Competitive Policy Optimization Supplementary material

Manish Prajapat[1,3] **Kamyar Azizzadenesheli**[2] **Alexander Liniger**[1] **Yisong Yue**[3] **Anima Anandkumar**[3]

[1]ETH Zurich,
[2]Purdue University,
[3]California Institute of Technology

Abstract

A core challenge in policy optimization in competitive Markov decision processes is the design of efficient optimization methods with desirable convergence and stability properties. We propose competitive policy optimization (CoPO), a novel policy gradient approach that exploits the game-theoretic nature of competitive games to derive policy updates. Motivated by the competitive gradient optimization method, we derive a bilinear approximation of the game objective. In contrast, off-the-shelf policy gradient methods utilize only linear approximations, and hence do not capture players' interactions. We instantiate CoPO in two ways: (i) competitive policy gradient, and (ii) trust-region competitive policy optimization. We theoretically study these methods, and empirically investigate their behavior on a set of comprehensive, yet challenging, competitive games. We observe that they provide stable optimization, convergence to sophisticated strategies, and higher scores when played against baseline policy gradient methods.

1 INTRODUCTION

Reinforcement learning (RL) in competitive Markov decision processes CoMDP [Filar and Vrieze, 2012] is the study of competitive players, sequentially making decisions in an environment. In CoMDPs, the competing agents (players) interact with each other within the environment, and through their interactions, learn how to develop their behavior and improve their policy. In this paper, we consider the rich and fundamental class of zero-sum two-player games.

A core challenge in CoMDP is to design optimization procedures with desirable convergence and stability properties. Policy gradient (PG) is a prominent RL approach that is widely used in single agent optimization and derives policy update using the first order (linear) approximation of the objective function [Robbins and Monro, 1951, Aleksandrov et al., 1968, Sutton et al., 2000]. A straightforward extension of conventional single-agent PG approaches to two-player min-max games results in the gradient descent ascent (GDA) PG algorithm. This approximation is linear in agents' parameters and does not take their interaction into account. Therefore, GDA directly optimizes the policy of each agent, assuming the policy of the opponent is fixed which some times leads to divergence even in simple scenarios and hence considered undesirable in competitive optimization.

We propose a new paradigm, competitive policy optimization (CoPO) for solving two-player CoMDPs. CoPO exploits the game-theoretic and competitive nature of CoMDPs, and, inspired by the competitive gradient descent approach [Schaefer and Anandkumar, 2019], deploys a local bilinear approximation of the game objective to derive policy updates. This local bilinear approximation can be viewed as the simultaneous two-player generalization of the local linear approximation used in single-agent policy gradient approaches (holding the other agent's policy fixed). To compute the policy updates, CoPO computes the Nash equilibrium of the local bilinear approximation of the game objective. In CoPO, each agent derives its update with the consideration of what the other agent's current move and moves in the future time steps might be. In addition, each agent considers how the environment, as the result of the agents' current and future moves, evolves in favor of each agent. Therefore, each agent hypothesizes about what the other agent's and the environment's responses would be, resulting in the *recursive reasoning* in game theory [Keynes, 2018] and temporal recursion in CoMDPs.

We instantiate CoPO in two ways to arrive at practical algorithms. We propose competitive policy gradient (CoPG), a novel PG algorithm that exploits value functions and the structure of CoMDPs to efficiently obtain policy updates. We further extend our approach to the case where each agent does not have direct access to the opponent's policy parameters, and must (approximate) it. We also propose trust region

Supplement for the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021).

competitive policy optimization (TRCoPO), a novel trust region based PG method [Schulman et al., 2015]. TRCoPO updates agents' parameters simultaneously by deriving the Nash equilibrium of a bilinear (in contrast to linear approximation in off-the-shelf trust region methods) approximation to the surrogate objective within a defined trust region in the parameter space.

We empirically validate our approach in several settings. We construct the main empirically study on competitive games, such as Markov soccer, Linear dynamical systems, and Racing cars. We show in all these environments that CoPO leads to superior policies. We observe many cases where standard policy gradient approaches do not exhibit stable learning behavior and can diverge. In our case studies, we show that CoPO can be applied to self-play setting, where an agent is playing against itself. We further show that CoPO can be applied to improve performance in other competitive algorithms such as generative adversarial imitation learning (GAIL) (where one player is the policy and the other is the discriminator) [Ho and Ermon, 2016]. We further extend our case study and show that CoPO remains effective even when one needs to learn a model of the opponent's policy rather than having direct access.

2 PRELIMINARIES

A two player CoMDP is a tuple of $\langle \mathcal{S}, \mathcal{A}^1, \mathcal{A}^2, \mathcal{R}, \mathcal{T}, \mathcal{P}, \gamma \rangle$, where $\mathcal{S}$ is the state space, $s \in \mathcal{S}$ is a state, for player $i \in \{1,2\}$, $\mathcal{A}^i$ is the player i's action space with $a^i \in \mathcal{A}^i$. $\mathcal{R}$ is the reward kernel with probability distribution $R(\cdot|s,a^1,a^2)$ and mean function $r(s,a^1,a^2)$ on $\mathbb{R}$. For a probability measure $\mathcal{P}$, p denotes the probability distribution of initial state, and for the transition kernel $\mathcal{T}$, $T(s'|s,a^1,a^2)$ is the distribution of successive state s' after taking actions a^1, a^2 simultaneously at state s, with discount factor $\gamma \in [0,1]$. We consider episodic environments with reachable absorbing state s_T almost surely in finite time. An episode starts at $s_0 \sim p$, and at each time step $k \geq 0$ at state s_k, each player i draws its action a^i_k according to policy $\pi(a^i_k|s_k;\theta^i)$ parameterized by $\theta^i \in \Theta^i$, where $\Theta^i \subset \mathbb{R}^l$ is a compact metric space. Players $1,2$ receive $(r_k, -r_k)$ with $r_k \sim R(s_k, a^1_k, a^2_k)$, and the environment evolves to a new state s_{k+1}. A realization of this stochastic process is a trajectory $\tau = \big((s_k, a^1_k, a^2_k, r_k)_{k=0}^{|\tau|-1}, s_{|\tau|}\big)$, an ordered sequence with random length $|\tau|$, where $|\tau|$ is determined by episode termination time and state $s_{|\tau|} = s_T$. Let $f(\tau;\theta^1,\theta^2)$ denote the probability distribution of the trajectory τ following players' policies $\pi(\theta^i)$,

$$f(\tau;\theta^1,\theta^2) = p(s_0)\prod_{k=0}^{|\tau|-1}\pi(a^1_k|s_k;\theta^1)\pi(a^2_k|s_k;\theta^2) R(r_k|s_k,a^1_k,a^2_k)T(s_{k+1}|s_k,a^1_k,a^2_k). \quad (1)$$

For $\mathsf{R}(\tau) = \sum_{k=0}^{|\tau|}\gamma^k r(s_k,a^1_k,a^2_k)$, the Q-function, V-functions, and game objective are defined,

$$Q(s_k,a^1_k,a^2_k;\theta^1,\theta^2)=\mathbb{E}_{\tau\sim f(\cdot;\theta^1,\theta^2)}\Big[\sum_{j=k}^{|\tau|-1}\gamma^{j-k}r(s_j,a^1_j,a^2_j)|s_k,a^1_k,a^2_k\Big],$$

$$V(s_k;\theta^1,\theta^2)=\mathbb{E}_{\tau\sim f(\cdot;\theta^1,\theta^2)}\Big[\sum_{j=k}^{|\tau|-1}\gamma^{j-k}r(s_j,a^1_j,a^2_j)|s_k\Big],$$

$$\eta(\theta^1,\theta^2) = \int_\tau f(\tau;\theta^1,\theta^2)\mathsf{R}(\tau)d\tau \quad (2)$$

We assume V, Q, and η are differentiable and bounded in (Θ^1,Θ^2) and for f on (Θ^1,Θ^2), $D_{\theta^i}f = \frac{\partial}{\partial\theta'^i}f(\theta'^1,\theta'^2)\big|_{(\theta'^1,\theta'^2)=(\theta^1,\theta^2)}$, and $D_{\theta^i\theta^j}f = \frac{\partial}{\partial\theta'^i}\big(\frac{\partial}{\partial\theta'^j}f(\theta'^1,\theta'^2)\big)\big|_{(\theta'^1,\theta'^2)=(\theta^1,\theta^2)}$, for $i,j\in\{1,2\}$.

3 COMPETITIVE POLICY OPTIMIZATION

Player 1 aims to maximize the game objective η Eq. (2), and player 2 aims to minimize it, i.e., simultaneously solving for $\max_{\theta^1}\eta(\theta^1,\theta^2)$ and $\min_{\theta^2}\eta(\theta^1,\theta^2)$ respectively with,

$$\theta^{1*} \in \underset{\theta^1\in\Theta^1}{\operatorname{argmax}}\,\eta(\theta^1,\theta^2), \text{ and } \theta^{2*} \in \underset{\theta^2\in\Theta^2}{\operatorname{argmin}}\,\eta(\theta^1,\theta^2). \quad (3)$$

As discussed in the introduction, a straightforward generalization of PG methods to CoMDP, results in GDA (Alg.1). Given players' parameters (θ^1,θ^2), GDA prescribes to optimize a linear approximation of the game objective in the presence of a regularization for the policy updates,

$$\theta^1 \leftarrow \theta^1 + \underset{\Delta\theta^1:\Delta\theta^1+\theta^1\in\Theta^1}{\operatorname{argmax}}\ \Delta{\theta^1}^\top D_{\theta^1}\eta - \frac{1}{2\alpha}||\Delta\theta^1||^2, \text{ and}$$

$$\theta^2 \leftarrow \theta^2 + \underset{\Delta\theta^2:\Delta\theta^2+\theta^2\in\Theta^2}{\operatorname{argmin}}\ \Delta{\theta^2}^\top D_{\theta^2}\eta + \frac{1}{2\alpha}||\Delta\theta^2||^2, \quad (4)$$

where α represent the step size. The parameter updates in Eq. 4 result in greedy updates along the directions of maximum change, assuming the other player stays constant. These updates are myopic, and ignore the agents' interactions. In other words, player 1 does not take player's 2 potential move into consideration and vice versa. While GDA might be an approach of interest in decentralized CoMDP, it mainly falls short in the problem of competitive and centralized optimization in a priori unknown CoMDPs, i.e., the focus of this work. In fact, this behaviour is far from optimal and is shown to diverge in many simple cases e.g. plain bilinear or linear quadratic games [Schaefer and Anandkumar, 2019, Mazumdar et al., 2019]. While single agent policy gradient methods generalize gradient descent [Robbins and Monro, 1951] to single player RL settings, in this paper, we generalize competitive gradient descent [Schaefer and Anandkumar, 2019] to zero-sum RL settings.

We propose competitive policy optimization CoPO, a policy gradient approach for optimization in unknown CoMDPs.

In contrast to standard PG methods, such as GDA, CoPO considers a bilinear approximation of the game objective, and takes the interaction between players into account. Following the competitive gradient updates, CoPO incorporates the game theoretic nature of the CoMDP optimization and derives parameter updates through finding the Nash equilibrium of the bilinear approximation of the game objective,

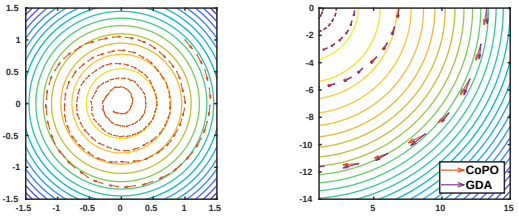
$$\theta^1 \leftarrow \theta^1 + \underset{\Delta\theta^1 : \Delta\theta^1 + \theta^1 \in \Theta^1}{\operatorname{argmax}} \Delta\theta^{1\top} D_{\theta^1} \eta + \Delta\theta^{1\top} D_{\theta^1\theta^2} \eta \Delta\theta^2 - \frac{1}{2\alpha} \|\Delta\theta^1\|^2,$$

$$\theta^2 \leftarrow \theta^2 + \underset{\Delta\theta^2 : \Delta\theta^2 + \theta^2 \in \Theta^2}{\operatorname{argmin}} \Delta\theta^{2\top} D_{\theta^2} \eta + \Delta\theta^{2\top} D_{\theta^2\theta^1} \eta \Delta\theta^1 + \frac{1}{2\alpha} \|\Delta\theta^2\|^2, \quad (5)$$

which has an extra term, the interaction term, in contrast to Eq. 4, and has the following closed-form solution,

$$\theta^1 \leftarrow \theta^1 + \alpha \big(I + \alpha^2 D_{\theta^1\theta^2} \eta D_{\theta^2\theta^1} \eta\big)^{-1} \big(D_{\theta^1} \eta - \alpha D_{\theta^1\theta^2} \eta D_{\theta^2} \eta\big),$$

$$\theta^2 \leftarrow \theta^2 - \alpha \big(I + \alpha^2 D_{\theta^2\theta^1} \eta D_{\theta^1\theta^2} \eta\big)^{-1} \big(D_{\theta^2} \eta + \alpha D_{\theta^2\theta^1} \eta D_{\theta^1} \eta\big), \quad (6)$$



(a) (b)

Figure 1: Bilinear games: (a) CoPO updates towards Nash equilibrium. (b) GDA updates point outward, leading to cycling/divergence.

where I is an identity matrix of appropriate size. Note that, the bilinear approximation in Eq. 5 is still linear in each player's action/parameters. Including any other terms, e.g., block diagonal Hessian terms from the Taylor expansion of the game objective, results in nonlinear terms in at least one player's parameters. As such, CoPO can be viewed as a natural linear generalization of PG. It is known that GDA-style co-learning approaches can often diverge or cycle indefinitely and never converge [Mertikopoulos et al., 2018a]. Fig. 1 shows that for a bilinear game, the gradient flow of GDA cycles or has gradient flow outward, while the CoPG flow, considering players' future moves, has gradient flow toward the Nash equilibrium and converges. In Apx.9, we deploy the Neumann series expansion of the inverses in Eq.5, and show that CoPO recovers the infinite recursion reasoning between players and the environment, while GDA correspond to the first term, and LOLA corresponds to the first two terms in the series. Next, we compute terms in Eq. 6 using the score function $g(\tau, \theta^i) := D_{\theta^i}(\log \prod_{k=0}^{|\tau|-1} \pi(a_k|s_k; \theta^i))$,

Proposition 1. *Given a* CoMDP, *players* $i, j \in \{1, 2\}$, $i \neq j$ *and the policy parameters* θ^i, θ^j,

$$D_{\theta^i} \eta = \int_\tau f(\tau; \theta^1, \theta^2) g(\tau, \theta^i) \mathsf{R}(\tau) d\tau,$$

$$D_{\theta^i\theta^j} \eta = \int_\tau f(\tau; \theta^1, \theta^2) g(\tau, \theta^i) g(\tau, \theta^j)^\top \mathsf{R}(\tau) d\tau.$$

Proof in Apx. 10.1. In practice, we use conjugate gradient and Hessian vector product to efficiently compute the updates in Eq.6, as explained in later sections. A closer look at CoPO shows that this paradigm does not require the knowledge of the environment if sampled trajectories are available. It neither requires full observability of the states, nor any structural assumption on CoMDP, but the Monte Carlo estimation suffer from high variance. In the following, we explicitly take the CoMDP structure into account to develop efficient algorithms.

3.1 COMPETITIVE POLICY GRADIENT

We propose competitive policy gradient (CoPG), an efficient algorithm that exploits the structure of CoMDPs to compute the parameter updates. The following is the CoMDP generalizing of the single agent PG theorem [Sutton et al., 2000]. For $\tau_{l:l'} = (s_k, a_k^1, a_k^2, r_k)_{k=l}^{l'}$, the events from time step l to l', we have:

Theorem 1. *Given a* CoMDP, *players* $i, j \in \{1, 2\}$, $i \neq j$, *and the policy parameters* θ^i, θ^j,

$$D_{\theta^i} \eta = \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k}; \theta^1, \theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k; \theta^i)) Q(s_k, a_k^1, a_k^2; \theta^1, \theta^2) d\tau, \quad (7)$$

$$D_{\theta^i\theta^j} \eta = \mathscr{T}_1 + \mathscr{T}_2 + \mathscr{T}_3. \quad (8)$$

Proof in Apx. 10.2. $\mathscr{T}_1$, $\mathscr{T}_2$ and $\mathscr{T}_3$ are described in Table 1. This theorem indicates that the terms in Eq. 6 can be computed using Q function. In Eq. 8, $\mathscr{T}_1$ is the immediate interaction between players, $\mathscr{T}_2$ is the interaction of player i's behavior up to time step k with player j's reaction at time step k, and the environment. $\mathscr{T}_3$ is the interaction of player j's behavior upto time step k with player i's reaction at time step k, and the environment.

CoPG operates in epochs. At each epoch, CoPG deploys $\pi(\theta^1)$, $\pi(\theta^2)$ to collect trajectories, exploits them to estimate the Q, $D_{\theta^i}\eta$, and $D_{\theta^i\theta^j}\eta$. Then CoPG follows the parameter updates in Eq. 6 and updates (θ^1, θ^2), and this process goes on to the next epoch (Alg.2). Many variants of PG approach uses baselines, and replace Q with, the advantage function $A(s, a^1, a^2; \theta^1, \theta^2) = Q(s, a^1, a^2; \theta^1, \theta^2) - V(s; \theta^1, \theta^2)$, Monte Carlo estimate of Q-V [Baird, 1993], empirical TD error or generalized advantage function

Table 1: Notations for the bilinear term in the competitive policy theorem

Symbol	Formulation
$\mathscr{T}_1:$	$\int_\tau \sum_{k=0}^{\lvert\tau\rvert-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log\pi(a_k^i\lvert s_k;\theta^i)) D_{\theta^j}(\log\pi(a_k^j\lvert s_k;\theta^j))^\top Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau$
$\mathscr{T}_2:$	$\int_\tau \sum_{k=1}^{\lvert\tau\rvert-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log\prod_{l=0}^{k-1}\pi(a_l^i\lvert s_l;\theta^i)) D_{\theta^j}(\log\pi(a_k^j\lvert s_k;\theta^j))^\top Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau$
$\mathscr{T}_3:$	$\int_\tau \sum_{k=1}^{\lvert\tau\rvert-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log\pi(a_k^i\lvert s_k;\theta^i)) D_{\theta^j}(\log\prod_{l=0}^{k-1}\pi(a_l^j\lvert s_l;\theta^j))^\top Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau$

(GAE) [Schulman et al., 2016]. Our algorithm can be extended to this set up and in Apx. 11 we show how CoPG can be accompanied with all the mentioned variants.

3.2 OPPONENT PARAMETER ESTIMATION

In some settings, each learner does not have access to the opponent's policy. To apply CoPG in such settings, one natural approach is to estimate online the opponent's policy by the opponent's state-action pairs, as proposed in [Foerster et al., 2017b]. We thus propose a variant of CoPG that also infers the opponent's policy parameters (CoPG-OP), where each agent i also estimates the parameters $\hat{\theta}^j$ of its opponent j's policy, e.g., using maximum-likelihood estimator,

$$\hat{\theta}^j = \text{argmax}_{\theta^j}\, \mathbb{E}_{\tau\sim f(\cdot,\theta^1,\theta^2)} \sum_{k=0}^{\lvert\tau\rvert-1} \log\pi(a_k^j\lvert s_k,\theta^j) \quad (9)$$

Then, the agent utilizes $\hat{\theta}^j$ to derive its policy updates in Eq. 7, 8, in place of θ^j. In our empirical study, we observe that CoPG-OP training is as stable as CoPG and the policies learned using CoPG-OP are as competent as CoPG (refer to Apx. 16). We conclude that opponent parameter learning can be considered effective in online settings, which confirms the observation in [Foerster et al., 2017b].

3.3 TRUST REGION COMPETITIVE POLICY OPTIMIZATION

Trust region based policy optimization methods exploit the local Riemannian geometry of the parameter space to derive more efficient policy updates [Kakade and Langford, 2002, Kakade, 2002, Schulman et al., 2015]. In this section, we propose trust region competitive policy optimization (TRCoPO), the CoPO generalization of TRPO [Schulman et al., 2015], that exploits the local geometry of the competitive objective to derive more efficient parameter updates.

