Robust Multi-Agent Reinforcement Learning With Reward Uncertainty

Kaiqing Zhang Sahika Genc

July 29, 2019

Abstract

In this work, we study the problem of multi-agent reinforcement learning (MARL) in face of the uncertainty of reward functions and probably also transition dynamics, namely, *robust MARL*. This is naturally motivated from the application of multi-car DeepRacer, where each user may train their multi-car racing agent using their own developed reward function, which may vary among different users with different preferences. Therefore, to enable racing in a multi-car setting with unknown opponent, the learning agent needs to train a policy that is robust to the opponent's policy that may be the equilibrium of a game based on the its own reward function.

1 Introduction

2 Problem Formulation

In this section, we introduce the background and formulation of the robust MARL problem.

2.1 Markov Games and MARL

In order to model the interaction among agents, a general framework of *Markov games* has been used in the literature of MARL (Littman, 1994). In particular, a Markov game $\mathcal{G}$ is usually characterized by a tuple

$$\mathcal{G} := \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{R^i\}_{i\in\mathcal{N}}, P, \gamma \rangle,$$

where $\mathcal{N} = [N]$ denotes the set of N agents, $\mathcal{S}$ denotes the state space that is common to all agents, $\mathcal{A}^i$ denotes the action space of agent $i \in \mathcal{N}$. $R^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \mathbb{R}$ represents the reward function of agent i, which is dependent on the state and the joint action of all agents. $P : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \to \Delta(\mathcal{S})$ represents the state transition probability that is a mapping from the current state and the joint action to the probability distribution over the state space. $\gamma \in [0,1]$ is the discounting factor.

At each time t, each agent selects its own action $a_t^i \in \mathcal{A}^i$ in face of the system state s_t, according to its own policy $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$, which is a mapping from the state space to the probability distribution over action space $\mathcal{A}^i$. Note that here we only consider the *Markov policies* that depend on the current state s_t at time t. Then the system transits to the next state s_{t+1} and each agent i receives the instantaneous reward $r_t^i = R^i(s_t, a_1^1, \cdots, a_t^N)$. The goal of each agent i is to maximize the long-term return J^i calculated using r_t^i, i.e.,

$$\max_{\pi^i} \quad J^i(\pi^i, \pi^{-i}) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i \,\middle|\, s_0, a_t^i \sim \pi^i(\cdot|s_t)\right] \tag{2.1}$$

where $-i$ represents the indices of all agents except agent i, and $\pi^{-i} := \prod_{j \neq i} \pi^j$ refers to the joint policy of all agents except agent i. In the same vein, one can define the value and action-value (Q)-function for each agent i as follows

$$V^i(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i \,\middle|\, s_0 = s, a_t^i \sim \pi^i(\cdot|s_t)\right],$$

$$Q^i(s, a^1, \cdots, a^N) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i \,\middle|\, s_0 = s, a_0^i = a^i, a_t^i \sim \pi^i(\cdot|s_t)\right].$$

Due to the coupling of agents' policies in J^i, the solution concept of maximizing the return of a single agent is unattainable. Instead, one commonly used solution concept is the (Markov perfect) *Nash equilibrium* (NE) of the game. Specifically, the NE is defined as the point of a joint policy $\pi_* := (\pi_*^1, \cdots, \pi_*^N)$ at which

$$J^i(\pi_*^i, \pi_*^{-i}) \geq J^i(\pi^i, \pi_*^{-i}), \quad \forall i \in \mathcal{N}, \tag{2.2}$$

namely, given all other agents' equilibrium policy π_*^{-i}, there is no motivation for agent i to deviate from π_*^i. Hence, the goal of MARL is to solve for the NE of the Markov game $\mathcal{G}$ without the knowledge of the model.

2.2 Formulation 1: Robust Markov Game

In many practical applications, the agents may not have perfect information of the model, i.e., the reward function and/or the transition probability model. This is originally motivated from the application of multi-car racing for DeepRacer, where the reward function used to train the car in each user's multi-agent simulation environment may be different from each other. Thus, the desired policy should not only be able to robust to the opponent's policy, but also robust to the possible uncertainty of the MARL model it will compete in.

