|s_t, a^{-i}_t)} - \log\frac{\rho_\phi(a^{-i}_t|s_t)}{P_{\text{prior}}(a^{-i}_t|s_t)}\right)\right] \tag{3.7}$$

We will start off by considering the last time step, as we will progress backwards in time. Considering the last time step:

$$\mathbb{E}_q\left[\beta r(s_T, a^i_T, a^{-i}_T) - \log\frac{\pi_\theta(a^i_T|s_T, a^{-i}_T)}{P_{\text{prior}}(a^i_T|s_T, a^{-i}_T)} - \log\frac{\rho_\phi(a^{-i}_T|s_T)}{P_{\text{prior}}(a^{-i}_T|s_T)}\right] \tag{3.8}$$

We can show that the optimal policy is equal to

$$\begin{aligned}\pi_\theta(a_T^i|s_T, a_T^{-i}) &= \frac{\exp\left(\beta r(s_t, a_T^i, a_T^{-i})\right) P_{\text{prior}}(a_T^i|s_T, a_T^{-i})}{\exp\left(Q(s_T, a_T^{-i})\right)} \\ \text{where } Q(s_T, a_T^{-i}) &= \log \int \exp\left(\beta R(s_T, a_T^i, a_T^{-i}\right) P_{\text{prior}}(a_T^i|s_T, a_T^{-i})\ \mathrm{d}a_T^i\end{aligned} \tag{3.9}$$

The proof will be presented in the appendix B.1.2. This part is where we can assume that the agent policy is optimal and allow us to calculate the opponent model objective. By plugging the optimal policy back, we are left with the opponent's objective:

$$\mathbb{E}_{\rho_\phi(a_T^{-i}|s_T)P(s_T)}\left[Q(s_T, a_T^{-i}) - \log \frac{\rho_\phi(a_T^{-i}|s_T)}{P_{\text{prior}}(a_T^{-i}|s_T)}\right] \tag{3.10}$$

which will give us the optimal opponent model at time step T to be as the following:

$$\begin{aligned}\rho_\phi(a_T^{-i}|s_T) &= \frac{\exp\left(Q(s_T, a_T^{-i})\right) P_{\text{prior}}(a_T^{-i}|s_T)}{\exp\left(V(s_T)\right)} \\ \text{where } V(s_T) &= \log \int \exp\left(Q(s_T, a_T^{-i})\right) P_{\text{prior}}(a_T^{-i}|s_T)\ \mathrm{d}a_T^{-i}\end{aligned} \tag{3.11}$$

All the proofs including the opponent's objective is in appendix B.1.3. Given both agent and its opponent's model policy, we can consider the message passed to the time steps before, and we can see that we are left with

$$\mathbb{E}_{P(s_T)}\left[V(s_T)\right] \tag{3.12}$$

This quantity will be passed down to the later time. Now for time t, the objective that we are optimizing becomes

$$\mathbb{E}_q\left[\underbrace{\beta r(s_t, a_T^i, a_T^{-i}) + \gamma \mathbb{E}_{P(s_{t+1}|s_t, a_t^i, a_t^{-i})}\left[V(s_{t+1})\right]}_{Q(s_t, a_t^i, a_t^{-i})} - \log \frac{\pi_\theta(a_T^i|s_t, a_t^{-i})}{P_{\text{prior}}(a_t^i|s_t, a_t^{-i})} - \log \frac{\rho_\phi(a_t^{-i}|s_t)}{P_{\text{prior}}(a_t^{-i}|s_t)}\right] \tag{3.13}$$

We can see that this leads to very similar problem as the one we have solved before. Therefore, by using the same proving process, we can derive the optimal agent's policy and optimal opponent model policy to be

$$\pi_\theta(a_t^i|s_t, a_t^{-i}) = \frac{\exp\left(Q(s_t, a_t^i, a_t^{-i})\right) P_{\text{prior}}(a_t^i|s_t, a_t^{-i})}{\exp\left(Q(s_t, a_t^{-i})\right)} \qquad \rho_\phi(a_t^{-i}|s_t) = \frac{\exp\left(Q(s_t, a_t^{-i})\right) P_{\text{prior}}(a_t^{-i}|s_t)}{\exp\left(V(s_t)\right)} \tag{3.14}$$

where the analogous "Bellman equation" is the following (with the value and action value functions being)

$$\begin{aligned}Q(s_t, a_t^i, a_t^{-i}) &= \beta r(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E}_{P(s_{t+1}|s_t, a_t^i, a_t^{-i})}\left[V(s_{t+1})\right] \\ \text{where } Q(s_t, a_t^{-i}) &= \log \int \exp\left(Q(s_t, a_t^i, a_t^{-i}\right) P_{\text{prior}}(a_t^i|s_t, a_t^{-i})\ \mathrm{d}a_t^i \\ V(s_t) &= \log \int \exp\left(Q(s_t, a_t^{-i})\right) P_{\text{prior}}(a_t^{-i}|s_t)\ \mathrm{d}a_t^{-i}\end{aligned} \tag{3.15}$$

This concludes the derivation for ROMMEO in its most general form. If we consider PR2, then we have to maximize opponent model first, by using a similar process.

3.3 Interpretation of ROMMEO

3.3.1 Understanding ROMMEO

Before we examine the results, let's state some of the theoretical results for ROMMEO, which are discussed in [75]. The authors showed that the Bellman equation in equation 3.15 is a contraction mapping, which shows that there exists a fixed point of agent and its opponent model action value function, while the updating agent and its opponent model does improve the action value. These results are analogous to results from section 2.4.1. Thus ROMMEO is implemented via soft Actor-Critic like algorithm, see section 2.4.3 and [75] for more details.

Let's examine the results that we get from the derivation of ROMMEO. We start with the definition of value function $V(s)$. Let's expand the meaning of it:

$$V(s_t) = \mathbb{E}_{\pi_\theta, \rho_\phi}\left[Q(s_t, a_t^i, a_t^{-i}) - \log \frac{\rho_\phi(a_t^{-i}|s_t)}{P_{\text{prior}}(a_t^{-i}|s_t)} - \log \frac{\pi_\theta(a_t^i|s_t, a_t^{-i})}{P_{\text{prior}}(a_t^i|s_t, a_t^{-i})}\right] \tag{3.16}$$

We can see that the Bellman equation holds for the following:

$$\begin{aligned} Q(s_t, a_t^i, a_t^{-i}) &= \beta r(s_t, a_t^i, a_t^{-i}) \\ &+ \gamma \mathbb{E}_{s_{t+1}}\left[\mathbb{E}_{\pi_\theta, \rho_\phi}\left[Q(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) - \log \frac{\rho_\phi(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})} - \log \frac{\pi_\theta(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})}\right]\right] \end{aligned} \tag{3.17}$$

Please keep in mind that for the value function to be complete, we have to consider the regularizers for both policy and its opponent model, which is equivalent to the use of soft-max on action-value function, when we use the optional policies. Furthermore, the exciting part is how the agent's policy uses this function. Since we have expanded the equation, we can see that the value function is based on the expectation of "optimal" opponent model[1], which means that the agent is also considering the optimality of opponent model implicitly. By reaching the fixed point[2], we perfectly capture how the agent considers its opponent model to be optimal and how the opponent model considers an optimal agent, which can't be described probabilistically. One might wonder how does the agent $\pi_\theta(a_t^i|s_t, a_t^{-i})$ executes its action ? According to [75], we can simply plug the value of opponent models so that the agent can return the action as we assume the optimality of the opponent model onward i.e

$$\pi_\theta(a_t^i|s_t) = \int \pi_\theta(a_t^i|s_t, a_t^{-i})\rho_\phi(a_t^{-i}|s_t)\ \mathrm{d}a_t^{-i} \tag{3.18}$$

However, please note that the agent is best responding to its opponent model only and not the opponent itself (see the rollout or even ELBO in equation 3.17 and 3.7 relatively). This might be

[1] during training we don't have the exact form due to approximation error

[2] We don't think that it is the same as logistic stochastic best response equilibrium (LSBRE) that is proposed in [81] because we don't refer to any stationary distribution of any Markov chain. But they should be related somehow, and we left it for future work.

trivial, but it can be a very subtle thing to miss. Let's view the value function from other forms, starting with the action-value function

$$Q(s_t, a_t^{-i}) = \log \int \exp\left(Q(s_t, a_t^i, a_t^{-i}\right) P_{\text{prior}}(a_t^i | s_t, a_t^{-i}) \, \mathrm{d}a_t^i \tag{3.19}$$

This represents the action-value function for the opponent model as it only contains the state and opponent model's action. As mentioned before[1], this resembles "soft-max" of agent's action based on agent's prior. In other words, the opponent model's action-value function is the maximization of full action-value function that is also influenced by the opponent's[2] prior on agent policy. The opponent models doesn't simply use an expected value of the action value given agent's prior i.e $\mathbb{E}_{P_{\text{prior}}(a_t^i|s_t,a_t^{-i})}\left[Q(s_t, a_t^i, a_t^{-i})\right]$ nor uses the action that maximizes agent's value i,e $\max_{a_t^i} Q(s_t, a_t^i, a_t^{-i})$, but a mixture of both - prior based best response, since we know that the agent will try to increase its expected reward (a maximize) while trying to acts similarly to its prior (expected value). This is crucial, since the prior is not only a regularizer for the agent, but it also represents the opponent's belief on what the agent should do. In conclusion, MAPI framework doesn't only best response to the opponent model that is estimated via supervised learning but also best responding to optimal opponent the agent aware of.

3.4 Solving Balancing-Q learning

3.4.1 Policies Derivation

Instead of having a ELBO that we want to maximize, we initially have the following loss function from [22]:

$$\mathbb{E}_q \left[\sum_{t=0}^{T} \gamma^t \left(r(s_t, a_t, a_t^{-i}) - \frac{1}{\beta^i} \log \frac{\pi_\theta(a_t|s_t)}{P_{\text{prior}}(a_t^i|s_t)} - \frac{1}{\beta^{-i}} \log \frac{\rho_\phi(a_t^{-i}|s_t)}{P_{\text{prior}}(a_t^{-i}|s_t)} \right) \right] \tag{3.20}$$

However, as noted in the section before, we should consider the conditional opponent model $\rho_\phi(a_t^{-i}|s_t, a_t^i)$ to arrive at the final optimal policy. In [22], the authors don't explicitly show us the method of solving; we, therefore, are going to propose one of the ways, the derivation can be carried out. Note that the loss function above assumes no opponent model. However, to have similar formulation to MAPI approach, we shall consider similar formulation as PR2 [80] case i.e

$$\mathbb{E}_q \left[\sum_{t=0}^{T} \gamma^t \left(r(s_t, a_t, a_t^{-i}) - \frac{1}{\beta^i} \log \frac{\pi_\theta(a_t|s_t)}{P_{\text{prior}}(a_t^i|s_t)} - \frac{1}{\beta^{-i}} \log \frac{\rho_\phi(a_t^{-i}|s_t, a_t^i)}{P_{\text{prior}}(a_t^{-i}|s_t, a_t^i)} \right) \right] \tag{3.21}$$

Now, it is an objective that is optimized within the agent and doesn't require any opponent. We will see that this leads to the same final optimal policy for the agent. Starting with the last time

[1] in section 2.4.1

[2] or opponent model i.e agent's belief that opponent will use $P_{\text{prior}}(a_t^i|s_t, a_t^{-i})$ as its (refer to opponent) prior for the agent

step we have:

$$\mathbb{E}_{P(s_T)P(a_T,a_T^{-i}|s_T)}\left[r(s_t,a_t,a_t^{-i}) - \frac{1}{\beta^i}\log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)} - \frac{1}{\beta^{-i}}\log\frac{\rho_\phi(a_T^{-i}|s_T,a_T^i)}{P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}\right] \tag{3.22}$$

Using the rearrangement and minimizing KL-divergence, we can show that:

$$\begin{aligned}\rho(a_T^{-i}|s_T,a_T^i) &= \frac{\exp\left(\beta^{-i}r(s_t,a_t,a_t^{-i})\right)P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}{\exp\left(Q^{-i}(s_T,a_T^i)\right)}\\ \text{where } Q^{-i}(s_T,a_T^i) &= \log\int\exp\left(\beta^{-i}r(s_t,a_t,a_t^{-i})\right)P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)\ \mathrm{d}a^{-i}\end{aligned} \tag{3.23}$$

see appendix B.2.1 for full derivation. Now consider the message left by plugging the agent's policy back, which leads to the following agent's objective:

$$\mathbb{E}_{P(s_T,a_T,a_T^{-i})}\Bigg[\underbrace{\frac{\beta^i}{\beta^{-i}}Q^{-i}(s_T,a_T^i)}_{Q^i(s_T,a_T^i)} - \log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)}\Bigg] \tag{3.24}$$

Given this, we can consider the agent's optimal policy, which is equal to

$$\begin{aligned}\pi_\theta(a_T^i|s_T) &= \frac{\exp\left(Q^i(s_T,a_T^i)\right)P_{\text{prior}}(a_T^i|s_T)}{\exp\left(V^i(s_T)\right)}\\ \text{where } Q^i(s_T,a_T^i) &= \frac{\beta^i}{\beta^{-i}}\log\int\exp\left(\beta^{-i}R(s_T,a_T^i,a_T^{-i})\right)P_{\text{prior}}(a_T^{-i}|s_T)\ \mathrm{d}a^{-i}\\ \text{and } V^i(s_T) &= \log\int\exp\left(Q^i(s_T,a_T^i)\right)P_{\text{prior}}(a_T^i|s_T)\ \mathrm{d}a^i\end{aligned} \tag{3.25}$$

See appendix B.2.2 for full derivation of both agent's objective optimal agent's policy, while we are left with the following message:

$$\mathbb{E}_{P(s_T)}\left[\frac{1}{\beta^i}V^i(s_T)\right] = \mathbb{E}_{P(s_T)}\left[V(s_T)\right] \tag{3.26}$$

Let's consider the arbitrary time step t, which lead us to the following objective:

$$\mathbb{E}_{P(s_t,a_t,a_t^{-i})}\Bigg[\underbrace{r(s_t,a_t,a_t^{-i}) + \gamma\mathbb{E}_{P(s_{t+1}|s_t,a_t,a_t^{-i})}\left[V(s)\right]}_{Q(s_t,a_t^i,a_t^{-i})} - \frac{1}{\beta^i}\log\frac{\pi_\theta(a_t|s_t)}{P_{\text{prior}}(a_t^i|s_t)} - \frac{1}{\beta^{-i}}\log\frac{\rho_\phi(a_t^{-i}|s_t,a_t^i)}{P_{\text{prior}}(a_t^{-i}|s_t,a_t^i)}\Bigg] \tag{3.27}$$

By solving using the same method, we have the following optimal agent's policy

$$\begin{aligned}\pi_\theta(a_t^i|s_t) &= \frac{\exp\left(Q^i(s_t,a_t^i)\right)P_{\text{prior}}(a_t^i|s_t)}{\exp\left(V^i(s_t)\right)}\\ \text{where } Q^i(s_t,a_t^i) &= \frac{\beta^i}{\beta^{-i}}\log\int\exp\left(\beta^{-i}Q(s_t,a_t^i,a_t^{-i})\right)P_{\text{prior}}(a_t^{-i}|s_t)\ \mathrm{d}a^{-i}\\ V^i(s_t) &= \log\int\exp\left(Q^i(s_t,a_t^i)\right)P_{\text{prior}}(a_t^i|s_t)\ \mathrm{d}a^i\end{aligned} \tag{3.28}$$

The definition of $Q(s_t, a_t^i, a_t^{-i})$ will be defined at the end, after considering the definition of optimal opponent agent, which is a solution to the following objective:

$$\mathbb{E}_q \left[\sum_{t=0}^{T} \gamma^t \left(r(s_t, a_t, a_t^{-i}) - \frac{1}{\beta^i} \log \frac{\pi_\theta(a_t|s_t, a_t^{-i})}{P_{\text{prior}}(a_t^i|s_t, a_t^{-i})} - \frac{1}{\beta^{-i}} \log \frac{\rho_\phi(a_t^{-i}|s_t)}{P_{\text{prior}}(a_t^{-i}|s_t)} \right) \right] \tag{3.29}$$

which lead to the following results, using the same techniques as the derivation of policy:

$$\begin{aligned} \rho_\phi(a_t^{-i}|s_t) &= \frac{\exp\left(Q^{-i}(s_t, a_t^{-i})\right) P_{\text{prior}}(a_t^{-i}|s_t)}{\exp\left(V^{-i}(s_t)\right)} \\ \text{where } Q^{-i}(s_t, a_t^{-i}) &= \frac{\beta^{-i}}{\beta^i} \log \int \exp\left(\beta^i Q(s_t, a_t^i, a_t^{-i})\right) P_{\text{prior}}(a_t^i|s_t, a_t^{-i}) \, \mathrm{d}a^i \\ V^{-i}(s_t) &= \log \int \exp\left(Q^{-i}(s_t, a_t^{-i})\right) P_{\text{prior}}(a_t^{-i}|s_t) \, \mathrm{d}a^{-i} \end{aligned} \tag{3.30}$$

The authors [22] proved that the value function of the agent and the opponent are related as follows:

$$\gamma \mathbb{E}_{P(s_t)} \left[V(s) \right] = \gamma \mathbb{E}_{P(s_t)} \left[\frac{1}{\beta^i} V^i(s_t) \right] = \gamma \mathbb{E}_{P(s_t)} \left[\frac{1}{\beta^{-i}} V^{-i}(s_t) \right] \tag{3.31}$$

We can, therefore, learn the same Q-value function:

$$Q(s_t, a_t^i, a_t^{-i}) = r(s_t, a_t, a_t^{-i}) + \mathbb{E}_{P(s_{t+1}|s_t, a_t, a_t^{-i})} \left[V(s_{t+1}) \right] \tag{3.32}$$

Intuitively, we can see that β^i and β^{-i} are cancelling each other. This result is fascinating as we can control how the opponent model learns while able to control "correct" type of the game. The result is, indeed, a special case as we can't usually control the opponent's reward to be as effective as this.

3.4.2 Theoretical Properties

Now, we shall consider the theoretical property of the algorithm. We start with the result from [22], where the authors show that Balancing Q-learning Bellman equation:

$$\mathcal{B}_{\text{Balance}} Q(s_t, a_t^i, a_t^{-i}) = r(s_t, a_t, a_t^{-i}) + \mathbb{E}_{P(s_{t+1}|s_t, a_t, a_t^{-i})} \left[V(s_{t+1}) \right] \tag{3.33}$$

is a contraction mapping. We would like to further investigate more on how the agent's update affects the action-value function. Suppose a fixed policy π, how would an updated opponent affect the action-value function given the value β^{-i}

Theorem 8 *The action value function $Q^{-i,\pi,\rho}(s_t, a_t^i, a_t^{-i})$ defined by Bellman equation as:*

$$\begin{aligned} Q^{-i,\pi,\rho}(s_t, a_t^i, a_t^{-i}) &= r(s_t, a_t, a_t^{-i}) \\ &+ \gamma \mathbb{E}_{s_{t+1}} \left[\mathbb{E}_{\pi,\rho} \left[Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) - \frac{1}{\beta^i} \log \frac{\pi(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})}{P_{prior}(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})} - \frac{1}{\beta^{-i}} \log \frac{\rho(a_{t+1}^{-i}|s_{t+1})}{P_{prior}(a_{t+1}^{-i}|s_{t+1})} \right] \right] \end{aligned} \tag{3.34}$$

Given the opponent's updated policy $\tilde{\rho}$ given normal policy ρ as

$$\begin{aligned}\tilde{\rho}(a_t^{-i}|s_t) &= \arg\min_{\rho' \in \Pi} D_{\mathrm{KL}}\left(\rho'(a_t^{-i}|s_t) \middle\| \frac{\exp\left(Q^{-i,\pi,\rho}(s_t, a_t^{-i})\right) P_{prior}(a_t^{-i}|s_t)}{\exp Z^{-i,\pi,\rho}(s_t)}\right) \\ &= \arg\min_{\rho' \in \Pi} J_\rho(\rho')\end{aligned} \tag{3.35}$$

The action-value function is affected in a difference way according to the sign of β^{-i}

$$\begin{cases} Q^{-i,\pi,\rho}(s_t, a_t^i, a_t^{-i}) \geq Q^{-i,\pi,\tilde{\rho}}(s_t, a_t^i, a_t^{-i}) & \text{if } \beta^{-i} < 0 \\ Q^{-i,\pi,\rho}(s_t, a_t^i, a_t^{-i}) \leq Q^{-i,\pi,\tilde{\rho}}(s_t, a_t^i, a_t^{-i}) & \text{if } \beta^{-i} > 0 \end{cases} \tag{3.36}$$

PROOF:

The proof follows closely from ROMMEO proof on policy improvement [75] which is based on soft Actor-Critic [24] proof for more details on the method see appendix A.3.4. We start by comparing these two quantities:

$$\mathbb{E}_{\pi,\rho}\left[Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) - \frac{1}{\beta^i}\log\frac{\pi(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})} - \frac{1}{\beta^{-i}}\log\frac{\rho(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})}\right]$$

$$\mathbb{E}_{\pi,\tilde{\rho}}\left[Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) - \frac{1}{\beta^i}\log\frac{\pi(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})} - \frac{1}{\beta^{-i}}\log\frac{\tilde{\rho}(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})}\right]$$

The differences are shown in red letter. Please note that the action-value function definition of the second equation still based on the old opponent ρ. We start by finding the KL-divergence between arbitrary ρ^+ the objective for updating is the following, which can be expanded as

$$\begin{aligned} & D_{\mathrm{KL}}\left(\rho^+(a_t^{-i}|s_t) \middle\| \frac{\exp\left(Q^{-i,\pi,\rho}(s_t, a_t^{-i})\right) P_{\text{prior}}(a_t^{-i}|s_t)}{\exp Z^{-i,\pi,\rho}(s_t)}\right) \\ &= \int \rho^+(a_{t+1}^{-i}|s_{t+1}) \log \frac{\rho^+(a_{t+1}^{-i}|s_{t+1}) \exp(Z^{-i,\pi,\rho}(s_{t+1}))}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1}) \exp\left(Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^{-i})\right)} \, \mathrm{d}a_{t+1}^{-i} \\ &= \int \rho^+(a_{t+1}^{-i}|s_{t+1}) \log \frac{\rho^+(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})} \, \mathrm{d}a_{t+1}^{-i} + Z^{-i,\pi,\rho}(s_{t+1}) \\ &\quad - \int \rho^+(a_{t+1}^{-i}|s_{t+1}) Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^{-i}) \, \mathrm{d}a_{t+1}^i \end{aligned}$$

