Model-based Multi-agent Policy Optimization with Adaptive Opponent-wise Rollouts

Weinan Zhang , Xihuai Wang , Jian Shen , Ming Zhou
Shanghai Jiao Tong University
{wnzhang, leoxhwang, r_ocky, mingak}@sjtu.edu.cn

Abstract

This paper investigates the model-based methods in multi-agent reinforcement learning (MARL). We specify the dynamics sample complexity and the opponent sample complexity in MARL, and conduct a theoretic analysis of return discrepancy upper bound. To reduce the upper bound with the intention of low sample complexity during the whole learning process, we propose a novel decentralized model-based MARL method, named Adaptive Opponent-wise Rollout Policy Optimization (AORPO). In AORPO, each agent builds its multi-agent environment model, consisting of a dynamics model and multiple opponent models, and trains its policy with the adaptive opponent-wise rollout. We further prove the theoretic convergence of AORPO under reasonable assumptions. Empirical experiments on competitive and cooperative tasks demonstrate that AORPO can achieve improved sample efficiency with comparable asymptotic performance over the compared MARL methods.

1 Introduction

Multi-agent reinforcement learning (MARL) has been recently paid much attention and preliminarily applied to various control scenarios including robot system, autonomous driving, resource utilization etc. [Du and Ding, 2020]. One main technical challenge MARL brings over single-agent reinforcement learning (SARL) is that the agents need to interact with other agents, and their returns depend on the behavior of all the agents, which usually requires a considerable amount of samples, i.e., high sample complexity.

In a general MARL setting, one agent can access the actions taken by other agents in any state through some communication protocol, without any knowledge of their specific policies [Tian *et al.*, 2020]. In such a situation, we claim that the sample complexity in MARL comes from two parts: *dynamics sample complexity* and *opponent sample complexity*. To achieve some policy performance, dynamics sample complexity represents the number of interactions made with the environment dynamics while collecting the samples. On the other hand, opponent sample complexity, which is unique in MARL, represents the times of communications to access one opponent's action. Thus one goal of MARL is to find an effective policy with the two sample complexities being low.

In SARL scenarios, it is well known that model-based reinforcement learning (MBRL) can achieve lower dynamics sample complexity than model-free reinforcement learning (MFRL) empirically [Wang *et al.*, 2019] or achieve at least competitive complexity theoretically [Jin *et al.*, 2018]. Specifically, by building a dynamics model, the agent can also be trained with the model simulation samples and the ones collected from the environment, which can reduce the need for the environment samples [Luo *et al.*, 2019]. In multi-agent scenarios, however, to utilize the dynamics model for data simulation, we need to ask for the opponents' actions through a communication protocol, which will count as the opponent sample complexity. Via building opponent models [He *et al.*, 2016], we can replace the real opponents in the data simulation stage to reduce the opponent sample complexity.

In this paper, we investigate model-based methods for MARL and propose a novel method called Adaptive Opponent-wise Rollout Policy Optimization (AORPO). Specifically, from the perspective of each ego agent[1], we build a multi-agent environment model, which consists of a dynamics model and opponent models for each opponent agent. The multi-agent environment model can then be used to perform simulation rollout for MARL training to reduce both sample complexities. To our knowledge, however, in literature, there is no theoretical guidance regarding how to perform the multi-agent simulated rollouts. We provide a theoretical analysis about the upper bound of the return discrepancy w.r.t. policy distribution shift and the generalization errors of the environment dynamics and each opponent model. The upper bound reveals that when the environment dynamics model performs rollout with multiple opponents models, different magnitudes of opponent model errors may lead to different contributions to the compounding error of the multi-step simulated rollouts. Based on the theoretic analysis, we design the adaptive opponent-wise rollout scheme for our policy optimization algorithm, called AORPO, and prove its convergence under reasonable assumptions.

Experiments show that AORPO outperforms several state-of-the-art MARL methods in terms of sample efficiency in both cooperative tasks and competitive tasks.

[1] In this paper, each studied agent is called ego agent while the other agents are called opponent agents. See Figure 2 for illustrations.

2 Related Work

There are two important training paradigms in MARL, i.e., centralized training with decentralized execution (CTDE) and fully decentralized training. Typical algorithms using CTDE paradigm are value function decomposition algorithms [Whiteson, 2018], which deal with the non-stationarity and credit-assignment challenges by a centralized value function. Fully decentralized MARL algorithms include consensus learning over networked agents and communicating agents [Zhang *et al.*, 2018; Qu *et al.*, 2019]. In decentralized scenarios, where the agents can communicate with each other, Iqbal and Sha [2019] proposed a method that integrates the received messages to learn a value function and a policy. In this paper, we propose a fully decentralized training method with a communication protocol.

Opponent modeling [He *et al.*, 2016] is a feasible solution to the non-stationarity problem in MARL, which models others' policies with interaction experience. Behavior cloning is a straightforward method to learn opponents' policies [Lowe *et al.*, 2017]. More advances include recursive reasoning with variational inference [Wen *et al.*, 2019], maximum-entropy objective [Tian *et al.*, 2019], and modeling the learning process of opponents [Foerster *et al.*, 2018] etc.

There are generally two major problems for model-based RL methods, i.e., model learning and model usage. For model learning, the most common solution is supervised learning [Nagabandi *et al.*, 2018], or non-parametric methods such as Gaussian processes [Kamthe and Deisenroth, 2018]. For model usage, a policy can be derived by exploiting the model with different algorithms such as Dyna [Sutton, 1990], shooting methods [Chua *et al.*, 2018] and policy search with back-propagation through paths [Clavera *et al.*, 2020]. We refer to Wang *et al.* [2019] for details of model-based RL. Theoretical analysis of model-based RL in single-agent scenarios has been investigated in recent literature. To name a few, Luo *et al.* [2019] provided a monotonic improvement guarantee by enforcing a distance constraint between the learning policy and the data-collecting policy. Janner *et al.* [2019] derived a return discrepancy bound with the branched rollout, which studies how the model generalization affects the model usage.