Formally, this problem can be modeled as a *robust Markov/stochastic game* problem (Kardeş et al., 2011), which can be formally characterized by the following tuple

$$\bar{\mathcal{G}} := \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{\bar{\mathcal{R}}_s^i\}_{(i,s)\in\mathcal{N}\times\mathcal{S}}, \{\bar{\mathcal{P}}_s\}_{s\in\mathcal{S}}, \gamma\rangle,$$

where $\mathcal{N}$, $\mathcal{S}$, and $\mathcal{A}^i$ denote the set of agents, the state, and the action space for each agent i, respectively, as before. For notational simplicity, let $\mathcal{A} := \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$. Moreover, $\bar{\mathcal{R}}_s^i \subseteq \mathbb{R}^{|\mathcal{A}|}$ and $\bar{\mathcal{P}}_s$ denote the uncertainty sets of all possible reward function values and that of all possible transition probabilities at state s, respectively. Note that the uncertainty set for the reward function $\bar{\mathcal{R}}_s^i$ may vary for different agent i. $\gamma \in [0,1]$ represents the discounting factor.

Each player considers a distribution-free incomplete information Markov game to be played using robust optimization. Such a formulation allows the use of simple uncertainty sets of the model, and requires no *a prior* probabilistic information, e.g., distribution of the class of models, about the uncertainty. Notice the following fact: if the player knows how to play in the robust Markov game optimally starting from the next stage on, then it would play to maximize not only the worst-case (minimal) expected immediate reward due to the model uncertainty set at the current stage, but also the worst-case expected reward incurred in the future stages. Formally, such a recursion property leads to the following Bellman-type equation:

$$\bar{V}^i(s) = \max_{\pi^i(\cdot\,|\,s)} \min_{\substack{\bar{R}_s^i\in\bar{\mathcal{R}}_s^i \\ \bar{P}(\cdot\,|\,s,\cdot)\in\bar{\mathcal{P}}_s}} \sum_{a\in\mathcal{A}} \prod_{j=1}^{N} \pi^j(a^j\,|\,s)\Big(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} \bar{P}(s'\,|\,s,a)\bar{V}^i(s')\Big), \quad (2.3)$$

where $\bar{V}^i : \mathcal{S} \to \mathbb{R}$ denotes the *optimal robust value*, $\bar{R}_s^i = [\bar{R}^i(s,a)]_{a\in\mathcal{A}}^\top \in \bar{\mathcal{R}}_s^i \subseteq \mathbb{R}^{|\mathcal{A}|}$ with $a = (a^1, \cdots, a^N)$, is the vector of possible reward values of agent i that lies in the uncertain set of vectors $\bar{\mathcal{R}}_s^i$ at state s. $\bar{P}(\cdot\,|\,s,\cdot) : \mathcal{A}^1 \times \cdots \mathcal{A}^N \to \Delta(\mathcal{S})$ denotes the possible transition probability lying in the uncertain set $\bar{\mathcal{P}}_s$. Note that the uncertainty here can be viewed as the decision made by an implicit player, *the nature*, who always plays against each agent i by selecting the worst-case data at every state. If such an optimal robust value exists, then it also leads to the definition of *robust Markov perfect Nash equilibrium* (RMPNE), the solution concept for robust Markov games, as follows.

Definition 2.1. A joint policy $\pi_* = (\pi_*^1, \pi_*^2, \cdots, \pi_*^N)$ is the *robust Markov perfect Nash equilibrium*, if for any $s \in \mathcal{S}$, and all $i \in [N]$, there exists a vector of value functions $\bar{V} = (\bar{V}^1, \cdots, \bar{V}^N)$ with each $\bar{V}^i : \mathcal{S} \to \mathbb{R}$, such that

$$\pi_*^i(\cdot\,|\,s) \in \operatorname*{argmax}_{\pi^i(\cdot\,|\,s)} \min_{\substack{\bar{R}_s^i\in\bar{\mathcal{R}}_s^i \\ \bar{P}(\cdot\,|\,s,\cdot)\in\bar{\mathcal{P}}_s}} \sum_{a\in\mathcal{A}} \pi^i(a^i\,|\,s) \prod_{j\neq i} \pi_*^j(a^j\,|\,s)\Big(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} \bar{P}(s'\,|\,s,a)\bar{V}^i(s')\Big).$$

By Definition 2.1, the perfect Nash equilibrium we consider is *stationary*, i.e., time-invariant. We then show in the following proposition that such a stationary robust Markov perfect Nash equilibrium exists under certain conditions.

Proposition 2.2. Suppose the state and action spaces $\mathcal{S}$ and $\mathcal{A}$ are finite, and the uncertain sets of both the transition probabilities and rewards of the robust Markov game $\bar{\mathcal{G}}$, namely, $\bar{\mathcal{R}}_s^i$ and $\bar{\mathcal{P}}_s$ for all $s \in \mathcal{S}$ and $i \in \mathcal{N}$, belong to compact sets. Then, a robust Markov perfect Nash equilibrium exists.