Now the performance gap can be established due to the KL-divergence between ρ and

$\tilde{\rho}$. Please note that by definition $J_\rho(\rho) \geq J_\rho(\tilde{\rho})$ which leads to the following inequality

$$\begin{aligned}
&\int \rho(a_{t+1}^{-i}|s_{t+1}) \log \frac{\rho(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})} \, \mathrm{d}a_{t+1}^{-i} + \cancel{Z^{-i,\pi,\rho}(s_{t+1})} \\
&\quad - \int \rho(a_{t+1}^{-i}|s_{t+1}) Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^{-i}) \, \mathrm{d}a_{t+1}^{i} \\
\geq &\int \tilde{\rho}(a_{t+1}^{-i}|s_{t+1}) \log \frac{\tilde{\rho}(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})} \, \mathrm{d}a_{t+1}^{-i} + \cancel{Z^{-i,\pi,\rho}(s_{t+1})} \\
&\quad - \int \tilde{\rho}(a_{t+1}^{-i}|s_{t+1}) Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^{-i}) \, \mathrm{d}a_{t+1}^{i}
\end{aligned}$$

Let's investigate more into the definition of the last term, for $\tilde{\rho}$. We will have to consider the opponent model of the agent

$$\int \tilde{\rho}(a_{t+1}^{-i}|s_{t+1}) \frac{\beta^{-i}}{\beta^i} \left(\beta^i Q(s_t, a_t^i, a_t^{-i}) - \log \frac{\pi(a_t^i|a_t^{-i}, s_t)}{P_{\text{prior}}(a_t^i|s_t)} \right) \, \mathrm{d}a_{t+1}^{-i} \tag{3.37}$$

which has the following definition of opponent's agent model:

$$\pi(a_t^i|a_t^{-i}, s_t) = \frac{\exp(\beta^i Q^{-i,\pi,\rho}(s_t, a_t^i, a_t^{-i})) P_{\text{prior}}(a_t^i|s_t)}{\exp\left(Q^{i,\pi,\rho}(s_t, a_t^{-i})\right)}$$

We expand the definition to be:

$$\begin{aligned}
&\int \tilde{\rho}(a_{t+1}^{-i}|s_{t+1}) Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^{-i}) \, \mathrm{d}a_{t+1}^{-i} \\
=&\int \rho(a_{t+1}^{-i}|s_{t+1}) \int \pi(a_{t+1}^i|a_{t+1}^{-i}, s_{t+1}) \left[\beta^{-i} Q(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) - \frac{\beta^{-i}}{\beta^i} \log \frac{\pi(a_{t+1}^i|a_{t+1}^{-i}, s_{t+1})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1})} \right] \, \mathrm{d}a_{t+1}^i \, \mathrm{d}a_{t+1}^{-i} \\
=&\mathbb{E}_{a_{t+1}^i, a_{t+1}^{-i} \sim \pi, \tilde{\rho}} \left[\beta^{-i} Q(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) - \frac{\beta^{-i}}{\beta^i} \mathbb{E}_{a_{t+1}^i} \log \frac{\pi(a_{t+1}^i|a_{t+1}^{-i}, s_{t+1})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1})} \right]
\end{aligned}$$

Plugging this back into the equation, which some arrangement, we have

$$\begin{aligned}
&\mathbb{E}_{a_{t+1}^i, a_{t+1}^{-i} \sim \pi, \rho} \left[\beta^{-i} Q(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) - \log \frac{\rho(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})} - \frac{\beta^{-i}}{\beta^i} \mathbb{E}_{a_{t+1}^i} \log \frac{\pi(a_{t+1}^i|a_{t+1}^{-i}, s_{t+1})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1})} \right] \\
\leq &\mathbb{E}_{a_{t+1}^i, a_{t+1}^{-i} \sim \pi, \tilde{\rho}} \left[\beta^{-i} Q(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) - \log \frac{\tilde{\rho}(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})} - \frac{\beta^{-i}}{\beta^i} \mathbb{E}_{a_{t+1}^i} \log \frac{\pi(a_{t+1}^i|a_{t+1}^{-i}, s_{t+1})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1})} \right]
\end{aligned}$$

In the case that $\beta^{-i} < 0$, we have the following inequality:

$$\begin{aligned}
&\mathbb{E}_{a_{t+1}^i, a_{t+1}^{-i} \sim \rho} \Big[- \frac{1}{\beta^i} \log \frac{\pi(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})} - \frac{1}{\beta^{-i}} \log \frac{\rho(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})} + Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) \Big] \\
\geq &\mathbb{E}_{a_{t+1}^i, a_{t+1}^{-i} \sim \tilde{\rho}} \Big[- \frac{1}{\beta^i} \log \frac{\pi(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})}{P_{\text{prior}}(a_{t+1}^i|s_{t+1}, a_{t+1}^{-i})} - \frac{1}{\beta^{-i}} \log \frac{\tilde{\rho}(a_{t+1}^{-i}|s_{t+1})}{P_{\text{prior}}(a_{t+1}^{-i}|s_{t+1})} + Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) \Big]
\end{aligned}$$

Now, let's consider the recursive definition of action value function $Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i})$,

which leads to the fact that

$$Q^{-i,\pi,\rho}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1}) \geq Q^{-i,\pi,\tilde{\rho}}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1}) \tag{3.38}$$

The recursive step will be shown in appendix B.2.3 due to space constrain. Similarly, $\beta^{-i} > 0$, we have the inequality:

$$\begin{aligned}
&\mathbb{E}_{a^i_{t+1}, a^{-i}_{t+1} \sim \rho}\Big[-\frac{1}{\beta^i} \log \frac{\pi(a^i_{t+1}|s_{t+1}, a^{-i}_{t+1})}{P_{\text{prior}}(a^i_{t+1}|s_{t+1}, a^{-i}_{t+1})} - \frac{1}{\beta^{-i}} \log \frac{\rho(a^{-i}_{t+1}|s_{t+1})}{P_{\text{prior}}(a^{-i}_{t+1}|s_{t+1})} + Q^{-i,\pi,\rho}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1}) \Big] \\
\leq &\mathbb{E}_{a^i_{t+1}, a^{-i}_{t+1} \sim \tilde{\rho}}\Big[-\frac{1}{\beta^i} \log \frac{\pi(a^i_{t+1}|s_{t+1}, a^{-i}_{t+1})}{P_{\text{prior}}(a^i_{t+1}|s_{t+1}, a^{-i}_{t+1})} - \frac{1}{\beta^{-i}} \log \frac{\tilde{\rho}(a^{-i}_{t+1}|s_{t+1})}{P_{\text{prior}}(a^{-i}_{t+1}|s_{t+1})} + Q^{-i,\pi,\rho}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1}) \Big]
\end{aligned}$$

this leads to $Q^{-i,\pi,\rho}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1}) \leq Q^{-i,\pi,\tilde{\rho}}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1})$. We have proven the way β^{-i} affects the update of opponent policy. □

Similarly, we can show for the policy that

Theorem 9 *The action value function $Q^{i,\pi,\rho}(s_t, a^i_t, a^{-i}_t)$ defined by Bellman equation as:*

$$\begin{aligned}
&Q^{i,\pi,\rho}(s_t, a^i_t, a^{-i}_t) = r(s_t, a_t, a^{-i}_t) \\
&\quad + \gamma \mathbb{E}_{s_{t+1}} \left[\mathbb{E}_{\pi,\rho} \left[Q^{i,\pi,\rho}(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1}) - \frac{1}{\beta^i} \log \frac{\pi(a^i_{t+1}|s_{t+1})}{P_{prior}(a^i_{t+1}|s_{t+1})} - \frac{1}{\beta^{-i}} \log \frac{\rho(a^{-i}_{t+1}|s_{t+1}, a^i_{t+1})}{P_{prior}(a^{-i}_{t+1}|s_{t+1}, a^i_{t+1})} \right] \right]
\end{aligned} \tag{3.39}$$

Given the agent's updated policy $\tilde{\rho}$ given normal policy π as

$$\begin{aligned}
\tilde{\rho}(a^i_t|s_t) &= \arg\min_{\rho' \in \Pi} D_{\text{KL}} \left(\rho'(a^{-i}_t|s_t) \middle\| \frac{\exp\left(Q^{i,\pi,\rho}(s_t, a^i_t)\right) P_{prior}(a^i_t|s_t)}{\exp Z^{i,\pi,\rho}(s_t)} \right) \\
&= \arg\min_{\rho' \in \Pi} J_\rho(\rho')
\end{aligned} \tag{3.40}$$

The action-value function for $\beta^i > 0$ is

$$Q^{i,\tilde{\pi},\rho}(s_t, a^i_t, a^{-i}_t) \leq Q^{i,\pi,\rho}(s_t, a^i_t, a^{-i}_t) \tag{3.41}$$

PROOF:

The prove follows the same step as theorem 8. □

Now, we will present a policy improvement results for Balancing Q-learning as

Corollary 10 *The update of $Q(s_t, a^i_t, a^{-i}_t)$ is denoted as $\bar{Q}(s_t, a^i_t, a^{-i}_t)$. Suppose the update is based on improved policy when $\beta^i > 0$,*

$$Q(s_t, a^i_t, a^{-i}_t) \geq \bar{Q}(s_t, a^i_t, a^{-i}_t) \tag{3.42}$$

Similarly, if we fix the policy and update opponent then we have:

$$\begin{cases}
Q(s_t, a^i_t, a^{-i}_t) \geq \bar{Q}(s_t, a^i_t, a^{-i}_t) & \text{if } \beta^{-i} < 0 \\
Q(s_t, a^i_t, a^{-i}_t) \leq \bar{Q}(s_t, a^i_t, a^{-i}_t) & \text{if } \beta^{-i} > 0
\end{cases} \tag{3.43}$$

PROOF:

We use the results from theorem 8 and 9, with the result from [22]'s Corollary 1 (shown in equation 3.31) that shows that both action value function are the same. □

Finally if we would like just to consider the PR2 like opponent model i.e $\rho(a_t^{-i}|s_t, a_t^i)$ given the value $\beta^{-i} > 0$ or $\beta^{-i} < 0$, we can follows the same procedure with similar step, which yields the same outcome as theorem 8. Note that this can be used to prove the improvement of PR2 and ROMMEO too, as they are a subset of Balancing Q-learning. Now, we have shown that Balancing Q-learning does control the opponent's policy via β^{-i}. Now, we shall consider how can we interpret Balancing Q-learning via graphical model and variational inference.

3.5 Probabilistic Balancing-Q

3.5.1 Graphical Model Representations

Before we start, we can simply consider the variational distribution to be:

$$\frac{\pi_\theta(a_t^i|s_t,)^{\frac{1}{\beta^i}}}{\int \pi_\theta(a_t^i|s_t)^{\frac{1}{\beta^i}}\ \mathrm{d}a_t^i} \qquad \frac{\rho_\phi(a_t^{-i}|s_t)^{\frac{1}{\beta^{-i}}}}{\int \rho_\phi(a_t^{-i}|s_t)^{\frac{1}{\beta^{-i}}}\ \mathrm{d}a_t^i} \tag{3.44}$$

This might satisfy the requirement. However, it doesn't give any other information or insight into Balancing Q-learning ability to works with both cooperative and competitive game. Now, let's show the graphical model of the opponent model. We will follow the insight from solving agent's policy in ROMMEO and the need for the intermediate opponent model. Let's start with the graphical model representation of the opponent model problem. We will consider the case, which the opponent assumes an optimal agent's policy and updates itself towards it. The graphical model is depicted in figure 3.5. The graphical model gives the following joint probability. The difference between PR2 and Balancing Q-learning is that we assume to have the knowledge of optimal agent:

$$\begin{aligned} &P(s_{1:T}, a_{1:T}^i, a_{1:T}^{-i}, \mathcal{O}_{1:T}^i = 1, \mathcal{O}_{1:T}^{-i} = 1) \\ &\quad = P(s_0) \prod_{t=0}^{T} \pi_\theta(a_t^i|s_t, \mathcal{O}_t^i = 1) P_{\text{prior}}(a_t^{-i}|s_t, a_t^i) P(s_{t+1}|s_t, a_t^i, a_t^{-i}) P(\mathcal{O}_t^{-i} = 1|s_t, a_t^i, a_t^{-i}) P(\mathcal{O}_t^{-i} = 1) \end{aligned} \tag{3.45}$$

The variational joint probability is depicted as the following, the graphical model is very similar to figure 3.4 with difference condition on actions:

$$q(s_{1:T}, a_{1:T}^i, a_{1:T}^{-i}) = P(s_0) \prod_{t=0}^{T} \pi_\theta(a_t^i|s_t, \mathcal{O}_t^i = 1) \rho_\phi(a_t^{-i}|s_t, a_t^i) P(s_{t+1}|s_t, a_t^i, a_t^{-i}) P(\mathcal{O}_t^{-i} = 1) \tag{3.46}$$

Given this, the optimality random variable is $P(\mathcal{O}_t^{-i} = 1|s_t, a_t^i, a_t^{-i}, \mathcal{O}_t^i = 1)$ is defined as follows:

$$P(\mathcal{O}_t^{-i} = 1|s_t, a_t^i, a_t^{-i}, \mathcal{O}_t^i = 1) \propto \exp\left(\beta^{-i} r(s_t, a_t^i, a_t^{-i})\right) \tag{3.47}$$

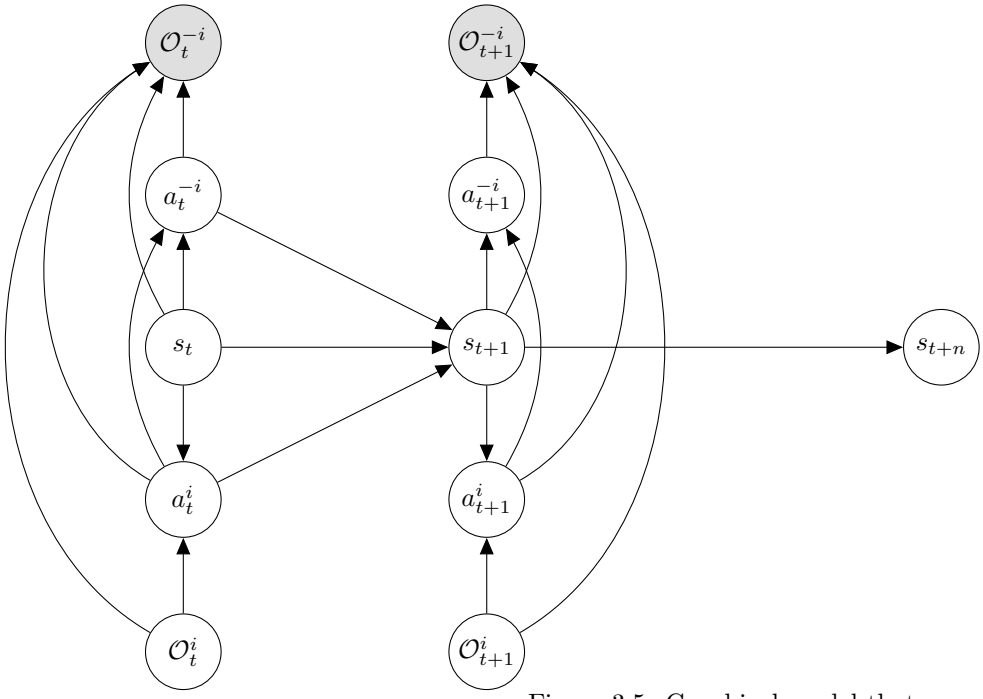


Figure 3.5: Graphical model that we want to approximate. For Balancing Q-learning, this bear a resemblance of PR2 graphical model.

Now, we can see that the ELBO is simply:

$$\mathbb{E}\left[\sum_{t=0}^{T}\beta^{-i}r(s_t,a_t^i,a_t^{-i})-\log\frac{\rho_\phi(a_t^{-i}|s_t,a_t^i)}{P_{\text{prior}}(a_t^{-i}|s_t,a_t^i)}\right] \tag{3.48}$$

We can solve only for the opponent model since we assume we know agent's optimal policy. With the same working as ROMMEO and Balancing Q-learning, we arrived at the following policy definition:

$$\begin{aligned}\rho_\phi(a_t^{-i}|s_t,a_t^i)&=\frac{\exp(Q^{-i}(s_t,a_t^i,a_t^{-i}))P_{\text{prior}}(a_t^{-i}|s_t,a_t^i)}{\exp(Q(s_t,a_t^i))}\\ \text{where } Q(s_t,a_t^i)&=\log\int\exp(Q^{-i}(s_t,a_t^i,a_t^{-i}))P_{\text{prior}}(a_t^{-i}|s_t,a_t^i)\ \mathrm{d}a_t^{-i}\end{aligned} \tag{3.49}$$

Now, please note that the definition of $Q(s_t,a_t^i,a_t^{-i})$ isn't entirely correct because we are missing the regularization of the policy. We will leave the action-value function as it is for now, once we have derived the agent's policy, let's re-calculate the action-value function for a more accurate value. Now, for policy, we have the following graphical model representation represented in figure 3.6. Given this, we define the optimality of an agent to be:

$$P(\mathcal{O}_t^i=1|s_t,a_t^{-i},\mathcal{O}_t^{-i}=1)=\exp\left(\frac{\beta^i}{\beta^{-i}}Q(s_t,a_t^i))\right) \tag{3.50}$$

Note that now, we assume the optimality of the opponent model via the use of soft-max of action-value function this works when we are aware that the opponent will "maximize" its action-value

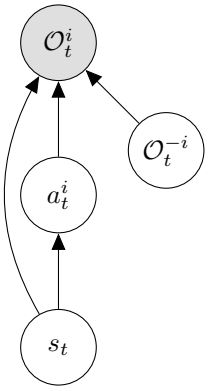


Figure 3.6: Graphical model that we want to approximate. This follows the VIREL [14] type of graphical model rather than traditional probabilistic as inference framework.

function. The fraction represents a conversion to the agent's reward term. The variational probability is simply

$$q(s_t, a_t^i) = P(s_t)\pi_\theta(a_t^i|s_t) \tag{3.51}$$

Therefore, the optimal opponent model is equal to

$$\begin{aligned} \pi_\theta(a_t^i|s_t) &= \frac{\exp\left(\frac{\beta^i}{\beta^{-i}} Q(s_t, a_t^i)\right) P_{\text{prior}}(a_t^i|s_t)}{\exp\left(V^i(s_t)\right)} \\ \text{where } V^i(s_t) &= \log \int \exp\left(\frac{\beta^i}{\beta^{-i}} Q(s_t, a_t^i)\right) P_{\text{prior}}(a_t^i|s_t)\ \mathrm{d}a_t^i \end{aligned} \tag{3.52}$$

Now, let's consider the relationship between the $V^i(s_t)$, $Q(s_t, a_t^i)$ and $Q(s_t, a_t^i, a_t^{-i})$:

$$V^i(s_t) = \mathbb{E}_{\pi,\rho}\left[\frac{\beta^i}{\beta^{-i}} Q(s_t, a_t^i, a_t^{-i}) - \frac{\beta^i}{\beta^{-i}} \log \frac{\rho_\phi(a_t^{-i}|s_t, a_t^i)}{P_{\text{prior}}(a_t^{-i}|s_t, a_t^i)} - \log \frac{\pi_\theta(a_t^i|s_t)}{P_{\text{prior}}(a_t^i|s_t)}\right] \tag{3.53}$$

Given this, we can consider the corrected Q value as defined in the following Bellman equation:

$$Q^{-i}(s_t, a_t^i, a_t^{-i}) = \beta^{-i} R(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E}_{s_{t+1}}\left[\frac{\beta^{-i}}{\beta^i} V^i(s_{t+1})\right] \tag{3.54}$$

We can use this action-value function for the calculation of the opponent model. Now, we can see that by the use of the ratio between β^{-i} and β^i, we can switch between the agent's action-value function and the opponent's action-value function. Furthermore, from the use of extremal operator in Bellman equation [22] of Balancing Q-learning denotes similar notion to friend-or-foe learning [44], which we can say that the Balancing Q-learning is the "soft" version of friend-or-foe learning. The distinction between friend or foe is needed for the optimization of the agent. Now, we haven't lost the property of Balancing Q-learning thus maintain the same form of value function and action-value function, where $Q^{-i}(s_t, a_t^i, a_t^{-i}) = \beta^{-i} Q(s_t, a_t^i, a_t^{-i})$ in equation 3.32. This only works because of the guarantee that the value functions are the same as in corollary 1 of [22].