Lemma 1. *Given the advantage function of policies $\pi(\theta^1),\pi(\theta^2)$, $\forall(\theta'^1,\theta'^2)\in\Theta^1\times\Theta^2$ we have,*

$$\eta(\theta'^1,\theta'^2) = \eta(\theta^1,\theta^2) + \mathbb{E}_{\tau\sim f(\cdot;\theta'^1,\theta'^2)} \sum_{k=0}^{\lvert\tau\rvert-1} \gamma^k A(s,a^1,a^2;\theta^1,\theta^2). \quad (10)$$

Proof in Apx. 13.1. Eq.10 indicates that, considering our current policies $(\pi(\theta^1),\pi(\theta^2))$, having access to their advantage function, and also samples from $f(\cdot;\theta'^1,\theta'^2)$ of any (θ'^1,θ'^2) (without rewards), we can directly compute $\eta(\theta'^1,\theta'^2)$ and optimize for (θ'^1,θ'^2). However, in practice, we might not have access to $f(\cdot;\theta'^1,\theta'^2)$ for all (θ'^1,θ'^2) to accomplish the optimization task, therefore, direct use of Eq.10 is not favorable. Instead, we define a surrogate objective function, $L_{\theta^1,\theta^2}(\theta'^1,\theta'^2)$,

$$L_{\theta^1,\theta^2}(\theta'^1,\theta'^2) = \eta(\theta^1,\theta^2) + \mathbb{E}_{\tau\sim f(\cdot;\theta^1,\theta^2)}\Big[\sum_{k=0}^{\lvert\tau\rvert-1}\gamma^k \mathbb{E}_{\pi(a_k'^1\lvert s_k;\theta'^1),\pi(a_k'^2\lvert s_k;\theta'^2)} A(s_k,a_k'^1,a_k'^2;\theta^1,\theta^2)\Big], \quad (11)$$

which can be computed using trajectories of our current polices $\pi(\theta^1),\pi(\theta^2)$. $L_{\theta^1,\theta^2}(\theta'^1,\theta'^2)$ is an object of interest in PG [Kakade and Langford, 2002, Schulman et al., 2015] since mainly its gradient is equal to that of $\eta(\theta^1,\theta^2)$ at (θ^1,θ^2), and as stated in the following theorem, it can carefully be used as a surrogate of the game value. For KL divergence $D_{KL}((\theta^1,\theta^2),(\theta'^1,\theta'^2)) := \int_\tau f(\tau,\theta^1,\theta^2)\log\big(f(\tau,\theta^1,\theta^2)/f(\tau,\theta'^1,\theta'^2)\big)d\tau$, we have,

Theorem 2. *$L_{\theta^1,\theta^2}(\theta'^1,\theta'^2)$ approximate $\eta(\theta'^1,\theta'^2)$ up to the following error upper bound, with constant ϵ*

$$\big\lvert\eta(\theta'^1,\theta'^2) - L_{\theta^1\theta^2}(\theta'^1,\theta'^2)\big\rvert \le \epsilon/\sqrt{2}\sqrt{D_{KL}((\theta'^1,\theta'^2),(\theta^1,\theta^2))}, \quad (12)$$

$$\epsilon := \max_\tau \sum_k^{\lvert\tau\rvert} \gamma^k \mathbb{E}_{\pi(a_k'^1\lvert s_k;\theta'^1),\pi(a_k'^2\lvert s_k;\theta'^2)} A(s_k,a_k'^1,a_k'^2;\theta^1,\theta^2).$$

Proof in Apx. 13.2. This theorem states that we can use $L_{\theta^1\theta^2}(\theta'^1,\theta'^2)$ to optimize for $\eta(\theta'^1,\theta'^2)$ as long as we keep the (θ'^1,θ'^2) in the vicinity of θ^1,θ^2. Similar to single agent TRPO [Schulman et al., 2015], we optimize for $L_{\theta^1\theta^2}(\theta'^1,\theta'^2)$, while constraining the KL divergence, $D_{KL}((\theta^1,\theta^2),(\theta'^1,\theta'^2)) \le \delta'$, i.e.,

$$\max_{\theta'^1\in\Theta^1}\min_{\theta'^2\in\Theta^2} L_{\theta^1\theta^2}(\theta'^1,\theta'^2), \text{with } D_{KL}((\theta^1,\theta^2),(\theta'^1,\theta'^2)) \le \delta'. \quad (13)$$

The GDA generalizing of TRPO uses a linear approximation of $L_{\theta^1\theta^2}(\theta'^1,\theta'^2)$ to approach this optimization, which again ignores the players' interactions. In contrast, TRCoPO considers the game theoretic nature of this optimization, and uses a bilinear approximation. TRCoPO operates in epochs.

At each epoch, TRCoPO deploys $(\pi(\theta^1), \pi(\theta^2))$ to collect trajectories, exploits them to estimate A (or GAE), then updates parameters accordingly, Alg.4. (For more details, please refer to Apx. 13.3.)

4 EXPERIMENTS

We empirically study the performance of CoPG and TRCoPO and their counterparts GDA and TRGDA, on six games, ranging from single-state repeated games to general sequential games, and tabular games to infinite/continuous high dimensional states/action games. They are 1) linear-quadratic(LQ) game, 2) bilinear game, 3) matching pennies (MP), 4) rock paper scissors (RPS), 5) Markov soccer, and 6) car racing. These games are representative enough that their study provides insightful conclusions, and challenging enough to highlight the core difficulties and interactions in competitive games.

We show that CoPG and TRCoPO converge to stable points, and learn opponent aware strategies, whereas GDA's and TRGDA's greedy approach shows poor performance and even diverge in bi-linear, MP, and RPS games. For the LQ game, when GDA does not diverge, it almost requires 1.5 times the amount of samples, and is 1.5 times slower than CoPG. In highly strategic games, where players' policies are tightly coupled, we show that CoPG and TRCoPO learn much better interactive strategies. In the soccer game, CoPG and TRCoPO players learn to defend, dodge and score goals, whereas GDA and TRGDA players learn how to score when they are initialized with the ball, and give way to the other player otherwise. In the car racing, while GDA and TRGDA show poor performance, CoPG and TRCoPO produce competing players, which learn to block and fake each other. Overall, we observe CoPG and TRCoPO considerably outperform their counterparts in terms of convergence, learned strategies, and gradient dynamics.

We implemented all algorithms in Pytorch [Paszke et al., 2019], and made the code and the videos public[1]. In our implementation, we deploy Pytorch's autograd package and Hessian vector product to efficiently obtain gradients and Hessian vector products to compute the bilinear terms in the optimizer. Moreover, we use the conjugate gradient trick to efficiently computed the inverses-matrix vector product in Eq.6 which incurs a minimal computational overhead (see [Shewchuk, 1994] for more details). To improve computation times, we compute inverse-matrix vector product only for one player strategy, and use optimal counter strategy for other player $\Delta\theta^2$ which is computed without an inverse matrix vector product. Also, the last optimal strategy can be used to warm start the conjugate gradient method which improves convergence times. We provide efficient implementation for both CoPO- and GDA-based methods, where CoPO incurs 1.5 times extra computation per batch.

Zero-sum LQ game is a continuous state-action linear dynamical game between two players, where GDA, with considerably small learning rate, has favorable convergence guarantees [Zhang et al., 2019]. This makes the LQ game an ideal platform to study the range of allowable step sizes and convergence rate of CoPG and GDA. We show that, with increasing learning rate, GDA generates erratic trajectories and policy updates, which cause instability (see "⊘" in Table 4), whereas CoPG is robust towards this behavior. Fig. 2a shows that CoPG dynamics are not just faster at the same learning rate but more importantly, CoPG can potentially take even larger steps.

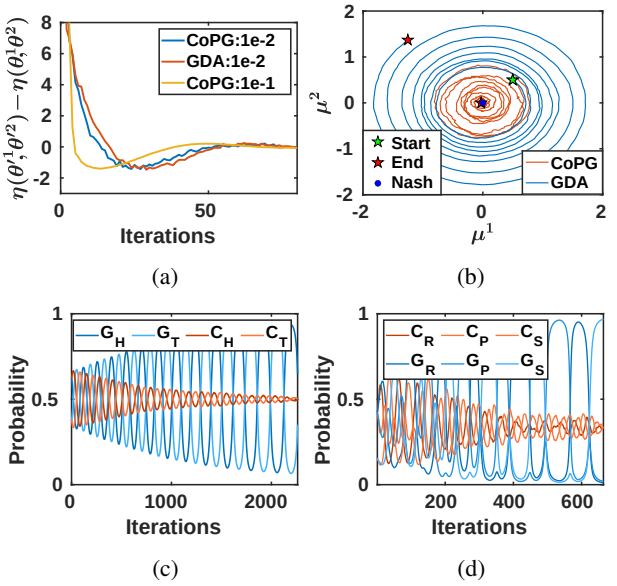


Figure 2: During training CoPG (C) vs GDA (G) on a) LQ game, difference in game objective due to policy update for $\alpha = 1e-1, 1e-2$ b) Bilinear game, μ^1 vs μ^2 c) MP, probability of selecting Head(H) and Tail(T) d) RPS, probability of selecting rock(R), paper(P) and scissors(S).

Bilinear game is a state-less strongly non-cooperative game, where GDA is known to diverge [Balduzzi et al., 2018]. In this game, reward $r(a^1, a^2) = \langle a^1, a^2 \rangle$ where $a^1 \sim \mathcal{N}(\mu_1, \sigma_1)$ and $a^2 \sim \mathcal{N}(\mu_2, \sigma_2)$, and (μ_1, σ_1), (μ_2, σ_2) are policy parameters. We show that GDA diverges for all learning rates, whereas CoPG converges to the unique Nash equilibrium Fig. 2b.

Matching pennies and **Rock paper scissors**, are challenging matrix games with mixed strategies as Nash equilibria, demand opponent aware optimization.[2] We show that CoPG and TRCoPO both converge to the unique Nash equilibrium of MP Fig. 2c and RPS Fig. 2d, whereas GDA and TRGDA

[1]Link to Videos: https://sites.google.com/view/rl-copo

[2]Traditionally, fictitious and counterfactual regret minimization approaches have been deployed [Neller and Lanctot, 2013].

diverges (to a sequence of polices that are exploitable by deterministic strategy). Detailed empirical study, formulation and explanation of these 4 games can be found in Apx. 15.

Markov soccer game, Fig. 4, consists of players A and B that are randomly initialised in the field, that are supposed to pick up the ball and put it in the opponent's goal [Littman, 1994, He et al., 2016]. The winner is awarded with +1 and the loser with -1 (see Apx. 15.5 for more details).

Since all methods converge in this game, it is a suitable game to compare learned strategies. In this game, we expect a reasonable player to learn sophisticated strategies of defending, dodging and scoring. For each method playing against their counterparts, e.g., CoPG against CoPG and GDA, Fig. 3 shows the number of times the ball is seized between the players before one player finally scores a goal, or time-out. In (3a), CoPG vs CoPG, we see agents seize ball, twice the times as compared to (3b),GDA vs GDA (see A_2 and B_2 in (3a) and (3b)). In the matches CoPG vs GDA (3c), CoPG trained agent could seize the ball from GDA agent (A_2) nearly 12 times more due to better seizing and defending strategy, but GDA can hardly take the ball back from CoPG (B_2) due to a better dodging strategy of the CoPG agent. Playing CoPG agent against GDA one, we observe that CoPG wins more than 74% of the games Fig. 3d. We observe a similar trend for trust region based methods TRCoPO and TRGDA, playing against each other (a slightly stronger results of 85% wins, A_2 column in Figs 3g, 3c).

We also compared CoPG with MADDPG [Lowe et al., 2017] and LOLA [Foerster et al., 2017b]. We observe that the MADDPG learned policy behaves similar to GDA, and loses 80% of the games to CoPG's (Fig. 15b). LOLA learned policy, with its second level reasoning, performs better than GDA, but lose to CoPG 72% of the matches. For completeness, we also compared GDA-PG, CoPG, TRGDA, and TRCoPO playing against each other. TRCoPO performs best,CoPG was runner up, then TRGDA, and lastly GDA (see Apx. 15.5).

Car Racing is another interesting game, with continuous state-action space, where two race cars competing against each other to finish the race first [Liniger and Lygeros, 2020]. The game is accompanied by two important challenges, 1) learning a policy that can maneuver the car at the limit of handling, 2) strategic interactions with opponents. The track is challenging, consisting of 13 turns with different curvature (Fig. 6). The game is formulated as a zero-sum, with reward $r(s_k, a_k^1, a_k^2) = \Delta\rho_{car_1} - \Delta\rho_{car_2}$, where $\Delta\rho = \rho_{k+1} - \rho_k$ and ρ_k is the car's progress along the track at the k^{th} time step. If a car crosses track boundaries (e.g., hit the wall), it is penalized, and the opponent receives rewards, this encourages cars to play rough and push each other into the track boundaries. When a collision happens, the rear car is penalized, and the car in the front receives a reward; it promotes blocking by the car in front and overtaking by the car in the rear. We study agents trained with all GDA, TRGDA, MADDPG, LOLA, CoPG and TRCoPO in this game, and show that even though all players were able to learn to "drive" only CoPG and TRCoPO were able to learn how to "race". Using GDA, only one player was able to learn, which manifested in either one player completely failing and the other finishing the track (Fig. 5a), or by an oscillation behavior where one player learns to go ahead, the other agent stays at lower progress(Fig. 5c, Fig. 5e). Even if one of the players learns to finish one lap at some point, this player does not learn to interact with its opponent(https://youtu.be/rxkGW02GwvE). In contrast, players trained with CoPG and TRGDA, both progress, learn to finish the lap, and race (interact with each other) (See Fig. 5b and Fig. 5d). To test the policies, we performed races between CoPG and GDA, TRCoPO and TRGDA, and CoPG and TRCoPO. As shown in Table 2, CoPG and TRCoPO win almost all races against their counterparts. Overall, we see that both CoPG and TRCoPO are able to learn policies that are faster, more precise, and interactive with the other player (e.g., learns to overtake).

Table 2: Trained agents competing against each others in car racing. Ratio of Wins(W), Overtakes(O) and Collisions(C).

	CoPG v GDA		TRCoPO v TRGDA		CoPG v TRCoPO	
W	**1**	0	**1**	0	0.24	**0.76**
O	**1.28**	0.78	**1.28**	0.78	1.80	**2.07**
C	**0.17**	16.11	**0.25**	1.87	**0.30**	0.31

4.1 CASE STUDIES

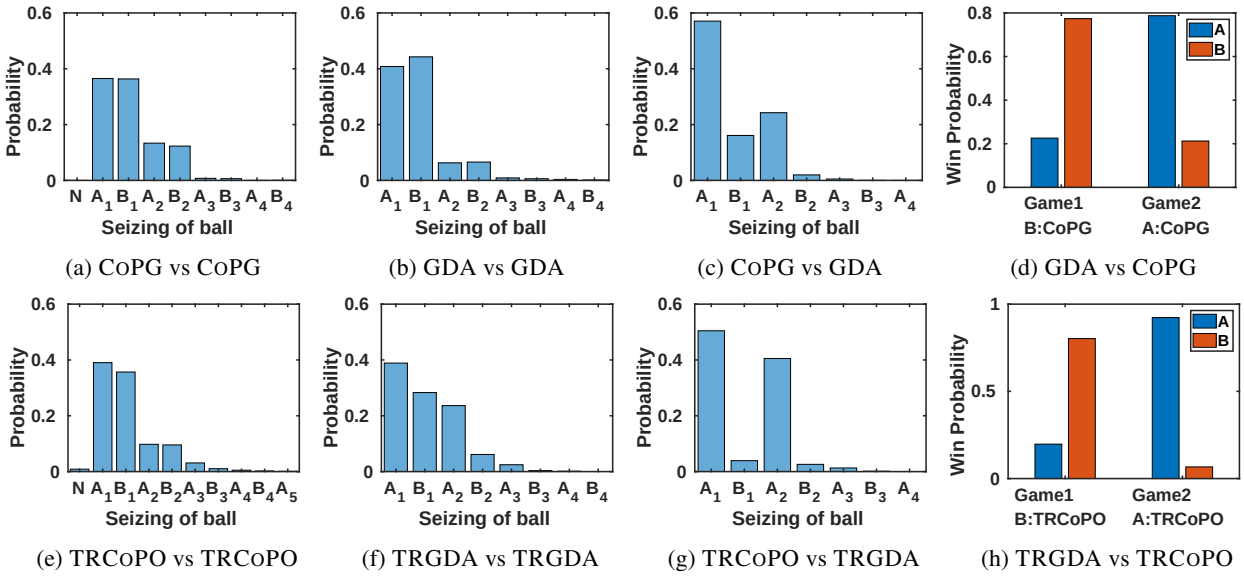
Generative Adversarial Imitation Learning: One can also apply our approach in asymmetric games, such as learning the agent policy and the discriminator in generative adversarial imitation learning (GAIL) [Ho and Ermon, 2016]. In GAIL, to imitate the expert, the agent (θ player) plays a game with a discriminator $\mathbb{D}$ (ϕ player), i.e.,

$$\min_{\theta}\max_{\phi}\mathbb{E}_{\tau_\theta}[\log\mathbb{D}_\phi(s,a)]+\mathbb{E}_{\tau_e}[\log(1-\mathbb{D}_\phi(s,a))]-\lambda H(\theta),$$

where τ_θ is a trajectory and $H(\theta)$ is the casual entropy.

We conduct this study on the car racing game with a single car, where the aim is to learn to drive a full lap. Given a long track (Fig. 6), exploration and reward formulation can be challenging. We train the agent using CoPO to learn to drive by imitating an expert. The expert trajectories are collected using a pure pursuit (PP) controller [Coulter, 1992] with different sets of parameters Apx. 17. We evaluate agent's policy using lap time (t^{lap}), and:

$$\zeta = E_{\tau_\theta}[\sum_{k}^{|\tau|-1}\|a_k - a_k^e\|^2/|\tau|], \tag{14}$$



(a) CoPG vs CoPG (b) GDA vs GDA (c) CoPG vs GDA (d) GDA vs CoPG

(e) TRCoPO vs TRCoPO (f) TRGDA vs TRGDA (g) TRCoPO vs TRGDA (h) TRGDA vs TRCoPO

Figure 3: a-c), e-g) Interaction plots, representing probability of seizing ball in the game between A vs B. X-axis convention, for player A. A_1: A scored goal, A_2: A scored goal after seizing ball from B, A_3: A scored goal by seizing ball from B which took the ball from A and so forth. Vice versa for player B. N: No one scored goal both kept on seizing ball. d),h) Probability of games won.

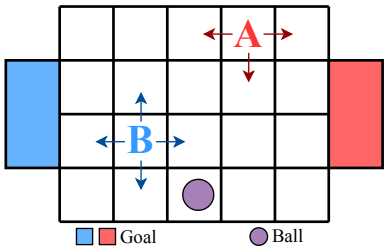


Figure 4: Markov Soccer

where a_k, a_k^e are the agent and the expert actions evaluated at the same state s_k, collected from an agent rollout. We compare the results of CoPO-GAIL with GDA-GAIL, Behaviour cloning (BC) [Ho and Ermon, 2016] and controllers such as PID and PP. The agent trained with CoPO learns to follow the reference path of the expert and even drives better than the average expert policy, achieving a performance similar to the best expert (https://youtu.be/DtGWZubjcf4). The CoPO-based agent achieves the lowest ζ value and learns a better imitation policy compared to GDA and BC Table 3.

Table 3: Comparing lap time and ζ (Eq. 14) of the policies learnt using imitation learning and the baseline controllers.

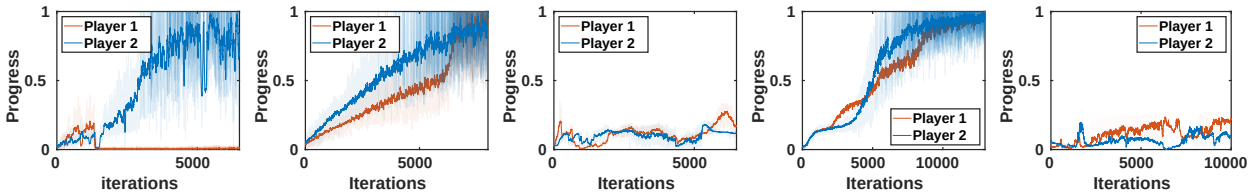
score	PP_{avg}	CoPO-GAIL	GDA-GAIL	PID	BC
ζ x 10^{-2}	0	0.82	1.14	2.35	2.02
$t^{lap}(s)$	13.07	11.82	12.18	14.67	DNF

Opponent learning: We next explore the case where one does not directly have access to the opponent's parameters and they have to be inferred through interaction with the other agent and the environment. We propose to use CoPG-OP (Sec. 3.2), the opponent learning variant of CoPG.

Fig. 7 shows interaction plots of CoPG-OP conducted on the Markov Soccer game (setting explained in Apx. 15.5). The opponent's parameters are estimated by observing state-action pairs of the opponent using Eq. 9. The interaction plot of the CoPG-OP agent with the estimated opponent (Fig. 7a) shows that the CoPG-OP player learns to seize the ball and interact with the opponent(A_3, A_4). Fig. 18b shows the interaction plot of CoPG-OP with CoPG, where we observe that CoPG-OP also learns a policy similar to CoPG, which is able to defend, escape and score goals (A_3). When directly competing with CoPG, we observe that CoPG-OP can win 46.5% of the games against CoPG.

The experiment details and numerical results on other setups can be found in Apx. 16. They show that CoPG-OP achieves the performance of CoPG in terms of stability and convergence to sophisticated strategies.

Training by Self-play: In self-play, one player plays against itself using the same policy model for both players. Each player then samples actions from this policy for their respective state and updates the policy using CoPG-SP (Alg. 5) which is a special case of CoPG. We observe that CoPG-SP can successfully learn competing strategies similarly to CoPG. We provide the CoPG-SP algorithm and a detailed empirical study in Apx. 12.