For model-based MARL, there are relatively limited works of literature, to our knowledge. Park *et al.* [2019] proposed to use a centralized auxiliary prediction network to model the environment dynamics to alleviate the non-stationary dynamics problem. Kamra *et al.* [2020] and Li *et al.* [2020] proposed interaction graph-based trajectory prediction methods, without considering policies. Krupnik *et al.* [2019] built a centralized multi-step generative model with a disentangled variational auto-encoder to predict the environment dynamics and the opponent actions and then performed trajectory planning. Unlike previous work, in our proposed AORPO each agent builds its environment model consisting of a dynamics model and opponent models, then learns its policy using the model rollouts. To our knowledge, AORPO is the first Dyna-style method in MARL. More importantly, AORPO is supported by the theoretical bound of return discrepancy and convergence guarantee, which provides a principle to design further Dyna-style model-based MARL methods.

3 Problem Definition and Sample Complexity

3.1 Problem Definition

We formulate the MARL problem as a stochastic game [Shapley, 1953]. An n-agent stochastic game can be defined as a tuple $(\mathcal{S}, \mathcal{A}^1, \ldots, \mathcal{A}^n, R^1, \ldots, R^n, \mathcal{T}, \gamma)$, where $\mathcal{S}$ is the state space of the stochastic game, $\mathcal{A}^i$ is the action space of agent $i \in \{1, \ldots, n\}$, $\mathcal{A} = \Pi_{i=1}^n \mathcal{A}^i$ is the joint action space, $R^i : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is the reward function of agent i, and γ is the discount factor. State transition proceeds according to the dynamics function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$. For agent i, denote its policy as $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$, which is a probability distribution over its action space, and $\pi^i(a_t|s_t)$ is the probability of taking action a_t at state s_t. By denoting other agents' actions as $a^{-i} = \{a^j\}_{j \neq i}$, we can formulate the joint policy of other agents' as $\pi^{-i}(a_t^{-i}|s_t) = \Pi_{j \in \{-i\}} \pi^j(a_t^j|s_t)$. At each timestep, actions are taken simultaneously. Each agent i aims at finding its optimal policy to maximize the expected return (cumulative reward), defined as

$$\max_{\pi^i} \eta_i[\pi^i, \pi^{-i}] = \mathbb{E}_{(s_t, a_t^i, a_t^{-i}) \sim \mathcal{T}, \pi^i, \pi^{-i}} \Big[\sum_{t=1}^{\infty} \gamma^t R^i(s_t, a_t^i, a_t^{-i}) \Big].$$

We consider a general scenario that each ego agent i has no knowledge of other agents' policies, but can observe the histories of other agents, i.e., $\{s_{1:t-1}, a_{1:t-1}^{-i}\}$ at timestep t. This scenario is common in decentralized MARL, such as in Zhang *et al.* [2018] and Qu *et al.* [2019].

3.2 Two Parts of Sample Complexity

From the perspective of the ego agent i, the state transition in MARL involves sub-processes, i.e., the action sampling process $\pi^{-i}(a^{-i}|s)$ of opponent agents given the current state s, and the transition process to the next state $\mathcal{T}(s'|s, a^i, a^{-i})$ given the current state s and the joint action (a^i, a^{-i}).

As such, for an ego agent to achieve some policy performance in a multi-agent environment, the sample complexity is specified in two parts: (i) *dynamics sample complexity*, i.e., the number of state transition interactions between the agent group and the dynamics environment, and (ii) *opponent sample complexity*, i.e., the total number of opponent action sampling given the conditioned state. Figure 1 illustrates such two parts of sample complexity.

We argue that it is necessary to study such two parts of sample complexity explicitly since the two sub-processes'

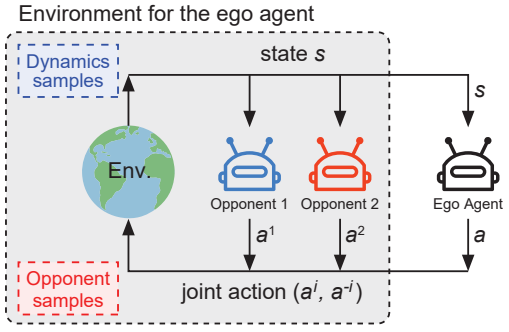

Figure 1: Two parts of the sample complexity in MARL.

approximations meet different challenges. Environment dynamics modeling is usually computationally expensive as the state representation and the joint actions are usually of high dimensions. Opponent modeling usually aims at approximating the opponents' policies and eases MARL training in a decentralized manner [Tian *et al.*, 2020].

Opponent Sample Complexity as Communication. If we leverage model-based techniques derived from SARL in multi-agent cases with decentralized setting, we may need to call real opponents for the action given the current state, which can be formulated as a communication protocol. For example, the agents communicate via calling for the opponent actions [Tian *et al.*, 2020]: the ego agent sends a state (no matter real or simulated) to an opponent agent and receives the action sampled from the opponent's policy. As such, we can regard the opponent sample complexity as communication load. To lower the opponent sample complexity, we replace the real opponents with learned opponent models in data simulation, which is analogous to selectively call some (or none) of opponents for useful information to reduce the communication load (or bandwidth) in multi-agent interactions [Ryu *et al.*, 2020].