3.5.2 Implementation

Now, we shall consider the implementation of the probabilistic Balancing Q-learning. Note that the current version of Balancing Q-learning based on discrete action, which we can find the exact expected value. However, Balancing Q-learning can be implemented in soft Actor-Critic like algorithm. The extension is trivial, while we are going to present a similar algorithm, which is derived from the section before. This would require a total of 4 neural networks: $\pi_\theta(a_t^i|s_t, a_t^{-i})$, $\rho_\phi(a_t^{-i}|s_t)$, $Q_\chi(s_t, a_t^i, a_t^{-i})$ and $V_\psi(s_t)$. Let's consider the value function first. We will follow the definition of the value function that we have defined in equation 3.53, leading the following loss function:

$$\mathcal{L}_V(\phi) = \mathbb{E}_{s_t\sim\mathcal{D}}\left[\frac{1}{2}\left(V^i(s_t) - \mathbb{E}_{\pi_\theta,\rho_\phi}\left[\frac{\beta^i}{\beta^{-i}}Q_\chi(s_t, a_t^i, a_t^{-i}) - \frac{\beta^i}{\beta^{-i}}\log\frac{\rho_\phi(a_t^{-i}|s_t, a_t^i)}{P_{\text{prior}}(a_t^{-i}|s_t, a_t^i)} - \log\frac{\pi_\theta(a_t^i|s_t)}{P_{\text{prior}}(a_t^i|s_t)}\right]\right)^2\right] \tag{3.55}$$

Now, we can use the Bellman equation of the action value function as the target for training:

$$\mathcal{L}_{Q_1}(\chi) = \mathbb{E}_{s_t,a_t^i,a_t^{-i},r_t,s_{t+1}\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\chi(s_t, a_t^i, a_t^{-i}) - \beta^{-i}r_t - \frac{\gamma\beta^{-i}}{\beta^i}V_\psi(s_{t+1})\right)^2\right] \tag{3.56}$$

Given the action value function, we can try to train the opponent model, which is in the form of $\rho_\phi(a_t^{-i}|s_t, a_t^i) = f_\phi^{-i}(\xi; s_t, a_t^i)$ where the noise $\xi \sim \mathcal{N}(0, I)$. Given the function, we would like to minimizing the KL-divergence between the network's output and the closed form solution in equation 3.49 i.e minimizing the following objective

$$\mathcal{L}_\rho(\phi) = D_{\text{KL}}\left(\rho_\phi(a_t^{-i}|s_t, a_t^i)\middle\|\frac{\exp(Q(s_t, a_t^i, a_t^{-i}))P_{\text{prior}}(a_t^{-i}|s_t, a_t^i)}{\exp(Q(s_t, a_t^i))}\right) \tag{3.57}$$

Now before we consider agent's objective, we will have to estimate the value $Q(s_t, a_t^i)$

$$\mathcal{L}_{Q_2} = \mathbb{E}_{(s_t,a_t^i)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t, a_t^i) - \mathbb{E}_{a_t^i\sim\pi_\theta}\left[\frac{\beta^{-i}}{\beta^i}V^i(s_t) + \frac{\beta^{-i}}{\beta^i}\log\frac{\pi_\theta(a_t^i|s_t)}{P_{\text{prior}}(a_t^i|s_t)}\right]\right)^2\right] \tag{3.58}$$

Alternatively, we can estimate the value $Q(s_t, a_t^i)$, by using Monte-Carlo estimate with importance sampling

$$Q^i(s_t, a_t^i) = \frac{\beta^i}{\beta^{-i}}\log\mathbb{E}_{a_t^{-i}\sim q}\left[\frac{\exp\left(Q(s_t, a_t^i, a_t^{-i})\right)P_{\text{prior}}(a_t^{-i}|s_t)}{q(a_t^{-i})}\right] \tag{3.59}$$

Given our estimation, we can now optimize the agent's policy $f_\theta^i(\xi; s_t)$ by KL-divergence minimization based on equation 3.52 as follows:

$$\mathcal{L}_\pi(\theta) = D_{\text{KL}}\left(\pi_\theta(a_t^i|s_t)\middle\|\frac{\exp\left(Q^i(s_t, a_t^i)\right)P_{\text{prior}}(a_t^i|s_t)}{\exp\left(V_\psi(s_t)\right)}\right) \tag{3.60}$$

This concludes the implementation of probabilistic Balancing Q-learning.

3.6 Conclusion

We showed that ROMMEO and PR2 don't work in an adversarial setting due to its inability to represent its opponent correctly. To fix this, we have reinterpreted Balancing Q-learning as probabilistic inference with some theoretical guarantee. Finally, we proposed an algorithm that is similar to Balancing Q-learning, which can be implemented using soft Actor-Critic like algorithm, while based on control as probabilistic inference framework.

Chapter 4

Hierarchy and Communication in Multi-Agent Reinforcement Learning

After we have a unified view on MAPI framework, which we have reinterpreted Balancing Q-learning [22] as probabilistic inference. We are now going to consider extensions to the agent's functionality. In this chapter, we are going to consider the case of temporal abstraction and communication in a cooperative multi-agent reinforcement learning problem. We will start by finding a single-agent option learning problem and solving it together with soft actor-critic like algorithm, as a minor contribution. After considering the single-agent problem, we are going to define a model in which the agent can plan in a more extended period of time and communicate with each others.

4.1 Single Agent Soft-Hierarchical Reinforcement Learning

We will associate our model with option learning paradigm, and extends the result in [32] by cooperating Soft-Actor Critic type solution as it is a standard method for solving probabilistic reinforcement learning. Furthermore, our approach is similar to [7], in terms of using a similar probabilistic technique, however, in the paper, the authors concerning more on inferring policies from expert data, while didn't explicitly solve for each policy given the optimally condition (what we are trying to do now). Folloing [32], we have a master policy π^H that suggesting the "mode of execution" or an option to lower-level policy π based on termination policy π^T, which tells what the next option should be. The joint distribution of actions i.e $P(s_{1:T}, a_{1:T}, \mathcal{O}_{1:T}, z_{1:T}, b_{1:T})$ is depicted in figure 4.1. The joint distribution is then

$$P(s_{1:T}, a_{1:T}, z_{1:T}, b_{1:T}) = P(s_0)P(z_0)\prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)P_{\text{prior}}(a_t|s_t, z_t) \\ P^H_{\text{prior}}(z_t|s_t, z_{t-1}, b_t)P^T_{\text{prior}}(b_t|s_t, z_{t-1})P(\mathcal{O}_t = 1|s_t, a_t) \tag{4.1}$$

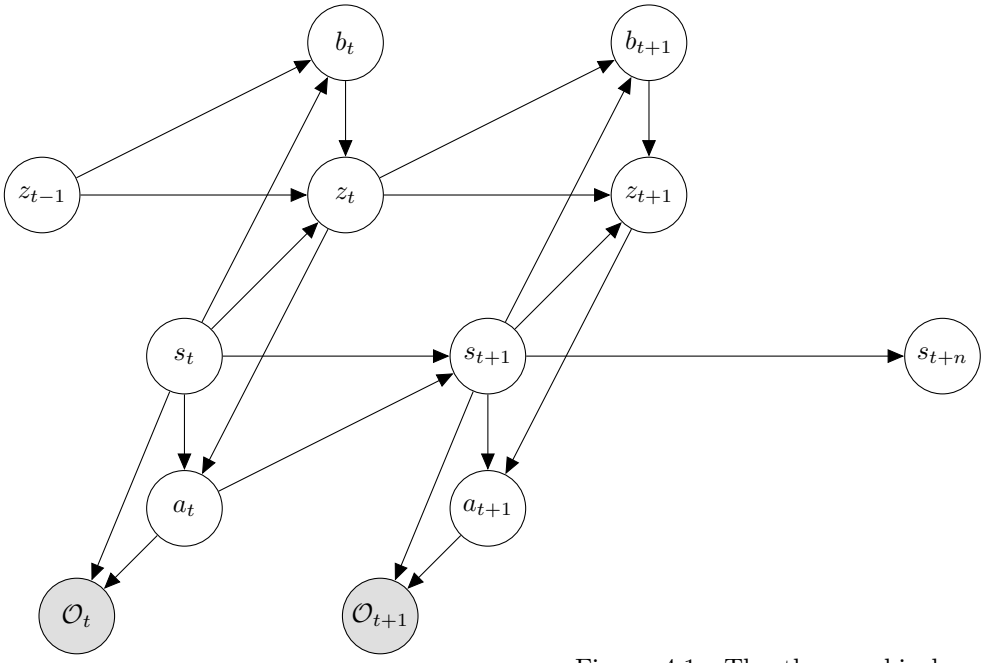


Figure 4.1: The the graphical model of hierarchical agent. We have the master policy which produces z_t while having switch goal policy producing policy b_t

The variations joint distribution is defined as

$$q(s_{1:T}, a_{1:T}, z_{1:T}, b_{1:T}) = P(s_0)P(z_0)\prod_{t=0}^{T} P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t, z_t) \\ \pi_\phi^H(z_t|s_t, z_{t-1}, b_t)\pi_\phi^T(b_t|s_t, z_{t-1}) \tag{4.2}$$

Usually, the master policy and its prior are defined as.

$$\begin{aligned} \pi_\phi^H(z_t|s_t, z_{t-1}, b_t) &= (1 - b_t)\delta(z_t - z_{t-1}) + b_t q_\phi^H(z_t|s_t) \\ P_{\text{prior}}^H(z_t|s_t, z_{t-1}, b_t) &= (1 - b_t)\delta(z_t - z_{t-1}) + b_t \frac{1}{m} \end{aligned} \tag{4.3}$$

As it takes the value of b_t to suggest when to change the option z_t. Now, the ELBO is equal to

$$\mathbb{E}\left[\sum_{t=1}^{T} \beta r(s_t, a_t) - \log \frac{\pi_\theta(a_t|s_t, z_t)}{P_{\text{prior}}(a_t|s_t, z_t)} - \log \frac{\pi_\phi^H(z_t|s_t, z_{t-1}, b_t)}{P_{\text{prior}}^H(z_t|s_t, z_{t-1}, b_t)} - \log \frac{\pi_\phi^T(b_t|s_t, z_{t-1})}{P_{\text{prior}}^T(b_t|s_t, z_{t-1})}\right] \tag{4.4}$$

This objective is the same as the objective defined in [32]. Given the ELBO, we will consider the closed-form solution of the low-level policy/master policy and termination policy. For the low-level

policy, we have

$$\begin{aligned} \pi_\theta(a_t|s_t,z_t) &= \frac{P_{\text{prior}}(a_t|s_t,z_t)\exp\left(Q(s_t,a_t,z_t)\right)}{\exp V(s_t,z_t)} \\ \text{where } V(s_t,z_t) &= \log\int P_{\text{prior}}(a_t|s_t,z_t)\exp\left(Q(s_t,a_t,z_t)\right)\ \mathrm{d}a_t \end{aligned} \tag{4.5}$$

For the high-level policy, we have

$$\begin{aligned} \pi^H(z_t|s_t) &= \frac{\frac{1}{m}\exp\left(Q(s_t,z_t)\right)}{\exp\left(V(s_t)\right)} \\ \text{where } Q(s_t,z_t) &= V(s_t,z_t) \\ V^H(s_t) &= \frac{1}{m}\log\int\exp\left(Q(s_t,z_t)\right)\ \mathrm{d}z_T \end{aligned} \tag{4.6}$$

Finally, the termination policy is

$$\begin{aligned} \pi^T(b_t|s_t,z_{t-1}) &= \frac{P^T_{\text{prior}}(b_t|s_t,z_{t-1})\exp V(s_t,z_{t-1},b_t)}{\exp V(s_t,z_{t-1})} \\ \text{where } V(s_t,z_{t-1},b_t) &= b_t\left[V^H(s_t)\right]+(1-b_t)V(s_t,z_{t-1}) \\ V(s_t,z_{t-1}) &= \log\int P^T_{\text{prior}}(b_t|s_t,z_{t-1})\exp\left(b_t\left[V^H(s_t)\right]+(1-b_t)V(s_t,z_{t-1})\right)\ \mathrm{d}b_t \end{aligned} \tag{4.7}$$

Finally, the Bellman equation for soft-hierarchical single-agent reinforcement learning is

$$Q(s_t,a_t,z_t) = \beta r(s_t,a_t)+\gamma\mathbb{E}_{s_{t+1}}\left[V(s_{t+1},z_t)\right] \tag{4.8}$$

The proof will be in appendix C.1. We can see that this is analogous to the options framework [74, 1]. Now, we have aware the work on similar topic [46], which the authors have almost the same solution as us but miss specific points on value function calculation, notably

$$\begin{aligned} Q(s_t,a_t,z_t) = \beta r(s_t,a_t)+\mathbb{E}_{s_{t+1}}\Big[\mathbb{E}_{b_{t+1}\sim P(b_{t+1}|s_{t+1},z_t),z'_{t+1}\sim P(z_{t+1}|s_{t+1})}\Big[&(1-b_{t+1})V(s_{t+1},z_t) \\ &+b_{t+1}V(s_{t+1},z'_t)\Big]\Big] \end{aligned} \tag{4.9}$$

The authors fails to consider the case in which the master policy soft-max the value function $V(s_{t+1},z_{t+1})$ and assuming this

$$\mathbb{E}_{z'_{t+1}\sim P(z_{t+1}|s_{t+1})}\left[V(s_{t+1},z_{t+1})\right] \tag{4.10}$$

Instead, which we can see that this misses the regularization of the policy π^H_ϕ. By using the soft-max, we have implicitly included the regularization.

4.2 Multi-Agent Delayed Soft-Option Communication

Now, we are going to consider the multi-agent case in which we have to consider the other agent's option at the last time step, which is a type of delayed communication, where the message is the option of the opponent. This algorithm would be a direct extension to the single-agent HRL

problem. The graphical model can be too complex to draw we, therefore, going to show only the factorization of joint probabilities. Consider the following joint probability distribution of the problem:

$$\begin{aligned}
&P(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T}, z^i_{1:T}, z^{-i}_{1:T}, b^i_{1:T}, b^{-i}_{1:T}, \mathcal{O}^i_{1:T} = 1, \mathcal{O}^{-i}_{1:T} = 1) \\
&\quad = P(s_1)P(z^{-i}_0, z^i_0) \prod_{t=1}^{T} P(s_{t+1}|s_t, a^i_t, a^{-i}_t) P_{\text{prior}}(a^{-i}_t|s_t, z^i_t, a^i_t, z^{-i}_t) P^{H,-i}_{\text{prior}}(z^{-i}_t|s_t, a^i_t, z^i_t, z^{-i}_{t-1}, b^i_t) \\
&\quad P^{T,-i}_{\text{prior}}(b^{-i}_t|s_t, z^{-i}_{t-1}, a^i_t, z^i_t) P_{\text{prior}}(a^i_t|s_t, z^i_t, z^{-i}_{t-1}) P^{H,i}_{\text{prior}}(z^i_t|s_t, z^{-i}_{t-1}, z^i_{t-1}, b^i_t) P^{T,i}_{\text{prior}}(b^i_t|s_t, z^i_{t-1}, z^{-i}_{t-1}) \\
&\quad P(\mathcal{O}^i_t = 1, \mathcal{O}^{-i}_t = 1|s_t, a^i_t, a^{-i}_t)
\end{aligned} \tag{4.11}$$

Now, we shall consider the joint variational distribution.

$$\begin{aligned}
&q(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T}, z^i_{1:T}, z^{-i}_{1:T}, b^i_{1:T}, b^{-i}_{1:T}, \mathcal{O}^{-i}_{1:T} = 1) \\
&\quad = P(s_1)P(z^{-i}_0, z^i_0) \prod_{t=1}^{T} q(s_{t+1}|s_t, a^i_t, a^{-i}_t) \rho_\phi(a^{-i}_t|s_t, z^i_t, a^i_t, z^{-i}_t) q^{H,-i}(z^{-i}_t|s_t, a^i_t, z^i_t, z^{-i}_{t-1}, b^i_t) \\
&\quad q^{T,-i}(b^{-i}_t|s_t, z^{-i}_{t-1}, a^i_t, z^i_t) \pi_\theta(a^i_t|s_t, z^i_t, z^{-i}_{t-1}) q^{H,i}(z^i_t|s_t, z^{-i}_{t-1}, z^i_{t-1}, b^i_t) q^{T,i}(b^i_t|s_t, z^i_{t-1}, z^{-i}_{t-1})
\end{aligned} \tag{4.12}$$

Where we have the optimal random variable to be

$$P(\mathcal{O}^i_t = 1, \mathcal{O}^{-i}_{1:T} = 1|s_t, a^i_t, a^{-i}_t) \propto \exp\left(\beta r(s_t, a^i_t, a^{-i}_t)\right) \tag{4.13}$$

Before we move on, let's specifically define the master policy's conditional probability and its prior:

$$\begin{aligned}
&q^{H,i}(z^i_t|s_t, z^{-i}_{t-1}, z^i_{t-1}, b^i_t) = (1 - b^i_t)\delta(z^i_t - z^i_{t-1}) + b^i_t q^{H,i}_\phi(z^i_t|s_t, z^{-i}_{T-1}) \\
&P_{\text{prior}}(z^i_t|s_t, z^{-i}_{t-1}, z^i_{t-1}, b^i_t) = (1 - b^i_t)\delta(z^i_t - z^i_{t-1}) + b^i_t \frac{1}{m} \\
&q^{H,-i}(z^{-i}_t|s_t, a^i_t, z^i_t, z^{-i}_{t-1}, b^i_t) = (1 - b^{-i}_t)\delta(z^{-i}_t - z^{-i}_{t-1}) + b^{-i}_t q^{H,-i}_\phi(z^{-i}_t|s_t, a^i_t, z^i_t) \\
&P_{\text{prior}}(z^{-i}_t|s_t, a^i_t, z^i_t, z^{-i}_{t-1}, b^i_t) = (1 - b^{-i}_t)\delta(z^{-i}_t - z^{-i}_{t-1}) + b^{-i}_t \frac{1}{m}
\end{aligned} \tag{4.14}$$

Given both of them, we can derive the ELBO by finding KL-divergence, which is equal to

$$\begin{aligned}
\mathbb{E}_q \Bigg[\sum_{t=0}^{T} \beta r(s_t, a^i_t, a^{-i}_t) &- \log \frac{\pi_\theta(a^i_t|s_t, z^i_t, z^{-i}_{t-1})}{P_{\text{prior}}(a^i_t|s_t, z^i_t, z^{-i}_{t-1})} - \log \frac{\rho_\theta(a^{-i}_t|s_t, z^i_t, a^i_t, z^{-i}_t)}{P_{\text{prior}}(a^{-i}_t|s_t, z^i_t, a^i_t, z^{-i}_t)} \\
&- \log \frac{q^{H,i}(z^i_t|s_t, z^{-i}_{t-1}, z^i_{t-1}, b^i_t)}{P^{H,i}_{\text{prior}}(z^i_t|s_t, z^{-i}_{t-1}, z^i_{t-1}, b^i_t)} - \log \frac{q^{H,-i}(z^{-i}_t|s_t, a^i_t, z^i_t, z^{-i}_{t-1}, b^i_t)}{P^{H,-i}_{\text{prior}}(z^{-i}_t|s_t, a^i_t, z^i_t, z^{-i}_{t-1}, b^i_t)} \\
&- \log \frac{q^{T,i}(b^i_t|s_t, z^i_{t-1}, z^{-i}_{t-1})}{P^{T,i}_{\text{prior}}(b^i_t|s_t, z^i_{t-1}, z^{-i}_{t-1})} - \log \frac{q^{T,-i}(b^{-i}_t|s_t, z^{-i}_{t-1}, a^i_t, z^i_t)}{P^{T,-i}_{\text{prior}}(b^{-i}_t|s_t, z^{-i}_{t-1}, a^i_t, z^i_t)} \Bigg]
\end{aligned} \tag{4.15}$$

We will show the full working for solving in appendix C.2. Now, the policies are: starting with opponent model's policy

$$\rho(a_t^{-i}|s_t, z_t^i, a_t^i, z_t^{-i}) = \frac{\exp Q(s_t, z_t^i, a_t^i, z_t^{-i}, a_t^{-i}) P_{\text{prior}}(a_t^{-i}|s_t, z_t^i, a_t^i, z_t^{-i})}{\exp Q(s_t, z_t^i, a_t^i, z_t^{-i})}$$
$$\text{where } Q(s_t, z_t^i, a_t^i, z_t^{-i}) = \log \int \exp Q(s_t, z_t^i, a_t^i, z_t^{-i}, a_t^{-i}) P_{\text{prior}}(a_t^{-i}|s_t, z_t^i, a_t^i, z_t^{-i}) \, \mathrm{d}a_t^{-i}$$

Now, the opponent's master policy is

$$q^{H,-i}(z_t^{-i}|s_t, a_t^i, z_t^i) = \frac{\exp Q(s_t, z_t^i, a_t^i, z_t^{-i}) P_{\text{prior}}(z_t^{-i}|s_t, a_t^i, z_t^i)}{\exp Q(s_t, z_t^i, a_t^i)}$$
$$\text{where } Q(s_t, z_t^i, a_t^i) = \log \int \exp Q(s_t, z_t^i, a_t^i, z_t^{-i}) P_{\text{prior}}(z_t^{-i}|s_t, a_t^i, z_t^i) \, \mathrm{d}z_t^{-i} \tag{4.16}$$

And, the opponent's terminal policy is

$$q^{T,-i}(b_t^{-i}|s_t, z_{t-1}^{-i}, a_t^i, z_t^i) = \frac{\exp Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i, b_t^{-i}) P_{\text{prior}}(b_t^{-i}|s_t, z_{t-1}^{-i}, a_t^i, z_t^i)}{\exp Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i)}$$
$$\text{where } Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i, b_t^{-i}) = b_t^{-i} \left[Q(s_t, z_t^i, a_t^i)\right] + (1 - b_t^{-i}) \left[Q(s_t, z_t^i, a_t^i, z_{t-1}^{-i})\right] \tag{4.17}$$
$$Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i) = \log \int \exp Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i, b_t^{-i}) P_{\text{prior}}(b_t^{-i}|s_t, z_{t-1}^{-i}, a_t^i, z_t^i) \, \mathrm{d}b_t^{-i}$$

Now, we shifted to agent's policy, which is equal to

$$\pi(a_t^i|s_t, z_t^i, z_{t-1}^{-i}) = \frac{\exp Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i) P_{\text{prior}}(a_t^i|s_t, z_t^i, z_{t-1}^{-i})}{\exp Q(s_t, z_{t-1}^{-i}, z_t^i)}$$
$$\text{where } Q(s_t, z_{t-1}^{-i}, z_t^i) = \log \int \exp Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i) P_{\text{prior}}(a_t^i|s_t, z_t^i, z_{t-1}^{-i}) \, \mathrm{d}a_t^i \tag{4.18}$$

The master policy for the agent follows the agent's value function

$$q^{H,i}(z_t^i|s_t, z_{t-1}^{-i}) = \frac{\exp Q(s_t, z_{t-1}^{-i}, z_t^i) P_{\text{prior}}(z_t^i|s_t, z_{t-1}^{-i})}{\exp Q(s_t, z_{t-1}^{-i})}$$
$$\text{where } Q(s_t, z_{t-1}^{-i}) = \log \int \exp Q(s_t, z_{t-1}^{-i}, z_t^i) P_{\text{prior}}(z_t^i|s_t, z_{t-1}^{-i}) \, \mathrm{d}z_t^i \tag{4.19}$$

Finally, ends with the agent termination policy with left us the the message.

$$q^{T,i}(b_t^i|s_t, z_{t-1}^i, z_{t-1}^{-i}) = \frac{\exp Q(s_t, z_{t-1}^i, z_{t-1}^{-i}, b_t^i) P_{\text{prior}}(b_t^i|s_t, z_{t-1}^i, z_{t-1}^{-i})}{\exp Q(s_t, z_{t-1}^i, z_{t-1}^{-i})}$$
$$\text{where } Q(s_t, z_{t-1}^i, z_{t-1}^{-i}, b_t^i) = b_t^i \left[Q(s_t, z_{t-1}^{-i})\right] + (1 - b_t^i) \left[Q(s_t, z_{t-1}^{-i}, z_{t-1}^i)\right] \tag{4.20}$$
$$Q(s_t, z_{t-1}^i, z_{t-1}^{-i}) = \log \int \exp Q(s_t, z_{t-1}^i, z_{t-1}^{-i}, b_t^i) P_{\text{prior}}(b_t^i|s_t, z_{t-1}^i, z_{t-1}^{-i}) \, \mathrm{d}b_t^i$$

Now, finally the bellman equation is

$$Q(s_t, a_t^i, a_t^{-i}, z_t^i, z_t^{-i}) = \beta r(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E}_{s_{t+1}} \left[Q(s_{t+1}, z_t^i, z_t^{-i})\right] \tag{4.21}$$

Note that this is similar to the one proposed to [26]. The agent's policy can be executed by message z_{t-1}^{-i} sent from the opponent. This process is also exciting since by having the message from an opponent, the agent can anticipate what the opponent can do next, which is shown in the calculation of the action-value function, and the agent can also understand how the message would affect the opponent's action.