Proof. The proof follows almost the same routine as that for Theorem 4 in (Kardeş et al., 2011). The main difference in the model is that the reward uncertainty set $\bar{\mathcal{R}}_s^i$ may vary across agents. However, the proof in (Kardeş et al., 2011) is done for each agent i individually, and for each agent i the argument holds by replacing the C_s therein by the $\bar{\mathcal{R}}_s^i$ here. The rest of the proof follows directly. We omit the proof here for brevity. □

For both simplicity and better implication for DeepRacer, we will focus on the uncertainty in the reward function only hereafter, i.e., the set $\bar{\mathcal{P}}_s$ only has one element, the exact transition $P(\cdot\,|\,s,\cdot)$. Therefore, the Bellman-type equation in (2.3) can be written as

$$\bar{V}^i(s) = \max_{\pi^i(\cdot|s)} \min_{\bar{R}_s^i \in \bar{\mathcal{R}}_s^i} \sum_{a\in\mathcal{A}} \pi^i(a^i\,|\,s) \prod_{j\neq i} \pi_*^j(a^j\,|\,s) \Big(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'\,|\,s,a) \bar{V}^i(s') \Big) \tag{2.4}$$

$$= \max_{\pi^i(\cdot|s)} \min_{\pi^{0,i}(\cdot|s)} \sum_{\bar{R}_s^i \in \bar{\mathcal{R}}_s^i} \pi^{0,i}\big(\bar{R}_s^i\,\big|\,s\big) \sum_{a\in\mathcal{A}} \pi^i(a^i\,|\,s) \prod_{j\neq i} \pi_*^j(a^j\,|\,s) \Big(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'\,|\,s,a) \bar{V}^i(s') \Big)$$

$$= \max_{\pi^i(\cdot|s)} \min_{\pi^{0,i}(\cdot|s)} \mathbb{E}_{\bar{R}_s^i \sim \pi^{0,i}(\cdot|s), a^i \sim \pi^i(\cdot|s), a^{-i} \sim \pi^{-i}(\cdot|s)} \Big(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'\,|\,s,a) \bar{V}^i(s') \Big), \tag{2.5}$$

where $\pi^{0,i} : \mathcal{S} \to \Delta(\bar{\mathcal{R}})$ denotes the policy of the nature against agent i, a mapping from the state space to the probability distribution over the joint uncertain set of the reward $\bar{\mathcal{R}} := \prod_{s\in\mathcal{S}} \bar{\mathcal{R}}_s$. Since the agents are not symmetric in the sense that the reward uncertainty and the role they play in the transition P may be different, the nature's policy against each agent i can be different. Thus, the policy for the nature is in fact a set, $\pi^0 := \{\pi^{0,i}\}_{i\in\mathcal{N}}$. The equivalence between (2.4) and (2.5) is due to the fact that the inner-loop minimization over a deterministic choice of $\bar{R}_s^i$ can be achieved by the minimization over the stochastic strategy, i.e., a probability distribution, over the support $\bar{\mathcal{R}}_s$.

By introduction of the *nature player* and its policy set $\pi^0 = \{\pi^{0,i}\}_{i\in\mathcal{N}}$, we further define the solution concept of *RMPNE with nature (NRMPNE)* as follows.

Definition 2.3. A joint policy $\widetilde{\pi}_* = (\pi_*^0, \pi_*^1, \pi_*^2, \cdots, \pi_*^N)$ is the *robust Markov perfect Nash equilibrium with nature*, where $\pi_*^0 = \left\{\pi_*^{0,i}\right\}_{i\in\mathcal{N}}$, if for any $s \in \mathcal{S}$, and all $i \in \mathcal{N}$, there exists a vector of value functions $\bar{V} = (\bar{V}^1, \cdots, \bar{V}^N)$ with each

$\bar{V}^i : \mathcal{S} \to \mathbb{R}$, such that

$$\left(\pi_*^i(\cdot\,|\,s), \pi_*^{0,i}(\cdot\,|\,s)\right) \in \underset{\pi^i(\cdot|s),\pi^{0,i}(\cdot|s)}{\operatorname{argmaxmin}} \sum_{\bar{R}_s^i \in \bar{\mathcal{R}}_s^i} \pi^{0,i}\left(\bar{R}_s^i \,\middle|\, s\right) \sum_{a\in\mathcal{A}} \pi^i(a^i\,|\,s) \prod_{j\neq i} \pi_*^j(a^j\,|\,s)\left(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'\,|\,s,a)\bar{V}^i(s')\right).$$

By Proposition 2.2, the existence of a RMPNE π_* is equivalent to the existence of a NRMPNE $\widetilde{\pi}_*$ with $\pi_*^{0,i}$ being a *deterministic* policy for all i. Thus, we will only consider the nature's policy as a deterministic one hereafter, i.e., $\pi_*^{0,i} : \mathcal{S} \to \bar{\mathcal{R}}$. Now the goal is to find the NRMPNE $\widetilde{\pi}_*$.

