Interactive Robust Policy Optimization for Multi-Agent Reinforcement Learning

Videh Raj Nema *
Department of Computer Science & Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore 575025
videhrn25@gmail.com

Balaraman Ravindran
Department of Computer Science
Indian Institute of Technology Madras
Chennai, 600036
ravi@cse.iitm.ac.in

Abstract

As machine learning is applied more to real-world problems like robotics, control of autonomous vehicles, drones, and recommendation systems, it becomes essential to consider the notion of agency where multiple agents with local observations start impacting each other and interact to achieve their goals. Multi-agent reinforcement learning (MARL) is concerned with developing learning algorithms that can discover effective policies in multi-agent environments. In this work, we develop algorithms for addressing two critical challenges in MARL - *non-stationarity* and *robustness*. We show that naive independent reinforcement learning does not preserve the strategic game-theoretic interaction between the agents, and we present a way to realize the classical infinite order recursion reasoning [Schaefer and Anandkumar, 2019] in a reinforcement learning setting. We refer to this framework as Interactive Policy Optimization (IPO) and derive four MARL algorithms using *centralized-training-decentralized-execution* that generalize the widely used single-agent policy gradient methods to multi-agent settings. Finally, we provide a method to estimate opponent's parameters in adversarial settings using maximum likelihood and integrate IPO with an adversarial learning framework to train agents robust to destabilizing disturbances from the environment/adversaries and for better *sim2real* transfer from simulated multi-agent environments to the real world.

1 Introduction

Multi-agent systems consist of multiple intelligent autonomous entities (artificial agents or humans or both) with computational capabilities interacting in an environment with distributed and decentralized information with similar, conflicting, or mixed interests. A fundamental difference from single-agent systems is that here the agents should take into account the presence of other agents in the environment. One of the frameworks or fields of study to solve the problem of learning in multi-agent systems is multi-agent reinforcement learning (MARL). MARL is concerned with developing and analyzing learning rules and algorithms that can discover effective policies in multi-agent settings [Foerster, 2018]. A key characteristic associated with MARL is that the environment is typically *non-stationary*, meaning that for a single agent learning in the environment, all the other agents are non-stationary since they also learn simultaneously. Considering this is essential to develop effective algorithms for MARL. Furthermore, for applying MARL to real-world systems, it is essential to incorporate robustness against (unexpected) destabilizing disturbances from environment and potential adversarial agents. We tackle both the problems in this work. Here we consider two-agent general-sum systems, but our method is generic and can be extended for more than two agents.

*Website: https://vrn25.github.io/. Work done during internship at the Robert Bosch Centre for Data Science and Artificial Intelligence, IIT Madras.

Accepted to the Workshop on Deep Reinforcement Learning at the 35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

A standard optimization workhorse for single-agent reinforcement learning is gradient ascent, where the agent maximizes its objective by following the steepest direction of ascent. This corresponds to maximizing a local linear approximation of the objective, subject to an L_2 quadratic penalty. An intuitive and commonly used extension of this to a multi-agent setting is a local linear approximation of the whole game subject to a quadratic penalty for each agent. However, this seemingly natural choice is flawed as it loses the interactive game-theoretic aspect of the multi-agent problem (discussed in section 2). Considering this, Schaefer and Anandkumar [2019] proposed the optimization framework of competitive gradient descent (CGD) that instead considers a local bilinear approximation of the game objective and is the natural generalization of gradient descent to multi-agent settings. In terms of game theory, CGD recovers the classical infinite order recursive reasoning, which indicates that the agents update their parameters with full awareness of what the other agents are doing.

In this work, we first present the extension of CGD to a multi-agent reinforcement learning setting. We then instantiate and derive a suite of four policy optimization algorithms (stochastic, deterministic, and natural policy gradients) that generalize the existing single-agent reinforcement learning algorithms to any type of cooperative, competitive, or mixed setting using the CGD framework and revert back to standard policy gradients when the interaction between the agents is close to 0. We call these altogether Interactive Policy Optimization (IPO) and use *centralized training* of the agents with *decentralized execution* of the trained policies to ease the learning process. We present a game-theoretic interpretation of IPO, explaining how the bilinear approximation captures interactions between the agents. We then provide a method to estimate the opponent's parameters via maximum likelihood for the application of IPO in adversarial settings and environments where we have access to only the trajectories but not the opponent parameters (hence centralized training is not possible). Following this, we discuss how to adversarially train IPO agents while being robust to potential destabilizing disturbances from the environment or other adversarial agents. We integrate IPO with robust adversarial reinforcement learning (RARL) [Pinto et al., 2017] to propose a novel robust algorithm that can be used for better sim2real transfer of agents trained in a simulated interactive multi-agent environment to real world and as a defense against adversarial attacks at the test time. Finally, we conclude by providing future works.