4.3 Implementation

Now, we have considered the implementation of these two algorithms, starting with the implementation of single-agent learning, which can be easily extended to the multi-agent algorithm. The purpose of this section is to show that Soft Actor-Critic like algorithm can be applied directly to the implementation of the algorithms. Finally, we will not go through the tricks that would stabilize the training since we are focusing on the control as probabilistic framework extensions and proves that the implementation is possible by constructing one.

4.3.1 Single agent problem

Starting with the value function $V(s_t, z_t)$, which we can train using the following :

$$\mathcal{L}_{V_1} = \mathbb{E}_{s_t, z_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V(s_t, z_t) - \mathbb{E}_{a_t \sim \pi_\theta(a_t|s_t, z_t)} \left[Q(s_t, a_t, z_t) - \log \frac{\pi_\theta(a_t|s_t, z_t)}{P_{\text{prior}}(a_t|s_t, z_t)} \right] \right)^2 \right] \tag{4.22}$$

Similarly, the value function $V(s_t)$ can be trained to

$$\mathcal{L}_{V_2} = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\frac{1}{2} \left(V(s_t) - \mathbb{E}_{z_t \sim \pi^H(z_t|s_t)} \left[V(s_t, z_t) - \log \left(m \cdot \pi^H(z_t|s_t) \right) \right] \right)^2 \right] \tag{4.23}$$

Finally, we can train the value function for the termination policy $V(s_t, z_{t-1})$ as:

$$\mathcal{L}_{V_3} = \mathbb{E}_{s_t, z_{t-1} \sim \mathcal{D}} \left[\frac{1}{2} \left(V(s_t, z_{t-1}) - \mathbb{E}_{b_t \sim \pi^T(b_t|s_t, z_{t-1})} \left[V(s_t, z_{t-1}, b_t) - \log \frac{\pi^T(b_t|s_t, z_{t-1})}{P_{\text{prior}}(b_t|s_t, z_{t-1})} \right] \right) \right] \tag{4.24}$$

Now, lower level policy is in the form of $a_t = f^L(\xi; s_t, z_t)$ where $\xi \sim \mathcal{N}(0, I)$, we can train it to minimize the KL-divergence between this to the optimal policy that we have shown in equation 4.5 as

$$\mathcal{L}_{\pi^L} = D_{\text{KL}} \left(\pi^L(a_t|s_t, z_t) \middle\| \frac{P_{\text{prior}}(a_t|s_t, z_t) \exp\left(Q(s_t, a_t, z_t)\right)}{\exp V(s_t, z_t)} \right) \tag{4.25}$$

Similarly, the higher-level policy will be in the form of $z_t = f^H(\xi; s_t)$ where $\xi \sim \mathcal{N}(0, I)$ with the following KL-divergence objective from the optimal policy equation 4.6:

$$\mathcal{L}_{\pi^H} = D_{\text{KL}} \left(\pi^H(z_t|s_t) \middle\| \frac{\exp\left(Q(s_t, z_t)\right)}{m \exp\left(V(s_t)\right)} \right) \tag{4.26}$$

And the termination policy is in the form of $b_t = \sigma(f^T(\xi; s_t, z_{t-1}))$, where σ denotes the Sigmoid function, which transforms the output to be between 0 and 1. We still use the KL-divergence

objective derived from equation 4.7:

$$\mathcal{L}_{\pi^T} = D_{\mathrm{KL}}\left(\pi^T(b_t|s_t, z_{t-1}) \middle\| \frac{P^T_{\mathrm{prior}}(b_t|s_t, z_{t-1}) \exp V(s_t, z_{t-1}, b_t)}{\exp V(s_t, z_{t-1})}\right) \tag{4.27}$$

Finally, we can train the action value function $Q(s_t, a_t, z_t)$, which is calculated via the Bellman equation 4.8 as:

$$\mathcal{L}_Q = \mathbb{E}_{(s_t, a_t, z_t, r_t, s_{t+1})\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t, a_t, z_t) - \beta r(s_t, a_t) + \gamma V(s_{t+1}, z_t)\right)^2\right] \tag{4.28}$$

4.3.2 Multi agent problem

Since there are lots of objective to follows, we will list them down here. The technique used is the same as one proposed in single agent case. We shall start with the definition of the value function

- Opponent's lower level policy value function $Q(s_t, z^i_t, z^{-i}_t, a^i_t)$:

  $$\mathcal{L}_{Q_1} = \mathbb{E}_{(s_t, z^i_t, z^{-i}_t, a^i_t, a^i_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t, z^i_t, z^{-i}_t) - \mathbb{E}_{a^{-i}_t}\left[Q(s_t, z^i_t, a^i_t, z^{-i}_t, a^{-i}_t) - \log\frac{\rho(a^{-i}_t|s_t, z^i_t, z^{-i}_t, a^i_t)}{P_{\mathrm{prior}}(a^{-i}_t|s_t, z^i_t, z^{-i}_t, a^i_t)}\right]\right)^2\right] \tag{4.29}$$

  where the opponent model action is sampled from $a^{-i}_t \sim \rho(a^{-i}_t|s_t, z^i_t, z^{-i}_t, a^i_t)$.

- Opponent's master policy value function $Q(s_t, z^i_t)$:

  $$\mathcal{L}_{Q_2} = \mathbb{E}_{(s_t, z^i_t, z^{-i}_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t, z^i_t) - \mathbb{E}_{z^{-i}_t}\left[Q(s_t, z^i_t, z^{-i}_t) - \log\frac{q^{H,-i}(z^{-i}_t|s_t, z^i_t)}{P_{\mathrm{prior}}(z^{-i}_t|s_t, z^i_t)}\right]\right)^2\right] \tag{4.30}$$

  where the option is sampled from $z^{-i}_t \sim q^{H,-i}(z^{-i}_t|s_t, z^i_t)$.

- Opponent's termination policy value function $Q(s_t, z^{-i}_{t-1}, a^i_t, z^i_t)$:

  $$\mathcal{L}_{Q_3} = \mathbb{E}_{(s_t, z^{-i}_{t-1}, a^i_t, z^i_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t, z^{-i}_{t-1}, a^i_t, z^i_t) - \mathbb{E}_{b^{-i}_t}\left[Q(s_t, z^{-i}_{t-1}, a^i_t, z^i_t, b^{-i}_t) - \log\frac{q^{T,-i}(b^{-i}_t|s_t, z^{-i}_{t-1}, a^i_t, z^i_t)}{P_{\mathrm{prior}}(b^{-i}_t|s_t, z^{-i}_{t-1}, a^i_t, z^i_t)}\right]\right)^2\right] \tag{4.31}$$

  where the terminating signal is sampled from $b^{-i}_t \sim q^{T,-i}(b^{-i}_t|s_t, z^{-i}_{t-1}, a^i_t, z^i_t)$.

- Agent's lower level policy value function $Q(s_t, z^{-i}_{t-1}, z^i_t)$:

  $$\mathcal{L}_{Q_4} = \mathbb{E}_{(s_t, z^{-i}_{t-1}, z^i_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t, z^{-i}_{t-1}, z^i_t) - \mathbb{E}_{a^i_t}\left[Q(s_t, z^{-i}_{t-1}, a^i_t, z^i_t) - \frac{\pi(a^i_t|s_t, z^i_t, z^{-i}_{t-1})}{P_{\mathrm{prior}}(a^i_t|s_t, z^i_t, z^{-i}_{t-1})}\right]\right)^2\right] \tag{4.32}$$

  where the lower level action is sampled from $a^i_t \sim \pi(a^i_t|s_t, z^i_t, z^{-i}_{t-1})$

- Agent's master policy value function $Q(s_t, z^{-i}_{t-1})$:

  $$\mathcal{L}_{Q_5} = \mathbb{E}_{(s_t, z^{-i}_{t-1})\sim\mathcal{D}}\left[\frac{1}{2}\left(Q(s_t, z^{-i}_{t-1}) - \mathbb{E}_{z^i_t}\left[Q(s_t, z^{-i}_{t-1}, z^i_t) - \log\frac{q^{H,i}(z^i_t|s_t, z^{-i}_{t-1})}{P_{\mathrm{prior}}(z^i_t|s_t, z^{-i}_{t-1})}\right]\right)^2\right] \tag{4.33}$$

where the agent's option is sampled from $z_t^i \sim q^{H,i}(z_t^i|s_t, z_{t-1}^{-i})$

- Finally, agent's termination policy value function $Q(s_t, z_{t-1}^i, z_{t-1}^{-i})$:

$$\mathcal{L}_{Q_6} = \mathbb{E}\left[\frac{1}{2}\left(Q(s_t, z_{t-1}^i, z_{t-1}^{-i}) - \mathbb{E}_{b_t^i}\left(Q(s_t, z_{t-1}^i, z_{t-1}^{-i}, b_t^i) - \log\frac{q^{T,i}(b_t^i|s_t, z_{t-1}^i, z_{t-1}^{-i})}{P_{\text{prior}}(b_t^i|s_t, z_{t-1}^i, z_{t-1}^{-i})}\right)\right)^2\right] \tag{4.34}$$

where the agent's terminating condition is sampled from $b_t^i \sim q^{T,i}(b_t^i|s_t, z_{t-1}^i, z_{t-1}^{-i})$

Now, we have listed all the agent's and its opponent model's value function, we can now consider the policies objective.

- Opponent Model's lower level policy $a_t^{-i} = f(\xi; s_t, z_t^i, a_t^i, z_t^{-i})$

$$\mathcal{L}_\rho = D_{\text{KL}}\left(\rho_\theta(a_t^{-i}|s_t, z_t^i, a_t^i, z_t^{-i})\middle\|\frac{\exp Q(s_t, z_t^i, a_t^i, z_t^{-i}, a_t^{-i})P_{\text{prior}}(a_t^{-i}|s_t, z_t^i, a_t^i, z_t^{-i})}{\exp Q(s_t, z_t^i, a_t^i, z_t^{-i})}\right) \tag{4.35}$$

- Opponent Model's master policy $z_t^{-i} = f^{H,-i}(\xi; s_t, a_t^i, z_t^i)$

$$\mathcal{L}_{q^{H,-i}} = D_{\text{KL}}\left(q^{H,-i}(z_t^{-i}|s_t, a_t^i, z_t^i)\middle\|\frac{\exp Q(s_t, z_t^i, a_t^i, z_t^{-i})P_{\text{prior}}(z_t^{-i}|s_t, a_t^i, z_t^i)}{\exp Q(s_t, z_t^i, a_t^i)}\right) \tag{4.36}$$

- Opponent Model's termination policy $b_t^{-i} = f^{T,-i}(\xi; s_t, z_{t-1}^{-i}, a_t^i, z_t^i)$

$$\mathcal{L}_{q^{T,-i}} = D_{\text{KL}}\left(q^{T,-i}(b_t^{-i}|s_t, z_{t-1}^{-i}, a_t^i, z_t^i)\middle\|\frac{\exp Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i, b_t^{-i})P_{\text{prior}}(b_t^{-i}|s_t, z_{t-1}^{-i}, a_t^i, z_t^i)}{\exp Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i)}\right) \tag{4.37}$$

- Agent lower level policy $a_t^i = f(\xi; s_t, z_t^i, z_{t-1}^{-i})$

$$\mathcal{L}_\pi = D_{\text{KL}}\left(\pi_\theta(a_t^i|s_t, z_t^i, z_{t-1}^{-i})\middle\|\frac{\exp Q(s_t, z_{t-1}^{-i}, a_t^i, z_t^i)P_{\text{prior}}(a_t^i|s_t, z_t^i, z_{t-1}^{-i})}{\exp Q(s_t, z_{t-1}^{-i}, z_t^i)}\right) \tag{4.38}$$

- Agent master policy $z_t^i = f^{H,i}(\xi; s_t, z_{t-1}^{-i})$

$$\mathcal{L}_{q^{H,i}} = D_{\text{KL}}\left(q^{H,i}(z_t^i|s_t, z_{t-1}^{-i})\middle\|\frac{\exp Q(s_t, z_{t-1}^{-i}, z_t^i)P_{\text{prior}}(z_t^i|s_t, z_{t-1}^{-i})}{\exp Q(s_t, z_{t-1}^{-i})}\right) \tag{4.39}$$

- agent termination policy $b_t^i = f^{T,i}(\xi; s_t, z_{t-1}^i, z_{t-1}^{-i})$

$$\mathcal{L}_{q^{T,i}} = D_{\text{KL}}\left(q^{T,i}(b_t^i|s_t, z_{t-1}^i, z_{t-1}^{-i})\middle\|\frac{\exp Q(s_t, z_{t-1}^i, z_{t-1}^{-i}, b_t^i)P_{\text{prior}}(b_t^i|s_t, z_{t-1}^i, z_{t-1}^{-i})}{\exp Q(s_t, z_{t-1}^i, z_{t-1}^{-i})}\right) \tag{4.40}$$

Note that all the noises is sampled from normal distribution $\xi \sim \mathcal{N}(0, I)$. Finally, the action value function $Q(s_t, z_t^i, a_t^i, z_t^{-i}, a_t^{-i})$:

$$\mathcal{L}_Q = \mathbb{E}_{(s_t, z_t^i, a_t^i, z_t^{-i}, a_t^{-i}, r_t, s_{t+1}) \sim \mathcal{D}} \left[\frac{1}{2} \left(Q(s_t, z_t^i, a_t^i, z_t^{-i}, a_t^{-i}) - \beta r(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E} Q(s_{t+1}, z_t^i, z_t^{-i}) \right)^2 \right] \tag{4.41}$$

4.4 Conclusion

We have shown that soft option framework in the single-agent scenario can be solved in closed-form, which allows us to train it using soft Actor-Critic while clearing some misconception about the final results. We then move to a multi-agent setting, where we successfully derived an algorithm for training agent that can do temporal abstraction and communicate with each other.

Chapter 5

Probabilistic perspective on Public Belief MDP

Another crucial extension to the multi-agent policy would be the ability to infer the hidden state via its local information. We will consider the cooperative case in which the agent's action is available as part of "public observation". The main observation from [57, 17] is that we can turn a partially observable multi-agent problem into single-agent POMDP by considering the public agent that forms a belief of the state given histories of public observation. Given its belief over the state, the public agent constructs a placeholder policy that acts according to each agents' individual observation. By having the same placeholder policy for all agents, we ensure that all the agents are coordinated correctly. We will start by introducing the Public Belief MDP framework. Then we introduce Variational Sequential Monte Carlo (VSMC) [40, 49, 54], which is a basis to Deep Variational Reinforcement Learning (DVRL) [33] - an algorithm that captures the belief overstate. We will follow [68] on the control as a probabilistic framework interpretation of DVRL to construct our multi-agent algorithms.

5.1 Public Belief MDP

As we aware, recursive learning in many steps can be complex and expensive even in a cooperation setting. By using public agent formulation, we can reduce the problem into a single agent POMDP task, which would be easier to manage. This technique is proposed in [57] and extended to scale over by approximation and neural network in [17].

The public belief MDP is based on the fact that every agent can somehow infer the underlying state given a public observation (can be an action too). After forming a belief on the state, each agent can execute an action, which conditions on its private observation and the belief over the state. Given a decoupled process, we can have a centralized agent that only infer the underlying state and distribute the common belief to each lower-level agent to execute its action with some instruction. The process can be summarized to fit our perspective as follows:

- The public agent observes a public observation and construct a message $m_t \sim \pi_t^{\text{pri}}(\cdot|o_t^{\text{pub}}, a_t^{\text{pub}})$ with belief over state $\mathcal{B}_t = P(s_t|o_{\leq t}^{\text{pub}})$ where $o_{\leq t}^{\text{pub}}$ is the history of public policy till the time t

- The lower-level agent after receives the message from the higher-level agent will execute its action based on its private observation, the message given, and belief over state calculated using higher-level agent i.e $a_t^i \sim \pi^i(\cdot|m_t, \mathcal{B}_t, o_t^{\text{pri}})$.

Please note that by having a centralized public belief agent makes the coordination problem much easier since each agent can reason about the other's action based on its the belief of other agent's private observation conditioned on message m_t. And, both of the agent and its higher-level counterpart will try to optimize the same reward. Note that we can train the public agent individually for each agent, however, we have to make sure that they are the same, which can be implemented by a public seed as a common-knowledge [17].

5.2 Variational Sequential Monte Carlo and Deep Variational Reinforcement Learning

5.2.1 VSMC

Let's start with Importance-Weight Auto Encoder [5]. Consider the Jensen equality derivation of ELBO, where we can show that

$$\log P(X) = \log\left(\int P(X,z)\ \mathrm{d}z\right) = \log\left(\int P(z|X)P(X)\ \mathrm{d}z\right) = \log\left(\mathbb{E}_{P(z|X)}\left[P(X)\right]\right) \tag{5.1}$$

Now, given the expected value, we can use our variational distribution $q_\phi(z)$ to be our importance sampling distribution i.e

$$\begin{aligned}\log \mathbb{E}_{q_\phi(z|x)}\left[\frac{P(z|X)}{q_\phi(z|x)}P(X)\right] &= \log \mathbb{E}_{q_\phi(z|x)}\left[\frac{P(X,z)}{q_\phi(z|x)}\right] \\ &= \log \mathbb{E}_{z_1,\ldots,z_K}\left[\frac{1}{K}\sum_{k=1}^{K}\frac{P(X,z)}{q_\phi(z|x)}\right] \geq \mathbb{E}\left[\log\frac{1}{K}\sum_{k=1}^{K}\frac{P(X,z)}{q_\phi(z|x)}\right]\end{aligned} \tag{5.2}$$

We will now generalize the result from IWAE by using the notion of Monte Carlo objective [49], where we define the denote positive estimator of $P(x)$ as $\hat{P}_N(x)$. Then, the Monte Carlo objective (which is always a lower bound of $P(x)$) is defined as

$$\mathbb{E}[\log \hat{p}_N(x)] \tag{5.3}$$

For a hidden Markov model, we can simply calculate the $P(x_{1:T})$ where x_t is an observable variable using Sequential Monte Carlo (SMC) [12] we can train variational emission distribution and transition distribution using a lower bound in equation 5.3 via back-propagation.

5.2.2 DVRL

Typically, in reinforcement learning, we usually use a recurrent neural network (RNN) to encode the state to h_t condition of the past action and observation. Given the RNN state h_t we can then condition our policy on h_t. However, this only works because of memory and heuristic from RNN. DVRL aim to solve POMDP by

- Estimating the transition and observation model
- Perform an inference model
- Choosing an action based on inferred belief state.

The authors uses 3 elements triplet (h_t^k, z_t^k, w_t^k) where h_t^k is the latent of RNN, z_t^k is the additional stochastic latent state and w_t^k is the weight, as a particle. The belief is the updated, at time t, as follows

$$\begin{aligned} u_{t-1}^k &\sim \text{Categorical}\left(\frac{w_{t-1}^k}{\sum_{k=1}^K w_{t-1}^k}\right) \\ z_t^k &\sim q_\phi(z_t^k | h_{t-1}^{u_{t-1}^k}, a_{t-1}, o_t) \\ h_t^k &= \psi_\theta^{\text{RNN}}(h_{t-1}^{u_{t-1}^k}, z_t^k, a_{t-1}, o_t) \\ w_T^k &= \frac{P_\theta(z_t^k | h_{t-1}^{u_{t-1}^k}) P_\theta(o_t | h_{t-1}^{u_{t-1}^k}, z_t^k, a_{t-1})}{q_\phi(z_t^k | h_{t-1}^{u_{t-1}^k}, a_{t-1}, o_t)} \end{aligned} \tag{5.4}$$

The model of the world is trained via ELBO

$$\mathcal{L}_t^{ELBO}(\theta, \phi) = -\frac{1}{n_e n_s} \sum_{\text{envs}}^{n_e} \sum_{i=0}^{n_s - 1} \log\left(\frac{1}{K} \sum_{k=1}^K w_{t+i}^k\right) \tag{5.5}$$

The agent executes its action $\pi(a_t|\hat{h}_t)$ [1] by using another RNN to aggregate all the history of particles, which returns $\hat{h}_t$. The agent's reward and its model of the world are trained *jointly*f[2] using back-propagation through time (BPTT) via standard Actor-Critic framework. For more information on BPTT training for the agent, we refer to the paper for more details. [68] has proposed a graphical model representation of the problem, while showing that a similar algorithm could be derived from control as inference framework. The major difference is how the agent's action is calculated. As mentioned before [33] uses secondary RNN to aggregate the histories to inform the agent, however, [68] uses a weighted average on the action of the agent over its belief over state i.e

$$\sum_{k=1}^K \frac{w_t^k}{\sum_{k=1}^K w_t^k} \pi(a_t | h_t^k) \tag{5.6}$$

Another work on using control as inference framework is [31]. The authors also consider optimizing ELBO objective. However, we believe that [68] provides more general solution since once the number of particle is equal to one, ELBO derived in [31][3] is similar to [68][4]/ELBO from multiple. Furthermore, by having multiple possibilities of hidden state with its weight, we can consider a counter-factual in multi-agent scenario, thus being more useful than one state representation. Finally, the difference between [31] and [68] lies in the use of VAE like objective when considering

[1] and calculate its value, since it is Actor-Critic model

[2] We disagree with [31] when the authors claim that [33] "separate the planning algorithm and the inference of the hidden state.", since [33] claims in the abstract that DVRL "allowing the model to be trained jointly with the policy."

[3] See equation 19 in appendix of [31]

[4] See equation 4 in appendix of [68]

state transition, as [31] minimizes VAE objective for state representation as an auxiliary loss, while [68] doesn't and focuses on maximizing ELBO only.