(a) GDA (b) CoPG (c) TRGDA (d) TRCoPO (e) MADDPG

Figure 5: Normalised progress of agents in one lap of car racing game during training

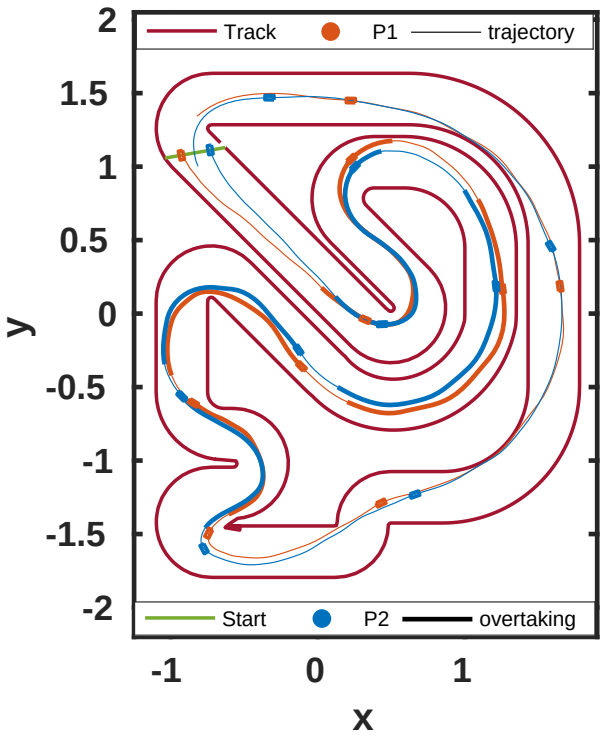
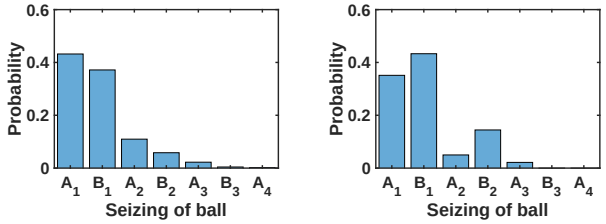


Figure 6: Overtaking maneuvers of CoPO agents in the Car Racing Game. The thin line shows the trajectory of the player in the game and the thick line shows trajectory when the trailing agent overtook.



(a) CoPG-OP (A) vs (B Est A) (b) CoPG-OP vs CoPG

Figure 7: Interaction plots evaluated by playing 5000 games. Matches played between a) CoPG-OP player A and player B estimated by A b) CoPG-OP and CoPG

5 RELATED WORK

In tabular CoMDP, Q-learning and actor-critic have been deployed [Littman, 1994, 2001b,a, Bowling and Veloso, 2002, Greenwald and Hall, 2003, Hu and Wellman, 2003, Frénay and Saerens, 2009, Pérolat et al., 2018, Srinivasan et al., 2018], and recently, deep RL methods have been extending to CoMDPs, with focus on modeling agents behaviour [Tampuu et al., 2017, Leibo et al., 2017, Raghu et al., 2017]. To mitigate the stabilization issues, centralized methods [Matignon et al., 2012, Lowe et al., 2017, Foerster et al., 2017a], along with opponent's behavior modeling [Raileanu et al., 2018, He et al., 2016] have been explored. Optimization in multi-agent learning can be interpreted as a game in the parameter space, and the main body of the mentioned literature does not take this aspect directly into account since they attempt to separately improve players' performance. Hence, they often fail to achieve desirable performance and oftentimes suffer from unstable training, especially in strategic games [Hernandez-Leal et al., 2019, Buşoniu et al., 2010]. In imperfect information games with known rules, e.g., poker [Moravcík et al., 2017], a series of works study algorithmically computing Nash equilibra [Shalev-shwartz and Singer, 2007, Koller et al., 1995, Gilpin et al., 2007, Zinkevich et al., 2008, Bowling et al., 2017]. Also, studies in stateless episodic games shown convergence to coarse correlated equilibrium [Hartline et al., 2015, Blum et al., 2008]. In contrast CoPO converges to the Nash equilibrium in such games. In two-player competitive games, self-play is an approach of interest where a player plays against itself to improve its behavior [Tesauro, 1995, Silver et al., 2016]. But, many of these approaches are limited to specific domains [Heinrich et al., 2015, Heinrich and Silver, 2016].

The closest approach to CoPO in the literature is LOLA [Foerster et al., 2017b] an opponent aware approach. LOLA updates parameters using a second-order correction term, resulting in gradient updates corresponding to the following shortened recursion: if a player thinks that the other player thinks its strategy stays constant [Schaefer and Anandkumar, 2019], whereas CoPO recovers the full recursion series until the Nash equilibrium is delivered. In contrast to [Foerster et al., 2017b] we also provide CoPO extension to value-based, and trust region-based methods, along with

their efficient implementation.

Our work is also related to GANs [Goodfellow et al., 2014], which involves solving a zero sum two-player competitive game (CoMDP with single state). Recent development in nonconvex-nonconcave problems and GANs training show GDA has undesirable convergence properties [Mazumdar et al., 2019] and exhibit strong rotation around fixed points [Balduzzi et al., 2018]. To overcome this rotation behaviour of GDA, various modifications have been proposed, including averaging [Yazıcı et al., 2019], negative momentum [Gidel et al., 2018] along many others [Mertikopoulos et al., 2018b, Daskalakis et al., 2017, Mescheder et al., 2017, Balduzzi et al., 2018, Gemp and Mahadevan, 2018]. Considering the game-theoretic nature of this problem, competitive gradient descent has been proposed as a natural generalization of gradient descent in two-players instead of GDA for GANs [Schaefer and Anandkumar, 2019]. This method, as the predecessor to CoPO, enjoys stability in training, robustness in choice of hyper-parameters, and has desirable performance and convergence properties.

6 DISCUSSION ON APPLICABILITY

CoPO is the paradigm of competitive policy optimization where the goal is to *jointly* find policies for agents. In CoPO, the optimization is centralized, and the execution of actions is decentralized. Applications of such setting are; *(i)* self-play: we train an agent to play against itself; *(ii)* adversarial robustness, inverse RL, and imitation learning: we aim to find a robust model; *(iii)* robust control [Zhou et al., 1996]: we train agents to be robust against attackers; *(iv)* athletic games analysis, e.g., soccer and basketball: we train models of teams in simulation, and let them play against each other to discover tactics and strategies; *(v)* Robocup World (robots soccer): we train our team in our lab before deploying it in the real match; *(vi)* AI economist [Zheng et al., 2020]: we run a game between workers along with rule-makers to discover new tax laws, and many more real-world problems.

Our empirical study shows that CoPG and TRCoPO can excel in this setting and have clear advantages compared to existing algorithms. While the centralized optimization setting in CoPO has a vast range of real-world applications, there are problems that require decentralized optimization. We showed that CoPG-OP, a decentralized extension of CoPG, where each agent also learns its opponent's parameters/model to compute its policy update, comes with the same benefits as CoPG in the centralized setting.

7 CONCLUSION

We presented competitive policy optimization CoPO, a novel PG-based RL method for two player strictly competitive game. In CoPO, each player optimizes strategy by considering the interaction with the environment and the opponent through game theoretic bilinear approximation to the game objective. This method is instantiated to competitive policy gradient (CoPG) and trust region competitive policy optimisation (TRCoPO) using value based and trust region approaches. We theoretically studied these methods and provided PG theorems to show the properties of these model-free RL approaches. We provided efficient implementation of these methods and empirically showed that they provide stable and faster optimization, and also converge to more sophisticated and competitive strategies. We performed case studies for CoPO based approach on self-play, asymmetric mini-max game GAIL, with opponent modelling and further discussed the general applicability of the CoPO paradigm in various real life settings. We dedicated this paper to two player zero-sum games, however, the principles provided in this paper can be used for multi-player general games. In the future, we plan to extend this study to multi-player general-sum games along with efficient implementation of methods. Moreover, we plan to use the techniques proposed in partially observable domains, and study imperfect information games [Azizzadenesheli et al., 2020].

Acknowledgements

The main body of this work took place when M. Prajapat was a visiting scholar at Caltech. The authors would like to thank Florian Schäfer for his support. M. Prajapat is thankful to Zeno Karl Schindler foundation for providing him with a Master thesis grant. K. Azizzadenesheli is supported in part by Raytheon and Amazon Web Service. A. Anandkumar is supported in part by Bren endowed chair, DARPA PAIHR00111890035 and LwLL grants, Raytheon, Microsoft, Google, and Adobe faculty fellowships.

References

V. M. Aleksandrov, V. I. Sysoyev, and V. V. Shemeneva. Stochastic optimization. *Engineering Cybernetics*, 1968.

K. Azizzadenesheli, Y. Yue, and A. Anandkumar. Policy gradient in partially observable environments: Approximation and convergence. *arXiv:1810.07900*, 2020.

L. C. Baird, III. Advantage updating. *WRIGHT LAB WRIGHT-PATTERSON AFB OH*, 1993.

E. Bakker, L. Nyborg, and H. B. Pacejka. Tyre modelling for use in vehicle dynamics studies. In *SAE Technical Paper*. SAE International, 02 1987. doi: 10.4271/870421.

D. Balduzzi et al. The mechanics of n-player differentiable games. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 354–363, 10–15 Jul 2018.

A. Blum et al. Regret minimization and the price of total anarchy. In *ACM Symposium on Theory of Computing*, page 373–382, 2008. ISBN 9781605580470.

M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002. ISSN 0004-3702.

M. Bowling et al. Heads-up limit hold'em poker is solved. *Commun. ACM*, 60(11):81–88, Oct. 2017. ISSN 0001-0782. doi: 10.1145/3131284.

L. Buşoniu, R. Babuška, and B. De Schutter. *Multi-agent Reinforcement Learning: An Overview*, pages 183–221. Springer Berlin Heidelberg, 2010.

R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, CMU RI, 1992.

C. Daskalakis et al. Training gans with optimism. *CoRR*, abs/1711.00141, 2017.

J. Filar and K. Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

J. N. Foerster et al. Counterfactual multi-agent policy gradients. *CoRR*, abs/1705.08926, 2017a.

J. N. Foerster et al. Learning with opponent-learning awareness. *CoRR*, abs/1709.04326, 2017b.

B. Frénay and M. Saerens. Ql2, a simple reinforcement learning scheme for two-player zero-sum markov games. *Neurocomput.*, page 1494–1507, 2009.

I. Gemp and S. Mahadevan. Global convergence to the equilibrium of gans using variational inequalities. *CoRR*, abs/1808.01531, 2018.

G. Gidel et al. Negative momentum for improved game dynamics. *CoRR*, abs/1807.04740, 2018.

A. Gilpin et al. Gradient-based algorithms for finding nash equilibria in extensive form games. In *WINE*, 2007.

I. Goodfellow et al. Generative adversarial nets. In *Advances in neural information processing systems*, 2014.

A. Greenwald and K. Hall. Correlated-q learning. In *International Conference on Machine Learning*, 2003.

J. Hartline, V. Syrgkanis, and E. Tardos. No-regret learning in bayesian games. In *Advances in Neural Information Processing Systems*, page 3061–3069, 2015.

H. He, J. L. Boyd-Graber, K. Kwok, and H. D. III. Opponent modeling in deep reinforcement learning. *CoRR*, abs/1609.05559, 2016.

J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *CoRR*, abs/1603.01121, 2016.

J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, ICML'15, page 805–813, 2015.

P. Hernandez-Leal, B. Kartal, and M. E. Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6): 750–797, Oct 2019. ISSN 1573-7454.

J. Ho and S. Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, 2016.

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

S. Iqbal and F. Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 2961–2970, 2019.

S. Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*, pages 267–274, 2002.

J. M. Keynes. *The general theory of employment, interest, and money*. Springer, 2018.

D. Koller, N. Megiddo, and B. von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 1995.

J. Z. Leibo et al. Multi-agent reinforcement learning in sequential social dilemmas. *CoRR*, abs/1702.03037, 2017.

A. Liniger and J. Lygeros. A noncooperative game approach to autonomous racing. *IEEE Transactions on Control Systems Technology*, 28(3):884–897, 2020.

A. Liniger, A. Domahidi, and M. Morari. Optimization-based autonomous racing of 1: 43 scale rc cars. *Optimal Control Applications and Methods*, 36(5):628–647, 2015.

M. Littman. Value-function reinforcement learning in markov games. *Cognitive Systems Research*, 2001a.

M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on International Conference on Machine Learning*, ICML'94, page 157–163, 1994. ISBN 1558603352.

M. L. Littman. Friend-or-foe q-learning in general-sum games. In *International Conference on Machine Learning*, page 322–328, 2001b.

R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in Neural Information Processing Systems*, 2017.

L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, page 1–31, 2012.

E. Mazumdar, L. J. Ratliff, M. I. Jordan, and S. S. Sastry. Policy-gradient algorithms have no guarantees of convergence in linear quadratic games, 2019.

P. Mertikopoulos, C. Papadimitriou, and G. Piliouras. Cycles in adversarial regularized learning. In *ACM-SIAM Symposium on Discrete Algorithms*, 2018a.

P. Mertikopoulos et al. Mirror descent in saddle-point problems: Going the extra (gradient) mile. *CoRR*, abs/1807.02629, 2018b.

L. M. Mescheder, S. Nowozin, and A. Geiger. The numerics of gans. *CoRR*, abs/1705.10461, 2017.

M. Moravcík et al. Deepstack: Expert-level artificial intelligence in no-limit poker. *CoRR*, abs/1701.01724, 2017.

T. W. Neller and M. Lanctot. An introduction to counterfactual regret minimization. *Educational Advances in Artificial Intelligence (EAAI-2013)*, 2013.

A. Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

J. Pérolat et al. Actor-critic fictitious play in simultaneous move multistage games. In *AISTATS*, volume 84 of *Proceedings of Machine Learning Research*, pages 919–928. PMLR, 2018.

M. Raghu et al. Can deep reinforcement learning solve erdos-selfridge-spencer games? *CoRR*, abs/1711.02301, 2017.

R. Raileanu et al. Modeling others using oneself in multi-agent reinforcement learning. *CoRR*, abs/1802.09640, 2018.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, 1951.

F. Schaefer and A. Anandkumar. Competitive gradient descent. In *Advances in Neural Information Processing Systems 32*, pages 7625–7635. Curran Associates, Inc., 2019.

J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR*, 2016.

J. Schulman et al. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, 07–09 Jul 2015.

F. Schäfer, H. Zheng, and A. Anandkumar. Implicit competitive regularization in gans, 2019.

S. Shalev-shwartz and Y. Singer. Convex repeated games and fenchel duality. In *Advances in Neural Information Processing Systems*, pages 1265–1272. MIT Press, 2007.

J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, Carnegie Mellon University, USA, 1994.

D. Silver et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

S. Srinivasan et al. Actor-critic policy optimization in partially observable multiagent environments. *CoRR*, abs/1810.09026, 2018.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

R. S. Sutton, D. A. McAllester, et al. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

A. Tampuu et al. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE*, 2017.

G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

J. L. Vázquez et al. Optimization-based hierarchical motion planning for autonomous racing. *ArXiv*, abs/2003.04882, 2020.

Y. Yazıcı et al. The unusual effectiveness of averaging in GAN training. In *International Conference on Learning Representations*, 2019.

K. Zhang, Z. Yang, and T. Başar. Policy optimization provably converges to nash equilibria in zero-sum linear quadratic games, 2019.

S. Zheng et al. The ai economist: Improving equality and productivity with ai-driven tax policies. *arXiv*, 2020.

K. Zhou et al. *Robust and optimal control*, volume 40. Prentice hall New Jersey, 1996.

M. Zinkevich et al. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems 20*, pages 1729–1736. Curran Associates, Inc., 2008.

Appendix

8 ALGORITHMS

In this section, we briefly present the algorithms discussed in the paper, namely gradient descent ascent (GDA), competitive policy gradient (CoPG), trust region gradient descent ascent (TRGDA) and trust region competitive policy optimization (TRCoPO).

Algorithm 1: Gradient Descent Ascent Policy Gradient

Initialize (θ^1, θ^2)
for $epoch : 1, 2, 3, ..$ *until termination* **do**
  Collect samples under $\pi(.|.; \theta^1)$, $\pi(.|.; \theta^2)$,
  Estimate Q, then $D_{\theta^i}\eta$ in Eq.7
  Update θ^1, θ^2 using Eq. 4
end

Algorithm 2: Competitive Policy Gradient

Initialize (θ^1, θ^2)
for $epoch : 1, 2, 3, ..$ *until termination* **do**
  Collect samples under $\pi(.|.; \theta^1)$, $\pi(.|.; \theta^2)$,
  Estimate Q, then $D_{\theta^i}\eta$, $D_{\theta^i\theta^j}\eta$ in Eqs. 7,8
  Update θ^1, θ^2 using Eq. 6
end

Algorithm 3: Trust Region Gradient Descent Ascent

Initialize (θ^1, θ^2)
for $epoch : 1, 2, 3, ..$ *until termination* **do**
  Collect samples under $\pi(.|.; \theta^1)$, $\pi(.|.; \theta^2)$,
  Estimate A, and $D_{\theta^i}L$ using Eq. 63,
  Update θ^i using Eq. 58, with bilinear term as a null matrix.
end

Algorithm 4: Trust Region Competitive Policy Optimisation

Initialize (θ^1, θ^2)
for $epoch : 1, 2, 3, ..$ *until termination* **do**
  Collect samples under $\pi(.|.; \theta^1)$, $\pi(.|.; \theta^2)$,
  Estimate A, and $D_{\theta^i}L$, $D_{\theta^i\theta^j}L$ using Eq. 63,
  Update θ^1, θ^2 using Eq. 58,
end

9 RECURSION REASONING

When players play strategic games, in order to maximize their payoffs, they need to consider their opponents' strategies which in turn may depend on their strategies (and so on), resulting in the well known infinite recursion in game theory. This is the recursive reasoning of the form what do I think that you think that I think (and so on) and arises while acting rationally in multi-player games. In this section we elaborate on the recursion in CoMDP and latter show how CoPO recovers the infinite recursion reasoning Fig. 8. Recursion reasoning in CoMDP works as follows:

- Level 1 recursion: Each player considers how the environment, as the results of the agents' current and future moves temporally evolves in favor of each agent,
- Level 2 recursion: Considering that the other agent also considers how the environment, as the results of the agents' current and future moves temporally evolves in favor of each agent,
- Level 3 recursion: Considering that the other agent also considers that the other agent also considers how the environment, as the results of the agents' current and future moves temporally evolves in favor of each agent,
- .....

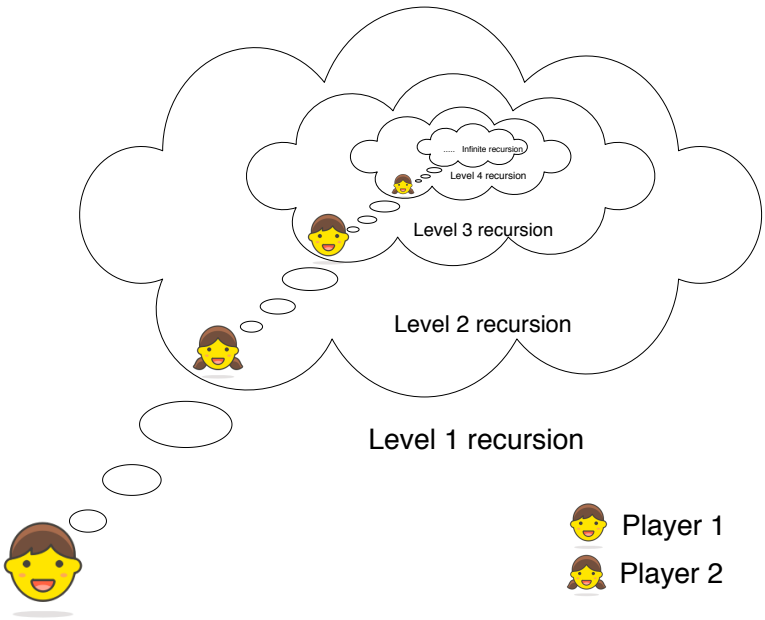


Figure 8: Recursion reasoning. The notion of "*Level N recursion*" in the figure assumes that there is no further recursive thinking (cloud) beyond that level.

In the special case of single state games (non sequential setting), this recursive reasoning boils down to what each player thinks, the other player thinks that the other player thinks, ... (For readers interested in the recursion reasoning in non-sequential but repeated episodic competitive games (single state (or stateless) episodic CoMDP), stability and convergence of competitive gradient optimization and the derivation of Eq. 6, we refer to [Schaefer and Anandkumar, 2019]).