4 Return Discrepancy Bounds

In this section, we conduct theoretical analysis to better understand model-based MARL in the decentralized setting, where each ego agent learns a dynamics model and opponent models based on observed data. For a comprehensive analysis, we also discuss the bound in centralized model-based MARL in Appendix A[2].

Usually, the dynamics model and opponent models have prediction errors, known as generalization errors. We investigate the influence of the learned models over the agent's performance. From the perspective of agent i, we aim to bound the discrepancy between the expected return $\eta_i[\pi^i, \pi^{-i}]$ of running the agent's policy π^i in the real dynamics $\mathcal{T}$ with real opponents π^{-i} and the expected return $\hat{\eta}_i[\pi^i, \hat{\pi}^{-i}]$ on a learned dynamics model $\hat{\mathcal{T}}(s'|s, a^i, a^{-i})$ with opponent models $\hat{\pi}^{-i}$. The return discrepancy upper bound C can be expressed as

$$\left|\eta_i[\pi^i, \pi^{-i}] - \hat{\eta}_i[\pi^i, \hat{\pi}^{-i}]\right| \leq C\,. \tag{1}$$

Once the upper bound C is derived, it may indicate the key influence from the dynamics model and the opponent models, and thus helps design model-based algorithms accordingly.

In MARL with iterative policy optimization, we denote data-collecting policies in the previous iterations as (π_D^i, π_D^{-i}). Then we measure the distribution shift between the current policies and the data-collecting policies with the total variation distance $\epsilon_\pi^i = \max_s D_{TV}(\pi^i(\cdot|s)\|\pi_D^i(\cdot|s))$, and similarly for the opponent agent $j \in \{-i\}$. Besides, interacting with (π_D^i, π_D^{-i}), we denote the generalization error of the learned dynamics model $\hat{\mathcal{T}}$ compared to the real environment dynamics $\mathcal{T}$ as $\epsilon_m = \max_t \mathbb{E}_{(s_t, a_t^i, a_t^{-i}) \sim \pi_D^i, \pi_D^{-i}}$ $[D_{TV}(\mathcal{T}(\cdot|s_t, a_t^i, a_t^{-i})\|\hat{\mathcal{T}}(\cdot|s_t, a_t^i, a_t^{-i}))]$ and the error of the opponent model for agent j as $\epsilon_{\hat{\pi}}^j = \max_s D_{TV}(\pi^j(\cdot|s)\|$ $\hat{\pi}^j(\cdot|s))$. Now we are ready to provide an upper bound.

[2]The full version of this paper with a detailed appendix can be found at https://arxiv.org/abs/2105.03363.

Theorem 1. Assume that the error of the learned dynamics model is bounded at each timestep by ϵ_m, and the distance of the policies are bounded by ϵ_π^i, ϵ_π^j, and the errors of the opponent models are bounded by $\epsilon_{\hat{\pi}}^j$, for $j \in \{-i\}$. The return discrepancy upper bound can be expressed as

$$\begin{aligned}&\left|\eta_i[\pi^i, \pi^{-i}] - \hat{\eta}_i[\pi^i, \hat{\pi}^{-i}]\right| \\ &\leq 2r_{\max}\Big[\frac{\gamma(\epsilon_m + 2\epsilon_\pi^i + 2\sum_{j\in\{-i\}}\epsilon_\pi^j + \sum_{j\in\{-i\}}\epsilon_{\hat{\pi}}^j)}{1-\gamma} \\ &\qquad + 2\epsilon_\pi^i + 2\sum_{j\in\{-i\}}\epsilon_\pi^j + \sum_{j\in\{-i\}}\epsilon_{\hat{\pi}}^j\Big].\end{aligned} \tag{2}$$

Proof. The proof is provided in Appendix B.2. □

This upper bound can be minimized by improving models' accuracy, which is the standard objective in previous literature. We investigate the model usage instead, and a more instructive and practical upper bound is needed.

Here we first investigate a k-step rollout scheme, which is a natural extension of MBPO [Janner *et al.*, 2019] to multi-agent scenario. This rollout scheme begins from a state collected by the previous data-collecting policies, and then run the current policy for k steps under the learned dynamics model and the opponent models. Denoting the return of using multi-agent branched rollouts as η_i^{branch}, and the bound of the generalization error of the learned dynamics model with the current policies (π^i, π^{-i}) as $\epsilon'_m = \max_t \mathbb{E}_{(s_t, a_t^i, a_t^{-i}) \sim \pi^i, \pi^{-i}}[D_{TV}$ $(\mathcal{T}(\cdot|s_t, a_t^i, a_t^{-i})\|\hat{\mathcal{T}}(\cdot|s_t, a_t^i, a_t^{-i}))]$, we derive the following return discrepancy upper bound.

Theorem 2. Assume the generalization error is bounded by ϵ'_m and the distance of the policies are bounded as ϵ_π^i, ϵ_π^j and $\epsilon_{\hat{\pi}}^j$ for $j \in \{-i\}$. The bound of the discrepancy between the return in real environment $\eta_i[\pi^i, \pi^{-i}]$ and the return using the dynamics model and opponent models with branched rollouts $\eta_i^{\text{branch}}[(\pi_D^1, \hat{\pi}^1), \ldots, (\pi_D^i, \pi^i), \ldots, (\pi_D^n, \hat{\pi}^n)]$ is