2.3 Formulation 2: MARL with Unknown Reward

If only the uncertainty of the reward function needs to be considered, it may not be necessary to use the general robust MARL model as in §2.2. Instead, motivated by the formulation in Eysenbach et al. (2019), we propose multi-agent RL with unknown reward functions. Compared to the robust model above, the setting here assumes that the unknown reward function follows some *a prior* distribution. The goal of the policy is to achieve a high return at the test time, when a new reward function may be drawn from this distribution. As in Eysenbach et al. (2019), one can address both the maximization of the *expected* and *worst-case* return. Specifically, consider the setting that can be characterized by the following tuple $\widetilde{G}$:

$$\widetilde{\mathcal{G}} := \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \{P_s^{R^i}\}_{(i,s)\in\mathcal{N}\times\mathcal{S}}, P, \gamma\rangle,$$

where $\mathcal{N}, \mathcal{S}, \mathcal{A}, \gamma$ are as defined before, $P : \mathcal{S}\times\mathcal{A}^1\times\cdots\times\mathcal{A}^N \to \Delta(\mathcal{S})$ represents the state transition probability, and $P_s^{R^i} \in \Delta(\bar{\mathcal{R}}_s)$ is the probability distribution over the support of $\bar{R}_s$, namely, $\bar{\mathcal{R}}_s$, of possible reward functions. Then for expected return maximization, each agent i aims to find the policy $\pi_*^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$ such that

$$\pi_*^i \in \underset{\pi^i}{\operatorname{argmax}}\ \mathbb{E}_{\bar{R}_{s_t}^i \sim P_{s_t}^{R^i}}\left\{\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i \,\middle|\, s_0, a_t^i \sim \pi^i(\cdot\,|\,s_t)\right]\right\},$$

for some given π^{-i}. By linearity of the expectation, one can instead find the policy for the new Markov game setting with the expected reward

$$\mathbb{E}_{\bar{R}_s^i \sim P_s^{R^i}}[R^i(s,a)]$$

as the reward. This essentially reduces to a standard Markov game, and can be solved by off-the-shelf MARL algorithms.

On the other hand, for worst-case return maximization, the goal is to find the policy $\pi_*^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$ such that

$$\pi_*^i \in \underset{\pi^i}{\text{argmax}} \min_{\bar{R}_{s_t} \in \bar{\mathcal{R}}_{s_t}} \left\{ \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t^i \,\middle|\, s_0, a_t^i \sim \pi^i(\cdot \,|\, s_t) \right] \right\},$$

where the reward function at each state s is taken as the worst one over the support $\bar{\mathcal{R}}_s$. In fact, this formulation will lead to essentially the same Bellman-type equation as (2.3), with the transition probability $\bar{P} = P$. This will further lead to the equivalent equation (2.5). Therefore, we will focus on solving the Bellman-type equation (2.5) hereafter.

2.4 Multi-Car Arena Platform

Besides the formal formulation above, AWS and the DeepRacer team are also thinking to build a platform API for the users, in order to provide services for the users to learn *robust policies* against opponents trained from various environments in a more efficient way. In particular, the platform, maybe named *multi-car arena*, is devised to help users to better train their multi-car policy against the policies collected by the platform. The platform will train several baseline policies, and will not disclose the policy to the user. Instead, the opponent policies will be embedded in the arena environment, and only provide feedback signal to the training agent. As a payment, the user has to submit their updated policy to the platform, so that the platform can update the policies in the pool on the fly. Note that this is more like an engineering problem, by adding a new feature of the product that may benefit the development of multi-car racing among users.

3 Algorithms

To find the Markov perfect Nash equilibrium defined in Definition 2.1, or equivalently in Definition 2.3, one has to solve the Bellman-type equation for the robust Markov game in (2.4), or (2.5). To this end, we first develop a value iteration algorithm, when the model is known to the agents. Based on this model-based algorithm, we propose a model-free Q-learning algorithm with convergence guarantees. In addition, viewing the nature as another agent, we also develop multi-agent policy gradient methods with function approximation.