2 Optimization in Multi-Agent Systems

Let us consider gradient ascent for single-agent systems where f is agent's maximization objective parameterized by $\theta \in \mathbb{R}^m$ and α is the step-size. This can be written as:

$$\theta_{k+1} = \arg\max_\theta f(\theta_k) + (\theta - \theta_k)^\top \nabla_\theta f(\theta_k) - \frac{1}{2\alpha}\|\theta - \theta_k\|_2^2 \tag{1}$$

This corresponds to maximizing a local linear (first-order) approximation of the objective, subject to an L_2 quadratic penalty that expresses the limited confidence in the approximation. This suggests that for multiple agents, the gradient ascent update should be the solution of a local first-order approximation of the full problem, with quadratic regularization terms on each agent that express their limited confidence in this approximation. But *what is the correct notion of local first-order approximation in multi-agent optimization?* Let us consider two agents with the following objectives:

$$\max_{\theta\in\mathbb{R}^m} f(\theta, \phi) \quad , \quad \max_{\phi\in\mathbb{R}^n} g(\theta, \phi) \tag{2}$$

, where $f, g : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ are two agents' objectives[2]. Note that each agent's objective is dependent on both agent's parameters. This is because we need to consider the effect of all agents on each agent's objective in a multi-agent setting. Extrapolating from (1), it is intuitive to use a linear approximation of both agents' objective for multi-agent settings. This corresponds to solving:

$$\begin{aligned}
\theta_{k+1} &= \arg\max_\theta f + (\theta - \theta_k)^\top \nabla_\theta f + (\phi - \phi_k)^\top \nabla_\phi f - \frac{1}{2\alpha}\|\theta - \theta_k\|_2^2 \\
\phi_{k+1} &= \arg\max_\phi g + (\theta - \theta_k)^\top \nabla_\theta g + (\phi - \phi_k)^\top \nabla_\phi g - \frac{1}{2\beta}\|\phi - \phi_k\|_2^2
\end{aligned} \tag{3}$$

[2]From now on, we consider the functions and derivatives to be evaluated at (θ_k, ϕ_k), unless specified otherwise.

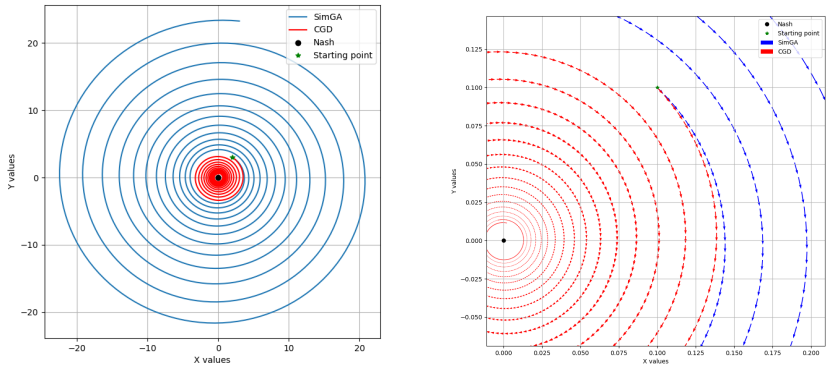

Figure 1: The left figure shows SimGA and CGD on bilinear game $f = -g = \theta^\top \phi, m = n = 1$. SimGA diverges to infinity while CGD converges to the Nash equilibrium $(0, 0)$. The right figure shows direction of gradients starting from the same point. The arrows converge inwards, i.e., towards $(0, 0)$ for CGD and diverge outwards i.e., away from $(0, 0)$ for SimGA. Learning rate used is 0.05.

In (3), the optimal strategy of f is independent of g and vice-versa. Hence it is equivalent to:

$$\begin{aligned} \theta_{k+1} &= \arg\max_\theta f + (\theta - \theta_k)^\top \nabla_\theta f - \frac{1}{2\alpha}\|\theta - \theta_k\|_2^2 \\ \phi_{k+1} &= \arg\max_\phi g + (\phi - \phi_k)^\top \nabla_\phi g - \frac{1}{2\beta}\|\phi - \phi_k\|_2^2 \end{aligned} \tag{4}$$

Let us call (4) simultaneous gradient ascent (SimGA). Although intuitive, this is not the correct generalization of gradient ascent to multi-agent settings. SimGA fails even for simple games like the bilinear game, i.e., $f = -g = \theta^\top \phi$, where $m = n = 1$. As shown in figure 1, θ and ϕ diverge to ∞ when using SimGA. At each iteration, θ and ϕ move farther away from the Nash equilibrium $(0, 0)$. An explanation for the poor convergence properties of SimGA is that the local game in (4) has completely lost the underlying game-theoretic structure and instead consists of both players myopically maximizing their own objective function.

Using SimGA in MARL is straightforward as it would just involve computing the standard policy gradient for each individual agent and myopic parameter updates. We call this independent reinforcement learning (IRL). IRL is a common algorithm for multi-agent optimization and has been used in Bansal et al. [2018] in a self-play manner to show the emergence of complex behavior and skills in two-agent zero-sum games. However, Gleave et al. [2020] show the existence of adversarial policies which can easily fail the agents trained via the naive gradient ascent-based self-play procedure. They argue that if the trained policy was to play a Nash, it would not be exploitable by an adversary. The optimization procedure that Bansal et al. [2018] use to approximate the Nash boils down similar to SimGA, which we saw does not perform well, and hence the agents are exploitable by adversaries. IRL inherits the problems associated with SimGA and hence considers other agents to be stationary in the environment (which is obviously not true).