In the following we provide an intuitive explanation for the updates prescribed by CoPO. Let us restate the Nash equilibrium of the bilinear game in Eq. 5 in its matrix form,

$$\begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} \leftarrow \begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} + \alpha \begin{bmatrix} I & -\alpha D_{\theta^1\theta^2}\eta \\ \alpha D_{\theta^2\theta^1}\eta & I \end{bmatrix}^{-1} \begin{bmatrix} D_{\theta^1}\eta \\ -D_{\theta^2}\eta \end{bmatrix}. \tag{15}$$

We rewrite the above mentioned update as

$$\begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} \leftarrow \begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} + \alpha(I-A)^{-1} \begin{bmatrix} D_{\theta^1}\eta \\ -D_{\theta^2}\eta \end{bmatrix}, \quad \text{with} \quad A = \begin{bmatrix} 0 & \alpha D_{\theta^1\theta^2}\eta \\ -\alpha D_{\theta^2\theta^1}\eta & 0 \end{bmatrix}. \tag{16}$$

If it holds that the spectral radius of A is smaller than one, then we can use the Neumann series to compute the inverse,

$$(I-A)^{-1} = \lim_{N\to\infty} \sum_{k=0}^{N} A^k = I + A + A^2 + A^3 + \dots . \tag{17}$$

Using this expansion and its different orders as an approximation, e.g., up to $k \in \{0, \dots, N\}$ for a finite N, we derive policy update rules that have different levels of reasoning. For example, consider the following levels of recursion reasoning:

- N = 0, $(I - A)^{-1} \rightarrow I$, i.e., we approximate $(I - A)^{-1}$ with the first term in the series presented in Eq. 17, therefore, the update rule in Eq. 15 reduces to the GDA update rule. In this case, each agent updates its parameters considering its current and future moves to gain a higher total cumulative reward in the environment assuming the policy of the opponent stays unchanged,

$$\begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} \leftarrow \begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} + \alpha \begin{bmatrix}D_{\theta^1}\eta\\ -D_{\theta^2}\eta\end{bmatrix}.$$

- N = 1, $(I-A)^{-1} \rightarrow I+A$ and the update rule in Eq. 15 reduce to the LOLA update rule. In this case, each agent updates its parameters considering that the opponent updates its parameters considering the other player stays unchanged. For example, player 1 updates its parameter, assuming that player 2 also updates its parameter, but player 1 assumes that the player 2 updates its parameter assuming the player 1 stays unchanged.

$$\begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} \leftarrow \begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} + \alpha \begin{bmatrix}D_{\theta^1}\eta - \alpha D_{\theta^1\theta^2}\eta D_{\theta^2}\eta\\ -D_{\theta^2}\eta - \alpha D_{\theta^2\theta^1}\eta D_{\theta^1}\eta\end{bmatrix}.$$

- N = 2, $(I - A)^{-1} \rightarrow I + A + A^2$, and the update rule is given by Eq. 18. In this case, the agent updates its parameters considering that the opponent also considering that the agent is considering to update parameters, such that its current and future moves increase the probability of earning a higher total cumulative reward in the environment.

$$\begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} \leftarrow \begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} + \alpha \begin{bmatrix}D_{\theta^1}\eta - \alpha D_{\theta^1\theta^2}\eta D_{\theta^2}\eta - \alpha^2 D_{\theta^1\theta^2}\eta D_{\theta^2\theta^1}\eta D_{\theta^1}\eta\\ -D_{\theta^2}\eta - \alpha D_{\theta^2\theta^1}\eta D_{\theta^1}\eta + \alpha^2 D_{\theta^2\theta^1}\eta D_{\theta^1\theta^2}\eta D_{\theta^2}\eta\end{bmatrix}. \tag{18}$$

- $\lim \mathrm{N} \rightarrow \infty$, $(I - A)^{-1} \rightarrow I + A + A^2 + A^3 + ...$, we recover the Nash equilibrium in the limit which corresponds to infinite recursion reasoning. In this case, the update rule in Eq. 15 is given by Eq. 19 which resembles the Eq. 6.

$$\begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} \leftarrow \begin{bmatrix}\theta^1\\ \theta^2\end{bmatrix} + \alpha \begin{bmatrix}\big(Id + \alpha^2 D_{\theta^1\theta^2}\eta D_{\theta^2\theta^1}\eta\big)^{-1}\big(D_{\theta^1}\eta - \alpha D_{\theta^1\theta^2}\eta D_{\theta^2}\eta\big)\\ -\big(Id + \alpha^2 D_{\theta^2\theta^1}\eta D_{\theta^1\theta^2}\eta\big)^{-1}\big(D_{\theta^2}\eta + \alpha D_{\theta^2\theta^1}\eta D_{\theta^1}\eta\big)\end{bmatrix}. \tag{19}$$

We provided this discussion to motivate the importance of taking the game theoretic nature of a given problem into account, in order to design an update.

10 PROOF OF COMPETITIVE POLICY GRADIENT THEOREM

In this section, we first derive the gradient and bilinear terms in the form of score function. In the subsequent subsections we provide complete proof of the competitive policy gradient theorem.

10.1 PROOF FOR PROP. 1:

Proof. Given that the agents' policies are parameterized with θ^1 and θ^2, we know that the game objective is,

$$\eta(\theta^1, \theta^2) = \int_\tau f(\tau; \theta^1, \theta^2)\mathsf{R}(\tau)d\tau, \tag{20}$$

thus, the gradient with respect to θ_i is given by

$$D_{\theta^i}\eta = \int_\tau D_{\theta_i} f(\tau; \theta^1, \theta^2)\mathsf{R}(\tau)d\tau,\ i \in \{1, 2\}\,, \tag{21}$$

for any $f(\tau; \theta^1, \theta^2) \neq 0$ using $D_{\theta^i} \log f(\tau; \theta^1, \theta^2) = \frac{D_{\theta^i} f(\tau; \theta^1, \theta^2)}{f(\tau; \theta^1, \theta^2)}$ from standard calculus and the definition of $f(\tau; \theta^1, \theta^2)$ in Eq. 1, we can compute the gradient of the game objective,

$$D_{\theta^i}\eta = \int_\tau f(\tau; \theta^1, \theta^2)(D_{\theta^i} \log(\prod_{k=0}^{|\tau|-1} \pi(a_k|s_k; \theta^i)))\mathsf{R}(\tau)d\tau\,,$$

Let us define the score function as $g(\tau, \theta^i) = D_{\theta^i}(\log(\prod_{k=0}^{|\tau|-1} \pi(a_k|s_k; \theta^i)))$ which results in,

$$D_{\theta^i}\eta = \int_\tau f(\tau; \theta^1, \theta^2)g(\tau, \theta^i)\mathsf{R}(\tau)d\tau. \tag{22}$$

By definition we know that the second order bilinear approximation of the game objective is $D_{\theta^2\theta^1}\eta(\theta^1,\theta^2) = D_{\theta^2}D_{\theta^1}\eta(\theta^1,\theta^2)$. Hence,

$$\begin{aligned}
D_{\theta^2\theta^1}\eta &= \int_\tau D_{\theta^2} f(\tau;\theta^1,\theta^2) g(\tau,\theta^1)^\top \mathsf{R}(\tau) d\tau \,. \\
&= \int_\tau f(\tau;\theta^1,\theta^2)(D_{\theta^2}\log(\prod_{k=0}^{|\tau|-1}\pi(a_k|s_k;\theta^2)))g(\tau,\theta^1)^\top \mathsf{R}(\tau) d\tau \,, \\
&= \int_\tau f(\tau;\theta^1,\theta^2) g(\tau,\theta^2) g(\tau,\theta^1)^\top \mathsf{R}(\tau) d\tau \,.
\end{aligned}$$

which concludes the proof. □

10.2 PROOF FOR THM. 1

The competitive policy gradient theorem (Thm. 1) consists of two parts. We first provide the derivation for the gradient and utilise this result to derive the bilinear term in the subsequent subsection.

10.2.1 Derivation of gradient term $D_{\theta^1}\eta$

Proof. Using Eq. 2, which defines $V(s_k;\theta^1,\theta^2)$ and $Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)$. We can write $V(s_0;\theta^1,\theta^2)$ as,

$$V(s_0;\theta^1,\theta^2) = \int_{a_0^1}\int_{a_0^2}\pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)da_0^1 da_0^2 \,, \tag{23}$$

$$\text{where, } Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2) = r(s_0,a_0^1,a_0^2) + \gamma\int_{s_1} T(s_1|s_0,a_0^1,a_0^2)V(s_0;\theta^1,\theta^2)ds_1 \,.$$

Hence, the gradient of the state-value V is,

$$\begin{aligned}
D_{\theta^1}V(s_0;\theta^1,\theta^2) = &\int_{a_0^1}\int_{a_0^2}\frac{\partial\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}\pi(a_0^2|s_0;\theta^2)Q(s_0,a_0^1,a_0^2;\theta^1\theta^2)da_0^1 da_0^2 \\
&+\int_{a_0^1}\int_{a_0^2}\pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)\frac{\partial Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)}{\partial\theta^1}da_0^1 da_0^2.
\end{aligned} \tag{24}$$

The gradient of the state-action value Q, in term of the gradient of the state-value V is given by

$$D_{\theta^1}Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2) = \gamma\int_{s_1} T(s_1|s_0,a_0^1,a_0^2)\frac{\partial V(s_1;\theta^1,\theta^2)}{\partial\theta^1}ds_1 \,. \tag{25}$$

Substituting Eq. 25 in Eq. 24 results in,

$$\begin{aligned}
D_{\theta^1}V(s_0;\theta^1,\theta^2) = &\int_{a_0^1}\int_{a_0^2}\frac{\partial\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}\pi(a_0^2|s_0;\theta^2)Q(s_0,a_0^1,a_0^2;\theta^1\theta^2)da_0^1 da_0^2 \\
&+\int_{a_0^1}\int_{a_0^2}\pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)\gamma\int_{s_1}T(s_1|s_0,a_0^1,a_0^2)\frac{\partial V(s_1;\theta^1,\theta^2)}{\partial\theta^1}ds_1 da_0^1 da_0^2 \,.
\end{aligned} \tag{26}$$

By exploring the recursive nature of the state-value V and $D_{\theta^1}V(s_0;\theta^1,\theta^2)$, we can unroll Eq. 26 as,

$$\begin{aligned}
D_{\theta^1}V(s_0;\theta^1,\theta^2) = &\int_{a_0^1}\int_{a_0^2}\frac{\partial\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}\pi(a_0^2|s_0;\theta^2)Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)da_0^1 da_0^2 \\
&+\int_{a_0^1}\int_{a_0^2}\pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)\gamma\int_{s_1}T(s_1|s_0,a_0^1,a_0^2) \\
&\quad\Bigg(\int_{a_1^1}\int_{a_1^2}\frac{\partial\pi(a_1^1|s_1;\theta^1)}{\partial\theta^1}\pi(a_1^2|s_1;\theta^2)Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)da_1^1 da_1^2 \\
&\qquad+\int_{a_1^1}\int_{a_1^2}\pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)\gamma\int_{s_2}T(s_2|s_1,a_1^1,a_1^2)
\end{aligned} \tag{27}$$

$$\frac{\partial V(s_2;\theta^1,\theta^2)}{\partial \theta^1} ds_2 da_1^1 da_1^2 \Bigg) ds_1 da_0^1 da_0^2 \,.$$

To find the derivative of the expected return, we can use the definition of the expected return $\eta(\theta^1,\theta^2)$ and Eq. 27, which results in

$$\begin{aligned}
D_{\theta^1}\eta(\theta^1,\theta^2) &= \int_{s_0} p(s_0) D_{\theta^1} V(s_0;\theta^1,\theta^2) ds_0 \\
&= \int_{s_0} p(s_0) \int_{a_0^1}\int_{a_0^2} \frac{\partial \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} \pi(a_0^2|s_0;\theta^2) Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2) da_0^1 da_0^2 ds_0 \\
&\quad + \gamma \int_{\tau} f(\tau_{0:0}, s_1;\theta^1,\theta^2) \frac{\partial \pi(a_1^1|s_1;\theta^1)}{\partial \theta^1} \pi(a_1^2|s_1;\theta^2) Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2) d\tau_{0:1} \\
&\quad + \gamma^2 \int_{\tau} f(\tau_{0:1}, s_2;\theta^1,\theta^2) \frac{\partial V(s_2;\theta^1,\theta^2)}{\partial \theta^1} ds_2 d\tau_{0:1}. \qquad (28)
\end{aligned}$$

Similar to the two previous step, unrolling and marginalisation can be continued for the entire length of the trajectory, which results in,

$$\begin{aligned}
D_{\theta^1}\eta(\theta^1,\theta^2) &= \int_{\tau} \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k-1}, s_k;\theta^1,\theta^2) \int_{a_k^1}\int_{a_k^2} \frac{\partial \pi(a_k^1|s_k;\theta^1)}{\partial \theta^1} \pi(a_k^2|s_k;\theta^2) Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau \qquad (29) \\
&= \int_{\tau} \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) \frac{\partial \log \pi(a_k^1|s_k;\theta^1)}{\partial \theta^1} Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau. \qquad (30)
\end{aligned}$$

This concludes our derivation of the gradient term of the game objective for the two agent case. □

10.2.2 Derivation of Bilinear term $D_{\theta^1\theta^2}\eta$

Proof: By definition we know that $D_{\theta^1\theta^2}V(s_0;\theta^1,\theta^2) = D_{\theta^1}(D_{\theta^2}V(s_0;\theta^1,\theta^2))$. Thus we can use $D_{\theta^2}V(s_0;\theta^1,\theta^2)$ as defined in Eq. 24 and differentiating with respect to θ^1, thereby we get,

$$\begin{aligned}
D_{\theta^1\theta^2}V(s_0;\theta^1,\theta^2) &= \int_{a_0^1}\int_{a_0^2} \frac{\partial \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} \frac{\partial \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2}^{\top} Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2) da_0^1 da_0^2 \\
&\quad + \int_{a_0^1}\int_{a_0^2} \frac{\partial \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} \pi(a_0^2|s_0;\theta^2) \frac{\partial Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)}{\partial \theta^2}^{\top} da_0^1 da_0^2 \\
&\quad + \int_{a_0^1}\int_{a_0^2} \pi(a_0^1|s_0;\theta^1) \frac{\partial Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)}{\partial \theta^1} \frac{\partial \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2}^{\top} da_0^1 da_0^2 \\
&\quad + \int_{a_0^1}\int_{a_0^2} \pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2) \frac{\partial^2 Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)}{\partial \theta^1 \partial \theta^2} da_0^1 da_0^2. \qquad (31)
\end{aligned}$$

Exploiting the recursive structure of $D_{\theta^1\theta^2}V(s_0;\theta^1,\theta^2)$, we can evaluate $D_{\theta^1\theta^2}V(s_1;\theta^1,\theta^2)$ and substitute it back in Eq. 31,we get,

$$\begin{aligned}
D_{\theta^1\theta^2}V(s_0;\theta^1,\theta^2) &= \int_{a_0^1}\int_{a_0^2} \frac{\partial \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} \frac{\partial \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2}^{\top} Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2) da_0^1 da_0^2 \qquad (32) \\
&\quad + \int_{a_0^1}\int_{a_0^2} \frac{\partial \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} \pi(a_0^2|s_0;\theta^2) \frac{\partial Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)}{\partial \theta^2}^{\top} da_0^1 da_0^2 \\
&\quad + \int_{a_0^1}\int_{a_0^2} \pi(a_0^1|s_0;\theta^1) \frac{\partial Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)}{\partial \theta^1} \frac{\partial \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2}^{\top} da_0^1 da_0^2 \\
&\quad + \int_{a_0^1}\int_{a_0^2} \pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2) \Bigg(\gamma \int_{s_1} T(s_1|s_0,a_0^1,a_0^2)
\end{aligned}$$

$$\Big(\int_{a_1^1}\int_{a_1^2}\frac{\partial\pi(a_1^1|s_1;\theta^1)}{\partial\theta^1}\frac{\partial\pi(a_1^2|s_1;\theta^2)}{\partial\theta^2}^\top Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)da_1^1da_1^2$$
$$+\int_{a_1^1}\int_{a_1^2}\frac{\partial\pi(a_1^1|s_1;\theta^1)}{\partial\theta^1}\pi(a_1^2|s_1;\theta^2)\frac{\partial Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)}{\partial\theta^2}^\top da_1^1da_1^2$$
$$+\int_{a_1^1}\int_{a_1^2}\pi(a_1^1|s_1;\theta^1)\frac{\partial Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)}{\partial\theta^1}\frac{\partial\pi(a_1^2|s_1;\theta^2)}{\partial\theta^2}^\top da_1^1da_1^2$$
$$+\int_{a_1^1}\int_{a_1^2}\pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)\gamma\int_{s_2}T(s_2|s_1,a_1^1,a_1^2)\frac{\partial^2V(s_2;\theta^1,\theta^2)}{\partial\theta^2\partial\theta^1}ds_2da_1^1da_1^2\Big)ds_1\Big)da_0^1da_0^2.$$

Exploring the recursive structure, quickly leads to long sequences of terms, to deal with this let us define a short hand notation, the first term in Eq. 31 is denoted by b_0, second term by c_0, third term by d_0, forth term as e_0, note that the subscript indicates the time step and let $Pr_k = Pr(s_k \to s_{k+1}, \theta^1, \theta^2)$ be the state transition probability summed over all actions. When investigating Eq. 32 note that the forth term of Eq. 31 e_0, expanded to $e_0 = \gamma\int_{s_1} Pr_0(b_1 + c_1 + d_1 + e_1)ds_1$. Thus, recursively applying this insight and using our short hand notation results in,

$$\begin{aligned}
D_{\theta^1\theta^2}V(s_0;\theta^1,\theta^2) =& b_0+c_0+d_0+\gamma\int_{s_1}Pr_0\Big(b_1+c_1+d_1+\gamma\int_{s_2}Pr_1\Big(b_2+c_2+d_2\\
&+\gamma\int_{s_3}Pr_2\Big(b_3+c_3+d_3+\gamma\int_{s_4}Pr_3\Big(...\Big)ds_4\Big)ds_3\Big)ds_2\Big)ds_1\\
=& b_0+\gamma\int_{s_1}Pr_0\Big(b_1+\gamma\int_{s_2}Pr_1\Big(b_2+\gamma\int_{s_3}Pr_2\Big(b_3+\gamma\int_{s_4}Pr_3\Big(...\Big)ds_4\Big)ds_3\Big)ds_2\Big)ds_1\\
&+c_0+\gamma\int_{s_1}Pr_0\Big(c_1+\gamma\int_{s_2}Pr_1\Big(c_2+\gamma\int_{s_3}Pr_2\Big(c_3+\gamma\int_{s_4}Pr_3\Big(...\Big)ds_4\Big)ds_3\Big)ds_2\Big)ds_1\\
&+d_0+\gamma\int_{s_1}Pr_0\Big(d_1+\gamma\int_{s_2}Pr_1\Big(d_2+\gamma\int_{s_3}Pr_2\Big(d_3+\gamma\int_{s_4}Pr_3\Big(...\Big)ds_4\Big)ds_3\Big)ds_2\Big)ds_1.
\end{aligned}$$

Summing over the initial state distribution results in,

$$\begin{aligned}
D_{\theta^1\theta^2}\eta(\theta^1,\theta^2) =& \int_{s_0}p(s_0)D_{\theta^1\theta^2}V(s_0;\theta^1,\theta^2)ds_0\\
=&\int_{s_0}p(s_0)\Big(b_0+\gamma\int_{s_1}Pr_0\Big(b_1+\gamma\int_{s_2}Pr_1\Big(b_2+\gamma\int_{s_3}Pr_2\Big(b_3+\gamma\int_{s_4}Pr_3\Big(...\Big)ds_4\Big)ds_3\Big)ds_2\Big)ds_1\Big)ds_0\\
&+\int_{s_0}p(s_0)\Big(c_0+\gamma\int_{s_1}Pr_0\Big(c_1+\gamma\int_{s_2}Pr_1\Big(c_2+\gamma\int_{s_3}Pr_2\Big(c_3+\gamma\int_{s_4}Pr_3\Big(...\Big)ds_4\Big)ds_3\Big)ds_2\Big)ds_1\Big)ds_0\\
&+\int_{s_0}p(s_0)\Big(d_0+\gamma\int_{s_1}Pr_0\Big(d_1+\gamma\int_{s_2}Pr_1\Big(d_2+\gamma\int_{s_3}Pr_2\Big(d_3+\gamma\int_{s_4}Pr_3\Big(...\Big)ds_4\Big)ds_3\Big)ds_2\Big)ds_1\Big)ds_0. \qquad (33)
\end{aligned}$$

Eq. 33 is clearly build up by three terms the first depending on b_k, the second on c_k and the last on d_k, let us denote these terms as $\mathscr{T}_1$,$\mathscr{T}_2$ and $\mathscr{T}_3$ respectively. Thus we have $D_{\theta^1\theta^2}\eta(\theta^1,\theta^2) = \mathscr{T}_1 + \mathscr{T}_2 + \mathscr{T}_3$. Given this separation, we can tackle each part separately.