3.1 Value Iteration for Robust Markov Games

By Bellman equation (2.4), one straightforward approach is to develop *value iteration* (VI) algorithms when the model on $\bar{\mathcal{G}}$ is known. In particular, the goal is to learn a value function $\bar{V}$ by updating the Bellman equation (2.4):

$$\begin{aligned}\bar{V}_{t+1}^{i}(s) &= \max_{\pi^i(\cdot|s)} \min_{\bar{R}_s^i \in \bar{\mathcal{R}}_s} \sum_{a\in\mathcal{A}} \prod_{j=1}^{N} \pi^j(a^j|s)\Big(\bar{R}^i(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)\bar{V}_t^i(s')\Big) \\ &=: \mathcal{T}_{\mathcal{V}}^{i}(\bar{V}_t^i) \end{aligned} \tag{3.1}$$

As a result, the desired value function $\bar{V}^i$ is a fixed-point of the operator $\mathcal{T}_{\mathcal{V}}^{i}(\cdot): \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$.

3.2 Q-Learning for Robust MARL

Building upon the VI update in (3.1), one can develop Q-learning-based algorithms. In particular, the action-value function, i.e., Q-value function, of robust Markov games should satisfy the following Bellman equation

$$\bar{Q}^i(s,a) := \pi_*^{0,i}(s)[a] + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) \sum_{a'} \prod_{j=1}^{N} \pi_*^j(a'^j|s')\bar{Q}^i(s',a') \tag{3.2}$$

where recall that $a = (a^1, \cdots, a^N)$ and $a' = (a'^1, \cdots, a'^N)$, π_*^j is the policy of agent j at the equilibrium, and $\pi_*^{0,i}(s)[a]$ is the a-th element of the output of the nature's deterministic policy $\pi_*^{0,i}(s)$ at the equilibrium. Note that different from the standard Bellman equation for Q-value function, the equilibrium policy π_* cannot be obtained just from $\bar{Q}^i$ (which is just the greedy policy for the single-agent setting). In fact, the Q-values of all other agents are also required to determine the equilibrium policy π_*. This is actually the most challenging part in developing multi-agent Q-learning algorithms in general (not just in our robust setting).

As a consequence, the tabular-setting Q-learning update can be written as

$$\begin{aligned}\bar{Q}_{t+1}^i(s_t,a_t) &:= (1-\alpha_t)\cdot \bar{Q}_t^i(s_t,a_t) + \alpha_t \cdot [\bar{R}_{*,t}^i + \gamma \bar{Q}_t^i(s_{t+1},a_{t+1})], \\ \text{with } \bar{R}_{*,t}^i &= \pi_{*,t}^{0,i}(s_t)[a_t], \;\; a_{t+1}^i \sim \pi_{*,t}^i(\cdot|s_{t+1}), \end{aligned} \tag{3.3}$$

where $\pi_{*,t}^0 = \left\{\pi_{*,t}^{0,i}\right\}_{i\in\mathcal{N}}$ and $\pi_{*,t}^i$ denote the equilibrium policies of the nature and agent i, respectively, that are obtained based on the Q-value estimate $\bar{Q}_t = (\bar{Q}_t^1, \cdots, \bar{Q}_t^N)$ at iteration t, and $\pi_{*,t}^{0,i}(s_t)[a_t]$ denotes the element a_t of the output vector $\pi_{*,t}^{0,i}(s_t)$. Note that this update (3.3) needs to maintain the Q-value update of all agents, i.e., $\bar{Q}_t^i$ for all agents, increases the complexity of the algorithm. This is inevitable for value-based RL for general-sum Markov games, and the same issue also occurs in the design of Nash-Q learning (Hu and Wellman, 2003). Another issue is that it is not clear how to obtain the equilibrium policy $\pi_{*,t}^i$ using $\bar{Q}_t$, even in the simplest zero-sum case. In particular, this follows from the fact that for standard Markov games (without robustness concern), such an equilibrium policy $\pi^i_{*,t}(\cdot|s)$ at time t can be obtained by solving a linear program that solves for the Nash equilibrium of the matrix game with payoff matrix $\left(\bar{Q}^1_t(\cdot|s_{t+1}), \cdots, \bar{Q}^N_t(\cdot|s_{t+1})\right)$ (Hu and Wellman, 2003). This can be further simplified by solving a minimax problem using linear programming in the zero-sum setting. However, it is not clear how to obtain the nature's equilibrium policy π^0_* without without knowing the model P, which is required in solving (3.2).. Zero-sum assumption on the reward function also does not simplify the problem, since the nature always plays against each player, which makes this three-player game non-zero sum.