Let us go back to the question of finding the correct notion of local first-order approximation for multi-agent optimization. As observed before, using linear functions cannot express any interaction (via anticipation of other agents' moves) between the two agents and is thus unable to capture the interactive nature of the underlying problem. As a solution to this, Schaefer and Anandkumar [2019] propose competitive gradient descent or CGD (although we do ascent on the objective in this work, we defer to using the terminology of CGD) which is obtained by considering a bilinear approximation in the two-agent setting, which is the lowest order approximation that captures interaction between the two players. They argue that the natural generalization of gradient ascent to multi-agent optimization is not SimGA but rather CGD. The bilinear approximation uses derivatives up to first-order per agent and is obtained by including the "mixed" hessians (denoted as $D_{\theta\phi} f = \nabla_\theta \nabla_\phi f$) in the local game:

$$\begin{aligned} \theta_{k+1} &= \arg\max_\theta f + (\theta - \theta_k)^\top \nabla_\theta f + (\theta - \theta_k)^\top D_{\theta\phi} f (\phi - \phi_k) - \frac{1}{2\alpha}\|\theta - \theta_k\|_2^2 \\ \phi_{k+1} &= \arg\max_\phi g + (\phi - \phi_k)^\top \nabla_\phi g + (\phi - \phi_k)^\top D_{\phi\theta} g (\theta - \theta_k) - \frac{1}{2\beta}\|\phi - \phi_k\|_2^2 \end{aligned} \tag{5}$$

This local game preserves the interactive aspect of the underlying problem since the optimal action of f depends on the next move of g and vice versa. The mixed hessians $D_{\theta\phi}f, D_{\phi\theta}g$ represent the magnitude of the interaction between the agents. [Schaefer and Anandkumar, 2019] show that among all possibly randomized strategies with finite first moment, the only Nash equilibrium of the game is given by differentiating (5), equating to 0 and solving. Schaefer and Anandkumar [2019] show that there exist only one pair of optimal strategies for (5), and hence we can use these strategies as an update rule, generalizing the idea of optimality from a single to multi-agent setting:

$$\begin{aligned}\theta_{k+1} &= \theta_k + \alpha\left(I - \alpha\beta D_{\theta\phi}f D_{\phi\theta}g\right)^{-1}\left(\nabla_\theta f + \alpha D_{\theta\phi}f\nabla_\phi g\right)\\ \phi_{k+1} &= \phi_k + \beta\left(I - \beta\alpha D_{\phi\theta}g D_{\theta\phi}f\right)^{-1}\left(\nabla_\phi g + \beta D_{\phi\theta}g\nabla_\theta f\right)\end{aligned} \tag{6}$$

The matrix inverses above exist for all but one value of α, β in general-sum games and for all values of α, β in zero-sum games. By arranging (6) in a block matrix form and using the Neumann series to compute the inverse, we can show that the partial sums recover different levels of reasoning in game theory. The first partial sum (each agent thinks the other agent is not changing) recovers SimGA, and the full sum recovers CGD, which corresponds to infinite recursion reasoning in game theory (Appendix **??**). In practice, we use Krylov subspace methods to compute the update in (5).

3 Extension to Multi-Agent Reinforcement Learning

In this section, we discuss how to use the CGD optimization method in multi-agent reinforcement learning. We have seen that IRL does not capture the interactive aspect of a MARL setting. Inspired by Schaefer and Anandkumar [2019], we use CGD as the policy optimization procedure for MARL. In this framework, each agent derives its update with the full consideration of what the other agent's current move and moves in future time steps might be. First, let us define the notation:

We treat the environment with two agents as a finite-horizon MDP, which is defined as $M = \left\{\mathcal{S}, \mathcal{A}^1, \mathcal{A}^2, r^1, r^2, P, \gamma, \rho\right\}$. $\mathcal{S}$ represents the state space. $\mathcal{A}^i$ represents the action space of agent i, where $i \in \{0, 1\}$. The transition dynamics are described by $s' \sim P(\cdot|s, a^1, a^2)$. $r^i(s, a^1, a^2), \gamma \in [0, 1]$, and ρ represent the finite reward function for agent i, discount factor, and initial state distribution respectively. Policy is a mapping from states to distribution over actions, i.e., $\pi : \mathcal{S} \to P(\mathcal{A})$. We consider parameterized policies (θ^i for agent i). We consider a trajectory $\tau = \left(\left(s_t, a_t^1, a_t^2, r_t^1, r_t^2\right)_{t=0}^{T-1}, s_T\right)$. The probability distribution of truncated trajectory is defined as:

$$p(\tau_{0:t}) = \rho(s_0)\pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)P(s_1|s_0,a_0^1,a_0^2)....\pi(a_t^1|s_t;\theta^1)\pi(a_t^2|s_t;\theta^2) \tag{7}$$

Let $p(\tau)$ be the distribution of the full trajectory. The goal of agent i is to maximize the objective:

$$J^i(\theta^1,\theta^2) = \mathbb{E}_{\pi^1,\pi^2,M}\left[\sum_{t=0}^{T-1}\gamma^t r^i(s_t,a_t^1,a_t^2)\right] = \mathbb{E}_{\tau\sim p(\tau)}\left[\sum_{t=0}^{T-1}\gamma^t r^i(s_t,a_t^1,a_t^2)\right] \tag{8}$$