Solving for the $\mathscr{T}_1$:
To bring $\mathscr{T}_1$ in Eq. 33, to the final form in our theorem, let us substitute the terms for b_k back in,

$$\begin{aligned}
\mathscr{T}_1 =& \int_{s_0}p(s_0)\Bigg(\int_{a_0^1}\int_{a_0^2}\frac{\partial\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}\frac{\partial\pi(a_0^2|s_0;\theta^2)}{\partial\theta^2}^\top Q(s_0,a_0^1,a_0^2;\theta^1\theta^2)da_0^1da_0^2\\
&+\gamma\int_{s_1}Pr(s_1|s_0;\theta^1,\theta^2)\Big(\int_{a_1^1}\int_{a_1^2}\frac{\partial\pi(a_1^1|s_1;\theta^1)}{\partial\theta^1}\frac{\partial\pi(a_1^2|s_1;\theta^2)}{\partial\theta^2}^\top Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)da_1^1da_1^2\\
&+\int_{a_1^1}\int_{a_1^2}\pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)\gamma\int_{s_2}T(s_2|s_1,a_1^1,a_1^2)\Big(...\Big)ds_2da_1^1da_1^2\Big)ds_1da_0^1da_0^2\Bigg)ds_0. \qquad (34)
\end{aligned}$$

Similar to Eq. 28 we can unrolling and marginalizing, which allows us to bring it into a compact form,

$$\begin{aligned}\mathscr{T}_1 =& \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k-1}, s_k; \theta^1, \theta^2) \frac{\partial \pi(a_k^1|s_k;\theta^1)}{\partial \theta^1} \frac{\partial \pi(a_k^2|s_k;\theta^2)}{\partial \theta^2}^\top Q(s_k, a_k^1, a_k^2; \theta^1, \theta^2) d\tau \\ =& \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k}; \theta^1, \theta^2) \frac{\partial \log \pi(a_k^1|s_k;\theta^1)}{\partial \theta^1} \frac{\partial \log \pi(a_k^2|s_k;\theta^2)}{\partial \theta^2}^\top Q(s_k, a_k^1, a_k^2; \theta^1, \theta^2) d\tau. \end{aligned} \tag{35}$$

Solving for the $\mathscr{T}_2$:
For $\mathscr{T}_2$ in Eq. 33, we again substitute the c_k terms back in, which gets us,

$$\begin{aligned}\mathscr{T}_2 = \int_{s_0} p(s_0) \Bigg(& \int_{a_0^1}\int_{a_0^2} \frac{\partial \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} \pi(a_0^2|s_0;\theta^2) \frac{\partial Q(s_0, a_0^1, a_0^2; \theta^1, \theta^2)}{\partial \theta^2}^\top da_0^1 da_0^2 \\ & + \int_{a_0^1}\int_{a_0^2} \pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)\gamma \int_{s_1} T(s_1|s_0, a_0^1, a_0^2) \\ & \Bigg(\int_{a_1^1}\int_{a_1^2} \frac{\partial \pi(a_1^1|s_1;\theta^1)}{\partial \theta^1} \pi(a_1^2|s_1;\theta^2) \frac{\partial Q(s_1, a_1^1, a_1^2; \theta^1, \theta^2)}{\partial \theta^2}^\top da_1^1 da_1^2 \\ & + \int_{a_1^1}\int_{a_1^2} \pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)\gamma \int_{s_2} T(s_2|s_1, a_1^1, a_1^2) \\ & \Big(...\Big) ds_2\, da_1^1 da_1^2 \Bigg) ds_1 da_0^1 da_0^2 \Bigg) ds_0. \end{aligned} \tag{36}$$

Note that $\mathscr{T}_2$ contains $\frac{\partial Q(s_k, a_k^1, a_k^2; \theta^1, \theta^2)}{\partial \theta^2}$. We can unroll $\frac{\partial Q(s_0, a_0^1, a_0^2; \theta^1, \theta^2)}{\partial \theta^2}$ using Eq. 25 and Eq. 26, till the next state s_1, which results in,

$$\begin{aligned}\mathscr{T}_2 = \int_{s_0} p(s_0) \Bigg(& \int_{a_0^1}\int_{a_0^2} \frac{\partial \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} \pi(a_0^2|s_0;\theta^2) \int_{s_1} \gamma T(s_1|s_0, a_0^1, a_0^2) \\ & \Bigg(\int_{a_1^1}\int_{a_1^2} \pi(a_1^1|s_1;\theta^1) \frac{\partial \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2} Q(s_1, a_1^1, a_1^2; \theta^1, \theta^2) da_1^1 da_1^2 \\ & + \int_{a_1^1}\int_{a_1^2} \pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2) \frac{\partial Q(s_1, a_1^1, a_1^2; \theta^1, \theta^2)}{\partial \theta^2} da_1^1 da_1^2 \Bigg)^\top ds_1 da_0^1 da_0^2 \\ & + \int_{a_0^1}\int_{a_0^2} \pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)\gamma \int_{s_1} T(s_1|s_0, a_0^1, a_0^2) \\ & \Bigg(\int_{a_1^1}\int_{a_1^2} \frac{\partial \pi(a_1^1|s_1;\theta^1)}{\partial \theta^1} \pi(a_1^2|s_1;\theta^2) \frac{\partial Q(s_1, a_1^1, a_1^2; \theta^1, \theta^2)}{\partial \theta^2}^\top da_1^1 da_1^2 \\ & + \int_{a_1^1}\int_{a_1^2} \pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)\gamma \int_{s_2} T(s_2|s_1, a_1^1, a_1^2) \\ & \Big(...\Big) ds_2 da_1^1 da_1^2 \Bigg) ds_1 da_0^1 da_0^2 \Bigg) ds_0. \end{aligned}$$

Using log trick $\big(D_{\theta^i} \log \pi(a_k^i|s_k;\theta^i) = D_{\theta^i}\pi(a_k^i|s_k;\theta^i)/\pi(a_k^i|s_k;\theta^i), i \in 1, 2\big)$ and combining terms together, we get,

$$\begin{aligned}\mathscr{T}_2 = \int_{s_0} p(s_0) \Bigg(& \int_{a_0^1}\int_{a_0^2} \pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2) \int_{s_1} \gamma T(s_1|s_0, a_0^1, a_0^2) \int_{a_1^1}\int_{a_1^2} \pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2) \\ & \Bigg(\frac{\partial \log \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} \frac{\partial \log \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}^\top Q(s_1, a_1^1, a_1^2; \theta^1, \theta^2) \\ & + \Bigg(\frac{\partial \log \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1} + \frac{\partial \log \pi(a_1^1|s_1;\theta^1)}{\partial \theta^1} \Bigg) \frac{\partial Q(s_1, a_1^1, a_1^2; \theta^1, \theta^2)}{\partial \theta^2}^\top \end{aligned}$$

$$+\gamma\int_{s_2}T(s_2|s_1,a_1^1,a_1^2)\Big(...\Big)ds_2\Bigg)da_1^1da_1^2ds_1da_0^1da_0^2\Bigg)ds_0$$

$$=\int_\tau f(\tau_{0:1};\theta^1,\theta^2)\gamma\Bigg(\frac{\partial\log\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}\frac{\partial\log\pi(a_1^2|s_1;\theta^2)}{\partial\theta^2}^\top Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)$$
$$+\Bigg(\frac{\partial\log\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}+\frac{\partial\log\pi(a_1^1|s_1;\theta^1)}{\partial\theta^1}\Bigg)\frac{\partial Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)}{\partial\theta^2}^\top+\gamma\int_{s_2}T(s_2|s_1,a_1^1,a_1^2)\Big(...\Big)ds_2\Bigg)d\tau_{0:1}.$$

We can use the identical method for the remaining $\frac{\partial Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)}{\partial\theta^2}$. Thus unrolling this term for one time step and using the log trick to combine terms. For example unrolling the second step results in,

$$\mathscr{T}_2=\int_\tau f(\tau_{0:1};\theta^1,\theta^2)\gamma\frac{\partial\log\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}\frac{\partial\log\pi(a_1^2|s_1;\theta^2)}{\partial\theta^2}^\top Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)d\tau_{0:1}$$
$$+\int_\tau f(\tau_{0:2};\theta^1,\theta^2)\gamma^2\Bigg(\Bigg(\frac{\partial\log\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}+\frac{\partial\log\pi(a_1^1|s_1;\theta^1)}{\partial\theta^1}\Bigg)\frac{\partial\log\pi(a_2^2|s_2;\theta^2)}{\partial\theta^2}^\top Q(s_2,a_2^1,a_2^2;\theta^1,\theta^2)$$
$$+\Bigg(\frac{\partial\log\pi(a_0^1|s_0;\theta^1)}{\partial\theta^1}+\frac{\partial\log\pi(a_1^1|s_1;\theta^1)}{\partial\theta^1}+\frac{\partial\log\pi(a_2^1|s_2;\theta^1)}{\partial\theta^1}\Bigg)\frac{\partial Q(s_2,a_2^1,a_2^2;\theta^1,\theta^2)}{\partial\theta^2}^\top+\Big(...\Big)\Bigg)d\tau_{0:2}.$$

Finally, on unrolling for $|\tau|$ steps we get,

$$\mathscr{T}_2=\int_\tau\sum_{k=1}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta^1,\theta^2)\frac{\partial\log(\prod_{i=0}^{k-1}\pi(a_i^1|s_i;\theta^1))}{\partial\theta^1}\frac{\partial\log\pi(a_k^2|s_k;\theta^2))}{\partial\theta^2}^\top Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)d\tau. \tag{37}$$

Solving for the $\mathscr{T}_3$:

In $\mathscr{T}_3$ in Eq. 33, similar to the previous two terms we again substitute d_k back in, which results in,

$$\mathscr{T}_3=\int_{s_0}p(s_0)\Bigg(\int_{a_0^1}\int_{a_0^2}\pi(a_0^1|s_0;\theta^1)\frac{\partial Q(s_0,a_0^1,a_0^2;\theta^1,\theta^2)}{\partial\theta^1}\frac{\partial\pi(a_0^2|s_0;\theta^2)}{\partial\theta^2}^\top da_0^1da_0^2$$
$$+\int_{a_0^1}\int_{a_0^2}\pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)\gamma\int_{s_1}T(s_1|s_0;a_0^1,a_0^2)$$
$$\Bigg(\int_{a_1^1}\int_{a_1^2}\pi(a_1^1|s_1;\theta^1)\frac{\partial Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)}{\partial\theta^1}\frac{\partial\pi(a_1^2|s_1;\theta^2)}{\partial\theta^2}^\top da_1^1da_1^2$$
$$+\int_{a_1^1}\int_{a_1^2}\pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)\gamma\int_{s_2}T(s_2|s_1,a_1^1,a_1^2)$$
$$\Big(...\Big)ds_2da_1^1da_1^2\Bigg)ds_1da_0^1da_0^2\Bigg)ds_0.$$

Similar to $\mathscr{T}_2$, $\mathscr{T}_3$ contains $\frac{\partial Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)}{\partial\theta^1}$, using the same approach and unfold this term using Eq. 25 and Eq. 26 for one step results in,

$$\mathscr{T}_3=\int_{s_0}p(s_0)\Bigg(\int_{a_0^1}\int_{a_0^2}\pi(a_0^1|s_0;\theta^1)\int_{s_1}\gamma T(s_1|s_0;a_0^1,a_0^2)$$
$$\Bigg(\int_{a_1^1}\int_{a_1^2}\frac{\partial\pi(a_1^1|s_1;\theta^1)}{\partial\theta^1}\pi(a_1^2|s_1;\theta^2)Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)da_1^1da_1^2$$
$$+\int_{a_1^1}\int_{a_1^2}\pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)\frac{\partial Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)}{\partial\theta^1}da_1^1da_1^2$$

$$\Big) ds_1 \frac{\partial \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2}^\top da_0^1 da_0^2$$
$$+ \int_{a_0^1}\int_{a_0^2} \pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)\gamma \int_{s_1} T(s_1|s_0;a_0^1,a_0^2)$$
$$\Big(\int_{a_1^1}\int_{a_1^2} \pi(a_1^1|s_1;\theta^1)\frac{\partial Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)}{\partial \theta^1}\frac{\partial \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}^\top da_1^1 da_1^2$$
$$+ \int_{a_1^1}\int_{a_1^2} \pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)\gamma \int_{s_2} T(s_2|s_1,a_1^1,a_1^2)$$
$$\Big(...\Big) ds_2 da_1^1 da_1^2\Big) ds_1 da_0^1 da_0^2\Big) ds_0.$$

Using log trick $\big(D_{\theta^i} \log \pi(a_k^i|s_k;\theta^i) = D_{\theta^i}\pi(a_k^i|s_k;\theta^i)/\pi(a_k^i|s_k;\theta^i), i \in 1,2\big)$ and combining terms together, we get,

$$\mathscr{T}_3 = \int_{s_0} p(s_0)\Bigg(\int_{a_0^1}\int_{a_0^2}\pi(a_0^1|s_0;\theta^1)\pi(a_0^2|s_0;\theta^2)\int_{s_1}\gamma T(s_1|s_0;a_0^1,a_0^2)\int_{a_1^1}\int_{a_1^2}\pi(a_1^1|s_1;\theta^1)\pi(a_1^2|s_1;\theta^2)$$
$$\Bigg(\frac{\partial \log \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1}\frac{\partial \log \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}^\top Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)$$
$$+ \frac{\partial Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)}{\partial \theta^1}\Bigg(\frac{\partial \log \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2} + \frac{\partial \log \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}\Bigg)^\top$$
$$+ \gamma \int_{s_2} T(s_2|s_1,a_1^1,a_1^2)\Big(...\Big)\, ds_2\Bigg) da_1^1 da_1^2 ds_1 da_0^1 da_0^2\Bigg) ds_0,$$

by simplifying the notation we get,

$$\mathscr{T}_3 = \int_\tau f(\tau_{0:1};\theta^1,\theta^2)\gamma\Bigg(\frac{\partial \log \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1}\frac{\partial \log \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}^\top Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)$$
$$+\frac{\partial Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2)}{\partial \theta^1}\Bigg(\frac{\partial \log \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2}+\frac{\partial \log \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}\Bigg)^\top + \gamma\int_{s_2} T(s_2|s_1,a_1^1,a_1^2)\Big(...\Big) ds_2\Bigg) d\tau_{0:1}.$$

The remaining terms in the recursive structure again contain terms of the form $\frac{\partial Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)}{\partial \theta^1}$, which we simplify in a similar fashion by unfolding for one step, for example unfolding the second term for one step results in,

$$\mathscr{T}_3 = \int_\tau f(\tau_{0:1};\theta^1,\theta^2)\gamma\frac{\partial \log \pi(a_0^1|s_0;\theta^1)}{\partial \theta^1}\frac{\partial \log \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}^\top Q(s_1,a_1^1,a_1^2;\theta^1,\theta^2) d\tau_{0:1}$$
$$+\int_\tau f(\tau_{0:2};\theta^1,\theta^2)\gamma^2\Bigg(\frac{\partial \log \pi(a_2^1|s_2;\theta^1)}{\partial \theta^1}\Bigg(\frac{\partial \log \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2}+\frac{\partial \log \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}\Bigg)^\top Q(s_2,a_2^1,a_2^2;\theta^1,\theta^2)$$
$$+\frac{\partial Q(s_2,a_2^1,a_2^2;\theta^1,\theta^2)}{\partial \theta^1}\Bigg(\frac{\partial \log \pi(a_0^2|s_0;\theta^2)}{\partial \theta^2}+\frac{\partial \log \pi(a_1^2|s_1;\theta^2)}{\partial \theta^2}+\frac{\partial \log \pi(a_2^2|s_2;\theta^2)}{\partial \theta^2}\Bigg)^\top + \Big(...\Big)\Bigg) d\tau_{0:2}.$$

Finally on unrolling for $|\tau|$ steps we get,

$$= \int_\tau \sum_{k=1}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta^1,\theta^2)\frac{\partial \log \pi(a_k^1|s_k;\theta^1)}{\partial \theta^1}\frac{\partial \log(\prod_{l=0}^{k-1}\pi(a_l^2|s_l;\theta^2))}{\partial \theta^2}^\top Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau. \tag{38}$$

Combining all 3 terms:
By Eq. 33 we know that the bilinear term of the game objective is given by $D_{\theta^1\theta^2}\eta(\theta^1,\theta^2) = \mathscr{T}_1 + \mathscr{T}_2 + \mathscr{T}_3$, or in other words the sum of the three terms derived above. Thus, we derived that,

$$D_{\theta^1\theta^2}\eta(\theta^1,\theta^2) = \int_\tau \sum_{k=0}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta^1,\theta^2)\frac{\partial \log \pi(a_k^1|s_k;\theta^1)}{\partial \theta^1}\frac{\partial \log \pi(a_k^2|s_k;\theta^2)}{\partial \theta^2}^\top Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau$$

$$+\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) \frac{\partial \log(\prod_{l=0}^{k-1} \pi(a_l^1|s_l;\theta^1))}{\partial \theta^1} \frac{\partial \log \pi(a_k^2|s_k;\theta^2)}{\partial \theta^2}^\top Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau$$
$$+\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) \frac{\partial \log \pi(a_k^1|s_k;\theta^1)}{\partial \theta^1} \frac{\partial \log(\prod_{l=0}^{k-1} \pi(a_l^2|s_l;\theta^2))}{\partial \theta^2}^\top Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) d\tau. \tag{39}$$

This concludes our derivation of the bilinear term of the game objective.

11 GENERALIZATION OF COPG TO VARIANTS OF RETURN

11.1 ESTIMATION OF GRADIENT AND BILINEAR TERM USING A BASELINE

The gradient and bilinear terms can be estimated using Eq. 7 and Eq. 8 based on $Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)$ the state-action value function. But in practice, this can perform poorly due to variance and missing the notion of relative improvement. During learning, policy parameters are updated to increase the probability of visiting good trajectories and reducing chances of encountering bad trajectories. But, let us assume good samples have zero reward and bad samples have a negative reward. In this case, the policy distribution will move away from the negative samples but can move in any direction as zero reward does not provide information on being better a state-action pair. This causes high variance and slow learning. In another case, let us assume that a non zero constant is added to the reward function for all states-action pairs, ideally this constant factor should not affect the policy update but in the current formulation using $Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)$ it does because the notion of better or worse sample is with respect to zero. Hence, missing the notion of relative improvement.

To mitigate the issues, a baseline b can be subtracted from $Q(s,a_k^1,a_k^2;\theta^1\theta^2)$. The baseline can be any random variable as long as it is not a function of a_k^1 and a_k^2 [Sutton and Barto, 2018]. For an MDP, a good baseline b is a function of the state $b(s)$, because let say in some states all actions have high values and we need a high baseline to differentiate the higher valued actions from the less highly valued ones; in other states all actions will have low values and a low baseline is appropriate. This is equivalent to $Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) \leftarrow Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) - b(s_k)$ in Eq. 8 and Eq. 7.

11.2 EXTENDING THM. 1 FOR GRADIENT AND BILINEAR ESTIMATION USING ADVANTAGE FUNCTION

A popular choice for a baseline $b(s)$ is $V(s,\theta^1,\theta^2)$, which results in the advantage function defined in Sec. 3.1. In Thm. 3 we extend Thm. 1 to estimate gradient and bilinear terms using the advantage function.