5.3 Probabilistic Public Agent

Now, let's formalized the problem of public Belief into the graphical model setting. We will follow derivation from [68]. The graphical model is depicted in figure 5.1. The reason we split between

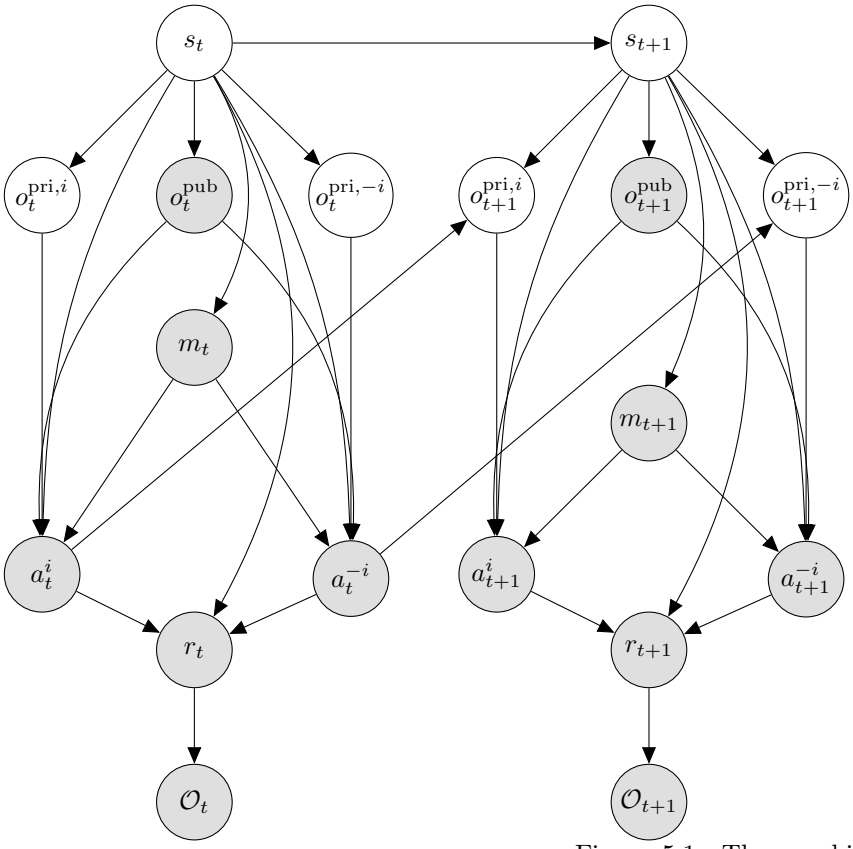


Figure 5.1: The graphical model represents the public belief POMDP o^{pub} representing the public observation, which depends on the current state and agents' action. We left the state transition function $P(s_{t+1}|s_t, a_t)$ to reduce the clusters. We assume a cooperative setting in which each agent (including the public belief agent) trying to optimize

reward r_t and optimality $\mathcal{O}_t$ is that the reward will depend on unknown state hence but when we are training agent we didn't know that state. This separation will be clear after we have derived the public agent. Let's start by defining our joint distribution for this:

$$P(s_0)\prod_{t=0}^{T} P(s_{t+1}|s_t, a_t^i, a_t^{-i})P(o_{t+1}^{\text{pri},i}|s_{t+1}, a_t^i)P(o_{t+1}^{\text{pri},-i}|s_{t+1}, a_t^{-i})P(o_{t+1}^{\text{pub}}|s_{t+1}, a_t^i, a_t^{-i}) \\ P(a_t^i|s_t, o_t^{\text{pri},i}, m_t, o_t^{\text{pub}})P(a_t^{-i}|s_t, o_t^{\text{pri},-i}, m_t, o_t^{\text{pub}})P(\mathcal{O}_t|r_t)P(r_t|s_t, a_t^i, a_t^{-i})P(m_t|s_t) \tag{5.7}$$

Let's consider the following scenario: In card games, usually, the public observation consists of action of agents and possibly a public card. The action of agents can also affect the state. The state is the configuration of the cards for all players, and the private observation process $P(o_{t+1}^{\text{pri},i}|s_{t+1}, a_t^i)$ is simply a mask to the state that allows us to see only our card, a similar process occurs with the public card $P(o_{t+1}^{\text{pub}}|s_{t+1}, a_t^i, a_t^{-i})$. This shows that the state is sufficiently statistic for individual private observation. Now, it's the time to consider the derivation of a public agent, which we can treat as POMDP problem, where the derivation follows [68]. We consider the variational distribution for the public agent:

$$\begin{aligned} P(s_0) \left(\prod_{t=0}^{T} q(s_{t+1}|s_t, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, \mathcal{O}_t, r_t)\right) \left(\prod_{t=0}^{T} q(o_{t+1}^{\text{pri},i}|s_t, a_t^i, o_{t+1}^{\text{pub}}, m_t, \mathcal{O}_t, r_t)\right) \\ \left(\prod_{t=0}^{T} q(o_{t+1}^{\text{pri},-i}|s_t, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, \mathcal{O}_t, r_t)\right) \left(\prod_{t=0}^{T} q(m_t|o_{1:t}^{\text{pub}}, a_{1:t}^i, a_{1:t}^{-i}, \mathcal{O}_{1:T}, r_{1:T})\right) \end{aligned} \tag{5.8}$$

Let's consider the ELBO for the problem. We have the following:

$$\begin{aligned} \mathbb{E}\Bigg[\sum_{t=1}^{T} \log \frac{P(s_{t+1}|s_t, a_t^i, a_t^{-i}) P(o_{t+1}^{\text{pri},i}|s_{t+1}, a_t^i) P(o_{t+1}^{\text{pri},-i}|s_{t+1}, a_t^{-i}) P(o_{t+1}^{\text{pub}}|s_{t+1}, a_t^i, a_t^{-i}) P(a_t^i|s_t, o_t^{\text{pri},i}, m_t, o_t^{\text{pub}})}{q(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}|s_t, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, \mathcal{O}_t, r_t)} \\ \cdot \frac{P(m_t|s_t) P(\mathcal{O}_t, r_t|r_t) P(r_t|s_t, a_t^i, a_t^{-i}) P(a_t^{-i}|s_t, o_t^{\text{pri},-i}, m_t, o_t^{\text{pub}})}{q(m_t|o_{1:t}^{\text{pub}}, a_{1:t}^i, a_{1:t}^{-i}, \mathcal{O}_{1:T}, r_{1:T})}\Bigg] \end{aligned} \tag{5.9}$$

Now, we want to infer the unknown given what we can observe:

$$\begin{aligned} P(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t | a_{1:t}^i, a_{1:t}^{-i}, \mathcal{O}_{1:T}, r_{1:T}, o_{1:t+1}^{\text{pub}}) \\ = \frac{P(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t, a_t^i, a_t^{-i}, \mathcal{O}_t, r_t, o_{t+1}^{\text{pub}} | a_{1:t-1}^i, a_{1:t-1}^{-i}, \mathcal{O}_{1:t-1}, r_{1:t-1}, o_{1:t}^{\text{pub}})}{P(a_t^i, a_t^{-i}, \mathcal{O}_t, r_t, o_{t+1}^{\text{pub}} | a_{1:t-1}^i, a_{1:t-1}^{-i}, \mathcal{O}_{1:t-1}, r_{1:t-1}, o_{1:t}^{\text{pub}})} \end{aligned} \tag{5.10}$$

We shall consider the numerator first. We can see that the current joint distribution at time t is conditional independent given the current state.

$$\begin{aligned} P(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t, a_t^i, a_t^{-i}, \mathcal{O}_t, r_t, o_{t+1}^{\text{pub}} | a_{1:t-1}^i, a_{1:t-1}^{-i}, \mathcal{O}_{1:t-1}, r_{1:t-1}, o_{1:t}^{\text{pub}}) \\ = \int P(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t, a_t^i, a_t^{-i}, \mathcal{O}_t, r_t, o_{t+1}^{\text{pub}} | s_t) P(s_t | a_{1:t-1}^i, a_{1:t-1}^{-i}, \mathcal{O}_{1:t-1}, r_{1:t-1}, o_{1:t}^{\text{pub}}) \, \mathrm{d}s_t \end{aligned} \tag{5.11}$$

Let's perform the importance sampling using the proposal distribution or the variational distribution:

$$\begin{aligned} \int P(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t, a_t^i, a_t^{-i}, \mathcal{O}_t, r_t, o_{t+1}^{\text{pub}} | s_t) P(s_t | a_{1:t-1}^i, a_{1:t-1}^{-i}, \mathcal{O}_{1:t-1}, r_{1:t-1}, o_{1:t}^{\text{pub}}) \, \mathrm{d}s_t \\ = \int \frac{P(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t, a_t^i, a_t^{-i}, \mathcal{O}_t, r_t, o_{t+1}^{\text{pub}} | s_t)}{q(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t)} q(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t) \\ P(s_t | a_{1:t-1}^i, a_{1:t-1}^{-i}, \mathcal{O}_{1:t-1}, r_{1:t-1}, o_{1:t}^{\text{pub}}) \, \mathrm{d}s_t \end{aligned} \tag{5.12}$$

We will denote the importance weight $w_{t+1}(s_t, s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t)$ as

$$w_{t+1}(s_t, s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t) = \frac{P(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t, a_t^i, a_t^{-i}, \mathcal{O}_t, r_t, o_{t+1}^{\text{pub}}|s_t)}{q(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t)} \tag{5.13}$$

Now given the following importance sampling, we can see that we can sample the state s_t by using ancestor index u_t^k. Let's approximate this using ancestor index type sampling:

$$\begin{aligned}
&\frac{1}{K}\sum_{k=1}^{K} w_{t+1}(s_t^{u_t^k}, s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t) \cdot q(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t|s_t^{u_t^k}, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}) \\
=&\frac{1}{K}\sum_{k=1}^{K} w_{t+1}(s_t^{u_t^k}, s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t) q(s_{t+1}|s_t^{u_t^k}, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}) q(o_{t+1}^{\text{pri},i}|s_t^{u_t^k}, a_t^i, m_t, \mathcal{O}_{1:T}, r_{1:T}, o_{t+1}^{\text{pub}}) \\
&\qquad\qquad q(o_{t+1}^{\text{pri},-i}|s_t^{u_t^k}, a_t^{-i}, m_t, \mathcal{O}_{1:T}, r_{1:T}, o_{t+1}^{\text{pub}}) q(m_t|o_{1:t}^{\text{pub}}, a_{1:t}^{-i}, a_{1:t}^i, \mathcal{O}_{1:T}, r_{1:T}) \\
\approx&\frac{1}{K}\sum_{k=1}^{K} w_{t+1}(s_t^{u_t^k}, s_{t+1}^k, o_{t+1}^{\text{pri},i,k}, o_{t+1}^{\text{pri},-i,k}, m_t^1)\delta(s_{t+1} - s_{t+1}^k)\delta(o_{t+1}^{\text{pri},i} - o_{t+1}^{\text{pri},i,k}) \\
&\qquad\qquad \delta(o_{t+1}^{\text{pri},-i} - o_{t+1}^{\text{pri},-i,k})\delta(m_t - m_t^1)
\end{aligned} \tag{5.14}$$

We can see above that we can further approximate our equation using sampling from our proposal/variational distribution. Now, we can find the denominator by marginalization. Given this, we have the final posterior being weighted mixture of particles:

$$\begin{aligned}
&P(s_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pri},-i}, m_t|a_{1:t}^i, a_{1:t}^{-i}, \mathcal{O}_{1:T}, r_{1:T}, o_{1:t+1}^{\text{pub}}) \\
&\qquad = \frac{1}{K}\sum_{k=1}^{K} W_{t+1}^k \delta(s_{t+1} - s_{t+1}^k)\delta(o_{t+1}^{\text{pri},i} - o_{t+1}^{\text{pri},i,k})\delta(o_{t+1}^{\text{pri},-i} - o_{t+1}^{\text{pri},-i,k})\delta(m_t - m_t^1)
\end{aligned} \tag{5.15}$$

where the W_{t+1}^k is the re-normalized weight. Now, the probability over the state is simply:

$$P(s_{t+1}|a_{1:t}^i, a_{1:t}^{-i}, \mathcal{O}_{1:T}, r_{1:T}, o_{1:t+1}^{\text{pub}}) = \frac{1}{K}\sum_{k=1}^{K} W_{t+1}^k \delta(s_{t+1} - s_{t+1}^k) \tag{5.16}$$

which we can sample future state using ancestor index similar to equation 5.14. Now, since the probability of the observable variable is:

$$P(a_{1:\tau}^i, a_{1:\tau}^{-i}, \mathcal{O}_{1:\tau}, r_{1:\tau} o_{1:\tau}^{\text{pub}}) = \prod_{t=0}^{\tau} \frac{1}{K}\sum_{k=1}^{K} w_{t+1}(s_t^{u_t^k}, s_{t+1}^k, o_{t+1}^{\text{pri},i,k}, o_{t+1}^{\text{pri},-i,k}, m_t^1) \tag{5.17}$$

which is a tighter variational lower bound. Now, the objective becomes:

$$\mathbb{E}\left[\sum_{t=0}^{\tau} \log \frac{1}{K}\sum_{k=1}^{K} w_{t+1}(s_t^{u_t^k}, s_{t+1}^k, o_{t+1}^{\text{pri},i,k}, o_{t+1}^{\text{pri},-i,k}, m_t^1)\right] \tag{5.18}$$

Let's recall the definition of the importance weight with suitable indexes:

$$\begin{aligned}
&\frac{P(s_{t+1}^k|s_t^{u_t^k}, a_t^i, a_t^{-i})P(o_{t+1}^{\text{pri},i,k}|s_{t+1}^k, a_t^i)P(o_{t+1}^{\text{pri},-i,k}|s_{t+1}^k, a_t^{-i})P(o_{t+1}^{\text{pub}}|s_{t+1}^k, a_t^i, a_t^{-i})}{q(s_{t+1}^k|s_t^{u_t^k}, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, \mathcal{O}_t, r_t)} \\
\times &\frac{P(a_t^i, a_t^{-i}|s_t^{u_t^k}, o_t^{\text{pri},i}, o_t^{\text{pri},-i}, m_t, o_t^{\text{pub}})P(m_t|s_t^{u_t^k})P(\mathcal{O}_t|r_t)}{q(o_{t+1}^{\text{pri},i,k}, o_{t+1}^{\text{pri},-i,k}|s_t^{u_t^k}, a_t^i, a_t^{-i}, m_t, o_{t+1}^{\text{pub}}, \mathcal{O}_t, r_t)} \\
\times &\frac{P(\mathcal{O}_t|r_t)}{q(m_t|o_{1:t}^{\text{pub}}, a_{1:t}^i, a_{1:t}^{-i}, \mathcal{O}_{1:T}, r_{1:T})}
\end{aligned} \tag{5.19}$$

We have the following objective that follows maximum entropy framework:

$$\begin{aligned}
\mathbb{E}\Bigg[\sum_{t=0}^{\tau} &\log P(\mathcal{O}_t|r_t) + \mathbb{H}\left[q(m_t|o_{1:t}^{\text{pub}}, a_{1:t}^i, a_{1:t}^{-i}, \mathcal{O}_{1:T}, r_{1:T})\right] \\
&+ \log \frac{1}{K}\sum_{k=1}^{K}\Bigg(\frac{P(s_{t+1}^k|s_t^{u_t^k}, a_t^i, a_t^{-i})P(o_{t+1}^{\text{pri},i,k}|s_{t+1}^k, a_t^i)P(o_{t+1}^{\text{pri},-i,k}|s_{t+1}^k, a_t^{-i})P(o_{t+1}^{\text{pub}}|s_{t+1}^k, a_t^i, a_t^{-i})}{q(s_{t+1}^k|s_t^{u_t^k}, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, \mathcal{O}_t, r_t)} \\
&\cdot \frac{P(a_t^i, a_t^{-i}|s_t^{u_t^k}, o_t^{\text{pri},i,k}, o_t^{\text{pri},-i,k}, m_t, o_t^{\text{pub}})P(m_t|s_t^{u_t^k})P(\mathcal{O}_t|r_t)}{q(o_{t+1}^{\text{pri},i,k}, o_{t+1}^{\text{pri},-i,k}|s_t^{u_t^k}, a_t^i, a_t^{-i}, m_t, o_{t+1}^{\text{pub}}, \mathcal{O}_t, r_t)}\Bigg)\Bigg]
\end{aligned} \tag{5.20}$$

Now, in practice the public agent shall be a weighted sum of underlying state, public observation, and agent actions i.e

$$\sum_{k=1}^{K} W_t^k \pi(m_t^k|s_t^k, o_t^{\text{pub}}, a_t^i, a_t^{-i}) \tag{5.21}$$

There are two minor issues that we want to consider (that we have mentioned above):

- The interpretability of $o_t^{\text{pri},i,k}$ and $o_t^{\text{pri},-i,k}$. There isn't a lot we can do with this apart from constructing a network that would translate the public agent's private observation into the agent's private observation. Or we can pretrain a neural network that outputs the agents' private observation. However, we have to make sure not to change the main public agent's network by backpropagating the local feature, since all the agents must have the same public agent.

- The joint action $P(a_t^i, a_t^{-i}|s_t^{u_t^k}, o_t^{\text{pri},i}, o_t^{\text{pri},-i}, m_t, o_t^{\text{pub}})$. After having all the private observation and public observation, we can try to optimize the reconstruction loss on the agents' actions. However, we can further assume that the agents' policies are based on Boltzmann distribution of action-value function. Given this assumption, the maximum-log-likelihood estimation should follow the usual adversarial imitation learning that have been extensively discussed in [15, 81]. For a normal scenario, we simply able to train its directly using supervised learning. As noted before, we shouldn't include any additional information from the agent to help us with the public agent (including the agent's action-value function).

We would like to state that all the problems above can be lifted if we assume that each agent has its own public agent and trying to model others using oneself or other methods. By ignoring the common knowledge of messages m_t, we lose a guarantee that the agents will be in "sync" with each other, thus returning to a problem of equilibrium selection. Finally, we shall consider the individual

agent that corresponds on the message, its private observation, and the state. Let's start with the definition of action-value function $Q(s_t, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})$, which is defined as:

$$Q(\{s_t\}_{k=1}^K, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) = \sum_{k=1}^K W_t^k Q(s_t^k, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) \tag{5.22}$$

The agent is based on single agent problem, which we can find its policy as:

$$\begin{aligned}
\pi(a_t^i|\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) &= \frac{P_{\text{prior}}(a_t^i|\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) \exp Q(\{s_t^k\}_{k=1}^K, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})}{V(\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})} \\
\text{where } Q(s_t, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) &= r_t + \gamma \mathbb{E}\left[Q(\{s_{t+1}^k\}_{k=1}^K, m_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pub}})\right] \\
V(\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) &= \int P_{\text{prior}}(a_t^i|\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) \exp Q(\{s_t^k\}_{k=1}^K, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) \, \mathrm{d}a_t^i
\end{aligned} \tag{5.23}$$

5.4 Implementation

Now, let's consider the implementation of public agent, which is similar to DVRL. Starting with the hidden state calculation. For each time step, we have 3 hidden variables that we have to infer

$$\begin{aligned}
u_{t-1}^k &\sim \text{Categorical}\left(\frac{w_{t-1}^k}{\sum_{k=1}^K w_{t-1}^k}\right) \\
x_{t+1}^k &\sim q_\psi(s_{t+1}|f_t^{u_t^k}, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, r_t) \\
y_{t+1}^k &\sim q_\psi(o_{t+1}^{\text{pri},i}|f_t^{u_t^k}, a_t^i, o_{t+1}^{\text{pub}}, m_t, r_t) \\
z_{t+1}^k &\sim q_\psi(o_{t+1}^{\text{pri},-i}|f_t^{u_t^k}, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, r_t) \\
f_t^k &= \psi_\theta^{\text{RNN }1}(x_{t+1}^k, f_t^{u_t^k}, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, \mathcal{O}_t, r_t) \\
g_t^k &= \psi_\theta^{\text{RNN }2}(y_{t+1}^k, g_t^{u_t^k}, a_t^i, o_{t+1}^{\text{pub}}, m_t, r_t) \\
h_t^k &= \psi_\theta^{\text{RNN }3}(z_{t+1}^k, h_t^{u_t^k}, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, r_t) \\
w_T^k = &\frac{P_\theta(s_{t+1}^k|s_t^{u_t^k}, a_t^i, a_t^{-i}) P_\theta(o_{t+1}^{\text{pri},i,k}|s_{t+1}^k, a_t^i) P_\theta(o_{t+1}^{\text{pri},-i,k}|s_{t+1}^k, a_t^{-i}) P_\theta(o_{t+1}^{\text{pub}}|s_{t+1}^k, a_t^i, a_t^{-i})}{q_\psi(s_{t+1}^k|s_t^{u_t^k}, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, r_t)} \\
&\cdot \frac{P_\theta(a_t^i, a_t^{-i}|s_t^{u_t^k}, o_t^{\text{pri},i}, o_t^{\text{pri},-i}, m_t, o_t^{\text{pub}}) P_\theta(m_t|s_t^{u_t^k}) P_\theta(\mathcal{O}_t|r_t)}{q_\psi(o_{t+1}^{\text{pri},i,k}, o_{t+1}^{\text{pri},-i,k}|s_t^{u_t^k}, a_t^i, a_t^{-i}, m_t, o_{t+1}^{\text{pub}}, r_t)} \cdot \frac{P_\theta(\mathcal{O}_t|r_t)}{q_\psi(m_t|o_{1:t}^{\text{pub}}, a_{1:t}^i, a_{1:t}^{-i}, r_{1:T})}
\end{aligned} \tag{5.24}$$

Now, we have the following particles, which is in tuple $(x_{t+1}^k, y_{t+1}^k, z_{t+1}^k, f_t^k, g_t^k, h_t^k, w_t^k)$. Now the agent have to be optimized the EBLO in equation 5.25, where we have

$$\mathbb{E}\left[\sum_{t=0}^\tau \log P(\mathcal{O}_t|r_t) + \mathbb{H}\left[q_\psi(m_t|o_{1:t}^{\text{pub}}, a_{1:t}^i, a_{1:t}^{-i}, \mathcal{O}_{1:T}, r_{1:T})\right] + \log \frac{1}{K}\sum_{k=1}^K \tilde{w}_t^k\right] \tag{5.25}$$

where

$$\begin{aligned}\tilde{w}_t^k =& \frac{P_\theta(s_{t+1}^k|s_t^{u_t^k}, a_t^i, a_t^{-i})P_\theta(o_{t+1}^{\text{pri},i,k}|s_{t+1}^k, a_t^i)P_\theta(o_{t+1}^{\text{pri},-i,k}|s_{t+1}^k, a_t^{-i})P_\theta(o_{t+1}^{\text{pub}}|s_{t+1}^k, a_t^i, a_t^{-i})}{q_\psi(s_{t+1}^k|s_t^{u_t^k}, a_t^i, a_t^{-i}, o_{t+1}^{\text{pub}}, m_t, \mathcal{O}_t, r_t)} \\ &\cdot \frac{P_\theta(a_t^i, a_t^{-i}|s_t^{u_t^k}, o_t^{\text{pri},i,k}, o_t^{\text{pri},-i,k}, m_t, o_t^{\text{pub}})P_\theta(m_t|s_t^{u_t^k})P_\theta(\mathcal{O}_t|r_t)}{q_\psi(o_{t+1}^{\text{pri},i,k}, o_{t+1}^{\text{pri},-i,k}|s_t^{u_t^k}, a_t^i, a_t^{-i}, m_t, o_{t+1}^{\text{pub}}, \mathcal{O}_t, r_t)}\end{aligned} \tag{5.26}$$

Now, let's consider the lower level agent's objective. We are going to start with the definition of the value function $V(\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})$, which trained to minimize:

$$\begin{aligned}\mathcal{L}_V = \mathbb{E}_{\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}} \sim \mathcal{D}} \Bigg[\frac{1}{2} \Bigg(& V(\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) \\ & - \mathbb{E}_{a_t^i} \left[q_\psi(\{s_t^k\}_{k=1}^K, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) - \log \frac{P_{\text{prior}}(a_t^i|\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})}{\pi(a_t^i|\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})} \right] \Bigg)^2 \Bigg]\end{aligned} \tag{5.27}$$

where the action is sampled from $a_t^i \sim \pi(a_t^i|\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})$. Now for the action value function, we follows from the Bellman equation:

$$\begin{aligned}\mathcal{L}_Q = \mathbb{E}_{\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}, r_t, \{s_{t+1}^k\}_{k=1}^K, m_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pub}}, \sim \mathcal{D}} \Bigg[\frac{1}{2} \Bigg(& q_\psi(\{s_t^k\}_{k=1}^K, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) \\ & - \left[r_t + \gamma V(\{s_{t+1}^k\}_{k=1}^K, m_{t+1}, o_{t+1}^{\text{pri},i}, o_{t+1}^{\text{pub}}) \right] \Bigg) \Bigg]\end{aligned} \tag{5.28}$$

Finally, the policy will be in the form of $f(\xi; \{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})$ where $\xi \sim \mathcal{N}(0, I)$ and trained using minimizing the KL-divergence:

$$\mathcal{L}_\pi = D_{\text{KL}} \left(\pi(a_t^i|\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) \middle\| \frac{P_{\text{prior}}(a_t^i|\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}}) \exp q_\psi(\{s_t^k\}_{k=1}^K, a_t^i, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})}{V(\{s_t^k\}_{k=1}^K, m_t, o_t^{\text{pri},i}, o_t^{\text{pub}})} \right) \tag{5.29}$$

Given this, we conclude the training for probabilistic public-agent.