For multi-agent training, we use the paradigm of *centralized-training-decentralized-execution* (CTDE) similar to Lowe et al. [2017]. This notion of multi-agent training is appealing as it allows using extra information to ease training, so long as this information is not used at the test time. This is commonly applicable in various scenarios, e.g., when training is carried out in a simulator while the final policies are later deployed in the real world. We consider an actor network for each agent, which is conditioned on the local observation or state and a critic network for each agent conditioned on the state information and actions of all the agents. The critics are not used at the test time and hence the execution at the test time is decentralized. This allows us to generalize to any type of cooperative, competitive, or mixed games. For agent i, we define the return of the trajectory R_t^i at time step t, state value function V, action value function Q, and advantage function A as:

$$\begin{aligned}R^i(\tau_t) &= \sum_{k=t}^{T-1}\gamma^{k-t}r^i(s_k,a_k^1,a_k^2)\ ,\ Q_\theta^i(s_t,a_t^1,a_t^2) = \mathbb{E}_{\tau\sim p(\tau)}\left[\sum_{k=t}^{T-1}\gamma^{k-t}r^i(s_k,a_k^1,a_k^2)|s_t,a_t^1,a_t^2\right]\\ V_\theta^i(s_t) &= \mathbb{E}_{\tau\sim p(\tau)}\left[\sum_{k=t}^{T-1}\gamma^{k-t}r^i(s_k,a_k^1,a_k^2)|s_t\right], A_\theta^i(s_t,a_t^1,a_t^2) = Q_\theta^i(s_t,a_t^1,a_t^2) - V_\theta^i(s_t)\end{aligned} \tag{9}$$

, where the subscript $\theta = (\theta^1, \theta^2)$ denotes dependence on both agents' parameters. Finally, the update rule for both agents is:

$$\begin{aligned}\theta^1_{k+1} &= \theta^1_k + \alpha \left(I - \alpha\beta D_{\theta^1\theta^2} J^1 D_{\theta^2\theta^1} J^2\right)^{-1} \left(\nabla_{\theta^1} J^1 + \alpha D_{\theta^1\theta^2} J^1 \nabla_{\theta^2} J^2\right) \\ \theta^2_{k+1} &= \theta^2_k + \beta \left(I - \beta\alpha D_{\theta^2\theta^1} J^2 D_{\theta^1\theta^2} J^1\right)^{-1} \left(\nabla_{\theta^2} J^2 + \beta D_{\theta^2\theta^1} J^2 \nabla_{\theta^1} J^1\right)\end{aligned} \tag{10}$$

When the interaction between the agents is 0, i.e., $D_{\theta^1\theta^2} J^1 = 0, D_{\theta^2\theta^1} J^2 = 0$, we get:

$$\theta^1_{k+1} = \theta^1_k + \alpha \nabla_{\theta^1} J^1 \quad , \quad \theta^2_{k+1} = \theta^2_k + \beta \nabla_{\theta^2} J^2 \tag{11}$$

, which is the standard policy gradient update widely used in reinforcement learning [Sutton et al., 1999]. This confirms that IRL does not capture any interactions between the agents and hence is prone to failure in multi-agent settings.

4 Interactive Policy Optimization algorithms

In this section, we discuss different ways to derive practical MARL algorithms that use CGD. To use (10), we require the gradients $\nabla_{\theta^i} J^i$ and the mixed hessians $D_{\theta^i\theta^j} J^i$ for $i, j \in \{1, 2\}, i \neq j$. We instantiate these using 4 algorithms: vanilla stochastic policy optimization, natural policy optimization, trust region policy optimization, and deterministic policy optimization. Altogether, we refer to these algorithms as Interactive Policy Optimization (IPO).

4.1 Vanilla Stochastic Policy Optimization

This instantiation of (10) extends the conventional single-agent vanilla policy gradients [Sutton et al., 1999] to a multi-agent setting. For $i, j \in \{1, 2\}, i \neq j$, the gradients and mixed hessians are given by:

$$\nabla_{\theta^i} J^i = \sum_{t=0}^{T-1} \mathbb{E}_{p(\tau_{0:t})} \left[\gamma^t \nabla_{\theta^i} \log \pi(a^i_t | s_t; \theta^i) Q^i_\theta(s_t, a^1_t, a^2_t)\right] \tag{12}$$

$$D_{\theta^i\theta^j} J^i = \mathbb{E}_{p(\tau)} \left[\sum_{t=0}^{T-1} \gamma^t r^i_t \left(\sum_{k=0}^{t} \nabla_{\theta^i} \log \pi(a^i_k | s_k; \theta^i)\right) \left(\sum_{k=0}^{t} \nabla_{\theta^j} \log \pi(a^j_k | s_k; \theta^j)^\top\right)\right] \tag{13}$$

$$\begin{aligned}D_{\theta^i\theta^j} J^i = &\sum_{t=0}^{T-1} \mathbb{E}_{p(\tau_{0:t})} \left[\gamma^t \nabla_{\theta^i} \log \pi(a^i_t | s_t; \theta^i) \nabla_{\theta^j} \log \pi(a^j_t | s_t; \theta^j)^\top Q^i_\theta(s_t, a^1_t, a^2_t)\right] \\ &+ \sum_{t=1}^{T-1} \mathbb{E}_{p(\tau_{0:t})} \left[\gamma^t \nabla_{\theta^i} \log \pi(a^i_t | s_t; \theta^i) \nabla_{\theta^j} \log \left(\prod_{k=0}^{t-1} \pi(a^j_k | s_k; \theta^j)\right)^\top Q^i_\theta(s_t, a^1_t, a^2_t)\right] \\ &+ \sum_{t=1}^{T-1} \mathbb{E}_{p(\tau_{0:t})} \left[\gamma^t \nabla_{\theta^i} \log \left(\prod_{k=0}^{t-1} \pi(a^i_k | s_k; \theta^i)\right) \nabla_{\theta^j} \log \pi(a^j_t | s_t; \theta^j)^\top Q^i_\theta(s_t, a^1_t, a^2_t)\right]\end{aligned} \tag{14}$$