Theorem 3. *Given that the advantage function is defined as $A(s,a^1,a^2;\theta^1,\theta^2) = Q(s,a^1,a^2;\theta^1,\theta^2) - V(s;\theta^1,\theta^2)$ and given Eq. 7, then for player $i,j \in \{1,2\}$ and $i \neq j$, it holds that,*

$$D_{\theta^i}\eta = \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) A(s_k,a_k^1,a_k^2;\theta^1\theta^2) d\tau, \tag{40}$$

$$\begin{aligned} D_{\theta^i\theta^j}\eta =& \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \pi(a_k^j|s_k;\theta^j))^\top A(s_k,a_k^1,a_k^2;\theta^1\theta^2) d\tau \\ &+\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \prod_{l=0}^{k-1} \pi(a_l^i|s_l;\theta^i)) D_{\theta^j}(\log \pi(a_k^j|s_k;\theta^j))^\top A(s_k,a_k^1,a_k^2;\theta^1\theta^2) d\tau \\ &+\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \prod_{l=0}^{k-1} \pi(a_l^j|s_l;\theta^j))^\top A(s_k,a_k^1,a_k^2;\theta^1\theta^2) d\tau. \end{aligned} \tag{41}$$

Proof. The RHS of Eq. 40 is,

$$\int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) A(s_k,a_k^1,a_k^2;\theta^1\theta^2) d\tau$$

$$
\begin{aligned}
&= \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i))(Q(s,a^1,a^2;\theta^1,\theta^2) - V(s;\theta^1,\theta^2))d\tau \\
&= D_{\theta^i}\eta - \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) V(s;\theta^1,\theta^2))d\tau \\
&= D_{\theta^i}\eta - \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k-1},s_k;\theta^1,\theta^2) D_{\theta^i}(\pi(a_k^i|s_k;\theta^i))\pi(a_k^j|s_k;\theta^j) V(s;\theta^1,\theta^2))d\tau \qquad (42) \\
&= D_{\theta^i}\eta.
\end{aligned}
$$

The last equality holds since the value function $V(s;\theta^1,\theta^2)$ is independent of the actions, and this concludes the proof for $D_{\theta^i}\eta$. Now consider, the RHS of Eq. 41,

$$
\begin{aligned}
&\int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \pi(a_k^j|s_k;\theta^j))^\top A(s_k,a_k^1,a_k^2;\theta^1,\theta^2)d\tau \qquad (43) \\
&+\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \prod_{l=0}^{k-1} \pi(a_l^i|s_l;\theta^i)) D_{\theta^j}(\log \pi(a_k^j|s_k;\theta^j))^\top A(s_k,a_k^1,a_k^2;\theta^1,\theta^2)d\tau \\
&+\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \prod_{l=0}^{k-1} \pi(a_l^j|s_l;\theta^j))^\top A(s_k,a_k^1,a_k^2;\theta^1,\theta^2)d\tau.
\end{aligned}
$$

Using definition of Advantage function, we can equivalently write,

$$
\begin{aligned}
&\int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \pi(a_k^j|s_k;\theta^j))^\top (Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) - V(s_k;\theta^1,\theta^2))d\tau \\
&+\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \prod_{l=0}^{k-1} \pi(a_l^i|s_l;\theta^i)) D_{\theta^j}(\log \pi(a_k^j|s_k;\theta^j))^\top (Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) - V(s_k;\theta^1,\theta^2))d\tau \\
&+\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \prod_{l=0}^{k-1} \pi(a_l^j|s_l;\theta^j))^\top (Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2) - V(s_k;\theta^1,\theta^2))d\tau.
\end{aligned}
$$

Using Eq. 8, we can equivalently separate the expression in $D_{\theta^i\theta^j}\eta$ and terms depending on the value function,

$$
\begin{aligned}
&D_{\theta^i\theta^j}\eta - \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \pi(a_k^j|s_k;\theta^j))^\top V(s_k;\theta^1,\theta^2)d\tau \\
&-\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \prod_{l=0}^{k-1} \pi(a_l^i|s_l;\theta^i)) D_{\theta^j}(\log \pi(a_k^j|s_k;\theta^j))^\top V(s_k;\theta^1,\theta^2)d\tau \\
&-\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k};\theta^1,\theta^2) D_{\theta^i}(\log \pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \prod_{l=0}^{k-1} \pi(a_l^j|s_l;\theta^j))^\top V(s_k;\theta^1,\theta^2)d\tau, \qquad (44)
\end{aligned}
$$

note that for any $p_\theta(\tau) \neq 0$ using $D_\theta \log p_\theta(\tau) = D_\theta p_\theta(\tau)/p_\theta(\tau)$, which allows us to further simplify the expression to,

$$
\begin{aligned}
&D_{\theta^i\theta^j}\eta - \int_\tau \sum_{k=0}^{|\tau|-1} \gamma^k f(\tau_{0:k-1},s_k;\theta^1,\theta^2) D_{\theta^i}(\pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\pi(a_k^j|s_k;\theta^j))^\top V(s_k;\theta^1,\theta^2)d\tau \\
&-\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k-1},s_k,a_k^i;\theta^1,\theta^2) D_{\theta^i}(\log \prod_{l=0}^{k-1} \pi(a_l^i|s_l;\theta^i)) D_{\theta^j}(\pi(a_k^j|s_k;\theta^j))^\top V(s_k;\theta^1,\theta^2)d\tau \\
&-\int_\tau \sum_{k=1}^{|\tau|-1} \gamma^k f(\tau_{0:k-1},s_k,a_k^j;\theta^1,\theta^2) D_{\theta^i}(\pi(a_k^i|s_k;\theta^i)) D_{\theta^j}(\log \prod_{l=0}^{k-1} \pi(a_l^j|s_l;\theta^j))^\top V(s_k;\theta^1,\theta^2)d\tau \\
&= D_{\theta^i\theta^j}\eta, \qquad (45)
\end{aligned}
$$

where the last equality holds since $V(s_k;\theta^1,\theta^2)$ is independent of the actions of any player. This concludes our proof. □

11.3 APPROACHES TO ESTIMATE ADVANTAGE FUNCTION

There exist multiple approaches to estimate $A(s_k,a_k^1,a_k^2;\theta^1,\theta^2)$, for example Monte Carlo (Eq. 46), 1-step TD residual (Eq. 47) and t-step TD residual (Eq. 48)

$$A(s_k,a_k^1,a_k^2;\theta^1,\theta^2) = \sum_{j=k}^{|\tau|-1}\gamma^{j-k}r(s_j,a_j^1,a_j^2) - V(s_k;\theta^1,\theta^2), \tag{46}$$

$$A(s_k,a_k^1,a_k^2;\theta^1,\theta^2) = r(s_k,a_k^1,a_k^2) + \gamma V(s_{k+1};\theta^1,\theta^2) - V(s_k;\theta^1,\theta^2), \tag{47}$$

$$\begin{aligned}A(s_k,a_k^1,a_k^2;\theta^1,\theta^2) =& r(s_k,a_k^1,a_k^2)+\gamma r(s_{k+1},a_{k+1}^1,a_{k+1}^2)+...+\gamma^{t-1}r(s_{k+t-1},a_{k+t-1}^1,a_{k+t-1}^2)\\ &+\gamma^t V(s_{k+t};\theta^1,\theta^2) - V(s_k;\theta^1,\theta^2).\end{aligned} \tag{48}$$

The Monte Carlo approach experience high variance in gradient estimation, particularly with long trajectories, which may result in slow learning or not learning at all. On the other hand 1-step TD residual has bias towards initialization of the value function. t-step TD residual can lower this effect up to an extent. To trade between bias and variance we propose to use generalized advantage estimation (GAE) [Schulman et al., 2016], which is an exponentially λ weighted average over t-step estimators. It is given by,

$$\begin{aligned}A^{GAE}(s_k,a_k^1,a_k^2;\theta^1,\theta^2) &= \sum_{l=0}^{|\tau|-1}(\lambda\gamma)^l\delta_{l+k}^V,\\ \text{where, } \delta_k^V &= r(s_k,a_k^1,a_k^2)+\gamma V(s_{k+1};\theta^1,\theta^2) - V(s_k;\theta^1,\theta^2).\end{aligned}$$

Note that at $\lambda=0$, $A^{GAE}(s_k,a_k^1,a_k^2;\theta^1,\theta^2)$ reduces to 1-step TD residual and at $\lambda=1$ it is equivalent to the Monte Carlo approach to estimate the advantage function. We refer the interested reader to [Schulman et al., 2016] for more details.

12 TRAINING BY SELF-PLAY

As proposed in Alg. 6, we use CoPG to learn a policy for each player, however, it is also possible to use the competitive policy gradient algorithm to learn game strategies by self-play. In contrast to two player training where each player has its own policy model, in self-play, we have one model for both players. Each player then samples actions from the same policy model for two different state instances $s^1=[s^{l1},s^{l2}]$ and $s^2=[s^{l2},s^{l1}]$, where s^{l1} and s^{l2} are the local states of player 1 and player 2 respectively. Let $\tau=\big((s_k^{l1},s_k^{l2},a_k^1,a_k^2,r_k)_{k=0}^{|\tau^1|-1},s_{|\tau|}\big)$ be a trajectory obtained in the game. Using Eq. 7 and Eq. 8, we can define the gradient and bilinear terms as $D_\theta\eta^i$ and $D_{\theta\theta}\eta^i$, $i,j\in\{1,2\}$, $i\neq j$,

$$D_\theta\eta^i = \int_\tau \sum_{k=0}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta)D_\theta(\log\pi(a_k^i|s_k^i;\theta))Q(s_k^1,a_k^1,a_k^2;\theta)d\tau, \tag{49}$$

$$\begin{aligned}D_{\theta\theta}\eta^i =& \int_\tau \sum_{k=0}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta)D_\theta(\log\pi(a_k^i|s_k^i;\theta))D_\theta(\log\pi(a_k^j|s_k^j;\theta))^\top Q(s_k^1,a_k^1,a_k^2;\theta)d\tau\\ &+\int_\tau\sum_{k=1}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta)D_\theta(\log\prod_{l=0}^{k-1}\pi(a_l^i|s_l^i;\theta))D_\theta(\log\pi(a_k^j|s_k^j;\theta))^\top Q(s_k^1,a_k^1,a_k^2;\theta)d\tau\\ &+\int_\tau\sum_{k=1}^{|\tau|-1}\gamma^k f(\tau_{0:k};\theta)D_\theta(\log\pi(a_k^i|s_k^i;\theta))D_\theta(\log\prod_{l=0}^{k-1}\pi(a_l^j|s_l^j;\theta))^\top Q(s_k^1,a_k^1,a_k^2;\theta)d\tau.\end{aligned} \tag{50}$$

Utilizing the gradients and bilinear term in Eq. 49 and Eq. 50, the parameters are updated twice which results in the following update rule for self-play,

$$\theta^t \leftarrow \theta + \alpha\big(Id+\alpha^2 D_{\theta\theta}\eta^1 D_{\theta\theta}\eta^2\big)^{-1}\big(D_\theta\eta^1 - \alpha D_{\theta\theta}\eta^1 D_\theta\eta^2\big),$$

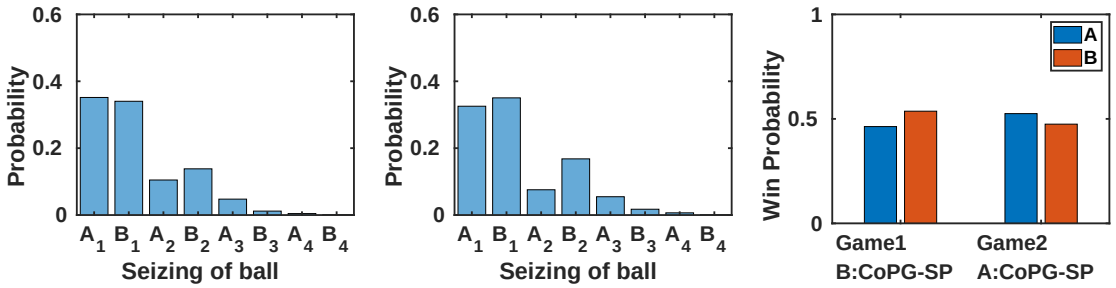
$$\theta \leftarrow \theta^t - \alpha\big(Id + \alpha^2 D_{\theta\theta}\eta^2 D_{\theta\theta}\eta^1\big)^{-1}\big(D_\theta\eta^2 + \alpha D_{\theta\theta}\eta^2 D_\theta\eta^1\big), \tag{51}$$

where θ^t is a temporary intermediate variable.[3]

Algorithm 5: Competitive Policy Gradient with self-play

Input: $\pi(.|.;\theta)$, N, B, α, Choice of $A(s^1, a^1, a^2; \theta)$ mentioned in Sec 3.1
for $epoch : e = 1$ ***to*** N **do**
  for $batch : b = 1$ ***to*** B **do**
    for $k = 0$ ***to*** $|\tau| - 1$ **do**
      Sample $a_k^1 \sim \pi(a_k^1|s_k^1;\theta)$, $a_k^2 \sim \pi(a_k^2|s_k^2;\theta)$.
      Execute actions $a_k = (a_k^1, a_k^2)$
      Record $\{s_k^{l1}, s_k^{l2}, a_k^1, a_k^2, r_k\}$ in τ_b
    end
    Record τ_b in *M*
  end
  1. Estimate $A(s^1, a^1, a^2; \theta)$ with τ in M
  2. Estimate $D_\theta\eta^1$, $D_\theta\eta^2$, $D_{\theta\theta}\eta^1$, $D_{\theta\theta}\eta^2$ using Eq. 49 and Eq. 50
  3. Update θ using Eq. 51
end

We tested the self-play algorithm on the Markov soccer game, which is discussed in more detail in Sec. 15.5.We focus in this comparison on the Markov soccer game since it is strategic enough while being representative. The game has two player A and B. The state vector in perspective of player A is $s^1 = [s^A, s^B]$ and in player B $s^2 = [s^B, s^A]$, in accordance with the self-play setting described above. The game description is given in Sec. 15.5. The player was trained while competing with itself using Alg. 5. Fig. 9a below show the interaction plot for CoPG agents trained with self-play with both players winning almost 50% of the games. On competing agents trained under self-play (CoPG-SP) against agents trained with two different networks (CoPG), we observe that self-play had slight edge in number of game wins. The is probably due to better generalisation of agents in self-play obtained by exploiting symmetricity in the game. We also trained a GDA agent using self-play and saw no clear improvements compared to normal GDA as tested in Sec. 15.5.



(a) CoPG-SP vs CoPG-SP (b) CoPG vs CoPG-SP (c) CoPG vs CoPG-SP

Figure 9: a-b) Interaction plots, representing probability of seizing ball in the game between A vs B. X-axis convention, for player A. A_1: A scored goal, A_2: A scored goal after seizing ball from B, A_3: A scored goal by seizing ball from B which took the ball from A and so forth. Vice versa for player B. N: No one scored goal both kept on seizing ball. c) probability of games won.

13 TRUST REGION COMPETITIVE POLICY OPTIMISATION

Following our discussion in the Sec. 3.3 on trust region competitive policy optimization, in this section we will go into more details on constraint competitive optimization and formally derive and proof all our results presented in Sec. 3.3. More precisely in this section we first proof Lem. 1 and Thm. 2, second we will derive how we can efficiently approximate Eq. 13

[3]Of course, there are other ways to come up with self-play updates. For example, one can use Eq. 6 with $\theta^1 = \theta^2 = \theta$ to update parameters, then set $\theta \leftarrow (\theta^1 + \theta^2)/2$.

using a bilinear approximation of the objective and a second order approximation of the KL divergence constraint and how this allows us to efficiently find an approximate solution.

13.1 PROOF FOR LEM. 1

Proof. The second term on the RHS of Eq. 10 can be simplified as,

$$\begin{aligned}
&\mathbb{E}_{\tau\sim f(.;\theta'^1,\theta'^2)}\Big[\sum_{k=0}^{|\tau|-1}\gamma^k A(s_k,a_k^1,a_k^2;\theta^1,\theta^2)\Big]\\
&\quad=\mathbb{E}_{\tau\sim f(.;\theta'^1,\theta'^2)}\Big[\sum_{k=0}^{|\tau|-1}\gamma^k\big(Q(s_k,a_k^1,a_k^2;\theta^1,\theta^2)-V(s_k;\theta^1,\theta^2)\big)\Big]\\
&\quad=\mathbb{E}_{\tau\sim f(.;\theta'^1,\theta'^2)}\Big[\sum_{k=0}^{|\tau|-1}\gamma^k\big(r(s_k,a_k^1,a_k^2)+\gamma V(s_{k+1};\theta^1,\theta^2)-V(s_k;\theta^1,\theta^2)\big)\Big]\\
&\quad=\mathbb{E}_{\tau\sim f(.;\theta'^1,\theta'^2)}\Big[\sum_{k=0}^{|\tau|-1}\gamma^k r(s_k,a_k^1,a_k^2)+\gamma^{k+1}V(s_{k+1};\theta^1,\theta^2)-\gamma^k V(s_k;\theta^1,\theta^2)\Big]\\
&\quad=\mathbb{E}_{\tau\sim f(.;\theta'^1,\theta'^2)}\Big[\sum_{k=0}^{|\tau|-1}\gamma^k r(s_k,a_k^1,a_k^2)\Big]-\mathbb{E}_{s_0}\Big[V^1(s_0;\theta^1,\theta^2)\Big]\\
&\quad=\eta(\theta'^1,\theta'^2)-\eta(\theta^1,\theta^2).
\end{aligned}\tag{52}$$

This concludes our proof that,

$$\eta(\theta'^1,\theta'^2)=\eta(\theta^1,\theta^2)+\mathbb{E}_{\tau\sim f(\cdot;\theta'^1,\theta'^2)}\sum\nolimits_{k=0}^{|\tau|-1}\gamma^k A(s,a^1,a^2;\theta^1,\theta^2).$$

□

13.2 PROOF FOR THM. 2

Proof. Let θ^1,θ^2 be the parametrization of the policies, when these parameters are updated to θ'^1,θ'^2, by Lem. 1 we know that the improved game objective $\eta(\theta'^1,\theta'^2)$ is,

$$\eta(\theta'^1,\theta'^2)=\eta(\theta^1,\theta^2)+\mathbb{E}_{\tau\sim f(.;\theta'^1,\theta'^2)}\sum_{k}^{|\tau|-1}\gamma^k A(s_k,a_k^1,a_k^2;\theta^1,\theta^2).\tag{53}$$

Furthermore, we defined $L_{\theta^1,\theta^2}(\theta'^1,\theta'^2)$ which denotes the surrogate game objective where the trajectory depends on policies (θ^1,θ^2) but actions are sampled from (θ'^1,θ'^2).

$$L_{\theta^1,\theta^2}(\theta'^1,\theta'^2)=\eta(\theta^1,\theta^2)+\mathbb{E}_{\tau\sim f(.;\theta^1,\theta^2)}\sum_{k=0}^{|\tau|-1}\gamma^k\mathbb{E}_{\pi(a_k'^1|s_k;\theta'^1),\pi(a_k'^2|s_k;\theta'^2)}A(s_k,a_k'^1,a_k'^2;\theta^1,\theta^2).\tag{54}$$

Note that $L_{\theta^1,\theta^2}(\theta'^1,\theta'^2)$ is an approximation of true game objective $\eta(\theta'^1,\theta'^2)$. Using the average advantage function,

$$\overline{A}(s;\theta^1,\theta^2)=\sum_{a^1,a^2}\pi(a^1|s;\theta'^1)\pi(a^2|s;\theta'^2)A(s,a^1,a^2;\theta^1,\theta^2),\tag{55}$$

one can write Eq. 53 and Eq. 54 as,

$$\eta(\theta'^1,\theta'^2)-\eta(\theta^1,\theta^2)=\mathbb{E}_{\tau\sim f(.;\theta'^1,\theta'^2)}\Big[\sum_{k=0}^{|\tau|-1}\gamma^k\overline{A}(s;\theta^1,\theta^2)\Big],\tag{56}$$

$$L_{\theta^1,\theta^2}(\theta'^1,\theta'^2) - \eta(\theta^1,\theta^2) = \mathbb{E}_{\tau\sim f(.;\theta^1,\theta^2)}\Big[\sum_{k=0}^{|\tau|-1}\gamma^k\overline{A}(s;\theta^1,\theta^2)\Big]. \tag{57}$$

To finish our proof, we subtract Eq. 57 from Eq. 56 and take the absolute value, this allows to bound $|\eta(\theta'^1,\theta'^2) - L_{\theta^1,\theta^2}(\theta'^1,\theta'^2)|$. To derive this bound let $\epsilon = \max_s \sum_k^{|\tau|-1}\gamma^k\overline{A}(s;\theta^1,\theta^2)$, which allows us to perform the following steps,

$$\begin{aligned}
|\eta(\theta'^1,\theta'^2) - L_{\theta^1,\theta^2}(\theta'^1,\theta'^2)| &= \Big|\mathbb{E}_{\tau\sim f(.;\theta'^1,\theta'^2)}\Big[\sum_k^{|\tau|-1}\gamma^k\overline{A}(s;\theta^1,\theta^2)\Big] - \mathbb{E}_{\tau\sim f(.;\theta^1,\theta^2)}\Big[\sum_k^{|\tau|-1}\gamma^k\overline{A}(s;\theta^1,\theta^2)\Big]\Big| \\
&= \int_\tau \Big|f(\tau;\theta'^1,\theta'^2) - f(\tau;\theta^1,\theta^2)\Big|\Big[\sum_k^{|\tau|-1}\gamma^k\overline{A}(s;\theta^1,\theta^2)\Big]d\tau \\
&\leq \epsilon D_{TV}\big(\tau\sim(\theta^1,\theta^2),\tau\sim(\theta'^1,\theta'^2)\big) \\
&\leq \epsilon\sqrt{\frac{1}{2}D_{KL}(\tau\sim(\theta^1,\theta^2),\tau\sim(\theta'^1,\theta'^2))} \\
&= \epsilon/\sqrt{2}\sqrt{D_{KL}((\theta'^1,\theta'^2),(\theta^1,\theta^2))}.
\end{aligned}$$

This concludes our proof. □

13.3 CONSTRAINT COMPETITIVE OPTIMIZATION OF A TRUST REGION

In this section, we approach the constrained optimization problem of TRCoPO given by Eq. 13 in 3 steps. First, motivated by CGD we seek for a Nash equilibrium of the bilinear approximation of the game objective. Secondly, in contrast to CoPG, where an L2 penalty term is used along with the bilinear approximation denoting limited confidence in parameter deviation, here a KL divergence constraint will influence the gradient direction. Note that we incorporate this KL divergence constraint into the objective using the method of Lagrange multipliers. This results in the game theoretic update rule defined in Eq. 58.

$$\begin{aligned}
\theta^1 &\leftarrow \theta^1 + \underset{\Delta\theta^1;\Delta\theta^1+\theta^1\in\Theta^1}{\operatorname{argmax}} \Delta\theta^{1\top}D_{\theta^1}L + \Delta\theta^{1\top}D_{\theta^1\theta^2}L\Delta\theta^2 - \frac{\lambda}{2}(\Delta\theta^{1\top}A_{11}\Delta\theta^1 + \Delta\theta^{2\top}A_{22}\Delta\theta^2 - 2\delta'), \\
\theta^2 &\leftarrow \theta^2 + \underset{\Delta\theta^2;\Delta\theta^2+\theta^2\in\Theta^2}{\operatorname{argmin}} \Delta\theta^{2\top}D_{\theta^2}L + \Delta\theta^{2\top}D_{\theta^2\theta^1}L\Delta\theta^1 + \frac{\lambda}{2}(\Delta\theta^{1\top}A_{11}\Delta\theta^1 + \Delta\theta^{2\top}A_{22}\Delta\theta^2 - 2\delta').
\end{aligned} \tag{58}$$

Here, λ is the Lagrange multiplier. Note that the same λ is used for both the agents and that we use a quadratic approximation of the KL divergence, which is $D_{KL}\big((\theta^1,\theta^2),(\theta'^1,\theta'^2)\big) \approx \frac{1}{2}\sum_i^2\sum_j^2 \Delta\theta^{i\top}A_{ij}\Delta\theta^j$, where $A_{ij} = D_{\theta^i\theta^j}D_{KL}\big((\theta^1,\theta^2),(\theta'^1,\theta'^2)\big)$, and it holds that for $i \neq j,\ A_{ij} = 0$. Note that we will go into more details later on the approximation of the KL divergence.