5.5 Conclusion

In the last chapter, we consider the problem of representing a belief of others. We relaxed the problem by allowing only a cooperative setting, which leads us to public-belief MDP formulation. By having this formulation, we can turn the problem into a POMDP problem, which can be solved via VSMC and control as an inference framework. We proposed an algorithm that would allow us to perform a public belief update and using this belief to act.

Chapter 6

Conclusion and Future Work

6.1 Summary

We have shown the *potential* of control as probabilistic inference framework in multi-agent reinforcement learning. We started by expanding the training capability of the current framework by reinterpreting Balancing Q-learning into a probabilistic framework. The reinterpretation allows us to create a training algorithm that can work in a competitive setting via probabilistic inference, see chapter 3. Then, we moved to expand the agent's capability so that it can perform temporal abstraction with communication via the use of soft-option framework [32] in chapter 4. Finally, we concluded our work on reasoning in an unknown state via public-belief MDP framework, which arises almost naturally given the graphical model representation of the problem, see chapter 5.

Given two examples, we have sufficiently showed that control as probability inference is *one of the prominent ways* multi-agent reinforcement learning problem can be tackled systematically while being inherently complex. The algorithms derived from this framework not only have theoretical grounded support but also serves as a "soft" counterpart to many algorithms that are derived before, for example, Delayed Soft-Option Communication algorithm is a direct counterpart to algorithm proposed in [26]. We hope that this framework will be developed further and extended to many other sub-classes of problems in multi-agent reinforcement learning literature.

6.2 Future Works

We have tackled two extensions to our MAPI framework, which allows the agent to develop a fundamental ability for any multi-agent system, which are the ability to communicate and the ability to reason under uncertainty. As shown in chapter 4 and chapter 5, the algorithm presented is slightly too large to implement via standard machine. One of the immediate task to complete in a future would be to reduce the size of the algorithm so that it is manageable. However, this isn't the primary goal of this thesis, in which our goal is to show that the multi-agent problem can be reduced down to the probabilistic model, which can be solved with ease. The logical next step would be to apply this framework on other problems that are untackled in this thesis, including a unified view of adversarial imitation learning and context-aware agent in a multi-agent setting, and more generalized recursive reasoning that includes competitive games with unknown environment

state. Furthermore, multi-tasking and distilling of multi-agent policy can also be exciting problems as they are related to control as probabilistic inference framework or even lifelong learning algorithm for multi-agent problems.

Looking back, investigating the effects of entropy regularization in the opponent model and agent through the lens of exploration can be fruitful, as shown in [50] that some of the multi-agent algorithms do suffer from exploration problem. Would this also occur in entropy regularized MARL algorithms? How the regularization affects the training trajectory of the agent? These are importance questions that require further investigation that are not investigated in this work.

In conclusion, there are two primary directions that one can take. Firstly, one can expand the family of MAPI algorithms by reinterpreting the single-agent algorithm and problems as a probabilistic model. Or, secondly, one can look inward on analyzing the performance that entropy regularization gives to a multi-agent algorithm.

Bibliography

[1] Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[2] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 449–458. JMLR. org, 2017.

[3] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[4] Ulrich Berger. Brown's original fictitious play. *Journal of Economic Theory*, 135(1):572–578, 2007.

[5] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[6] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998(746-752):2, 1998.

[7] Christian Daniel, Herke Van Hoof, Jan Peters, and Gerhard Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104(2-3):337–357, 2016.

[8] Constantinos Daskalakis. On the complexity of approximating a nash equilibrium. *ACM Transactions on Algorithms (TALG)*, 9(3):1–35, 2013.

[9] Constantinos Daskalakis, Paul W Goldberg, and Christos H Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009.

[10] Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pages 271–278, 1993.

[11] Prashant Doshi and Piotr J Gmytrasiewicz. Monte carlo sampling methods for approximating interactive pomdps. *Journal of Artificial Intelligence Research*, 34:297–337, 2009.

[12] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[13] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating functional knowledge in neural networks. *Journal of Machine Learning Research*, 10(Jun):1239–1262, 2009.

[14] Matthew Fellows, Anuj Mahajan, Tim GJ Rudner, and Shimon Whiteson. Virel: A variational inference framework for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 7120–7134, 2019.

[15] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

[16] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[17] Jakob N Foerster, Francis Song, Edward Hughes, Neil Burch, Iain Dunning, Shimon Whiteson, Matthew Botvinick, and Michael Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1811.01458*, 2018.

[18] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

[19] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[20] Tim Genewein, Felix Leibfried, Jordi Grau-Moya, and Daniel Alexander Braun. Bounded rationality, abstraction, and hierarchical decision-making: An information-theoretic optimality principle. *Frontiers in Robotics and AI*, 2:27, 2015.

[21] Piotr J Gmytrasiewicz and Prashant Doshi. A framework for sequential planning in multi-agent settings. *Journal of Artificial Intelligence Research*, 24:49–79, 2005.

[22] Jordi Grau-Moya, Felix Leibfried, and Haitham Bou-Ammar. Balancing two-player stochastic games with soft q-learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 268–274. AAAI Press, 2018.

[23] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org, 2017.

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

[25] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[26] Dongge Han, Wendelin Boehmer, Michael Wooldridge, and Alex Rogers. Multi-agent hierarchical reinforcement learning with dynamic termination. In *Pacific Rim International Conference on Artificial Intelligence*, pages 80–92. Springer, 2019.

[27] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[28] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[29] Hengyuan Hu and Jakob N Foerster. Simplified action decoder for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1912.02288*, 2019.

[30] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov):1039–1069, 2003.

[31] Shiyu Huang, Hang Su, Jun Zhu, and Ting Chen. Svqn: Sequential variational soft q-learning networks.

[32] Maximilian Igl, Andrew Gambardella, Nantas Nardelli, N Siddharth, Wendelin Böhmer, and Shimon Whiteson. Multitask soft option learning. *arXiv preprint arXiv:1904.01033*, 2019.

[33] Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep variational reinforcement learning for pomdps. *arXiv preprint arXiv:1806.02426*, 2018.

[34] Tommi Jaakkola, Michael I Jordan, and Satinder P Singh. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems*, pages 703–710, 1994.

[35] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

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

[37] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[38] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[39] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[40] Tuan Anh Le, Maximilian Igl, Tom Rainforth, Tom Jin, and Frank Wood. Auto-encoding sequential monte carlo. *arXiv preprint arXiv:1705.10306*, 2017.

[41] Felix Leibfried, Jordi Grau-Moya, and Haitham Bou-Ammar. An information-theoretic optimality principle for deep reinforcement learning. *arXiv preprint arXiv:1708.01867*, 2017.

[42] Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

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

[44] Michael L Littman. Friend-or-foe q-learning in general-sum games.

[45] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in neural information processing systems*, pages 2378–2386, 2016.

[46] Elita Lobo and Scott Jordan. Soft options critic. *arXiv preprint arXiv:1905.11222*, 2019.

[47] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, 2017.

[48] David JC MacKay and David JC Mac Kay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[49] Chris J Maddison, John Lawson, George Tucker, Nicolas Heess, Mohammad Norouzi, Andriy Mnih, Arnaud Doucet, and Yee Teh. Filtering variational objectives. In *Advances in Neural Information Processing Systems*, pages 6573–6583, 2017.

[50] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pages 7611–7622, 2019.

[51] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[52] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[53] Daniel Murfet. Lecture 14 : Banach fixed point theorem, Oct 2019.

[54] Christian A Naesseth, Scott W Linderman, Rajesh Ranganath, and David M Blei. Variational sequential monte carlo. *arXiv preprint arXiv:1705.11140*, 2017.

[55] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[56] John F Nash et al. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

[57] Ashutosh Nayyar, Aditya Mahajan, and Demosthenis Teneketzis. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Transactions on Automatic Control*, 58(7):1644–1658, 2013.

[58] Pedro A Ortega and Daniel A Braun. Thermodynamics as a theory of decision-making with information-processing costs. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 469(2153):20120683, 2013.

[59] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

[60] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[61] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

[62] I Rezek, SJ Roberts, A Rogers, RK Dash, and N Jennings. Unifying learning in games and graphical models. In *2005 7th International Conference on Information Fusion*, volume 2, pages 6–pp. IEEE, 2005.

[63] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[64] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[65] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

[67] Seymour Sherman et al. Solomon kullback, information theory and statistics. *Bulletin of the American Mathematical Society*, 66(6):472–472, 1960.

[68] Pavel Shvechikov, Alexander Grishin, Arsenii Kuznetsov, Alexander Fritzler, and Dmitry Vetrov. Joint belief tracking and reward optimization through approximate inference.

[69] David Silver. Ucl course on rl.

[70] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. 2014.

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

[72] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

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

[74] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[75] Zheng Tian, Ying Wen, Zhichen Gong, Faiz Punakkath, Shihao Zou, and Jun Wang. A regularized opponent model with maximum entropy objective. *arXiv preprint arXiv:1905.08087*, 2019.

[76] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[77] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3540–3549. JMLR. org, 2017.

[78] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[79] Ying Wen, Yaodong Yang, Rui Lu, and Jun Wang. Multi-agent generalized recursive reasoning. *arXiv preprint arXiv:1901.09216*, 2019.

[80] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and Wei Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint arXiv:1901.09207*, 2019.

[81] Lantao Yu, Jiaming Song, and Stefano Ermon. Multi-agent adversarial inverse reinforcement learning. *arXiv preprint arXiv:1907.13220*, 2019.

[82] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

Appendix A

Chapter 2: Appendix

A.1 Probabilistic Machine Learning

A.1.1 ELBO from KL-divergence

We will start with the initial equation that we get and then expand from that

$$
\begin{aligned}
-\int q_\phi(\theta) \log\left(\frac{q_\phi(\theta)}{P(\theta|X,Y)}\right)\ \mathrm{d}\theta &= \int q_\phi(\theta) \log\left(\frac{P(\theta|X,Y)}{q_\phi(\theta)}\right)\ \mathrm{d}\theta \\
&= \int q_\phi(\theta) \log\left(\frac{P(Y|X,\theta)P(\theta)}{P(Y|X)q_\phi(\theta)}\right)\ \mathrm{d}\theta \\
&= \int q_\phi(\theta) \log\left(\frac{P(\theta)}{q_\phi(\theta)}\right) + q_\phi(\theta)\log P(Y|X,\theta) - q_\phi(\theta)\log P(Y|X)\ \mathrm{d}\theta \\
&= \int q_\phi(\theta)\log\left(P(\theta|X,Y)\right)\ \mathrm{d}\theta - \int q_\phi(\theta)\log\left(\frac{q_\phi(\theta)}{P(\theta)}\right)\ \mathrm{d}\theta - \text{const.} \\
&= \mathbb{E}_{q_\phi(\theta)}\left[P(Y|X,\theta)\right] - D_{\mathrm{KL}}\left(q_\phi(\theta)\middle\|P(\theta)\right) - \text{const.}
\end{aligned}
$$

A.1.2 ELBO from Jensen's inequality

We will start with the log-likelihood

$$
l(X,Y) = \log\left(\int P(Y,\theta|X)\ \mathrm{d}\theta\right) \tag{A.1}
$$

Now we will introduce the variational distribution $q_\phi(\theta)$ that we would like to estimate

$$
\begin{aligned}
l(X,Y) = \log\left(\int P(Y,\theta|X)\ \mathrm{d}\theta\right) &= \log\left(\int P(Y,\theta|X)\frac{q_\phi(\theta)}{q_\phi(\theta)}\ \mathrm{d}\theta\right) \\
&\geq \int q_\phi(\theta)\log\left(\frac{P(Y,\theta|X)}{q_\phi(\theta)}\right)\ \mathrm{d}\theta
\end{aligned}
$$

Since log function is concave, we can use Jensen's inequality to derived the lower bounded on the log-likelihood.

A.1.3 Gap between ELBO and log-likelihood

We will show that the sum between the ELBO and KL-divergence between $P(\theta|X,Y)$ and $q_\phi(\theta)$ is indeed equal to the log-likelihood.

$$\int q_\phi(\theta) \log\left(\frac{P(Y,\theta|X)}{q_\phi(\theta)}\right) \, \mathrm{d}\theta + \int q_\phi(\theta) \log\left(\frac{q_\phi(\theta)}{P(\theta|X,Y)}\right) \, \mathrm{d}\theta$$

Starting by expanding the integral and algebraic manipulation, we can see that this is equal to

$$\begin{aligned}
&\int q_\phi(\theta) \log\left(\frac{P(Y,\theta|X)}{q_\phi(\theta)}\right) + q_\phi(\theta) \log\left(\frac{q_\phi(\theta)}{P(\theta|X,Y)}\right) \, \mathrm{d}\theta \\
&\quad = \int q_\phi(\theta) \log\left(\frac{P(Y,\theta|X)}{P(\theta|X,Y)}\right) \, \mathrm{d}\theta = \int q_\phi(\theta) \log\left(P(Y|X)\right) \, \mathrm{d}\theta \\
&\quad = \log\left(P(Y|X)\right) \int q_\phi(\theta) \, \mathrm{d}\theta = \log\left(P(Y|X)\right) \cdot 1
\end{aligned}$$

This can also be seen as putting the variational distribution using similar technique in appendix A.1.2.

A.2 Single Agent Reinforcement Learning

A.2.1 Policy Guarantee Improvement

We start with the definition of improved policy:

$$\pi'(a|s) = \begin{cases} 1 & \text{if } a = \arg\max\limits_{a \in A} Q^\pi(s,a) \\ 0 & \text{otherwise} \end{cases}$$

We have the following sequence of substitutions

$$\begin{aligned}
V^\pi(s) &\leq Q^\pi(s, \pi'(s)) \\
&= r(s,\pi'(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s'|s,a)} \left[V^\pi(s')\right] \\
&\leq r(s,\pi'(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s'|s,a)} \left[Q^\pi(s', \pi'(s'))\right] \\
&\vdots \\
&\leq r(s,\pi'(s)) + \gamma r(s', \pi'(s')) + \gamma^2 r(s'', \pi'(s'')) + \cdots \\
&= V^{\pi'}(s)
\end{aligned}$$

Please note that this improved policy is deterministic, therefore, we can simply substitute the output into the action value function.

A.2.2 Expected Bellman Operator is a Contraction Mapping

Recalling the definition of the Map:

$$\mathcal{B}^\pi V(s) = \mathbb{E}_{a \sim \pi(a|s)} \left[r(s,a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s'|s,a)} \left[V(s')\right]\right] \tag{A.2}$$

Let's consider the differences between 2 value functions $V_1(s)$ and $V_2(s)$ after being applied by the Bellman operator:

$$\begin{aligned}\mathcal{B}^\pi V_1(s) - \mathcal{B}^\pi V_2(s) &= \mathbb{E}_{a\sim\pi(a|s)}\left[\cancel{r(s,a)} + \gamma\mathbb{E}_{s'\sim\mathcal{T}(s'|s,a)}\left[V_1(s')\right] - \cancel{r(s,a)} - \gamma\mathbb{E}_{s'\sim\mathcal{T}(s'|s,a)}\left[V_2(s')\right]\right] \\ &= \gamma\mathbb{E}_{a,s'}\left[V_1(s') - V_2(s')\right]\end{aligned} \tag{A.3}$$

Now, we shall consider the ∞-norm of the equaltion, we have:

$$\begin{aligned}\max_s |\mathcal{B}^\pi V_1(s) - \mathcal{B}^\pi V_2(s)| &= \gamma \max_s |\mathbb{E}_{a,s'}\left[V_1(s') - V_2(s')\right]| \\ &\leq \gamma \max_s \left|\mathbb{E}_{a,s'}\left[\max_s |V_1(s') - V_2(s')|\right]\right| \\ &\leq \gamma \max_s |V_1(s) - V_2(s)|\end{aligned} \tag{A.4}$$

Given this, we finished the proof.

A.3 Control as Inference Framework

A.3.1 ELBO for Reinforcement Learning with KL-divergence

$$\begin{aligned}\mathbb{E}_{q(s_{1:T},a_{1:T})}\left[\log \frac{P(s_{1:T},a_{1:T}|\mathcal{O}_{1:T}=1)}{q(s_{1:T},a_{1:T})}\right] &= \mathbb{E}_{q(s_{1:T},a_{1:T})}\left[\log \frac{P(s_{1:T},a_{1:T},\mathcal{O}_{1:T}=1)}{q(s_{1:T},a_{1:T})P(\mathcal{O}_{1:T}=1)}\right] \\ &= \mathbb{E}_{q(s_{1:T},a_{1:T})}\left[\frac{P(s_{1:T},a_{1:T},\mathcal{O}_{1:T}=1)}{q(s_{1:T},a_{1:T})}\right] + \text{const.}\end{aligned}$$

Now, we can consider the ratio of joint probability

$$\begin{aligned}&\log \frac{\cancel{P(s_0)}\prod_{t=1}^T \cancel{P(s_{t+1}|s_t,a_t)}P_{\text{prior}}(a_t|s_t)P(\mathcal{O}_t=1|s_t,a_t)}{\cancel{P(s_0)}\prod_{t=1}^T \cancel{P(s_{t+1}|s_t,a_t)}\pi(a_t|s_t)} \\ &= \sum_{t=1}^T \log P(\mathcal{O}_t=1|s_t,a_t) - \frac{\pi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)} \\ &= \sum_{t=1}^T \beta r(s_t,a_t) - \log \frac{\pi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)}\end{aligned}$$

Plugging this back in yields the answer.

A.3.2 ELBO for Reinforcement Learning with Jensen's inequality

Starting, when we want the to maximize the log-probability of optimality

$$\begin{aligned}\log P(\mathcal{O}_{1:T}=1) &= \int P(s_{1:T},a_{1:T},\mathcal{O}_{1:T}=1)\ \mathrm{d}s_{1:T}\ \mathrm{d}a_{1:T} \\ &= \int P(s_{1:T},a_{1:T},\mathcal{O}_{1:T}=1)\frac{q(s_{1:T},a_{1:T})}{q(s_{1:T},a_{1:T})}\ ds_{1:T}\ da_{1:T} \\ &\geq \mathbb{E}_{q(s_{1:T},a_{1:T})}\left[\log \frac{P(s_{1:T},a_{1:T},\mathcal{O}_{1:T}=1)}{q(s_{1:T},a_{1:T})}\right] \\ &= \mathbb{E}_q\left[\sum_{t=1}^T \beta r(s_t,a_t) - \log \frac{\pi(a_t|s_t)}{P_{\text{prior}}(a_t|s_t)}\right]\end{aligned}$$

which is the same as ELBO for KL-divergence.