$$\begin{aligned}&\left|\eta_i[\pi^i, \pi^{-i}] - \eta_i^{\text{branch}}[(\pi_D^1, \hat{\pi}^1), \ldots, (\pi_D^i, \pi^i), \ldots, (\pi_D^n, \hat{\pi}^n)]\right| \\ &\leq 2r_{\max}\Big[\underbrace{k\epsilon'_m + (k+1)\sum_{j\in\{-i\}}\epsilon_{\hat{\pi}}^j}_{\text{model generalization error}} \\ &\qquad + \underbrace{\gamma^{k+1}\big(\epsilon_\pi^i + \sum_{j\in\{-i\}}\epsilon_\pi^j\big) + \frac{\gamma^{k+1}(\epsilon_\pi^i + \sum_{j\in\{-i\}}\epsilon_\pi^j)}{1-\gamma}}_{\text{policy distribution shift}}\Big] \\ &= C(\epsilon'_m, \epsilon_\pi^i, \epsilon_\pi^{-i}, \epsilon_{\hat{\pi}}^{-i}, k)\,,\end{aligned} \tag{3}$$

where $\epsilon_\pi^{-i} = \sum_{j\in\{-i\}}\epsilon_\pi^j$, $\epsilon_{\hat{\pi}}^{-i} = \sum_{j\in\{-i\}}\epsilon_{\hat{\pi}}^j$ and the pair $(\pi_D^j, \hat{\pi}^j)$ means that the policy π_D^j and opponent model $\hat{\pi}^j$ are used before and after the branch point respectively for agent j.

Proof. The proof is provided in Appendix B.2. □

The upper bound $C(\epsilon'_m, \epsilon_\pi^i, \epsilon_\pi^{-i}, \epsilon_{\hat{\pi}}^{-i}, k)$ in Eq. (3) consists of the generalization errors of both dynamics model and opponent models as well as the policy distribution shift. Intuitively, choosing the optimal $k^* = \arg\min_{k>0} C(\epsilon'_m, \epsilon_\pi^i, \epsilon_\pi^{-i}, \epsilon_{\hat{\pi}}^{-i}, k)$

with sufficiently low ϵ'_m and $\sum_{j\in\{-i\}}\epsilon^j_{\hat{\pi}}$ minimizes the discrepancy, which, however, cannot directly achieve a low discrepancy if any opponent model has relatively large error. So we need to reduce the upper bound such that the policy trained with the model rollouts will still perform well in real environment, leading to improved sample efficiency. To be more specific, we can optimize the target η_i through optimizing η_i^{branch} if the bound is tight. And improving η_i^{branch} by $C(\epsilon'_m, \epsilon^i_\pi, \epsilon^{-i}_\pi, \epsilon^{-i}_{\hat{\pi}}, k)$ guarantees to improve the target η_i. It means that the policy improved under the environment model and the opponent models will get improved performance in the real environment.

Remark. As the policy is trained using mainly the samples from the model rollouts, which do not contribute to the sample complexity. A reduced return discrepancy upper bound indicates that the policy will get more improvement in the real environment given that the policy is improved to the same degree using the models. In other words, the policy will obtain the same performance improvement in the real environment with fewer samples with a reduced discrepancy upper bound.

Now the problem becomes how to reduce the bound in order to achieve low sample complexity. Considering that $\epsilon^j_{\hat{\pi}}$ may be different across different opponent models, and sometimes the ego agent can call the real opponent agents for real actions, the rollout length k needs not to be the same across different opponents. Thus, in Section 5 we propose a method called *Adaptive Opponent-wise Rollout Policy Optimization* (AORPO) to reduce the upper bound.

5 The AORPO Method

Based on the bound analysis in Theorem 2, now we present the detailed AORPO algorithm and prove its convergence.

5.1 Algorithm Details of AORPO

We propose a model-based MARL algorithm named Multi-Agent Branched-Rollout Policy Optimization (AORPO). The overall algorithm of AORPO is shown in Algorithm 1. For simplicity, the detailed algorithm is described in the perspective of agent i. AORPO includes some key components, and the implementation of them is based on previous work, which serve as the preliminaries and are described in Appendix E.

In AORPO, the agent i learns a dynamics model and an opponent model for each of other agents and learns a policy based on the data collected from the model rollouts.

Agent Learning

For the dynamics model, in line 3 of Algorithm 1, a bootstrap ensemble of probabilistic dynamics models is trained to predict the environment dynamics as in Chua *et al.* [2018], and we select one of the models randomly from a uniform distribution when generating a prediction. For the opponent model, in line 6 of Algorithm 1, the policy of one opponent agent j can be modeled as a Gaussian distribution of the action. We further use the opponent models to encourage the coordination behaviors. Agent i makes a decision based on both the current state and the inferred opponent actions $\hat{a}^{-i}$.

For the policy learning, AORPO is implemented based on the Multi-Agent Soft Actor-Critic (MASAC) algorithm,

Algorithm 1: AORPO Algorithm

1 Initialize policy π_ζ, Q value function Q_ω, predictive model $\hat{\mathcal{T}}_\theta$, opponent models π_{ϕ^j} for $j \in \{-i\}$, environment dataset $\mathcal{D}_{\text{env}}$, model dataset $\mathcal{D}_{\text{model}}$.
2 **for** N *epochs* **do**
3 Train model $\hat{\mathcal{T}}_\theta$ on $\mathcal{D}_{\text{env}}$.
4 **for** E *steps* **do**
5 Take actions in environment via π_ζ with real opponents, add the transitions to $\mathcal{D}_{\text{env}}$.
6 Train all opponent models π_{ϕ^j}.
7 Compute the errors for each opponent model $\epsilon^j_{\hat{\pi}}$.
8 For each opponent, compute $n^j = \lfloor k\frac{\min_{j'}\epsilon^{j'}_{\hat{\pi}}}{\epsilon^j_{\hat{\pi}}}\rfloor$.
9 **for** M *model rollouts* **do**
10 Sample s_t uniformly from $\mathcal{D}_{\text{env}}$.
11 Perform k-step rollouts from s_t:
12 **for** $p = 1, \dots, k$ **do**
13 $a^i_{t+p-1} = \pi_\zeta(s_{t+p-1})$
14 For each opponent j:
15 **if** $p \le n^j$ **then**
16 $a^j_{t+p-1} = \pi_{\phi^j}(s_{t+p-1})$
17 **else**
18 $a^j_{t+p-1} = \texttt{Comm}\,(s_{t+p-1}, j)$
19 $s_{t+p} = \hat{\mathcal{T}}_\theta(s_{t+p-1}, a^i_{t+p-1}, a^{-i}_{t+p-1})$
20 Add the transitions to $\mathcal{D}_{\text{model}}$.
21 Train the Q function and the policy using data from $\mathcal{D}_{\text{model}}$ with the loss in Eq. (5) and Eq. (4).
22 **Function** Comm *(s, j)* **:**
23 Send state s to agent j.
24 **return** the received a^j