TRCoPO update: As said the update step in Eq. 58 is given by the Nash equilibrium of the game within Eq. 58. We can find the Nash equilibrium that fulfills the approximated KL divergence constraint efficiently by solving the following equation using a line-search approach,

$$\Delta\theta = -(B + \lambda A')^{-1}C, \ \text{ subject to: } \Delta\theta^\top A\Delta\theta \leq \delta, \tag{59}$$

where, let A', B, C, δ, and $\Delta\theta$ be defined as,

$$\begin{aligned}
A' &= \begin{bmatrix} -A_{11} & 0 \\ 0 & A_{22}\end{bmatrix},\ B = \begin{bmatrix} 0 & D_{\theta^1\theta^2}L \\ D_{\theta^2\theta^1}L & 0\end{bmatrix}, \\
C &= \begin{bmatrix} D_{\theta^1}L \\ D_{\theta^2}L\end{bmatrix},\ \delta = 2\delta', \text{ and } \Delta\theta = \begin{bmatrix}\Delta\theta^1 \\ \Delta\theta^2\end{bmatrix}.
\end{aligned} \tag{60}$$

TRCoPO constraint optimization: After this quick introduction on how to practically update the policy parameters using TRCoPO, let us derive in detail how to get to Eq. 59. As explained above Eq. 59 is the Nash equilibrium of an approximation

of Eq. 13. Before starting to approximate Eq. 13 let us rewrite it by using the definition of $L_{\theta^1\theta^2}(\theta'^1, \theta'^2)$ as given in Eq. 54. Note that we can neglect the constant term $\eta(\theta^1, \theta^2)$, which results in the following game,

$$\max_{\theta'^1} \min_{\theta'^2} \; \mathbb{E}_{\tau\sim f(\cdot;\theta^1,\theta^2)} \sum_{k=0}^{|\tau|-1} \gamma^k \mathbb{E}_{\pi(a_k'^1|s_k;\theta'^1),\pi(a_k'^2|s_k;\theta'^2)} A(s_k, a_k'^1, a_k'^2; \theta^1, \theta^2), \tag{61}$$
$$\text{subject to: } D_{KL}((\theta^1, \theta^2), (\theta'^1, \theta'^2)) \leq \delta'.$$

Furthermore, we can use an importance sampling factor instead of considering the expected value with respect to the policies parametrized with (θ'^1, θ'^2). This results in the following optimization problem,

$$\max_{\theta'^1} \min_{\theta'^2} \; E_{\tau\sim f(\cdot;\theta^1,\theta^2)} \sum_{k=0}^{|\tau|-1} \gamma^k \frac{\pi(a_k^1|s_k;\theta'^1)\pi(a_k^2|s_k;\theta'^2)}{\pi(a_k^1|s_k;\theta^1)\pi(a_k^2|s_k;\theta^2)} A(s_k, a_k^1, a_k^2; \theta^1, \theta^2), \tag{62}$$
$$\text{subject to: } D_{KL}((\theta^1, \theta^2), (\theta'^1, \theta'^2)) \leq \delta'.$$

For the game in Eq. 62 we can compute the gradient $D_{\theta^i} L$ and bilinear term $D_{\theta^i\theta^j} L$ given player $i, j \in \{1, 2\}$, $i \neq j$ as

$$\begin{aligned} D_{\theta^i} L &= E_{\tau\sim f(\cdot;\theta^1,\theta^2)} \sum_{k=0}^{|\tau|-1} \gamma^k \frac{D_{\theta^i}\pi(a_k^i|s_k;\theta^i)}{\pi(a_k^i|s_k;\theta^i)} A(s_k, a_k^1, a_k^2; \theta^1, \theta^2), \\ D_{\theta^i\theta^j} L &= E_{\tau\sim f(\cdot;\theta^1,\theta^2)} \sum_{k=0}^{|\tau|-1} \gamma^k \frac{D_{\theta^i}\pi(a_k^i|s_k;\theta^i) D_{\theta^j}\pi(a_k^j|s_k;\theta^j)}{\pi(a_k^i|s_k;\theta^i)\pi(a_k^j|s_k;\theta^j)} A(s_k, a_k^1, a_k^2; \theta^1, \theta^2), \end{aligned} \tag{63}$$

which allows us to formulate the following bilinear approximation of the optimization problem,

$$\max_{\Delta\theta^1;\Delta\theta^1+\theta^1\in\Theta^1} \min_{\Delta\theta^2;\Delta\theta^2+\theta^2\in\Theta^2} \Delta\theta^{1\top} D_{\theta^1} L + \Delta\theta^{1\top} D_{\theta^1\theta^2} L \Delta\theta^2 + \Delta\theta^{2\top} D_{\theta^2} L + \Delta\theta^{2\top} D_{\theta^2\theta^1} L \Delta\theta^1,$$
$$\text{subject to:} \qquad D_{KL}((\theta^1, \theta^2), (\theta'^1, \theta'^2)) \leq \delta'.$$

However, to efficiently solve the optimization problem we also need to approximate the KL divergence for the two agent case, similar to Schulman et al. [2015] we use a quadratic approximation of the constraint.

Approximation of KL divergence Let us derive an approximation of the KL divergence for the two player case that can be estimated efficiently using Monte Carlo samples. As defined in Sec. 3.3, the KL divergence between (θ^1, θ^2) and (θ'^1, θ'^2) is given by,

$$D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big) = \int_\tau f(\tau; \theta^1, \theta^2) \log\left(\frac{f(\tau; \theta^1, \theta^2)}{f(\tau; \theta'^1, \theta'^2)}\right) d\tau, \tag{64}$$

Let us define $\theta = \big(\theta^{1\top}, \theta^{2\top}\big)^\top$, and $\Delta\theta = \theta' - \theta$, thus, using a Taylor series to approximate D_{KL} around θ we get,

$$D_{KL}(\theta, \theta') = D_{KL}(\theta, \theta')\Big|_{\theta'=\theta} + \frac{\partial D_{KL}(\theta, \theta')}{\partial \theta'}\Big|_{\theta'=\theta} \Delta\theta + \frac{1}{2}\Delta\theta^\top A\big|_{\theta'=\theta}\Delta\theta + \varepsilon\big(||\Delta\theta||^3\big), \tag{65}$$

where,

$$A = \begin{bmatrix} D_{\theta'^1\theta'^1} D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big) & D_{\theta'^1\theta'^2} D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big) \\ D_{\theta'^2\theta'^1} D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big) & D_{\theta'^2\theta'^2} D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big) \end{bmatrix}. \tag{66}$$

In the Taylor expansion Eq. 65, the zero order term $D_{KL}(\theta, \theta')|_{\theta'=\theta} = 0$ and the first order term can be expanded as,

$$\frac{\partial D_{KL}(\theta, \theta')}{\partial \theta'}\Big|_{\theta'=\theta} = \Big(D_{\theta'^1} D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big)^\top, D_{\theta'^2} D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big)^\top\Big)^\top\Big|_{(\theta'^1,\theta'^2)=(\theta^1,\theta^2)}.$$

When solving for $D_{\theta'^1} D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big)$, we get,

$$D_{\theta'^1} D_{KL}\big((\theta^1, \theta^2), (\theta'^1, \theta'^2)\big)\Big|_{\theta'=\theta} = D_{\theta'^1}\left(\int_\tau f(\tau; \theta^1, \theta^2) \log\Big(\frac{f(\tau; \theta^1, \theta^2)}{f(\tau; \theta'^1, \theta'^2)}\Big) d\tau\right)\Big|_{(\theta'^1,\theta'^2)=(\theta^1,\theta^2)}$$

$$
\begin{aligned}
&= \int_\tau D_{\theta'^1}\big(f(\tau;\theta^1,\theta^2)\log f(\tau;\theta^1,\theta^2) - f(\tau;\theta^1,\theta^2)\log f(\tau;\theta'^1,\theta'^2)\big)d\tau \\
&= \int_\tau -\frac{f(\tau;\theta^1,\theta^2)}{f(\tau;\theta'^1,\theta'^2)} D_{\theta'^1} f(\tau;\theta'^1,\theta'^2)d\tau \Big|_{(\theta'^1,\theta'^2)=(\theta^1,\theta^2)} \\
&= -D_{\theta'^1}\Big(\int_\tau f(\tau;\theta'^1,\theta'^2)d\tau\Big) = -D_{\theta'^1}(1) = 0.
\end{aligned} \tag{67}
$$

Similarly, $D_{\theta'^2} D_{KL}\big((\theta^1,\theta^2),(\theta'^1,\theta'^2)\big) = 0$. Hence, we write that,

$$
\frac{\partial D_{KL}(\theta,\theta')}{\partial \theta'}\Big|_{\theta'=\theta} = 0.
$$

Ignoring higher order terms in the approximation, we get

$$
D_{KL}\big((\theta^1,\theta^2),(\theta'^1,\theta'^2)\big) \approx \frac{1}{2}\begin{bmatrix}\Delta\theta^1 & \Delta\theta^2\end{bmatrix}\begin{bmatrix}A_{11} & A_{12}\\ A_{21} & A_{22}\end{bmatrix}\begin{bmatrix}\Delta\theta^1\\ \Delta\theta^2\end{bmatrix}, \tag{68}
$$

where, A_{ij} is defined in Eq. 66. Let us now derive all the A_{ij} terms, starting with A_{11},

$$
\begin{aligned}
A_{11} &= D_{\theta'^1\theta'^1}\Big(\int_\tau f(\tau;\theta^1,\theta^2)\log\Big(\frac{f(\tau;\theta^1,\theta^2)}{f(\tau;\theta'^1,\theta'^2)}\Big)d\tau\Big) \\
&= \int_\tau D_{\theta'^1\theta'^1}\big(f(\tau;\theta^1,\theta^2)\log f(\tau;\theta^1,\theta^2) - f(\tau;\theta^1,\theta^2)\log f(\tau;\theta'^1,\theta'^2)\big)d\tau \\
&= \int_\tau D_{\theta'^1\theta'^1}\big(-f(\tau;\theta^1,\theta^2)\log f(\tau;\theta'^1,\theta'^2)\big)d\tau \\
&= -\int_\tau f(\tau;\theta^1,\theta^2) D_{\theta'^1\theta'^1}\log(f(\tau;\theta'^1,\theta'^2))d\tau,
\end{aligned} \tag{69}
$$

and now using the definition of $f(\tau;\theta^1,\theta^2)$ in Eq. 1, we get,

$$
\begin{aligned}
A_{11} &= -\int_\tau f(\tau;\theta^1,\theta^2) D_{\theta'^1\theta'^1}\Big(\log\big(p(s_0)\prod_{k=0}^{|\tau|-1}\pi(a_k^1|s_k;\theta'^1) \\
&\qquad\qquad \pi(a_k^2|s_k;\theta'^2)R(r_k|s_k,a_k^1,a_k^2)T(s_{k+1}|s_k,a_k^1,a_k^2)\big)\Big)d\tau \\
&= -\int_\tau f(\tau;\theta^1,\theta^2) D_{\theta'^1\theta'^1}\Big(\sum_k^{|\tau|-1}\log\pi(a_k^1|s_k;\theta'^1) + \log\pi(a_k^2|s_k;\theta'^2)\Big)d\tau \\
&= -\int_\tau f(\tau;\theta^1,\theta^2) D_{\theta'^1\theta'^1}\Big(\sum_k^{|\tau|-1}\log\pi(a_k^1|s_k;\theta'^1)\Big)d\tau.
\end{aligned} \tag{70}
$$

This formulation allows us to estimate A_{11} using Monte Carlo samples. Note that an interesting alternative to estimate A_{11} only using first order derivatives is in terms of the Fisher matrix. Using Eq. 69, we get,

$$
\begin{aligned}
A_{11} &= \int_\tau D_{\theta'^1}\Big(-f(\tau;\theta^1,\theta^2)D_{\theta'^1}\log(f(\tau;\theta'^1,\theta'^2))\Big)d\tau \\
&= \int_\tau D_{\theta'^1}\Big(-\frac{f(\tau;\theta^1,\theta^2)}{f(\tau;\theta'^1,\theta'^2)}D_{\theta'^1}f(\tau;\theta'^1,\theta'^2)\Big)d\tau \\
&= \int_\tau -\frac{f(\tau;\theta^1,\theta^2)}{f(\tau;\theta'^1,\theta'^2)}D_{\theta'^1\theta'^1}f(\tau;\theta'^1,\theta'^2) \\
&\qquad + \frac{f(\tau;\theta^1,\theta^2)}{f(\tau;\theta'^1,\theta'^2)^2}D_{\theta'^1}f(\tau;\theta'^1,\theta'^2)D_{\theta'^1}f(\tau;\theta'^1,\theta'^2)^\top d\tau.
\end{aligned}
$$

Similar to Eq. 67, using the above equation we can write A_{11} at $\theta'^1 = \theta^1, \theta'^2 = \theta^2$ as,

$$
A_{11}\Big|_{\theta'=\theta} = \int_\tau f(\tau;\theta^1,\theta^2)\frac{D_{\theta'^1}f(\tau;\theta'^1,\theta'^2)}{f(\tau;\theta'^1,\theta'^2)}\frac{D_{\theta'^1}f(\tau;\theta'^1,\theta'^2)}{f(\tau;\theta'^1,\theta'^2)}^\top d\tau
$$

$$= \mathbb{E}_{f(\tau;\theta^1,\theta^2)}\Big[D_{\theta^1}\log(f(\tau;\theta'^1,\theta'^2))D_{\theta^1}\log(f(\tau;\theta'^1,\theta'^2))^\top\Big].$$

Which by definition is the Fisher information matrix. Using the definition of $f(\tau;\theta^1,\theta^2)$ in Eq. 1 and further reformulations, we get the following estimate for A_{11},

$$A_{11}\Big|_{\theta'=\theta} = \mathbb{E}_{f(\tau;\theta^1,\theta^2)}\Big[D_{\theta^1}\sum_t^{|\tau|-1}\Big(\log(\pi(a^1_{t+1}|s_k;\theta'^1))\Big)D_{\theta^1}\sum_t^{|\tau|-1}\Big(\log(\pi(a^1_{t+1}|s_k;\theta'^1))^\top\Big)\Big].$$

Note that for our implementation, we use the first approach Eq. 70, but the second approach has the advantage that it avoids second order derivatives, which can potentially improve computation times. Using the same ideas as in the derivation of A_{11} in Eq. 70, we can find the A_{22} term, which is

$$A_{22} = -\int_\tau f(\tau;\theta^1,\theta^2)D_{\theta'^2\theta'^2}\Big(\sum_k^{|\tau|-1}\log\pi(a^2_k|s_k;\theta'^2)\Big)d\tau\ .$$

Next, in order to estimate the Hessian of the KL divergence, we want to derive a formulation for A_{12}. Similar to Eq. 70, A_{12} can be written as,

$$\begin{aligned}
A_{12} &= -\int_\tau f(\tau;\theta^1,\theta^2)D_{\theta'^1\theta'^2}\Big(\sum_t^{|\tau|-1}\log\pi(a^1_{t+1}|s_k;\theta'^1)+\log\pi(a^2_{t+1}|s_k;\theta'^2)\Big)d\tau\\
&= -\int_\tau f(\tau;\theta^1,\theta^2)\Big(D_{\theta'^1\theta'^2}\sum_t^{|\tau|-1}\Big(\log\pi(a^1_{t+1}|s_k;\theta'^1)\Big)+D_{\theta'^1\theta'^2}\sum_t^{|\tau|-1}\Big(\log\pi(a^2_{t+1}|s_k;\theta'^2)\Big)\Big)d\tau\\
&= 0+0\\
&= 0.
\end{aligned}$$

Using the same derivation we also get $A_{21}=0$.

Thus, the constraint on the KL divergence $D_{KL}((\theta^1,\theta^2),(\theta'^1,\theta'^2))\leq\delta'$ can be approximated by the following quadratic inequality constraint,

$$\Delta\theta^{1\top}A_{11}\Delta\theta^1+\Delta\theta^{2\top}A_{22}\Delta\theta^2\leq 2\delta', \tag{71}$$

which allows us to efficiently find an approximate solution for Eq. 13.

Derivation of the TRCoPO Update: Given the bilinear approximation of the objective Eq. 63 and the quadratic approximation of the KL divergence constraint Eq. 71, we can use the the method of Lagrange multiplier to reformulate the problem as the following optimization problem:

$$\begin{aligned}
&\max_{\Delta\theta^1;\Delta\theta^1+\theta^1\in\Theta^1}\Delta\theta^{1\top}D_{\theta^1}L+\Delta\theta^{1\top}D_{\theta^1\theta^2}L\Delta\theta^2-\frac{\lambda}{2}(\Delta\theta^{1\top}A_{11}\Delta\theta^1+\Delta\theta^{2\top}A_{22}\Delta\theta^2-2\delta'),\\
&\min_{\Delta\theta^2;\Delta\theta^2+\theta^2\in\Theta^2}\Delta\theta^{2\top}D_{\theta^2}L+\Delta\theta^{2\top}D_{\theta^2\theta^1}L\Delta\theta^1+\frac{\lambda}{2}(\Delta\theta^{1\top}A_{11}\Delta\theta^1+\Delta\theta^{2\top}A_{22}\Delta\theta^2-2\delta').
\end{aligned} \tag{72}$$

Note that in Eq. 58 the same game was used, and thus finding a Nash equilibrium $(\Delta\theta^{1,*},\Delta\theta^{2,*})$ to Eq. 72 allows to define the update rule as $(\theta'^1,\theta'^2)=(\theta^1,\theta^2)+(\Delta\theta^{1,*},\Delta\theta^{2,*})$. The Nash equilibrium $(\Delta\theta^{1,*},\Delta\theta^{2,*})$ of the game, for a fixed λ, can be obtained by differentiating Eq. 72 with respect to θ'^1, θ'^2 and setting the derivatives to zero:

$$\begin{aligned}
&D_{\theta^1}L+D_{\theta^1\theta^2}L\Delta\theta^2-\lambda(A_{11}\Delta\theta^1)=0,\\
&D_{\theta^2}L+D_{\theta^2\theta^1}L\Delta\theta^1+\lambda(A_{22}\Delta\theta^2)=0.
\end{aligned} \tag{73}$$

To solve this system of equations, we can formulate Eq. 73 in matrix form, using $\Delta\theta=[\Delta\theta^{1\top},\Delta\theta^{2\top}]^\top$ and $\delta=2\delta'$, which results in

$$\begin{bmatrix}D_{\theta^1}L\\D_{\theta^2}L\end{bmatrix}+\begin{bmatrix}0&D_{\theta^1\theta^2}L\\D_{\theta^2\theta^1}L&0\end{bmatrix}\Delta\theta+\lambda\begin{bmatrix}-A_{11}&0\\0&A_{22}\end{bmatrix}\Delta\theta=0\ .$$

This gives us a one dimensional manifold of solutions for $\Delta\theta$ when varying λ. Using the notation in Eq. 60, we can reformulate the system of equation as,

$$C \ + \ B\Delta\theta + \ \lambda A'\Delta\theta = 0,$$

which for non-singular $B + \lambda A'$ has the following solution,

$$\Delta\theta = -(B \ + \ \lambda A')^{-1} \, C$$

For any λ, in case of a unique solution, we obtain a point $\Delta\theta$. In the end we are interested in a $\Delta\theta$ solution that satisfies the approximate KL divergence constraint Eq. 71 or in other words,

$$\Delta\theta^{1\top} A_{11}\Delta\theta^1 + \Delta\theta^{2\top} A_{22}\Delta\theta^2 \leq 2\delta'$$

All λ which satisfy this constraint are legitimate solutions, but we prefer the one which assigns higher value to the left hand side of this equation. Therefore, considering $\frac{1}{\lambda}$ as step size, we select a λ that satisfy this constraint with a high value for the left hand side using line search. The solution is given by computing $\Delta\theta = -(B + \lambda A')^{-1}C$ inside a line search, and choosing the solution that satisfies the following constraint with the biggest margin,

$$C^\top (B + \lambda A')^{-1^\top} A (B + \lambda A')^{-1} C \leq \delta.$$

This concludes our derivation.

14 IMPLEMENTATION OF ALGORITHMS

In this section we talk about final setup of CoPG and TRCoPO algorithms and their efficient implementation.