A.3.3 Soft Bellman Operator is a Contraction Mapping

This has been proven in [23], but we will expand some of the details in this section for completeness. Let's first formally define the map $\mathcal{B}$ to be:

$$\mathcal{B}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p_s} \left[\log \int \exp(Q(s_{t+1}, a')) P_{\text{prior}}(a'|s_{t+1}) \, da' \right] \tag{A.5}$$

Let's consider the infinite norm and we want to show that

$$\left\| \mathcal{B}^\pi Q_1 - \mathcal{B}^\pi Q_2 \right\|_\infty \leq \gamma \left\| Q_1 - Q_2 \right\|_\infty \tag{A.6}$$

Let's start with just substitute the Bellman operator, we will first won't expand the V to save the space, and we will expand on it later on.

$$\begin{aligned}
\left\| \mathcal{B}^\pi Q_1 - \mathcal{B}^\pi Q_2 \right\|_\infty &= \left\| r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p_s} \left[V_1(s_{t+1}) \right] - r(s_t, a_t) - \gamma \mathbb{E}_{s_{t+1} \sim p_s} \left[V_2(s_{t+1}) \right] \right\|_\infty \\
&= \gamma \max_{s_t, a_t} \left| \mathbb{E}_{s_{t+1} \sim p_s} \left[V_1(s_{t+1}) - V_2(s_{t+1}) \right] \right| \\
&\leq \gamma \max_{s_t, a_t} \mathbb{E}_{s_{t+1} \sim p_s} \left| V_1(s_{t+1}) - V_2(s_{t+1}) \right|
\end{aligned}$$

Now we would want to show that $|V_1(s_{t+1}) - V_2(s_{t+1})| \leq \varepsilon$ where $\varepsilon = \left\| Q_1 - Q_2 \right\|_\infty$. Now there are 2 cases we can consider: **In the fist case**, we have that

$$\begin{aligned}
\log \int \exp Q_1(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' &\leq \log \int_{\mathcal{A}} \exp \big(Q_2(\mathbf{s}_{t+1}, \mathbf{a}') + \varepsilon \big) P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' \\
&= \log \int \exp \varepsilon \cdot \exp Q_2(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' \\
&= \varepsilon + \log \int \exp Q_2(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}'
\end{aligned}$$

Finally we have that

$$\log \int \exp Q_1(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' - \log \int \exp Q_2(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' \leq \varepsilon$$

In the second case, we have that:

$$\begin{aligned}
\log \int \exp Q_1(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' &\geq \log \int_{\mathcal{A}} \exp \big(Q_2(\mathbf{s}_{t+1}, \mathbf{a}') - \varepsilon \big) P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' \\
&= \log \int \exp(-\varepsilon) \cdot \exp Q_2(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' \\
&= -\varepsilon + \log \int \exp Q_2(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}'
\end{aligned}$$

And so we have

$$\log \int \exp Q_1(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' - \log \int \exp Q_2(\mathbf{s}_{t+1}, \mathbf{a}') P_{\text{prior}}(\mathbf{a}'|\mathbf{s}_{t+1}) d\mathbf{a}' \geq -\varepsilon$$

Therefore we can concluded that $|V_1(s_{t+1}) - V_2(s_{t+1})| \leq \varepsilon$ since $V_1(s_{t+1}) - V_2(s_{t+1}) \leq \epsilon$ and $V_1(s_{t+1}) - V_2(s_{t+1}) \geq -\epsilon$. Plug this into the the inequality as

$$\begin{aligned}\left\|\mathcal{B}^{\pi}Q_1 - \mathcal{B}^{\pi}Q_2\right\|_{\infty} &\leq \gamma \max_{s_t,a_t} \mathbb{E}_{s_{t+1}\sim p_s}\left|V_1(s_{t+1}) - V_2(s_{t+1})\right| \\ &\leq \gamma \left\|Q_1 - Q_2\right\|_{\infty}\end{aligned}$$

We can see that the soft-Bellman operator is a contraction mapping. Therefore, by repeatedly apply this mapping will make the action value function reaches the fixed point.

A.3.4 Soft Policy Update Improvement

This has been proven in [24], but we will expand some of the details in this section for completeness, since this is going to be useful in the latter chapters. Let arbitrary policy π_+ be

$$J_{\pi_{\text{old}}}(\pi_+) = D_{\text{KL}}\left(\pi_+(\cdot|s_t)\middle\|\frac{\exp(Q^{\pi_{\text{old}}}(s_t,\cdot))P_{\text{prior}}(\cdot|s_t)}{\exp Z^{\pi_{\text{old}}}(s_t)}\right) \tag{A.7}$$

Please note that $Z^{\pi_{\text{old}}}(s_t)$ isn't equal to $V^{\pi_{\text{old}}}(s_t)$ because π_{old} is an arbitrary policy, and it converges when the policy satisfies energy based form. We can show that by the definition of π_{new} that $J_{\pi_{\text{old}}}(\pi_{\text{old}}) \geq J_{\pi_{\text{old}}}(\pi_{\text{new}})$, given this we can expand the definition of KL-divergence above giving us the following:

$$\begin{aligned}&\mathbb{E}_{\pi_{\text{old}}}\left[\cancel{Z^{\pi_{\text{old}}}(s_t)} - Q^{\pi_{\text{old}}}(s_t,\cdot) - \log P_{\text{prior}}(\cdot|s_t) + \log \pi_{\text{old}}(\cdot|s_t)\right] \\ &\quad\geq \mathbb{E}_{\pi_{\text{new}}}\left[\cancel{Z^{\pi_{\text{old}}}(s_t)} - Q^{\pi_{\text{old}}}(s_t,\cdot) - \log P_{\text{prior}}(\cdot|s_t) + \log \pi_{\text{new}}(\cdot|s_t)\right]\end{aligned} \tag{A.8}$$

This would gives us

$$\begin{aligned}&-V^{\pi_{\text{old}}}(s_t) \geq \mathbb{E}_{\pi_{\text{new}}}\left[-Q^{\pi_{\text{old}}}(s_t,\cdot) - \log P_{\text{prior}}(\cdot|s_t) + \log \pi_{\text{new}}(\cdot|s_t)\right] \\ \iff &V^{\pi_{\text{old}}}(s_t) \leq \mathbb{E}_{\pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}(s_t,\cdot) - \log \frac{\pi_{\text{new}}(\cdot|s_t)}{P_{\text{prior}}(\cdot|s_t)}\right]\end{aligned} \tag{A.9}$$

Now, we can consider the recursive definition of Q learning as follows:

$$\begin{aligned}Q^{\pi_{\text{old}}} &= r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}}\left[V^{\pi_{\text{old}}}(s_{t+1})\right] \\ &\leq r(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}}\left[\mathbb{E}_{\pi_{\text{new}}}\left[Q^{\pi_{\text{old}}}(s_{t+1},\cdot) - \log \frac{\pi_{\text{new}}(\cdot|s_{t+1})}{P_{\text{prior}}(\cdot|s_{t+1})}\right]\right] \\ &\vdots \\ &\leq Q^{\pi_{\text{new}}}\end{aligned} \tag{A.10}$$

Appendix B

Chapter 3: Appendix

B.1 ROMMEO

B.1.1 ROMMEO ELBO derivation

Starting off with the KL-divergence (we have shown that this is equivalent to ELBO derived from Jensen's inequality) and we expand by doing a algebraic manipulation to get the following:

$$
\begin{aligned}
& D_{\mathrm{KL}}\left(q(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T}) \middle\| P(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T} | \mathcal{O}^i_{1:T} = 1, \mathcal{O}^{-i}_{1:T} = 1)\right) \\
=& \mathbb{E}_q \left[\log \frac{q(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T}) P(\mathcal{O}^i_{1:T} = 1, \mathcal{O}^{-i}_{1:T} = 1)}{P(s_{1:T}, a^i_{1:T}, a^{-i}_{1:T} | \mathcal{O}^i_{1:T} = 1, \mathcal{O}^{-i}_{1:T} = 1)}\right] \\
=& \mathbb{E}_q \left[\log \frac{P(\mathcal{O}^i_{1:T} = 1, \mathcal{O}^{-i}_{1:T} = 1) \cancel{P(s_0)} \prod_{t=0}^{T} \pi_\theta(a^i_t | s_t, a^{-i}_t) \rho_\phi(a^{-i}_t | s_t) \cancel{P(s_{t+1} | s_t, a^i_t, a^{-i}_t)}}{\cancel{P(s_0)} \prod_{t=0}^{T} P_{\text{prior}}(a^i_t | s_t, a^{-i}_t) P_{\text{prior}}(a^{-i}_t | s_t) \cancel{P(s_{t+1} | s_t, a^i_t, a^{-i}_t)} P(\mathcal{O}^i_t = 1, \mathcal{O}^{-i}_t = 1 | s_t, a^i_t, a^{-i}_t)}\right] \\
=& \mathbb{E}_q \left[\log P(\mathcal{O}^i_{1:T} = 1, \mathcal{O}^{-i}_{1:T} = 1) + \sum_{t=0}^{T} -\beta R(s_t, a^i_t, a^{-i}_t) + \log \frac{\pi_\theta(a^i_t | s_t, a^{-i}_t)}{P_{\text{prior}}(a^i_t | s_t, a^{-i}_t)} + \log \frac{\rho_\phi(a^{-i}_t | s_t)}{P_{\text{prior}}(a^{-i}_t | s_t)}\right] \\
=& -\mathbb{E}_q \left[\sum_{t=0}^{T} \beta R(s_t, a^i_t, a^{-i}_t) - \log \frac{\pi_\theta(a^i_t | s_t, a^{-i}_t)}{P_{\text{prior}}(a^i_t | s_t, a^{-i}_t)} - \log \frac{\rho_\phi(a^{-i}_t | s_t)}{P_{\text{prior}}(a^{-i}_t | s_t)}\right] - \text{const.}
\end{aligned}
$$

B.1.2 ROMMEO Optimal Agent's Policy

We start by expanding the equation and separate the opponent's model. We are going to see that the objective is equivalent to minimizing the KL-divergence, which by Gibbs' inequality, we can imply that the optimal agent's policy can be found by setting the agent's policy to be equal to the quantity in KL-divergence.

$$
\begin{aligned}
& \mathbb{E}_q \left[\beta R(s_T, a^i_T, a^{-i}_T) - \log \frac{\pi_\theta(a^i_T | s_T, a^{-i}_T)}{P_{\text{prior}}(a^i_T | s_T, a^{-i}_T)} - \log \frac{\rho_\phi(a^{-i}_T | s_T)}{P_{\text{prior}}(a^{-i}_T | s_T)} + Q(s_T, a^{-i}_T) - Q(s_T, a^{-i}_T)\right] \\
=& \mathbb{E}_{q(s_T, a^{-i}_T)} \Bigg[\underbrace{\mathbb{E}_{\pi_\theta} \left[\beta R(s_T, a^i_T, a^{-i}_T) - \log \frac{\pi_\theta(a^i_T | s_T, a^{-i}_T)}{P_{\text{prior}}(a^i_T | s_T, a^{-i}_T)} - Q(s_T, a^{-i}_T)\right]}_{①} - \log \frac{\rho_\phi(a^{-i}_T | s_T)}{P_{\text{prior}}(a^{-i}_T | s_T)} + Q(s_T, a^{-i}_T)\Bigg]
\end{aligned}
$$

We will consider the only part (1) as it depends only on the optimal policy.

$$
\begin{aligned}
&\mathbb{E}_{\pi_\theta(a_T^i|s_T,a_T^{-i})}\left[\beta R(s_T,a_T^i,a_T^{-i}) - \log\frac{\pi_\theta(a_T^i|s_T,a_T^{-i})}{P_{\text{prior}}(a_T^i|s_T,a_T^{-i})} - Q(s_T,a_T^{-i})\right]\\
=&\mathbb{E}_{\pi_\theta(a_T^i|s_T,a_T^{-i})}\left[\log\frac{\exp\left(\beta R(s_T,a_T^i,a_T^{-i})\right)P_{\text{prior}}(a_T^i|s_T,a_T^{-i})}{\pi_\theta(a_T^i|s_T,a_T^{-i})\exp\left(Q(s_T,a_T^{-i})\right)}\right]\\
=&-D_{\text{KL}}\left(\pi_\theta(a_T^i|s_T,a_T^{-i})\middle\|\frac{\exp\left(\beta R(s_T,a_T^i,a_T^{-i})P_{\text{prior}}(a_T^i|s_T,a_T^{-i})\right)}{\exp\left(Q(s_T,a_T^{-i})\right)}\right)
\end{aligned}
$$

We can maximize this value by minimizing the KL-divergence. And we can see that, we can set $Q(s_T,a_T^{-i})$ to be normalizing constant. Thus finishes the proof.

B.1.3 ROMMEO Optimal Opponent's Policy

Similar proving process to the Optimal agent's policy. We will start by plugging the optimal agent's policy into the equation first, then we can use the KL-divergence trick.

$$
\begin{aligned}
&\mathbb{E}_q\left[\beta R(s_T,a_T^i,a_T^{-i}) - \log\frac{\pi_\theta(a_T^i|s_T,a_T^{-i})}{P_{\text{prior}}(a_T^i|s_T,a_T^{-i})} - \log\frac{\rho_\phi(a_T^{-i}|s_T)}{P_{\text{prior}}(a_T^{-i}|s_T)}\right]\\
=&\mathbb{E}_q\left[\beta R(s_T,a_T^i,a_T^{-i}) - \log\frac{\exp\left(\beta R(s_T,a_T^i,a_T^{-i})\right)\cancel{P_{\text{prior}}(a_T^i|s_T,a_T^{-i})}}{\cancel{P_{\text{prior}}(a_T^i|s_T,a_T^{-i})}\exp\left(Q(s_T,a_T^{-i})\right)} - \log\frac{\rho_\phi(a_T^{-i}|s_T)}{P_{\text{prior}}(a_T^{-i}|s_T)}\right]\\
=&\mathbb{E}_q\left[\log\left(\exp\left(Q(s_T,a_T^{-i})\right)\right) + \cancel{\beta R(s_T,a_T^i,a_T^{-i})} - \cancel{\log\left(\exp\left(\beta R(s_T,a_T^i,a_T^{-i})\right)\right)} - \log\frac{\rho_\phi(a_T^{-i}|s_T)}{P_{\text{prior}}(a_T^{-i}|s_T)}\right]\\
=&\mathbb{E}_{\rho_\phi(a_T^{-i}|s_T)P(s_T)}\left[Q(s_T,a_T^{-i}) - \log\frac{\rho_\phi(a_T^{-i}|s_T)}{P_{\text{prior}}(a_T^{-i}|s_T)}\right]\\
=&\mathbb{E}_{\rho_\phi(a_T^{-i}|s_T)P(s_T)}\left[Q(s_T,a_T^{-i}) - \log\frac{\rho_\phi(a_T^{-i}|s_T)}{P_{\text{prior}}(a_T^{-i}|s_T)} + V(s_T) - V(s_T)\right]\\
=&\mathbb{E}_{\rho_\phi(a_T^{-i}|s_T)P(s_T)}\left[\log\frac{\exp\left(Q(s_T,a_T^{-i})\right)P_{\text{prior}}(a_T^{-i}|s_T)}{\rho_\phi(a_T^{-i}|s_T)\exp\left(V(s_T)\right)}\right] + \mathbb{E}_{P(s_T)}\left[V(s_T)\right]
\end{aligned}
$$

We will consider only the LHS of the equation, which is equal to the KL-divergence (technically expected KL-divergence since we also find expectation over state s_T)

$$
-D_{\text{KL}}\left(\rho_\phi(a_T^{-i}|s_T)\middle\|\frac{\exp\left(Q(s_T,a_T^{-i})\right)P_{\text{prior}}(a_T^{-i}|s_T)}{\exp\left(V(s_T)\right)}\right)
$$

This finishes the proof (where $V(s_T)$ is the normalizing factor). Similarly, we can replace $Q(s,a_T^{-i})$ with other functions and the result will be in the similar form.

B.2 Balancing-Q

B.2.1 Balancing-Q Opponent Model derivation

We start with the opponent model:

$$
\begin{aligned}
&\mathbb{E}_{P(s_T)P(a_T,a_T^{-i}|s_T)}\left[R(s_T,a_T,a_T^{-i})-\frac{1}{\beta^i}\log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)}-\frac{1}{\beta^{-i}}\log\frac{\rho_\phi(a_T^{-i}|s_T)}{P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}\right]\\
=&\mathbb{E}_{P(s_T,a_T^i)}\Bigg[\underbrace{\mathbb{E}_{\rho(a^{-i}|s_T,a_T^i)}\left[R(s_T,a_T,a_T^{-i})-\frac{1}{\beta^{-i}}\log\frac{\rho(a^{-i}|s_T,a_T^i)}{P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}\right]}_{①}-\frac{1}{\beta^i}\log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)}\Bigg]
\end{aligned}
$$

We will consider the quality ①, which is equivalent to KL-divergence as we introduce the normalizing factor $Q^{-i}(s_T,a_T^i)$ and move β^{-i} to the reward:

$$
\begin{aligned}
&\mathbb{E}_{\rho(a^{-i}|s_T,a_T^i)}\left[R(s_T,a_T,a_T^{-i})-\frac{1}{\beta^{-i}}\log\frac{\rho(a_T^{-i}|s_T,a_T^i)}{P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}\right]\\
=&\mathbb{E}_{\rho(a_T^{-i}|s_T,a_T^i)}\left[\beta^{-i}R(s_T,a_T,a_T^{-i})-\log\frac{\rho(a_T^{-i}|s_T,a_T^i)}{P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}-Q^{-i}(s_T,a_T^i)+Q^{-i}(s_T,a_T^i)\right]\\
=&\mathbb{E}_{\rho(a^{-i}|s_T,a_T^i)}\left[\log\frac{\exp\left(\beta^{-i}R(s_T,a_T,a_T^{-i})\right)P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}{\rho(a_T^{-i}|s_T,a_T^i)\exp\left(Q^{-i}(s,a^i)\right.}\right]+\mathbb{E}_{\rho(a_T^{-i}|s_T,a_T^i)}\left[Q^{-i}(s_T,a_T^i)\right]\\
=&-D_{\text{KL}}\left(\rho(a_T^{-i}|s_T,a_T^i)\middle\|\frac{\exp\left(\beta^{-i}R(s_T,a_T,a_T^{-i})\right)P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}{\exp\left(Q^{-i}(s_T,a_T^i)\right)}\right)+Q^{-i}(s_T,a_T^i)
\end{aligned}
$$

We can simply minimizing the KL-divergence error to get the policy.

B.2.2 Balancing-Q agent's objective and optimal value

Let's plug this into the original objective to find the optimal policy

$$
\begin{aligned}
&\mathbb{E}_{P(s_T,a_T,a_T^{-i})}\left[R(s_T,a_T,a_T^{-i})-\frac{1}{\beta^i}\log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)}-\frac{1}{\beta^{-i}}\log\frac{\exp\left(\beta^{-i}R(s_T,a_T,a_T^{-i})\right)\cancel{P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}}{\cancel{P_{\text{prior}}(a_T^{-i}|s_T,a_T^i)}\exp\left(Q^{-i}(s_T,a_T^i)\right)}\right]\\
=&\mathbb{E}_{P(s_T,a_T,a_T^{-i})}\left[\frac{1}{\beta^{-i}}Q^{-i}(s_T,a_T^i)-\frac{1}{\beta^i}\log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)}\right]\\
=&\mathbb{E}_{P(s_T,a_T,a_T^{-i})}\Bigg[\underbrace{\frac{\beta^i}{\beta^{-i}}Q^{-i}(s_T,a_T^i)}_{Q^i(s_T,a_T^i)}-\log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)}\Bigg]
\end{aligned}
$$

Using the same method, we can we get the final agent's policy by minimizing the KL-divergence

$$
\begin{aligned}
&\mathbb{E}_{P(s_T,a_T)}\left[Q^i(s_T,a_T^i)-\log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)}\right]\\
=&\mathbb{E}_{P(s_T,a_T)}\left[Q^i(s_T,a_T^i)-\log\frac{\pi_\theta(a_T^i|s_T)}{P_{\text{prior}}(a_T^i|s_T)}+V^i(s_T)-V^i(s_T)\right]\\
=&\mathbb{E}_{P(s_T,a_T)}\left[\log\frac{\exp\left(Q^i(s_T,a_T^i)\right)P_{\text{prior}}(a_T^i|s_T)}{\pi_\theta(a_T^i|s_T)\exp\left(V^i(s_T)\right)}\right]+\mathbb{E}_{P(s_T)}\left[V^i(s_T)\right]\\
=&-D_{\text{KL}}\left(\pi_\theta(a_T^i|s_T)\middle\|\frac{\exp\left(Q^i(s_T,a_T^i)\right)P_{\text{prior}}(a_T^i|s_T)}{\exp\left(V^i(s_T)\right)}\right)+\mathbb{E}_{P(s_T)}\left[V^i(s_T)\right]
\end{aligned}
$$

This can be maximize by setting agent policy to appropriate value.

B.2.3 Recursive Definition of action value function for Balancing-Q

Now consider the Bellman equation and its expansion:

$$
\begin{aligned}
Q^{-i,\pi,\rho}(s_t, a_t^i, a_t^{-i}) &= r(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}^i, a_{t+1}^{-i} \sim \rho} \Bigg[-\frac{1}{\beta^i} \log \frac{\pi(a_{t+1}^i | s_{t+1})}{P_{\text{prior}}(a_{t+1}^i | s_{t+1})} \\
&\quad - \frac{1}{\beta^{-i}} \log \frac{\rho(a_{t+1}^{-i} | s_{t+1}, a_{t+1}^i)}{P_{\text{prior}}(a_{t+1}^{-i} | s_{t+1}, a_{t+1}^i)} + Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) \Bigg] \\
&\geq r(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}^i, a_{t+1}^{-i} \sim \tilde{\rho}} \Bigg[-\frac{1}{\beta^i} \log \frac{\pi(a_{t+1}^i | s_{t+1}, a_{t+1}^{-i})}{P_{\text{prior}}(a_{t+1}^i | s_{t+1}, a_{t+1}^{-i})} \\
&\quad - \frac{1}{\beta^{-i}} \log \frac{\tilde{\rho}(a_{t+1}^{-i} | s_{t+1}, a_{t+1}^i)}{P_{\text{prior}}(a_{t+1}^{-i} | s_{t+1}, a_{t+1}^i)} + Q^{-i,\pi,\rho}(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) \Bigg] \\
&= r(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}^i, a_{t+1}^{-i} \sim \tilde{\rho}} \Bigg[-\frac{1}{\beta^i} \log \frac{\pi(a_{t+1}^i | s_{t+1}, a_{t+1}^{-i})}{P_{\text{prior}}(a_{t+1}^i | s_{t+1}, a_{t+1}^{-i})} \\
&\quad - \frac{1}{\beta^{-i}} \log \frac{\tilde{\rho}(a_{t+1}^{-i} | s_{t+1}, a_{t+1}^i)}{P_{\text{prior}}(a_{t+1}^{-i} | s_{t+1}, a_{t+1}^i)} + r(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) + \\
&\quad \gamma \mathbb{E}_{s_{t+2}, a_{t+2}^i, a_{t+2}^{-i} \sim \rho} \Bigg[-\frac{1}{\beta^i} \log \frac{\pi(a_{t+2}^i | s_{t+2})}{P_{\text{prior}}(a_{t+2}^i | s_{t+2})} \\
&\quad - \frac{1}{\beta^{-i}} \log \frac{\rho(a_{t+2}^{-i} | s_{t+2}, a_{t+2}^i)}{P_{\text{prior}}(a_{t+2}^{-i} | s_{t+2}, a_{t+2}^i)} + Q^{-i,\pi,\rho}(s_{t+2}, a_{t+2}^i, a_{t+2}^{-i}) \Bigg] \Bigg] \\
&\geq r(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E}_{s_{t+1}, a_{t+1}^i, a_{t+1}^{-i} \sim \tilde{\rho}} \Bigg[-\frac{1}{\beta^i} \log \frac{\pi(a_{t+1}^i | s_{t+1}, a_{t+1}^{-i})}{P_{\text{prior}}(a_{t+1}^i | s_{t+1}, a_{t+1}^{-i})} \\
&\quad - \frac{1}{\beta^{-i}} \log \frac{\tilde{\rho}(a_{t+1}^{-i} | s_{t+1}, a_{t+1}^i)}{P_{\text{prior}}(a_{t+1}^{-i} | s_{t+1}, a_{t+1}^i)} + r(s_{t+1}, a_{t+1}^i, a_{t+1}^{-i}) + \\
&\quad \gamma \mathbb{E}_{s_{t+2}, a_{t+2}^i, a_{t+2}^{-i} \sim \tilde{\rho}} \Bigg[-\frac{1}{\beta^i} \log \frac{\pi(a_{t+2}^i | s_{t+2})}{P_{\text{prior}}(a_{t+2}^i | s_{t+2})} \\
&\quad - \frac{1}{\beta^{-i}} \log \frac{\tilde{\rho}(a_{t+2}^{-i} | s_{t+2}, a_{t+2}^i)}{P_{\text{prior}}(a_{t+2}^{-i} | s_{t+2}, a_{t+2}^i)} + Q^{-i,\pi,\rho}(s_{t+2}, a_{t+2}^i, a_{t+2}^{-i}) \Bigg] \Bigg] \\
&\vdots \\
&\geq Q^{\pi,\tilde{\rho}}(s_t, a_t^i, a_t^{-i})
\end{aligned}
$$

This proves the inequality. Note that in the case of $\beta^{-i} > 0$ the proof is exactly the same except the inequality direction.