3.3 Policy Gradient/Actor-Critic for Robust MARL

The challenge in developing value-based RL algorithm motivates us to consider policy gradient/actor-critic-based methods. In particular, each agent i's policy π^i is parameterized as[1] π_{θ^i} for $i = 1, \cdots, N$, and the nature's policy is parameterized by a set of policies $\pi_{\theta^0} := \{\pi_{\theta^{0,i}}\}_{i\in\mathcal{N}}$. Also, we define the return objective of each agent i under the joint policy $\widetilde{\pi}_\theta := (\pi_{\theta^0}, \pi_{\theta^1}, \cdots, \pi_{\theta^N})$ as $\mathcal{J}^i(\theta) := \bar{V}^i_{\widetilde{\pi}_\theta}(s_0)$, where s_0 denotes the initial state[2], $\theta = (\theta^0, \theta^1, \cdots, \theta^N)$ is the concatenation of all policy parameters with $\theta^0 = (\theta^{0,1}, \cdots, \theta^{0,N})$, and $\bar{V}^i_{\widetilde{\pi}_\theta}$ is the *value function* under joint policy $\widetilde{\pi}_\theta$ that satisfies

$$\bar{V}^i_{\widetilde{\pi}_\theta}(s) = \sum_{a\in\mathcal{A}} \prod_{j=1}^{N} \pi_{\theta^j}(a^j|s) \left(\pi_{\theta^{0,i}}(s)[a] + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) \bar{V}^i_{\widetilde{\pi}_\theta}(s') \right). \tag{3.4}$$

$\pi_{\theta^{0,i}}(s)[a]$ is the a-th element of the output vector $\pi_{\theta^{0,i}}(s)$. Moreover, we also similarly define the Q-value under joint policy $\widetilde{\pi}_\theta$ to be the one that satisfies

$$\bar{Q}^i_{\widetilde{\pi}_\theta}(s,a) := \pi_{\theta^0}(s)[a] + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) \sum_{a'\in\mathcal{A}} \prod_{j=1}^{N} \pi_{\theta^j}(a'^j|s') \bar{Q}^i_{\widetilde{\pi}_\theta}(s',a'). \tag{3.5}$$

We first establish the policy gradient with respect to the parameter θ in the following lemma.

Lemma 3.1. For each agent $i = 1, \cdots, N$, the policy gradient of the objective $\mathcal{J}^i(\theta)$ with respect to the parameter θ has the following form[3]:

$$\nabla_{\theta^i} \mathcal{J}^i(\theta) = \mathbb{E}_{s\sim\rho^{s_0}_{\widetilde{\pi}_\theta}, a\sim\pi_\theta(\cdot|s)} \Big[\nabla \log \pi_{\theta^i}(a^i|s) \cdot \bar{Q}^i_{\widetilde{\pi}_\theta}(s,a) \Big], \tag{3.6}$$

$$\nabla_{\theta^{0,i}} \mathcal{J}^i(\theta) = \mathbb{E}_{s\sim\rho^{s_0}_{\widetilde{\pi}_\theta}, a\sim\pi_\theta(\cdot|s)} \Big[\nabla \pi_{\theta^{0,i}}(s)[a] \Big], \tag{3.7}$$

[1] For notational simplicity, we omit the superscript i since the index i can be identified by the parameter used.

[2] Note that the derivation below can be easily generalized to the setting that the initial state is randomly drawn from some distribution.

[3] For notational simplicity, we omit the parameter that some function takes gradient with respect to, if the function takes gradient with respect to the full parameter, e.g., we write $\nabla_{\theta^{0,i}} \pi_{\theta^{0,i}}(s)[a]$ as $\nabla \pi_{\theta^{0,i}}(s)[a]$, $\nabla_{\theta^i} \log \pi_{\theta^i}(a^i|s)$ as $\nabla \log \pi_{\theta^i}(a^i|s)$.

where $\pi_\theta(a|s) := \prod_{j=1}^{N} \pi_{\theta^j}(a^j|s)$, $\rho_{\widetilde{\pi}_\theta}^{s_0} := \sum_{t=0}^{\infty} \gamma^t Pr(s \to s', t, \pi_\theta)$ is the discounted ergodic state distribution under joint policy $\widetilde{\pi}_\theta$ with state starting from s_0, $Pr(s \to s', t, \pi_\theta)$ denotes the probability of transitioning from s to s' under joint policy π_θ with t-steps, and $\pi_{\theta^{0,i}}(s)[a]$ is the a-th element of the output of $\pi_{\theta^{0,i}}(s)$.