which is a multi-agent version of Soft Actor-Critic algorithm [Haarnoja *et al.*, 2018]. In line 21 of Algorithm 1, agent i alternates between optimizing its policy and updating the value function. With the reparameterization function f and a Gaussian $\mathcal{N}$, the policy optimization objective is

$$J(\pi) = \mathbb{E}_{s_t, a_t^{-i}\sim\mathcal{D}, \epsilon_t\sim\mathcal{N}}[\alpha \log \pi(f(\epsilon_t; s_t)|s_t, \hat{a}_t^{-i}) - Q(s_t, f(\epsilon_t; s_t), a_t^{-i})], \tag{4}$$

where $\mathcal{D}$ is the replay buffer, ϵ_t is the input noise and $\hat{a}_t^{-i} \sim \hat{\pi}^{-i}(\cdot|s_t)$.

The loss function of the Q function is

$$J(Q) = \mathbb{E}_{s_t, a^i_t, a_t^{-i}\sim\mathcal{D}}\Big[\frac{1}{2}\Big(Q(s_t, a^i_t, a_t^{-i}) - \big(r_t + \gamma Q(s_{t+1}, a^i_{t+1}, a^{-i}_{t+1}) - \alpha \log \pi(a^i_{t+1}|s_{t+1})\big)\Big)^2\Big], \tag{5}$$

where α is the temperature hyperparameter.

Adaptive Opponent-wise Rollout

In this work, a k-step model rollout begins from a real state collected in the real environment. We design the *adaptive*

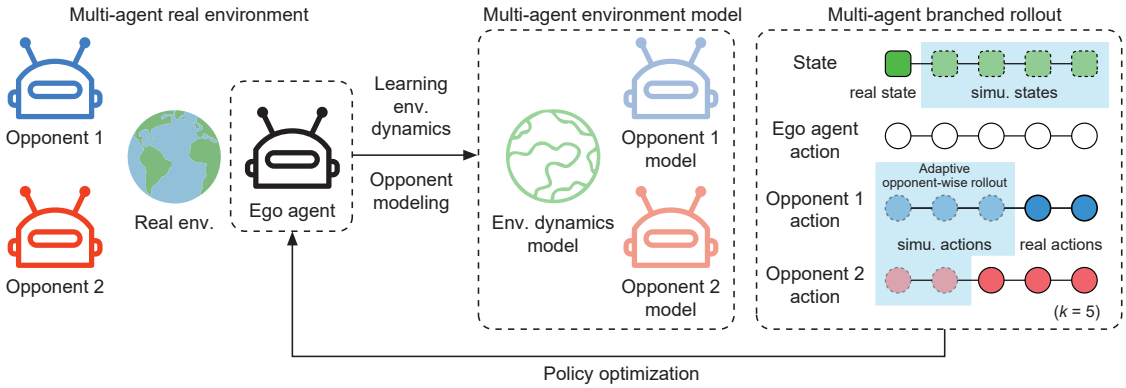

Figure 2: Illustration of the *adaptive opponent-wise rollout policy optimization* (AORPO) method from the perspective of the ego agent. The ego agent begins the rollouts from the states collected from the real environment interacting with real opponents, then runs k steps under the learned dynamics model. For each opponent agent, the ego agent interacts with it directly or with the opponent model for it in the k-step rollout, according to the prediction performance of its opponent model.

opponent-wise rollout scheme to reduce the bound. The opponent model generalization error term $(k+1)\sum_{j\in\{-i\}}\epsilon^j_{\hat{\pi}}$ of the bound in Eq. (3) reveals that different magnitudes of opponent model errors may lead to different contributions to the compounding error of the multi-step simulated rollouts. Intuitively, if too short rollouts are performed, the relatively accurate opponent models are not fully utilized, leading to low sample efficiency. In contrast, if the rollout is too long, the relatively inaccurate opponent models may cause the rollouts to depart from the real trajectory distribution heavily, leading to degraded performance in the environment and thereby low sample efficiency. Thus our proposed adaptive opponent-wise rollout scheme lowers the rollout length of the relatively inaccurate opponent models but keeps long rollout length for relatively accurate opponent models, as shown in Figure 2.

We can achieve this since the agent i can play with either the learned opponent model $\hat{\pi}^j$ or the real policy π^j of each opponent j. In detail, for the opponent j, $\hat{\pi}^j$ is used for the first $n^j = \lfloor k\frac{\min_{j'}\epsilon^{j'}_{\hat{\pi}}}{\epsilon^j_{\hat{\pi}}}\rfloor$ steps, then the agent i interacts with the real opponent π^j in the rest $k-n^j$ steps. The compounding error contribution of each opponent model is thus bounded by $k\min_{j'}\epsilon^{j'}_{\hat{\pi}}$. Note that the adaptive opponent-wise rollout scheme requires the ego agent i can obtain the actions a^j_t from the real policy in the last $k-n^j$ steps. In line 18 of Algorithm 1, the ego agent i sends the simulated states to the opponent j and requires the response, following the predefined communication protocol. One may argue that it will introduce extra opponent sample complexity since we need to communicate with the real opponent in the model rollouts. However, if we only use the opponent models, more errors are exploited, and it will lead to a poor policy and, finally, the sample complexity to achieve some good performance will be high. Our experiments in Section 6 will justify this claim.