B.3 Experimentation

B.3.1 Matrix Game Detailed Experiment

In this experiment, we use the learning rate α of 0.6, given the agent i and agent j's β value to both 0.1 in cooperative setting, and assign the value 0.1 and -0.1, respectively, in competitive setting. The reward in each time step is calculated by averaging 50 rollouts. Finally, the temperature is

scheduled to increase exponentially i.e ω^t, for each time step t, where ω is the scheduling weight and it is equal to 1.01 for all experiments. The pseudo-code is equal the following:

Algorithm 1 Balancing Q-Learning in Matrix Game for agent i

1: **Input**: β^{agent}, β^{opp}

2: Initialized the following function $Q(a^i, a^j) = 0$, $P_{\text{prior}}(a^j|a^i) = \mathcal{U}$ and $P_{\text{prior}}(a^i) = \mathcal{U}$ where $\mathcal{U}$ is a uniform distribution.

3: **for** $i = 1, 2, \cdots, T$ **do**

4: Calculated agent's q-function based of opponent's model

$$Q(a^i) = \frac{\beta^{\text{agent}}}{\beta^{\text{opp}}} \cdot \log \sum_{a^j} \exp(\beta^{\text{opp}} \cdot \omega \cdot Q(a^{\text{agent}}, a^j)) P_{\text{prior}}(a^j|a^i)$$

5: Calculate agent's policy

$$\pi(a^i) = \frac{P_{\text{prior}}(a^i) \cdot Q(a^i)}{\sum_{a^i} P_{\text{prior}}(a^i) \cdot Q(a^i)} \tag{B.1}$$

6: Acts on the environment

7: Given common reward r with opponent's action a^j and agent's own action a^i update the agent's action value function

$$Q(a^i, a^j) \leftarrow Q(a^i, a^j) + \alpha(r - Q(a^i, a^j))$$

8: Update ω

$$\omega \leftarrow \omega * 1.01$$

9: **end for**

The code for the experiment will be in `examples/experiments/` in the attached zip file.

Appendix C

Chapter 4: Appendix

C.1 Solution to Single agent Soft-Hierarchical Reinforcement Learning

Consider the last time step of the optimization problem:

$$\mathbb{E}_{s_T,z_T,b_T,z_{T-1}}\left[\beta r(s_T,a_T)-\log\frac{\pi_\theta(a_T|s_T,z_T)}{P_{\text{prior}}(a_T|s_T,z_T)}-\log\frac{\pi_\phi^H(z_T|s_T,z_{T-1},b_T)}{P_{\text{prior}}^H(z_T|s_T,z_{T-1},b_T)}-\log\frac{\pi_\phi^T(b_T|s_T,z_{T-1})}{P_{\text{prior}}^T(b_T|s_T,z_{T-1})}\right]$$

We can solve the lower-level policy first (similar to multi-agent where we optimize the conditional polity first i.e $\pi(a^i|a^{-i},s)$). We have the following equation

$$\begin{aligned}\pi_\theta(a_T|s_T,z_T)&=\frac{P_{\text{prior}}(a_T|s_T,z_T)\exp\left(Q(s_T,a_T,z_T)\right)}{\exp V(s_T,z_T)}\\ \text{where } Q(s_T,a_T,z_T)&=\beta r(s_T,a_T)\\ V(s_T,z_T)&=\log\int P_{\text{prior}}(a_T|s_T,z_T)\exp\left(Q(s_T,a_T,z_T)\right)\ \mathrm{d}a_T\end{aligned}\tag{C.1}$$

This leads to the objective, which we can split between the case where agent still uses z_{T-1} and the case where we still have to generate z_T from the state.

$$\begin{aligned}b_T&\left[V(s_T,z_T)-\log\frac{\pi_\phi^H(z_T|s_T)}{P_{\text{prior}}(z_T|s_T)}-\log\frac{\pi_\phi^T(b_T|s_T,z_{T-1})}{P_{\text{prior}}^T(b_T|s_T,z_{T-1})}\right]\\ &+(1-b_T)\left[V(s_T,z_{T-1})-\log\frac{\pi_\phi^T(1-b_T|s_T,z_{T-1})}{P_{\text{prior}}^T(1-b_T|s_T,z_{T-1})}\right]\end{aligned}\tag{C.2}$$

We can now consider $\pi_\phi^H(z_T|s_T)$, which leads to the following equation:

$$\begin{aligned}\pi^H(z_T|s_T)&=\frac{\frac{1}{m}\exp\left(V(s_T,z_T)\right)}{\exp\left(V(s_T)\right)}\\ \text{where } V(s_T)&=\frac{1}{m}\log\int\exp\left(V(s_T,z_T)\right)\ \mathrm{d}z_T\end{aligned}\tag{C.3}$$

Plugging the optimal master policy into the objective, we have the following objective left:

$$\begin{aligned} & b_T \left[V(s_T) - \log \frac{\pi_\phi^T(b_T|s_T, z_{T-1})}{P_{\text{prior}}^T(b_T|s_T, z_{T-1})} \right] \\ & + (1 - b_T) \left[V(s_T, z_{T-1}) - \log \frac{\pi_\phi^T(1 - b_T|s_T, z_{T-1})}{P_{\text{prior}}^T(1 - b_T|s_T, z_{T-1})} \right] \end{aligned} \tag{C.4}$$

We are left with the termination policy by maximizing the objective:

$$b_T \left[V(s_T) \right] + + (1 - b_T) \left[V(s_T, z_{T-1}) \right] - \log \frac{\pi_\phi^T(b_T|s_T, z_{T-1})}{P_{\text{prior}}^T(b_T|s_T, z_{T-1})} \tag{C.5}$$

We have the following optimal policy for the termination policy, which is:

$$\begin{aligned} \pi^T(b_T|s_T, z_{T-1}) &= \frac{P_{\text{prior}}^T(b_T|s_T, z_{T-1}) \exp U(s_T, z_{T-1}, b_T)}{\exp U(s_T, z_{T-1})} \\ \text{where } U(s_T, z_{T-1}, b_T) &= b_T \left[V^H(s_T) \right] + (1 - b_T) V(s_T, z_{T-1}) \\ U(s_T, z_{T-1}) &= \log \int P_{\text{prior}}^T(b_T|s_T, z_{T-1}) \exp \left(b_T \left[V^H(s_T) \right] + (1 - b_T) V(s_T, z_{T-1}) \right) \, \mathrm{d}b_T \end{aligned} \tag{C.6}$$

Now, we left with $U(s_T, z_{T-1})$ as the message, which we can use it to consider the arbitrary time step, which is:

$$\begin{aligned} \mathbb{E}_{s_t, z_t, b_t, z_{t-1}} \Bigg[& \beta r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}} \left[U(s_{t+1}, z_t) \right] - \log \frac{\pi_\theta(a_t|s_t, z_t)}{P_{\text{prior}}(a_t|s_t, z_t)} \\ & - \log \frac{\pi_\phi^H(z_t|s_t, z_{t-1}, b_t)}{P_{\text{prior}}^H(z_t|s_t, z_{t-1}, b_t)} - \log \frac{\pi_\phi^T(b_t|s_t, z_{t-1})}{P_{\text{prior}}^T(b_t|s_t, z_{t-1})} \Bigg] \end{aligned}$$

Now, we have the Bellman equation as:

$$Q(s_t, a_t, z_t) = \beta r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}} \left[U(s_{t+1}, z_t) \right] \tag{C.7}$$

Repeating the procedure, we reach the final solution as shown.

C.2 Solution to Multi-agent Soft-Hierarchical Reinforcement Learning

Starting with the objective, we have the following ELBO to solve at the final time step T. Note that solving this requires standard techniques:

$$\begin{aligned}\mathbb{E}_q\Bigg[\beta r(s_T, a_T^i, a_T^{-i}) &- \log\frac{\pi_\theta(a_T^i|s_T, z_T^i, z_{T-1}^{-i})}{P_{\text{prior}}(a_T^i|s_T, z_T^i, z_{T-1}^{-i})} - \log\frac{\rho_\theta(a_T^{-i}|s_T, z_T^i, a_T^i, z_T^{-i})}{P_{\text{prior}}(a_T^{-i}|s_T, z_T^i, a_T^i, z_T^{-i})} \\ &- \log\frac{q^{H,i}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)}{P_{\text{prior}}^{H,i}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)} - \log\frac{q^{H,-i}(z_T^{-i}|s_T, a_T^i, z_T^i, z_{T-1}^{-i}, b_T^i)}{P_{\text{prior}}^{H,-i}(z_T^{-i}|s_T, a_T^i, z_T^i, z_{T-1}^{-i}, b_T^i)} \\ &- \log\frac{q^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}{P_{\text{prior}}^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})} - \log\frac{q^{T,-i}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}{P_{\text{prior}}^{T,-i}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}\Bigg]\end{aligned} \tag{C.8}$$

Starting with the definition of the opponent model, which is equal to:

$$\begin{aligned}\rho_\theta(a_T^{-i}|s_T, z_T^i, a_T^i, z_T^{-i}) &= \frac{\exp Q(s_T, z_T^i, a_T^i, z_T^{-i}, a_T^{-i}) P_{\text{prior}}(a_T^{-i}|s_T, z_T^i, a_T^i, z_T^{-i})}{\exp Q(s_T, z_T^i, a_T^i, z_T^{-i})} \\ \text{where } Q(s_T, z_T^i, a_T^i, z_T^{-i}, a_T^{-i}) &= \beta r(s_T, a_T^i, a_T^{-i}) \\ Q(s_T, z_T^i, a_T^i, z_T^{-i}) &= \log\int \exp Q(s_T, z_T^i, a_T^i, z_T^{-i}, a_T^{-i}) P_{\text{prior}}(a_T^{-i}|s_T, z_T^i, a_T^i, z_T^{-i})\ \mathrm{d}a_T^{-i}\end{aligned}$$

Plugging this back and split for the opponent model's master policy, which lead to the following objective

$$\begin{aligned}b_T^{-i}\Bigg[Q(s_T, z_T^i, a_T^i, z_T^{-i}) &- \log\frac{\pi_\theta(a_T^i|s_T, z_T^i, z_{T-1}^{-i})}{P_{\text{prior}}(a_T^i|s_T, z_T^i, z_{T-1}^{-i})} - \log\frac{q^{H,i}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)}{P_{\text{prior}}^{H,i}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)} \\ &- \log\frac{q^{H,-i}(z_T^{-i}|s_T, a_T^i, z_T^i)}{P_{\text{prior}}^{H,-i}(z_T^{-i}|s_T, a_T^i, z_T^i)} - \log\frac{q^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}{P_{\text{prior}}^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})} \\ &- \log\frac{q^{T,-i}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}{P_{\text{prior}}^{T,-i}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}\Bigg] \\ +(1-b_T^{-i})\Bigg[Q(s_T, z_T^i, a_T^i, z_{T-1}^{-i}) &- \log\frac{\pi_\theta(a_T^i|s_T, z_T^i, z_{T-1}^{-i})}{P_{\text{prior}}(a_T^i|s_T, z_T^i, z_{T-1}^{-i})} - \log\frac{q^{H,i}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)}{P_{\text{prior}}^{H,i}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)} \\ &- \log\frac{q^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}{P_{\text{prior}}^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})} - \log\frac{q^{T,-i}(1-b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}{P_{\text{prior}}^{T,-i}(1-b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}\Bigg]\end{aligned} \tag{C.9}$$

Now, we can consider the opponent's master policy , which is equal to:

$$\begin{aligned}q^{H,-i}(z_T^{-i}|s_T, a_T^i, z_T^i) &= \frac{\exp Q(s_T, z_T^i, a_T^i, z_T^{-i}) P_{\text{prior}}(z_T^{-i}|s_T, a_T^i, z_T^i)}{\exp Q(s_T, z_T^i, a_T^i)} \\ \text{where } Q(s_T, z_T^i, a_T^i) &= \log\int \exp Q(s_T, z_T^i, a_T^i, z_T^{-i}) P_{\text{prior}}(z_T^{-i}|s_T, a_T^i, z_T^i)\ \mathrm{d}z_T^{-i}\end{aligned} \tag{C.10}$$

Plugging this back, we are left with the objective for termination policy, which is:

$$\mathbb{E}\Bigg[b_T^{-i}\left[Q(s_T, z_T^i, a_T^i)\right] + (1-b_T^{-i})\left[Q(s_T, z_T^i, a_T^i, z_{T-1}^{-i})\right] - \log\frac{q^{T,-i}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}{P^{T,-i}_{\text{prior}}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)} \\ - \log\frac{\pi_\theta(a_T^i|s_T, z_T^i, z_{T-1}^{-i})}{P_{\text{prior}}(a_T^i|s_T, z_T^i, z_{T-1}^{-i})} - \log\frac{q^{H,i}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)}{P^{H,i}_{\text{prior}}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)} \\ - \log\frac{q^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}{P^{T,i}_{\text{prior}}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}\Bigg] \tag{C.11}$$

Let's optimize the termination policy for opponent model, which is:

$$q^{T,-i}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i) = \frac{\exp Q(s_T, z_{T-1}^{-i}, a_T^i, z_T^i, b_T^{-i}) P_{\text{prior}}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}{\exp Q(s_T, z_{T-1}^{-i}, a_T^i, z_T^i)}$$
$$\text{where } Q(s_T, z_{T-1}^{-i}, a_T^i, z_T^i, b_T^{-i}) = b_T^{-i}\left[Q(s_T, z_T^i, a_T^i)\right] + (1-b_T^{-i})\left[Q(s_T, z_T^i, a_T^i, z_{T-1}^{-i})\right]$$
$$Q(s_T, z_{T-1}^{-i}, a_T^i, z_T^i) = \log\int \exp Q(s_T, z_{T-1}^{-i}, a_T^i, z_T^i, b_T^{-i}) P_{\text{prior}}(b_T^{-i}|s_T, z_{T-1}^{-i}, a_T^i, z_T^i)\ \mathrm{d}b_T^{-i} \tag{C.12}$$

Now, we have the objective for an agent:

$$\mathbb{E}\Bigg[Q(s_T, z_{T-1}^{-i}, a_T^i, z_T^i) - \log\frac{\pi_\theta(a_T^i|s_T, z_T^i, z_{T-1}^{-i})}{P_{\text{prior}}(a_T^i|s_T, z_T^i, z_{T-1}^{-i})} - \log\frac{q^{H,i}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)}{P^{H,i}_{\text{prior}}(z_T^i|s_T, z_{T-1}^{-i}, z_{T-1}^i, b_T^i)} \\ - \log\frac{q^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}{P^{T,i}_{\text{prior}}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}\Bigg] \tag{C.13}$$

Starting with agent's policy:

$$\pi_\theta(a_T^i|s_T, z_T^i, z_{T-1}^{-i}) = \frac{\exp Q(s_T, z_{T-1}^{-i}, a_T^i, z_T^i) P_{\text{prior}}(a_T^i|s_T, z_T^i, z_{T-1}^{-i})}{\exp Q(s_T, z_{T-1}^{-i}, z_T^i)}$$
$$\text{where } Q(s_T, z_{T-1}^{-i}, z_T^i) = \log\int \exp Q(s_T, z_{T-1}^{-i}, a_T^i, z_T^i) P_{\text{prior}}(a_T^i|s_T, z_T^i, z_{T-1}^{-i})\ \mathrm{d}a_T^i \tag{C.14}$$

Let's now consider the differences when the agent's master policy doesn't use the old option z_{T-1}^i

$$b_T^i\left[Q(s_T, z_{T-1}^{-i}, z_T^i) - \log\frac{q_\phi^{H,i}(z_T^i|s_T, z_{T-1}^{-i})}{P^{H,i}_{\text{prior}}(z_T^i|s_T, z_{T-1}^{-i})} - \log\frac{q^{T,i}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}{P^{T,i}_{\text{prior}}(b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}\right] \\ + (1-b_T^i)\left[Q(s_T, z_{T-1}^{-i}, z_{T-1}^i) - \log\frac{q^{T,i}(1-b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}{P^{T,i}_{\text{prior}}(1-b_T^i|s_T, z_{T-1}^i, z_{T-1}^{-i})}\right] \tag{C.15}$$

Now, we consider the agent's master policy

$$q_\phi^{H,i}(z_T^i|s_T, z_{T-1}^{-i}) = \frac{\exp Q(s_T, z_{T-1}^{-i}, z_T^i) P_{\text{prior}}(z_T^i|s_T, z_{T-1}^{-i})}{\exp Q(s_T, z_{T-1}^{-i})}$$
$$\text{where } Q(s_T, z_{T-1}^{-i}) = \log\int \exp Q(s_T, z_{T-1}^{-i}, z_T^i) P_{\text{prior}}(z_T^i|s_T, z_{T-1}^{-i})\ \mathrm{d}z_T^i \tag{C.16}$$

We are ended with the objective for agent's terminal policy:

$$\mathbb{E}_q \left[b_T^i \left[Q(s_T, z_{T-1}^{-i}) \right] + (1 - b_T^i) \left[Q(s_T, z_{T-1}^{-i}, z_{T-1}^i) \right] - \log \frac{q^{T,i}(b_T^i | s_T, z_{T-1}^i, z_{T-1}^{-i})}{P_{\text{prior}}^{T,i}(b_T^i | s_T, z_{T-1}^i, z_{T-1}^{-i})} \right] \tag{C.17}$$

Finally, we end with the policy

$$\begin{aligned} q^{T,i}(b_T^i | s_T, z_{T-1}^i, z_{T-1}^{-i}) &= \frac{\exp Q(s_T, z_{T-1}^i, z_{T-1}^{-i}, b_T^i) P_{\text{prior}}(b_T^i | s_T, z_{T-1}^i, z_{T-1}^{-i})}{\exp Q(s_T, z_{T-1}^i, z_{T-1}^{-i})} \\ \text{where } Q(s_T, z_{T-1}^i, z_{T-1}^{-i}, b_T^i) &= b_T^i \left[Q(s_T, z_{T-1}^{-i}) \right] + (1 - b_T^i) \left[Q(s_T, z_{T-1}^{-i}, z_{T-1}^i) \right] \\ Q(s_T, z_{T-1}^i, z_{T-1}^{-i}) &= \log \int \exp Q(s_T, z_{T-1}^i, z_{T-1}^{-i}, b_T^i) P_{\text{prior}}(b_T^i | s_T, z_{T-1}^i, z_{T-1}^{-i}) \, \mathrm{d}b_T^i \end{aligned} \tag{C.18}$$

We are left with the message $Q(s_T, z_{T-1}^i, z_{T-1}^{-i})$, which we can consider message as follows

$$\begin{aligned} \mathbb{E}_q \Bigg[\beta r(s_t, a_t^i, a_t^{-i}) &- \log \frac{\pi_\theta(a_t^i | s_t, z_t^i, z_{t-1}^{-i})}{P_{\text{prior}}(a_t^i | s_t, z_t^i, z_{t-1}^{-i})} - \log \frac{\rho_\theta(a_t^{-i} | s_t, z_t^i, a_t^i, z_t^{-i})}{P_{\text{prior}}(a_t^{-i} | s_t, z_t^i, a_t^i, z_t^{-i})} \\ &- \log \frac{q^{H,i}(z_t^i | s_t, z_{t-1}^{-i}, z_{t-1}^i, b_t^i)}{P_{\text{prior}}^{H,i}(z_t^i | s_t, z_{t-1}^{-i}, z_{t-1}^i, b_t^i)} - \log \frac{q^{H,-i}(z_t^{-i} | s_t, a_t^i, z_t^i, z_{t-1}^{-i}, b_t^i)}{P_{\text{prior}}^{H,-i}(z_t^{-i} | s_t, a_t^i, z_t^i, z_{t-1}^{-i}, b_t^i)} \\ &- \log \frac{q^{T,i}(b_t^i | s_t, z_{t-1}^i, z_{t-1}^{-i})}{P_{\text{prior}}^{T,i}(b_t^i | s_t, z_{t-1}^i, z_{t-1}^{-i})} - \log \frac{q^{T,-i}(b_t^{-i} | s_t, z_{t-1}^{-i}, a_t^i, z_t^i)}{P_{\text{prior}}^{T,-i}(b_t^{-i} | s_t, z_{t-1}^{-i}, a_t^i, z_t^i)} + \gamma \mathbb{E}_{s_{t+1}} \left[Q(s_{t+1}, z_t^i, z_t^{-i}) \right] \Bigg] \end{aligned} \tag{C.19}$$

Now, let's consider Bellman equation as follows:

$$Q(s_t, a_t^i, a_t^{-i}, z_t^i, z_t^{-i}) = \beta r(s_t, a_t^i, a_t^{-i}) + \gamma \mathbb{E}_{s_{t+1}} \left[Q(s_{t+1}, z_t^i, z_t^{-i}) \right] \tag{C.20}$$

Following the same procedure before, we can have the optimal policy as follows.