Proof. Note that $\mathcal{J}^i(\theta)$ can be viewed as the normal value in Markov games with reward function $R^i(s,a) = \pi_{\theta^{0,i}}(s)[a]$. Thus, the form of (3.6) follows by the derivation in either (Lowe et al., 2017, Eq. (4)) or (Zhang et al., 2018, Theorem 3.1).

Moreover, taking gradient with respect to $\theta^{0,i}$ on both sides of (3.4) yields

$$\begin{aligned}
\nabla_{\theta^{0,i}} \bar{V}^i_{\widetilde{\pi}_\theta}(s) &= \sum_{a\in\mathcal{A}} \pi_\theta(a|s)\Big(\nabla\pi_{\theta^{0,i}}(s)[a] + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)\cdot\nabla_{\theta^{0,i}}\bar{V}^i_{\widetilde{\pi}_\theta}(s')\Big) \\
&= \sum_{a\in\mathcal{A}} \pi_\theta(a|s)\Big[\nabla\pi_{\theta^{0,i}}(s)[a] + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)\cdot \sum_{a'\in\mathcal{A}} \pi_\theta(a'|s') \\
&\qquad \Big(\nabla\pi_{\theta^{0,i}}(s')[a'] + \gamma\sum_{s''\in\mathcal{S}} P(s''|s',a')\cdot\nabla_{\theta^{0,i}}\bar{V}^i_{\widetilde{\pi}_\theta}(s'')\Big)\Big] \\
&= \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\Big[\nabla\pi_{\theta^{0,i}}(s)[a]\Big] + \gamma\sum_{s'\in\mathcal{S}} Pr(s\to s',1,\pi_\theta)\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\Big[\nabla\pi_{\theta^{0,i}}(s')[a']\Big] \\
&\qquad + \gamma^2\sum_{s''\in\mathcal{S}} Pr(s\to s'',2,\pi_\theta)\cdot\nabla_{\theta^{0,i}}\bar{V}^i_{\widetilde{\pi}_\theta}(s''),
\end{aligned} \tag{3.8}$$

where the second equation follows by unrolling $\nabla_{\theta^{0,i}}\bar{V}^i_{\widetilde{\pi}_\theta}(s')$. By keeping unrolling (3.8), we further have

$$\begin{aligned}
\nabla_{\theta^{0,i}}\bar{V}^i_{\widetilde{\pi}_\theta}(s) &= \sum_{s'\in\mathcal{S}}\sum_{t=0}^{\infty}\gamma^t Pr(s\to s',t,\pi_\theta)\cdot\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\Big[\nabla\pi_{\theta^{0,i}}(s')[a']\Big] \\
&= \sum_{s'\in\mathcal{S}}\rho^s_{\widetilde{\pi}_\theta}(s')\cdot\mathbb{E}_{a'\sim\pi_\theta(\cdot|s')}\Big[\nabla\pi_{\theta^{0,i}}(s')[a']\Big],
\end{aligned} \tag{3.9}$$

which implies the formula in (3.7) and completes the proof. □

Lemma 3.1 lays foundations for developing policy gradient methods for each agent i. In particular, if the robust Q-value function $\bar{Q}^i_{\widetilde{\pi}_\theta}$ is also parameterized as $\bar{Q}_{\omega^i}: \mathcal{S}\times\mathcal{A}\to\mathbb{R}$ by some parameter $\omega^i\in\mathbb{R}^d$. Then, temporal difference (TD) learning algorithm can be applied to perform policy evaluation, i.e., the critic update. This gives us the online actor-critic algorithm as follows:

$$\delta^i_t = \pi_{\theta_t^{0,i}}(s_t)[a_t] + \gamma\bar{Q}_{\omega^i_t}(s_{t+1},a_{t+1}) - \bar{Q}_{\omega^i_t}(s_t,a_t) \tag{3.10}$$

$$\omega^i_{t+1} = \omega^i_t + \alpha_t\cdot\delta^i_t\cdot\nabla\bar{Q}_{\omega^i_t}(s_t,a_t) \tag{3.11}$$

$$\theta^i_{t+1} = \theta^i_t + \beta_t\cdot\nabla\log\pi_{\theta^i_t}(a^i_t|s_t)\cdot\bar{Q}_{\omega^i_t}(s_t,a_t) \tag{3.12}$$