With such n^j's, the generalization error caused by the opponent models becomes $\sum_{j\in\{-i\}}(n^j+1)\epsilon^j_{\hat{\pi}} \simeq (n-1)k\min_{j'}\epsilon^{j'}_{\hat{\pi}}$, which is remarkably reduced and makes a good balance of the contribution to overall generalization error from different opponent models. According to the remark of Theorem 2, improving the surrogate return η_i^{branch} with a tighter discrepancy bound will improve the target return η_i more efficiently. Note that the comparison of different model usage schemes is provided in Appendix G.6, where the above scheme yields the highest asymptotic performance and sample efficiency.

5.2 Convergence Guarantee

According to Wei *et al.* [2018], the optimal policy learned by MASQL is $\pi^*_{\text{MASQL}} = \exp\left(\frac{1}{\alpha}Q^*(s_t,\cdot) - V^*(s_t)\right) = \exp\left(\frac{1}{\alpha}Q^*(s_t,\cdot)\right)/\exp\left(V^*(s_t)\right)$, where Q^* and V^* are the optimal Q function and state value function respectively. As Haarnoja *et al.* [2018] showed, the optimal policy learned by MASAC should be $\pi^*_{\text{MASAC}} = \exp\left(\frac{1}{\alpha}Q^*(s_t,\cdot)\right)/Z(s_t)$. Since the partition function is $Z(s_t) = \exp\left(V^*(s_t)\right)$, with given the same optimal Q^* and V^*, MASQL is equivalent to MASAC from the perspective of policy learning. With this fact, we prove that (1) using the learned dynamics model and the opponent models in AORPO still guarantees the convergence of MASQL; (2) MASQL still guarantees the convergence of Nash Q-learning [Hu and Wellman, 2003]. Thus we prove that AORPO achieves the convergence solution of Nash Q-learning. The theorems and proofs can be found in Appendix D, Theorem 6 and Theorem 7.

The theorems guarantee AORPO's convergence under several assumptions. However, we show that the assumptions are not necessary conditions for the learning algorithm to converges by the following experimental results in Figure 4, in which the convergence is still guaranteed when the assumptions are not strictly satisfied. More empirical findings can be found in Yang *et al.* [2018].

6 Experiments

We evaluate the performance of AORPO in both competitive and cooperative tasks. We demonstrate that AORPO can converge to the Nash equilibrium and that AORPO is more sample-efficient than a series of baselines in both competitive and cooperative tasks. See Appendix G for task details.

Compared Methods. We compare our method AORPO with the strong model-free MARL algorithms, MADDPG

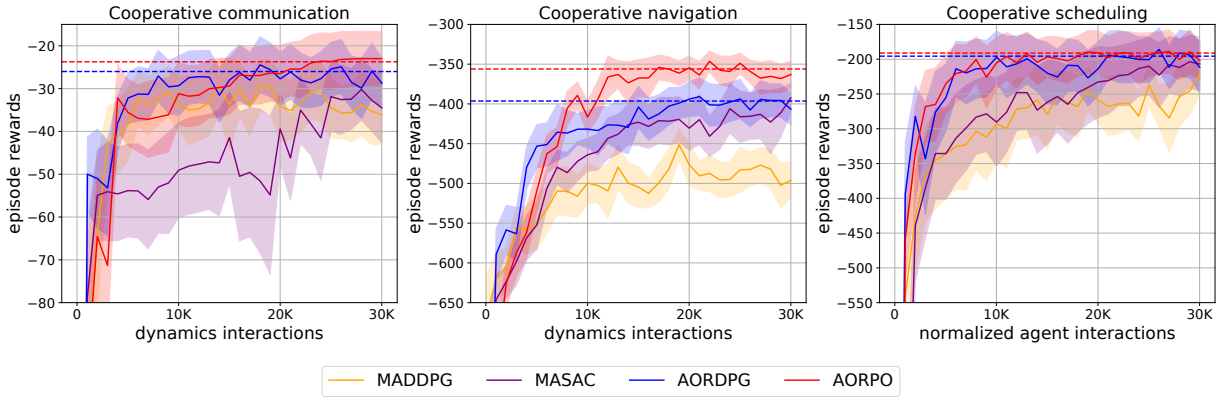

Figure 3: The average episode reward performance of five different random seeds in the cooperative tasks. *Dynamics interactions* means the episodes in the training stage (the agents interacts with the dynamics environment 25 times in an episode). *Normalized agent interactions* means the times the agent groups complete a full interaction, which is divided by 25.

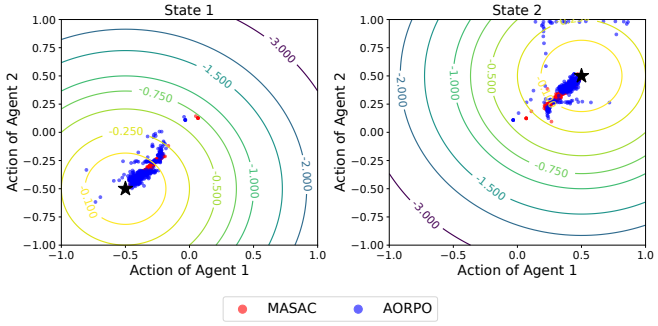

Figure 4: Climb task convergence.