The proofs are in Appendix **??**. (13) uses higher variance monte-carlo rollouts while (14) uses value functions to lower the variance. (14) has an elegant game-theoretic interpretation. The first term measures the immediate interaction (i.e., at time step t) between two agents multiplied by the quality of the actions (for agent i) taken at time step t (quantified via Q^i). The second term measures the interaction of agent j's behavior up to time step $t-1$ with the reaction of agent i at time step t times Q^i. The third term measures the interaction of agent i's behavior up to time step $t-1$ with the reaction of agent j at time step t times Q^i. This gives an intuitive explanation of the effectiveness of using a bilinear approximation in multi-agent settings, as opposed to a linear approximation.

4.2 Natural and Trust Region Policy Optimization

In this section, we provide an instantiation of the bilinear approximation using the natural gradient method [Amari, 1998]. (10) is the solution to the local game in (5) with bilinear local approximation

of each agent's objective and a Euclidean penalty in the parameter space. Alternatively, we can also use the information geometry by using a penalty in distribution space instead of Euclidean space (natural gradient descent [Amari, 1998]). Prior work in natural policy gradients (NPG) [Kakade, 2001] and trust region policy optimization (TRPO) [Schulman et al., 2015] has highlighted the importance of using such optimization methods in reinforcement learning. For parameterized policies, seemingly small changes in the parameter space can lead to unexpectedly large changes in the policy space, which can lead to a collapse in the performance. To address this problem, we extend the bilinear approximation to natural gradient and trust-region methods. First, let us consider the following relation between policy performance at $\theta = (\theta^1, \theta^2)$ and current policy parameters $\theta_k = (\theta_k^1, \theta_k^2)$:

$$J^i(\theta^1, \theta^2) = J^i(\theta_k^1, \theta_k^2) + \mathbb{E}_{\tau \sim p(\tau_\theta)} \left[\sum_{t=0}^{T-1} \gamma^t A^i_{\theta_k}(s_t, a_t^1, a_t^2) \right] \tag{15}$$

where $p(\tau_\theta), p(\tau_{\theta_k})$ are the trajectory distributions induced by policies $\pi(\theta^1), \pi(\theta^2)$ and $\pi(\theta_k^1), \pi(\theta_k^2)$ respectively. The proof is provided in Appendix **??**. (15) tells that given our current policy parameters θ_k, the advantage at θ_k, and samples from $p(\tau_\theta)$, we can compute $J^i(\theta^1, \theta^2)$ for any θ. However, in practice we want to reuse samples from the current policy and also we might not have access to $p(\tau_\theta)$. Hence we can define a surrogate objective similar to Schulman et al. [2015]:

$$L^i(\theta^1, \theta^2) = J^i(\theta_k^1, \theta_k^2) + \mathbb{E}_{\tau \sim p(\tau_{\theta_k})} \left[\sum_{t=0}^{T-1} \gamma^t \frac{\pi(a_t^1|s_t;\theta^1)}{\pi(a_t^1|s_t;\theta_k^1)} \frac{\pi(a_t^2|s_t;\theta^2)}{\pi(a_t^2|s_t;\theta_k^2)} A^i_{\theta_k}(s_t, a_t^1, a_t^2) \right] \tag{16}$$

(15) uses states and actions sampled from $p(\tau_\theta)$ in the advantage function, while (16) uses states and actions sampled from $p(\tau_{\theta_k})$ but has the same effect as sampling actions from $\pi(\theta^1), \pi(\theta^2)$ due to importance weight correction. The surrogate objective uses trajectories depending on the current policies $\pi(\theta_k^1), \pi(\theta_k^2)$ but actions are sampled from $\pi(\theta^1), \pi(\theta^2)$. Optimizing this surrogate is key to this method (as $\nabla_i L^i = \nabla_i J^i$ at $\theta = \theta_k$). We now derive a bound on how well L^i approximates J^i:

$$J^i(\theta^1, \theta^2) \geq L^i(\theta^1, \theta^2) - \epsilon \sqrt{2 D_{KL}\left(p(\tau_{\theta_k}), p(\tau_\theta)\right)} \tag{17}$$

$\epsilon = \max_s \left| \left[\sum_{t=0}^{T-1} \gamma^t \bar{A}^i_{\theta_k}(s) \right] \right|$, $D_{KL}\left(p(\tau_{\theta_k}), p(\tau_\theta)\right) = \int_\tau p(\tau_{\theta_k}) \log \frac{p(\tau_{\theta_k})}{p(\tau_\theta)} d\tau$ (Appendix **??**).