$$\theta^{0,i}_{t+1} = \theta^{0,i}_t - \beta_t\cdot\nabla\pi_{\theta^{0,i}_t}(s_t)[a_t], \tag{3.13}$$

where δ_t^i is the TD error for agent i, $\alpha_t, \beta_t > 0$ are both stepsizes that may diminish over time, i.e., $\lim_{t\to\infty} \alpha_t = \lim_{t\to\infty} \beta_t = 0$, and also satisfies:

$$\sum_{t\geq 0} \alpha_t^2 < \infty, \quad \sum_{t\geq 0} \beta_t^2 < \infty, \qquad \sum_{t\geq 0} \alpha_t = \sum_{t\geq 0} \beta_t = \infty.$$

Moreover, usually α_t is larger than β_t as $t \to \infty$, i.e., $\lim_{t\to\infty} \beta_t/\alpha_t = 0$, in order to ensure that the critic step performs faster than the actor step. This is also known as a *two-timescale* update.

In practice, both critic and actor can be updated in a mini-batch fashion. See Algorithm 1 for the details. Note that the minimization in line 11 in Algorithm 1 can be solved by any nonconvex optimization solvers, or by just several iterations of gradient steps w.r.t. ω^i.

Algorithm 1 Actor-Critic for Robust Multi-Agent RL:

```
1: Initialization of Q-value parameters $\{\omega_0^i\}_{i\in\mathcal{N}}$, and policy parameters
   $\{\theta_0^i\}_{i\in\mathcal{N}}$ and $\theta_0^0 := \{\theta_0^{0,i}\}_{i\in\mathcal{N}}$.
2: for episode = 1 to $M$ do
3:   Receive an initial state $s$
4:   for $t = 1, \cdots T$ do
5:     For each agent $i$, sample action $a^i \sim \pi_{\theta_t^i}$ with current policy $\pi_{\theta_t^i}$
6:     Execute joint $a = (a^1, \cdots, a^N)$, and observe new state $s'$
7:     Store $(s, a, s')$ in replay buffer $\mathcal{D}$, let $s' \leftarrow s$
8:     for agent $i = 1$ to $N$ do
9:       Sample a random mini-batch of $S$ samples of $(s_t, a_t, s_{t+1})$ from $\mathcal{D}$
10:      Set
             $y_t = \pi_{\theta^{0,i}}(s_t)[a_t] + \gamma \bar{Q}_{\omega^i}(s_{t+1}, a_{t+1}^1, \cdots, a_{t+1}^N)\big|_{a_{t+1}^i \sim \pi_{\theta^i}(\cdot|s_{t+1})}$,
11:      Update critic by minimizing the loss $\mathcal{L}(\theta^i) = \frac{1}{S}\sum_{t=1}^S \left(y_t - \bar{Q}_{\omega^i}(s_t, a_t)\right)^2$
12:      Update actor using the sampled policy gradient
             $\nabla_{\theta^i}\mathcal{J}^i(\theta) \approx \frac{1}{S}\sum_{t=1}^S \nabla \log \pi_{\theta^i}(a_t^i|s_t)\bar{Q}_{\widetilde{\pi}_\theta}^i(s_t, a_t)$, $\theta'^i = (1-\tau)\theta^i + \tau\nabla_{\theta^i}\mathcal{J}^i(\theta)$
             $\nabla_{\theta^{0,i}}\mathcal{J}^i(\theta) \approx \frac{1}{S}\sum_{t=1}^S \nabla \pi_{\theta^{0,i}}(s_t)[a_t]$,   $\theta'^{0,i} = (1-\tau)\theta^{0,i} + \tau\nabla_{\theta^{0,i}}\mathcal{J}^i(\theta)$
13:    end for
14:  end for
15: end for
```

References

Eysenbach, B., Tyo, J., Gu, S., Salakhutdinov, R., Lipton, Z. and Levine, S. (2019). Reinforcement learning with unknown reward functions. In *Proceedings of the Workshop on "Structure & Priors in Reinforcement Learning" at International Conference on Learning Representations*.

Hu, J. and Wellman, M. P. (2003). Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, **4** 1039–1069.

Kardeş, E., Ordóñez, F. and Hall, R. W. (2011). Discounted robust stochastic games and an application to queueing control. *Operations Research*, **59** 365–382.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *International Conference on Machine Learning*.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P. and Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*.

Zhang, K., Yang, Z., Liu, H., Zhang, T. and Başar, T. (2018). Fully decentralized multi-agent reinforcement learning with networked agents. In *International Conference on Machine Learning*.