[Lowe *et al.*, 2017] and MASAC. To show that our multi-agent models approach is generally helpful and to have fair comparisons, we also implement AORPO on top of MADDPG, named as AORDPG, for more comprehensive analysis.

Implementation. As for the practical implementation of AORPO, we first collect real experience data using model-free algorithms and use them to pre-train the dynamics model and the opponent models. Other implementation details, including network architectures and important hyperparameters, are provided in Appendix F.

6.1 Competitive Tasks

Climb is a modified *Battle of Sex* game. Agent 1 picks a number $a \in [-1, 1]$, and agent 2 picks a number $b \in [-1, 1]$. Their actions form a position (a, b). There are 2 states and 2 corresponding landmarks located in the lower-left $(-0.5, -0.5)$ and in the upper right $(0.5, 0.5)$ respectively. The only Nash equilibrium is that the agents go to the lower-left landmark at state 1 and go to the upper right landmark at state 2.

The reward surfaces of converged MASAC and AORPO for the two agents in the two states are shown in Figure 4. The result verifies that AORPO will converge to the Nash equilibrium. In Appendix G.3 and G.4, we further provide more detailed results of comparing model-based and model-free methods conducted in several competitive environments of the Multi-Agent Particle Environment [Lowe *et al.*, 2017] and the results of comparing AORPO with another model-based MARL, i.e., MAMSGM [Krupnik *et al.*, 2019].

6.2 Cooperative Tasks

Based on a multi-agent particle environment, we evaluate our method in two types of cooperative tasks: CESO tasks for complex environment dynamics and simple opponents, while SECO tasks for simple environment dynamics and complex opponents. CESO tasks include *Cooperative communication* with two agents and three landmarks and *Cooperative navigation* with three agents and three landmarks. The studied SECO task is *Cooperative scheduling* with three agents and three landmarks, as detailed in Appendix G.

The laws of mechanics and the collisions' stochastic outcomes make difficulties for the dynamics model prediction, so we mainly consider the efficiency of interacting with the environment in the CESO tasks. In the left and middle subfigures of Figure 3, both AORPO and AORDPG can reach the asymptotic performance of the state-of-the-art model-free baselines with fewer interactions with the environment, i.e., achieving lower dynamics sample complexity.

The *Cooperative scheduling* task is configured as a SECO task, with two kinds of agents and two kinds of action spaces. We consider the efficiency of interacting with the other agents in the SECO tasks. As shown in the right subfigure of Figure 3, our methods AORPO and AORDPG reach the asymptotic performance with fewer agents' interactions, which means the opponent sample complexity is lower. One may think using opponent models will indeed introduce extra agent interactions when running rollouts, but interestingly, AORPO uses fewer agent interactions to achieve some performance since it helps policy learn faster. More detailed experiments are presented in Appendix G.

6.3 Analysis on Model Usage

We further discuss the effect of the opponent model usage. Although the generalization errors caused by the opponent models in the return discrepancy are hard to estimate, the opponent models affect the performance through the model compounding errors. We investigate the model compounding

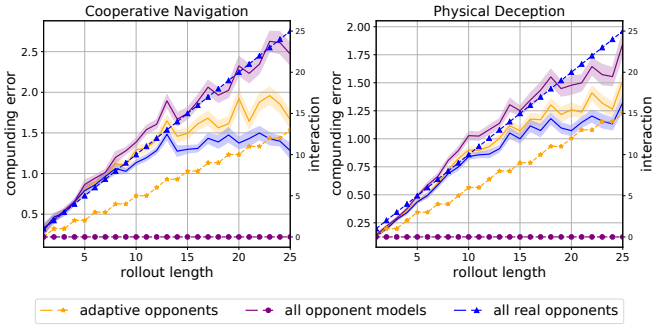

Figure 5: Model compounding errors and interactions numbers of different model usages.

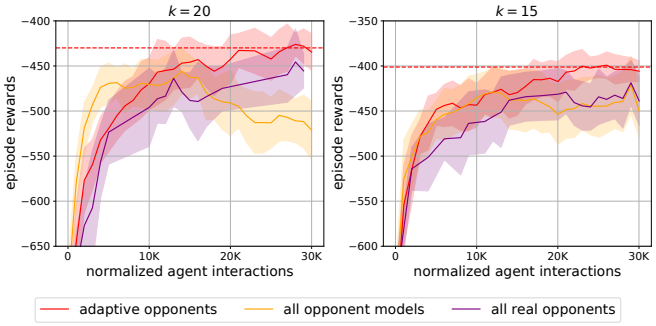

Figure 6: Performance curves of average episode reward of different model usages.

errors when using adaptive opponent-wise rollout or using the opponent models in a whole rollout. Specifically, for a real trajectory $(s_0, a_0, \ldots, s_h)$, a branched rollout from the start state s_0 under learned dynamics model and opponent models is denoted as $(s_0, \hat{a}_0, \ldots, \hat{s}_h)$. Following Nagabandi *et al.* [2018], the model compounding error is defined as $\epsilon_c = \frac{1}{h}\sum_{i=1}^{h} \|s_i - \hat{s}_i\|_2^2$.

Using opponent models reduces interactions among agents but causes extra compounding errors compared with real opponents. We show the trade-off between model errors and interaction times in Figure 5, where the left and right y-axes are for the solid lines and the dotted lines, respectively. We observe the adaptive opponent-wise rollout achieves lower model compounding error than rollout with all opponent models while reducing the number of interactions compared to rollout with real opponents.