Using this bound, we can optimize $L^i(\theta^1, \theta^2)$ instead of $J^i(\theta^1, \theta^2)$ provided $p(\tau_\theta)$ is close to $p(\tau_{\theta_k})$, i.e., $D_{KL}\left(p(\tau_{\theta_k}), p(\tau_\theta)\right) \leq \delta$, where δ is a hyperparameter and is small. Hence we solve the following optimization problem:

$$\max_{\theta^1} L^1(\theta^1, \theta^2) \;\; , \;\; \max_{\theta^2} L^2(\theta^1, \theta^2) \;\; \text{subject to } D_{KL}\left(p(\tau_{\theta_k}), p(\tau_\theta)\right) \leq \delta \tag{18}$$

The traditional NPG or TRPO algorithms use a linear approximation of the surrogate objective above. Instead, we use a bilinear approximation to exploit the game-theoretic interaction. Further, we use a quadratic approximation of D_{KL} constraint. Solving using Lagrangian duality, the final update rule for natural policy gradient in block matrix form is (refer Appendix **??** for proof):

$$\begin{bmatrix} \theta^1 - \theta_k^1 \\ \theta^2 - \theta_k^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 F_1 & -D_{\theta^1\theta^2} L^1 \\ -D_{\theta^2\theta^1} L^2 & \lambda_2 F_2 \end{bmatrix}^{-1} \begin{bmatrix} \nabla_{\theta^1} L^1 \\ \nabla_{\theta^2} L^2 \end{bmatrix} \tag{19}$$

$F_i = D_{\theta^i\theta^i} D_{KL}\left(p(\tau_{\theta_k}), p(\tau_\theta)\right)|_{\theta=\theta_k}$ is the Fisher Information Matrix (refer Appendix **??** for derivation). The expressions for $\nabla_i L^i$ and $D_{\theta^i\theta^j} L^i$ can be obtained by differentiating (16) and evaluating at $\theta = \theta_k$. Note that when the mixed hessians are 0 (interaction between the two agents is 0 i.e., $D_{\theta^1\theta^2} L^1 = 0, D_{\theta^2\theta^1} L^2 = 0$), we get back the original NPG algorithm:

$$\theta^1 - \theta_k^1 = \frac{1}{\lambda_1} {F_1}^{-1} \nabla_{\theta^1} L^1 \;\; , \;\; \theta^2 - \theta_k^2 = \frac{1}{\lambda_2} {F_2}^{-1} \nabla_{\theta^2} L^2 \tag{20}$$

For the natural policy gradient extension of CGD, the step sizes λ_1, λ_2 are chosen by substituting $(\theta^1 - \theta_k^1), (\theta^2 - \theta_k^2)$ from (19) into $\bar{D}_{KL} = \frac{1}{2}\left[(\theta^1 - \theta_k^1)^\top F_1 (\theta^1 - \theta_k^1) + (\theta^2 - \theta_k^2)^\top F_2 (\theta^2 - \theta_k^2)\right]$ (refer (**??**)), equating to δ and solving. However, due to various approximations made in the derivation (bilinear approximation of nonlinear L^i and quadratic approximation of nonlinear D_{KL}) and in using conjugate gradient for estimating (19), λ_1, λ_2 computed using the above method can violate the KL-constraint that $D_{KL}\left(p(\tau_{\theta_k}), p(\tau_\theta)\right) \leq \delta$. Hence like TRPO, we introduce a line search on λ_1, λ_2, to make sure we improve the original nonlinear objective while satisfying the nonlinear constraint. After computing $(\theta^1 - \theta_k^1), (\theta^2 - \theta_k^2)$ using (19), we do a line search by starting with the value of λ_1, λ_2 as described above and then exponentially decreasing it until $\bar{D}_{KL} \leq \delta$ and $L^1 \geq 0, L^2 \geq 0$.

4.3 Deterministic Policy Optimization

Here we consider deterministic policies, which are a mapping from states to deterministic actions, i.e., $\mu : \mathcal{S} \rightarrow \mathcal{A}$. Similar to DPG [Silver et al., 2014] and DDPG [Lillicrap et al., 2019], our formulation will work only for continuous action spaces. We consider parameterized policies (θ^i for agent i), which can be a neural network that takes continuous/discrete state as input and outputs the continuous action. The critic network (Q-function) takes in the continuous states and actions and outputs the value of the action. For agent i, we can define the return of the trajectory R_t^i at time step t, state value function V^i, action value function Q^i, and advantage function A^i similar to (9) with the difference that a_t^i will get replaced by $\mu(s_t; \theta^i)$. Here we define the distribution of a truncated trajectory as:

$$q(\tau_{0:t}) = \rho(s_0) \prod_{k=0}^{t-1} P\left(s_{k+1} | s_k, \mu(s_k; \theta^1), \mu(s_k; \theta^2)\right), \text{ with } q(\tau_{0:0}) = \rho(s_0) \tag{21}$$

This instantiation of (10) extends the conventional single-agent deterministic policy gradients [Silver et al., 2014, Lillicrap et al., 2019] to a multi-agent setting. For $i, j \in \{1, 2\}, i \neq j$, the gradients and mixed hessians are given by (the proofs of (22) and (23) can be found in Appendix **??**):

$$\nabla_{\theta^i} J^i = \sum_{t=0}^{T-1} \mathbb{E}_{q(\tau_{0:t})} \left[\gamma^t \nabla_{\theta^i} \mu(s_t; \theta^i) \nabla_{a_t^i} Q_\theta^i(s_t, a_t^1, a_t^2)|_{\substack{a_t^1 = \mu(s_t; \theta^1), \\ a_t^2 = \mu(s_t; \theta^2)}} \right] \tag{22}$$

$$D_{\theta^i \theta^j} J^i = \sum_{t=0}^{T-1} \mathbb{E}_{q(\tau_{0:t})} \left[\gamma^t \nabla_{\theta^i} \mu(s_t; \theta^i) D_{a_t^i a_t^j} Q_\theta^i(s_t, a_t^1, a_t^2)|_{\substack{a_t^1 = \mu(s_t; \theta^1), \\ a_t^2 = \mu(s_t; \theta^2)}} \nabla_{\theta^j} \mu(s_t; \theta^j)^\top \right] \tag{23}$$

5 IPO with Opponent Modeling

In the previous sections, the update rules for vanilla (10) and natural policy optimization (19) require access to the other agent's policy parameters. This poses a problem in applying IPO in adversarial settings, where the opponent's parameters are typically unavailable and have to be inferred from the opponent's state-action trajectories. We encounter such settings in section 6 and hence provide a way to model opponent's behavior for estimating parameters via maximum likelihood estimator as done in [Foerster et al., 2018]. We refer to this as IPO-Opponent Modeling (IPO-OM). For agent 1, the estimation of agent 2's parameters is done via maximizing the likelihood of its trajectories:

$$\hat{\theta}^2 = \arg\max_{\theta^2} \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{t=0}^{T-1} \log \pi(a_t^2 | s_t; \theta^2) \right] \tag{24}$$

We can do vice-versa for agent 2 and use $(\hat{\theta}^1, \hat{\theta}^2)$ for the updates in (10) and (19).

6 Bringing in Robust Control

For the application of IPO to safety-critical real-world systems like autonomous driving, robotics (e.g., RoboCup [Stone et al., 2005]), etc., a promising way is to train policies in simulation and transfer to the real environment (sim2real). This allows using large amounts of data, which is generally not possible by collecting real samples. However, simulated dynamics can be different from the real environment due to inaccuracies in simulator. Furthermore, deployment of agents at the test time can involve presence of adversarial agents that are intentionally trained to fail the deployed agents [Gleave et al., 2020]. To address both these issues, it is essential to train agents to be robust while simultaneously also preserving game-theoretic aspects of the multi-agent problem. Rajeswaran et al. [2017] propose training the agent over a distribution of dynamics parameters, optimizing only for the worst-performing trajectories. Doing so incorporates a form of adversarial training. However, this requires human experts to manually specify distribution over dynamics, which can be challenging.

Alternatively, Pinto et al. [2017] propose the RARL algorithm, which views modeling the differences between training and testing as extra destabilizing forces/disturbances in the system. They train the agent to operate in the presence of a destabilizing adversary that applies disturbance forces to it. The

Algorithm 1 IPO-RARL (IPO embedded in RARL framework)

Input: Environment $\mathcal{E}$; Parameterized policies $\pi(\theta^1)$ and $\pi(\theta^2)$
Initialize: Learnable parameters θ_0^1 for agent 1 and θ_0^2 for agent 2
for i=0,1,..$N_{\text{iter}} - 1$ **do**
  $\theta^{1\prime} \leftarrow \theta_i^1, \theta^{2\prime} \leftarrow \theta_i^2$
  for j=1,2,..N_1 **do**
    $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \text{rollout}\left(\mathcal{E}, \pi(\theta^{1\prime}), \pi(\theta^{2\prime}), N_{\text{traj}}, \text{SUP} = 1, \alpha_1\right)$
    $\theta^{1\prime}, \theta^{2\prime} \leftarrow \text{InteractivePolicyOptimizer}\left(\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\}, \theta^{1\prime}, \theta^{2\prime}\right)$
  end for
  for j=1,2,..N_2 **do**
    $\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\} \leftarrow \text{rollout}\left(\mathcal{E}, \pi(\theta^{1\prime}), \pi(\theta^{2\prime}), N_{\text{traj}}, \text{SUP} = 2, \alpha_2\right)$
    $\theta^{1\prime}, \theta^{2\prime} \leftarrow \text{InteractivePolicyOptimizer}\left(\{(s_t^i, a_t^{1i}, a_t^{2i}, r_t^{1i}, r_t^{2i})\}, \theta^{1\prime}, \theta^{2\prime}\right)$
  end for
  $\theta_{i+1}^1 \leftarrow \theta^{1\prime}, \theta_{i+1}^2 \leftarrow \theta^{2\prime}$
end for
Return: $\theta_{N_{\text{iter}}}^1, \theta_{N_{\text{iter}}}^2$

jointly trained adversary is reinforced (it gets a reward for failing the agent), and it learns an optimal destabilization policy. The policy learning is formulated as a zero-sum minimax game between the agent and the adversary. Inspired by this, we present IPO-RARL (Algorithm 1), which embeds IPO algorithms in the RARL framework. IPO-RARL inherits all properties of IPO and provides robustness against differences in training and testing conditions and against external adversaries.

The procedure is given in Algorithm 1 and consists of two phases inside each outer iteration. Like RARL, we define "superpower" as the ability of an agent to affect the opponent or environment in ways the opponent cannot (e.g., suddenly changing the friction or contact forces by applying forces). We can do so in simulation, which lets one agent to focus on opponent's weak points, thereby bringing adversarial training. We give superpower separately to agent 1 in phase 1 and agent 2 in phase 2 (indicated using SUP) and perform multiple trajectory rollouts. α_i controls the amount of superpower using magnitude of force available ($\alpha_i = 0$ means no superpower and $\alpha_i = 1$ means maximum superpower). An agent with no superpower cannot apply any destabilizing forces. We use IPO algorithms (or IPO-OM when opponent parameters are not accessible) for updating policies in each phase. The agent applying superpower in the corresponding phase gets an additional reward for failing the opponent. The total reward an agent gets is a combination of rewards it can get by using superpower and reward for accomplishing the actual objective of the multi-agent game.