Moreover, we investigate how different rollout schemes affect the sample efficiency. In Figure 6, we compare three kinds of usages in a Cooperative navigation scenario using two settings of rollout length k. We notice that using the adaptive opponent-wise rollout achieves the same performance with fewer interactions among the agents than interacting with all real opponents, verifying that although we introduce some sample complexity while simulating data, the reduced discrepancy bound ensures the model-based methods to improve the sample efficiency. It can also avoid performance drop caused by compounding error compared with using all opponent models, which means that the adaptive opponent-wise rollout scheme mitigates some bad opponent models' negative effect by leveraging different rollout lengths of opponent models.

7 Conclusion

In this paper, we investigated model-based MARL problems with both theoretical and empirical analyses. We specified two parts of sample complexity in MARL and derived an upper bound of the return discrepancy in model-based MARL w.r.t. the policy distribution shift and the generalization errors of the learned dynamics model and opponent models. Inspired by the theoretical analysis, we designed the AORPO algorithm framework, in which the return discrepancy can be reduced by adaptive opponent-wise rollout controlling and the two parts of sample complexity are strategically reduced by the learned dynamics and opponent models. We then proved that AORPO can converge to Nash Q-values with reasonable assumptions. In experiments, AORPO has shown highly comparable asymptotic performance with the model-free MARL baselines while achieving higher sample efficiency. For future work, we plan to look deeper into various adaptive opponent-wise rollout schemes for different settings of model-based MARL. We will also investigate theoretical and empirical results of multi-agent dynamics model learning to further ease model-based MARL and improve its sample efficiency.

Acknowledgements

This work is supported by "New Generation of AI 2030" Major Project (2018AAA0100900) and National Natural Science Foundation of China (62076161, 61632017). Xihuai Wang is supported by Wu Wen Jun Honorary Doctoral Scholarship, AI Institute, Shanghai Jiao Tong University. We thank Yaodong Yang for helpful discussion.

References

[Chua *et al.*, 2018] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.

[Clavera *et al.*, 2020] Ignasi Clavera, Yao Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. In *ICLR*, 2020.

[Du and Ding, 2020] Wei Du and Shifei Ding. A survey on multi-agent deep reinforcement learning: from the perspective of challenges and applications. *Artificial Intelligence Review*, 2020.

[Foerster *et al.*, 2018] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *AAMAS*. AAMAS, 2018.

[Haarnoja *et al.*, 2018] 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.

[He *et al.*, 2016] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *ICML*, 2016.

[Hu and Wellman, 2003] Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *JMLR*, (Nov), 2003.

[Iqbal and Sha, 2019] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *ICML*. PMLR, 2019.

[Janner *et al.*, 2019] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *NeurIPS*, 2019.

[Jin *et al.*, 2018] Chi Jin, Zeyuan Allen-Zhu, Sebastien Bubeck, and Michael I Jordan. Is q-learning provably efficient? In *NeurIPS*, 2018.

[Kamra *et al.*, 2020] Nitin Kamra, Hao Zhu, Dweep Trivedi, Ming Zhang, and Yan Liu. Multi-agent trajectory prediction with fuzzy query attention. In *NeurIPS*, 2020.

[Kamthe and Deisenroth, 2018] Sanket Kamthe and Marc Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *AISTATS*, 2018.

[Krupnik *et al.*, 2019] Orr Krupnik, Igor Mordatch, and Aviv Tamar. Multi agent reinforcement learning with multi-step generative models. In *CoRL*, 2019.

[Li *et al.*, 2020] Jiachen Li, Fan Yang, Masayoshi Tomizuka, and Chiho Choi. Evolvegraph: Multi-agent trajectory prediction with dynamic relational reasoning. *NeurIPS*, 2020.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*, 2017.

[Luo *et al.*, 2019] Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *ICLR*, 2019.

[Nagabandi *et al.*, 2018] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*. IEEE, 2018.

[Park *et al.*, 2019] Young Joon Park, Yoon Sang Cho, and Seoung Bum Kim. Multi-agent reinforcement learning with approximate model learning for competitive games. *PloS one*, (9), 2019.

[Qu *et al.*, 2019] Chao Qu, Shie Mannor, Huan Xu, Yuan Qi, Le Song, and Junwu Xiong. Value propagation for decentralized networked deep multi-agent reinforcement learning. In *NeurIPS*, 2019.

[Ryu *et al.*, 2020] Heechang Ryu, Hayong Shin, and Jinkyoo Park. Multi-agent actor-critic with hierarchical graph attention network. In *AAAI*, 2020.

[Shapley, 1953] Lloyd S Shapley. Stochastic games. *PNAS*, (10), 1953.

[Sutton, 1990] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*. Elsevier, 1990.

[Tian *et al.*, 2019] Zheng Tian, Ying Wen, Zhichen Gong, Faiz Punakkath, Shihao Zou, and Jun Wang. A regularized opponent model with maximum entropy objective. *IJCAI*, 2019.

[Tian *et al.*, 2020] Zheng Tian, Shihao Zou, Ian Davies, Tim Warr, Lisheng Wu, Haitham Bou-Ammar, and Jun Wang. Learning to communicate implicitly by actions. In *AAAI*, 2020.

[Wang *et al.*, 2019] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, Shunshi Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[Wei *et al.*, 2018] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. Multiagent soft q-learning. In *AAAI*, 2018.

[Wen *et al.*, 2019] Ying Wen, Yaodong Yang, Rui Luo, Jun Wang, and W Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *ICLR*, 2019.

[Whiteson, 2018] Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi- agent reinforcement learning. *ICML*, 2018.

[Yang *et al.*, 2018] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. In *ICML*, 2018.

[Zhang *et al.*, 2018] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, et al. Fully decentralized multi-agent reinforcement learning with networked agents. In *ICML*, 2018.