Note that another way to perform adversarial training with IPO is to pretrain the agents using IPO and then apply RARL to robustify each agent separately. However, this could result in the disturbance of the (approximate) Nash equilibrium established between the pretrained parameters trained using IPO. Combining both processes as in algorithm 1, lets us find interactive policies that are also robust to change in dynamics and adversarial disturbances. Moreover, we cannot give superpowers to both the agents in a single phase; otherwise, the agents would simply learn to maximize reward by utilizing superpowers instead of taking meaningful actions (actions that accomplish the multi-agent game objective). This would result in a lack of generalization at the test time when agents do not have superpowers. Our alternating procedure lets an agent use superpower only in its respective phase. Hence to continue getting high rewards, it has to learn to take meaningful as well as robust actions. IPO-RARL also incorporates a form of adversarial training by encouraging the agent with superpower to hammer at the weak points of the other agent and encourages the non-superpower agent to learn robust policies. Hence agent 2 learns robust parameters in phase 1 and agent 1 in phase 2. However, the magnitude of force (the hyperparameter α_i) should be controlled as making it very high makes one agent unreasonably strong, which can destabilize the process.

The adversarial learning procedure described in algorithm 1 is non-conventional since the agent playing the role of the adversary (i.e., having superpowers) keeps switching between phases. This is in contrast to RARL and other common methods where only one of the agents acts as an adversary throughout the training procedure. This is because, unlike RARL, we do not just want to train a single agent to achieve a particular objective in the environment while being robust to disturbances. Instead, we have a multi-agent competitive objective, which can only be achieved via interactive play between

the agents, which is enabled by the game-theoretic aspects of IPO. Giving superpowers to both agents alternatively enables inheriting properties of IPO as well as adversarial training. Additionally, we can gradually anneal the hyperparameter α_i with time so that the agents eventually learn to accomplish the task objective via meaningful actions while simultaneously being robust to disturbances.

In RARL, the number of dimensions and the locations to apply destabilizing forces on are predefined, which is realized by bringing in human intuition. For e.g., in a walker robot, RARL picks specific joints and dimensions such as forces on both feet. We can further automate the process by just fixing the number of dimensions that can be perturbed using superpower and let the agent itself learn the locations (x, y, z coordinates) to apply destabilizing forces on (we just fix the upper limit of force to prevent the agent from becoming unreasonably strong). Hence the agents need to learn an optimal perturbation distribution that would fail the other agent (for which it gets a positive reward), thereby sampling worst-case trajectories. To realize this using neural network policies, in addition to the usual action distribution, each agent outputs mean vector and diagonal covariance of the perturbation distribution. It then samples from this distribution to get the locations to apply disturbance forces, the magnitude of which is controlled using α_i. Doing so gives freedom to the learning algorithm to pick sensitive locations to apply forces on, which human intuition cannot account for.

Finally, the key takeaway from IPO-RARL is that RARL is a container, inside which lies the MARL optimization process. Using standard policy gradients (SimGA) does not capture any interaction between agents, which could result in an attacker training an adversarial policy against our trained agent at test time [Gleave et al., 2020]. However, if the two agents in the multi-agent competition play a Nash equilibrium, no agent would be exploitable by any adversarial strategy. IPO helps us converge better to the Nash equilibrium, and integration with its RARL makes the agents robust, which can be used for sim2real transfer and as a defense against adversaries at test time.

7 Conclusion and Future Work

In this work, we presented an interactive policy optimization (IPO) framework for multi-agent reinforcement learning that preserves game-theoretic nature of the problem and takes into account non-stationarity in multi-agent systems. We showed how CGD can be used in a reinforcement learning setting and how important it is to consider a bilinear approximation of the game objective instead of linear. We then instantiated and derived IPO using stochastic, deterministic, and natural policy gradient methods, presented a maximum likelihood estimator for opponent's parameters and integrated it with RARL for robustification to differences between training in simulation and testing in real environment and adversarial attacks. Further, we discussed IPO-RARL only in competitive zero-sum environments. However, the IPO algorithms and derivations are applicable to more general settings and it is promising to extend it to robustification and sim2real transfer in general-sum environments. It is also promising to scale this work to more than two agents. Finally, we do not foresee any immediate potential negative societal impacts of our work.

References

Florian Schaefer and Anima Anandkumar. Competitive gradient descent. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/56c51a39a7c77d8084838cc920585bd0-Paper.pdf`.

Jakob N Foerster. *Deep multi-agent reinforcement learning*. PhD thesis, University of Oxford, 2018.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2817–2826. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/pinto17a.html`.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. In *ICLR*, 2018.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. In *ICLR*, 2020.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/68a9750337a418a86fe06c1991a1d64c-Paper.pdf.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 1057–1063, Cambridge, MA, USA, 1999. MIT Press.

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural Comput.*, 10(2):251–276, February 1998. ISSN 0899-7667. doi:10.1162/089976698300017746. URL https://doi.org/10.1162/089976698300017746.

Sham Kakade. A natural policy gradient. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, page 1531–1538, Cambridge, MA, USA, 2001. MIT Press.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/silver14.html.

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. In *ICLR*, 2019.

Jakob Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, page 122–130, Richland, SC, 2018. International Foundation for Autonomous Agents and Multiagent Systems.

Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. Reinforcement learning for robocup-soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005. URL http://nn.cs.utexas.edu/?AB05.

Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. In *ICLR*, 2017.