14.1 COMPETITIVE POLICY GRADIENT

Algorithm 6: Competitive Policy Gradient

Input: $\pi(.|.;\theta^1)$, $\pi(.|.;\theta^2)$, $V_\phi(s)$, N, B, α, Choice of $A(s, a^1, a^2; \theta^1, \theta^2)$ mentioned in Sec 3.1
for $epoch : e = 1$ ***to*** N **do**
  for $batch : b = 1$ ***to*** B **do**
    for $k = 0$ ***to*** $|\tau| - 1$ **do**
      Sample $a_k^1 \sim \pi(a_k^1|s_k;\theta^1)$, $a_k^2 \sim \pi(a_k^2|s_k;\theta^2)$.
      Execute actions $a_k = (a_k^1, a_k^2)$
      Record $\{s_k, a_k^1, a_k^2, r_k^1, r_k^2\}$ in τ_b
    end
    Record τ_b in ***M***
  end
  1. Estimate $A(s, a^1, a^2; \theta^1, \theta^2)$ with τ in M
  2. Estimate $D_{\theta^1}\eta$, $D_{\theta^2}\eta$, $D_{\theta^2\theta^1}\eta$, $D_{\theta^1\theta^2}\eta$ using Thm. 1
  3. Simultaneously update (θ^1, θ^2) using Eq. 6
end

Based on our discussion in Sec. 11 we propose Alg. 6 which is an extension of Alg. 2 to utilise advantage function estimation. In this algorithm, simultaneous update of (θ^1, θ^2) using Eq. 6 requires computing inverse matrix product twice, one for each agent. Instead we can compute infinite recursion strategy using Eq. 6 for one player only and apply the counter strategy for the other player as proposed in Schaefer and Anandkumar [2019]. From Eq. 6, for player 1 $\Delta\theta^1 = \alpha\big(Id + \alpha^2 D_{\theta^1\theta^2}\eta D_{\theta^2\theta^1}\eta\big)^{-1}\big(D_{\theta^1}\eta - \alpha D_{\theta^1\theta^2}\eta D_{\theta^2}\eta\big)$. Using this, one can derive an optimal counter strategy $\Delta\theta^2$ for player 2 by solving the argmin in Eq. 5. This results in,

$$\Delta\theta^2 = -\alpha\big(D_{\theta^2\theta^1}\eta\Delta\theta^1 + D_{\theta^2}\eta\big),$$

which is computed without evaluating an inverse. Similarly, if one evaluates $\Delta\theta^2$ from Eq. 6, the optimal counter strategy $\Delta\theta^1$ can be obtain by solving the argmax in Eq. 5, which is given by,

$$\Delta\theta^1 = \alpha\big(D_{\theta^1\theta^2}\eta\Delta\theta^2 + D_{\theta^1}\eta\big).$$

This reduces the requirement of computing inverse to only once per epoch. Further, CoPG has been extended to utilise RMSProp to iteratively adapt the step size as proposed in Schäfer et al. [2019]. In RMSProp, the learning rate is adapted based on the moving average of the square of gradients. The code for CoPG along with RMS Prop is also available in the code repository.

14.2 TRUST REGION COMPETITIVE POLICY OPTIMIZATION

Based on our discussion in Sec. 13.3, we propose the trust region competitive policy optimization algorithm given in Alg. 7.

Algorithm 7: Trust Region Competitive Policy Optimisation

Input: $\pi(.|.;\theta^1)$, $\pi(.|.;\theta^2)$, N, B, δ, Choice of $A(s,a^1,a^2;\theta^1,\theta^2)$ mentioned in Sec 3.1
for $epoch : e = 1$ ***to*** N **do**
  for $batch : b = 1$ ***to*** B **do**
    for $k = 0$ ***to*** $|\tau| - 1$ **do**
      Sample $a_k^1 \sim \pi(a_k^1|s_k;\theta_1)$, $a_k^2 \sim \pi(a_k^2|s_k;\theta_2)$.
      Execute actions $a_k = (a_k^1, a_k^2)$
      Record $\{s_k, a_k^1, a_k^2, r_k^1, r_k^2\}$ in τ_b
    end
    Record τ_b in ***M***
  end
  1. Estimate $A(s,a^1,a^2;\theta^1,\theta^2)$ with τ in M
  2. Estimate $D_{\theta^1}L,\ D_{\theta^1}L,\ D^2_{\theta^1\theta^2}L\ D^2_{\theta^2\theta^1}L$ using Eq. 63
  3. Solve for $\Delta\theta = [{\Delta\theta^1}^\top, {\Delta\theta^2}^\top]^\top$ in Eq. 59 using line search,
  while $\Delta\theta^\top A\Delta\theta < \delta$ **do**
    $\lambda \leftarrow 2\lambda$
    $\Delta\theta \leftarrow -(B + \lambda A')^{-1}C$
  end
  $\theta \leftarrow \theta + \Delta\theta$
end

In the experiments, we used Eq. 70 to estimate A'. As given in the Alg. 7, **3.**, we suggest to fulfill the constraint using a line search method. Start with small a value of $\lambda = \lambda_0$, and solve for $\Delta\theta = -(B + \lambda A')^{-1}C$. Check for $\Delta\theta^T A\Delta\theta < \delta$. If not satisfied change λ to 2λ and repeat it until the constraint is satisfied.

There could be multiple heuristics to initialize the λ value for a fast convergence. We propose to solve for λ_0 assuming 2 agents solve for their objective independently (TRGDA). The λ value can be obtained in closed form using single-agent TRPO [Schulman et al., 2015]. Let us say we obtain λ_{min} and λ_{max} corresponding to the minimizing and maximizing player. We propose to use $\lambda_0 = \min\{\lambda_{min}, \lambda_{max}\}$ as an initialisation.

Similar to single-agent TRPO, often the resulting step size from the algorithm might be large and thus probably does not benefit either of the agents with policy updates. During such cases, we can further refine the line search iteratively for a smaller step size and update the policies only if both players gain an advantage.

15 EXPERIMENT DETAILS

In this section, we provide all the details and explanations of the experiment platforms used in this work. First, we start with the games with a known closed-form Nash equilibrium. Later we talk about the Markov Soccer game and the car racing game.

15.1 LINEAR QUADRATIC (LQ) GAME

The zero sum LQ game is defined by:

$$J(K^1, K^2) = \max_{K^1} \min_{K^2} \mathbb{E}\left[\sum_{k=0}^{|\tau|} \gamma^k \left({s_k}^T Q s_k + {a_k^1}^T R_{11} a_k^1 + {a_k^2}^T R_{22} a_k^2\right)\right], \tag{74}$$
$$\text{where: } s_{k+1} = A s_k + B_1 a_k^1 + B_2 a_k^2, \qquad a_{k+1}^1 = -K^1 s_k,\ a_{k+1}^2 = -K^2 s_k,$$

where, $s_k \in R^n$ is the system state vector, $a^1 \in R^{d_1}$ and $a^2 \in R^{d_2}$ are the control inputs of player 1 and 2 respectively. The matrix $A \in R^{n \times n}$, $B_1 \in R^{n \times d_1}$ and $B_1 \in R^{n \times d_2}$ describe the system dynamics. Here, K^1 and K^2 denotes the control policies. We considered a simple game where, $A = 0.9,\ B_1 = 0.8,\ B_2 = 1.5,\ R_{11} = 1,\ R_{22} = 1$. These environment parameters are not known to the players. The policy of each player is linear in the state and actions are sampled from Gaussian distribution $a^1 \sim \mathcal{N}(\mu^1, \sigma^1)$, $a^2 \sim \mathcal{N}(\mu^2, \sigma^2)$, with initial values $\mu_0^1 = 0.1$ and $\mu_0^2 = -0.1$ and $\log(\sigma_0^1) = \log(\sigma_0^2) = 0.1$. The game dynamics and rewards follows Eq. 74. We collected a batch of 1000 trajectories, each 5 time steps long. The optimal control policy $(K^{1*}, K^{2*} = -0.5735, -0.3059)$, obtained using coupled riccatti equations [Zhang et al., 2019]. The experiment is performed for 5 different random seeds. As per discussion in Apx. 11, we used GAE for advantage estimation with $\lambda = 0.95$ and $\gamma = 0.99$. Fig. 10 shows the difference in game objective due to policy update from (θ^1, θ^2) to (θ'^1, θ'^2) given by $\eta(\theta^1, \theta^2) - \eta(\theta^1, \theta^2) = \mathbb{E}_{\tau \sim f(\cdot; \theta'^1, \theta'^2)} \sum_{k=0}^{|\tau|-1} \gamma^k A(s, a^1, a^2; \theta^1, \theta^2)$ of GDA and CoPG for different learning rates. The number of iterations it takes to converge for different learning rates is given in Table 4.

learning rate	0.1	0.05	0.01	0.005	0.001	0.0001
GDA	⊘	⊘	113	139	223	611
CoPG	75	87	105	127	195	590

Table 4: Number of iterations to converge to optimal policies

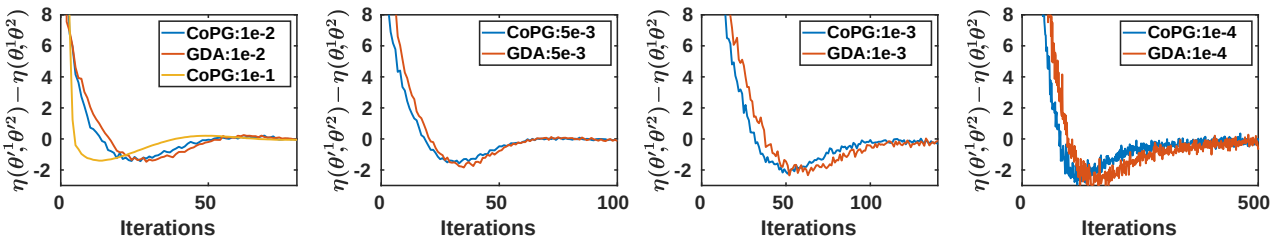


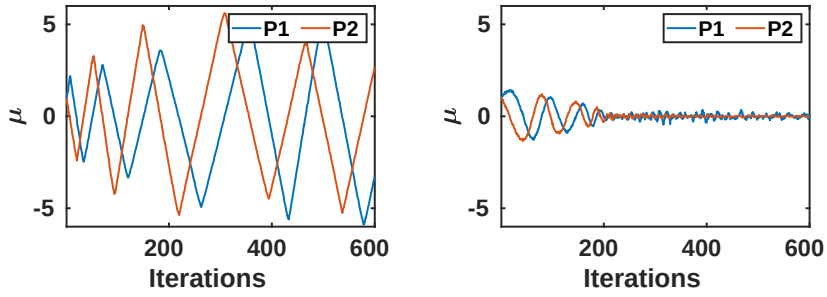
Figure 10: Difference in game objective due to policy updates for LQ game at different learning rates

15.2 BILINEAR GAME

In a bilinear game both players play actions simultaneously and receive rewards based on the product of their actions,

$$r^1(a^1, a^2) = a^1 a^2, \qquad r^2(a^1, a^2) = -a^1 a^2, \tag{75}$$

where $a^1 \sim \mathcal{N}(\mu_1, \sigma_1)$ and $a^2 \sim \mathcal{N}(\mu_2, \sigma_2)$ are actions of player 1 and player 2 respectively. In our experimental setup, a player's policy is modelled as a Gaussian distribution with $\mu_i, \sigma_i, i \in \{1, 2\}$ as mean and variance respectively. The policy is randomly initialised. We collected batch of 1000 trajectories. The experiment was performed for learning rates between 0.5 and 0.05 for 8 random seeds. The main paper features results of CoPG vs GDA, but even for the trust region approach we observe a similar contrast between TRCoPO and TRGDA. Starting from some random policy initialization, TRCoPO converges to the unique Nash equilibrium of the game whereas TRGDA diverges for all δ. Fig. 11 shows such behavior for δ of 0.001.



(a) (b)

Figure 11: Policy trajectories of player 1 in a bilinear game. (a) TRGDA (b) TRCoPO

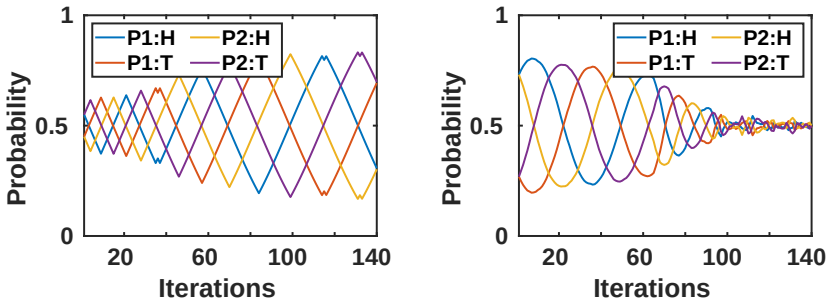
15.3 MATCHING PENNIES

The matching pennies game is played between two players, each player holds a coin and hence has two possible actions {Head, Tail}. Both the players secretly turn their coin to either head or tail. If the pennies of both the players matches then player 1 wins and player 2 looses. Otherwise, if pennies do not match then player 2 wins and player 1 loses. The game matrix is given in Table 5 captures win and lose of players for every possible action pair.

	H	T
H	1,-1	-1,1
T	-1,1	1,-1

Table 5: Game matrix for matching pennies game

Players' policy is modeled with a two-class categorical distribution, which is randomly initialized. They sample actions from a softmax probability function over the categorical distribution and receive rewards for their actions according to the game matrix. We collect a batch of 1000 trajectories, to estimate gradient and bilinear terms in every epoch. CoPG converges to the Nash equilibrium of the game $(H, T) = \left(\frac{1}{2}, \frac{1}{2}\right)$. The experiment was performed for learning rates between 0.5 and 0.05 for 8 random seeds. The main paper features plot of CoPG vs GDA, but even for trust region approach we observe a similar contrast between TRCoPO and TRGDA. Starting from random policy initialization, TRCoPO converges to the unique Nash equilibrium of the game $(H, T) = \left(\frac{1}{2}, \frac{1}{2}\right)$, whereas TRGDA diverges for all KL-divergence bounds δ. Fig. 12 shows this behaviour for a δ of 0.01. This experiment was performed also performed for 8 different random seeds and the results where consistent.



(a) TRGDA (b) TRCoPO

Figure 12: Policy trajectory in matching pennies game

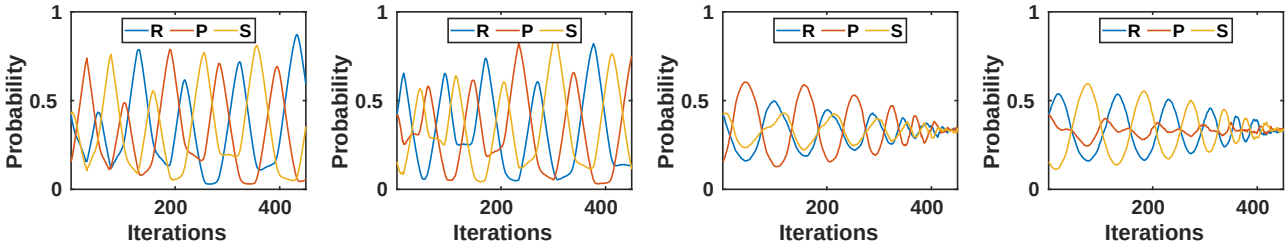
15.4 ROCK PAPER SCISSORS

The rock paper scissors game is played between two players, each player has three possible actions {Rock, Paper, Scissors}. Both players will simultaneously display one of three actions: a rock, a paper, or a scissors. Rock beats scissors, scissors beat paper by cutting it, and paper beats rock by covering it. This is captured in the game matrix given by Table 6. Players'

	Rock	Paper	Scissors
Rock	0,0	-1,1	1,-1
Paper	1,-1	0,0	-1,1
Scissors	-1,1	1,-1	0,0

Table 6: Game matrix for rock paper scissors game

policies are modeled with 3 class categorical distribution which is randomly initialized. Players sample action from a softmax probability function over categorical distribution. They receive a reward for their actions according to the game matrix. We collect a batch of 1000 trajectories. CoPG converges to the unique Nash equilibrium of the game, where probability of playing $(R, P, S) = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$. The experiment was performed for learning rates between 0.5 and 0.05 for 8 random seeds where CoPG converged in all cases and GDA diverged. Similarly for the trust region approach, the TRCoPO agent converges to a unique Nash equilibrium of the game (R, P, S)= $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$, where as TRGDA diverges. Fig. 13 shows this behavior for δ of 0.001. The experiment was performed for 8 different random seeds.



(a) Player 1 TRGDA (b) Player 2 TRGDA (c) Player 1 TRCoPO (d) Player 2 TRCoPO

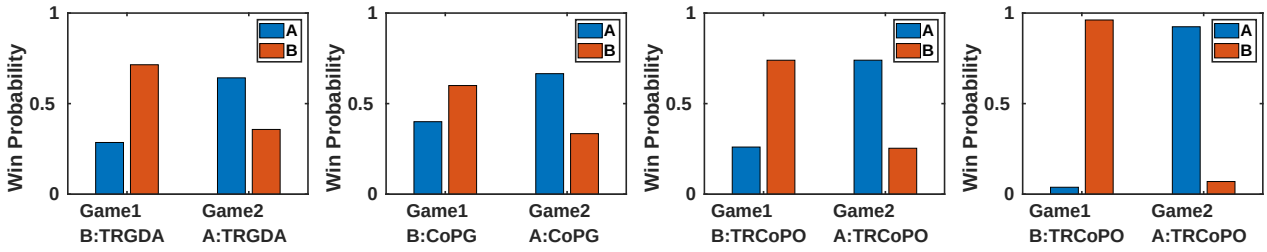
Figure 13: Policy trajectory in a rock paper scissors game

15.5 MARKOV SOCCER GAME

The soccer game setup is shown in Fig. 4. It is played between two players A and B, both are randomly initialized in one of the 4x5 grid cells. The ball is also randomly initialised in one of the 4x5 grid cells, this contrast to the previous studies, where one of the players was randomly initialised with the ball. Players are supposed to pick up the ball and place it in the opponent's goal. The players are allowed to move *up, down, left, right* or stand still. The player without the ball seizes the ball, if both players move simultaneously to the same grid cell. The game ends if a player scores a goal or none of them scores within 1000 steps. The winner receives a +1 reward and -1 is awarded to the losing player, hence the game is formulated as a zero sum game. This game does not have an optimal deterministic policy. This game depicts heavy interactions, where strategy depends on what the other player plays. Potentially, a good player can learn defending, dodging and scoring.

The state vector of a player P with respect to opponent O is $s^P = [x_{G_O}, y_{G_O}, x_{ball}, y_{ball}, x_O, y_O]$, where $P, O \in \{A, B\}$. G_O is goal of the opponent, x, y refers to the relative position from the player to the subscript, e.g., x_O, y_O is the relative position to the opponent O. The state vector of the game used during training is $s = [s^A, s^B]$, $s \in \mathcal{R}^{12}$, it captures the position of each player relative to the goal, the ball and relative to the opponent. The players' policy maps state vector of the game to a categorical distribution with 5 categories using a network with two hidden layers one with 64 and other with 32 neurons. Players sample actions from a softmax probability function over the categorical distribution. The players were trained using GDA and CoPG. In both the algorithms, players were trained for roughly 30,000 episodes until the number of wins against any good player saturates. In each epoch we collected a batch which consists of 10 trajectories. All the parameters were the same throughout the training for CoPG and GDA. The experiment was tested with 6 different random seeds. We used learning rate of 0.01 and GAE for advantage estimation with λ of 0.95 and γ of 0.99.

Similarly, for the trust region approach the same game setting was used, with a maximum KL divergence δ of 0.0001. The players were trained with TRCoPO or TRGDA. Both the agents were trained for 5,000 episodes, each episode denotes a batch update where each batch consists of 10 trajectories. For comparison, we played 10000 games between agents trained using TRCoPO and with that using TRGDA, in which for 50% of the games TRCoPO was player A and for remaining TRGDA was player A. TRGDA generates unequal players, on competing with the stronger TRGDA agent, the TRCoPO agent still won more than 80% games against TRGDA as shown in Fig. 3h. TRCoPO was able to learn better tactics for snatching the ball and dodging and defending its goal post. The histogram in Fig. 14, depicts a comparison of GDA, CoPG, TRGDA, and TRCoPO in the soccer game playing against each other.



(a) GDA vs TRGDA (b) TRGDA vs CoPG (c) CoPG vs TRCoPO (d) GDA vs TRCoPO

Figure 14: Win probability in soccer game played between GDA, CoPG, TRGDA and TRCoPO

15.6 THE CAR RACING GAME

Our final experimental setup is a racing game between two miniature race cars, where the goal is to finish the one lap race first. This involves both, learning a policy that can control a car at the limit of handling, as well as strategic interactions with the opposing car. Only if a player has the correct relative position to the opponent, this player is able to overtake or to block an overtaking. Our simulation study mimics the experimental platform located at ETH Zurich, which use miniature autonomous race cars.Following [Liniger et al., 2015] the dynamics of one car is modeled as a dynamic bicycle model with Pacejka tire models [Bakker et al., 1987]. However, compared to [Liniger et al., 2015] we formulate the dynamics in curvilinear coordinates Vázquez et al. [2020] where the position and orientation are represented relative to a reference path. This change in coordinates significantly simplifies the definition of our reward, and simplified the policy learning. The resulting state of a single car is given as $z = [\rho, d, \mu, V_x, V_y, \psi]^T$, where ρ is the progress along a reference path, d is the deviation from a reference path, μ is the local heading with respect to the reference path, V_x and V_y are the longitudinal and the lateral velocity respectively in car frame and ψ is the yawrate of the car. The inputs to the car is $[D, \delta]^T$, where $D \in [-1, 1]$ is duty cycle input to the electric motor varying from full braking at -1 to full acceleration at 1 and $\delta \in [-1, 1]$ is the steering angle. The test track which consists of 13 turns with different curvature can be seen in Figure 6. For the reference path we used the X-Y path obtained by a laptime optimization tool Vázquez et al. [2020], note that it is not necessary to use a pre-optimize reference path, but we saw that it helped the convergence of the algorithm. Finally, to get a discrete time MDP we discretize the continuous time dynamics using an RK4 integrator with a sampling time of 0.03s.

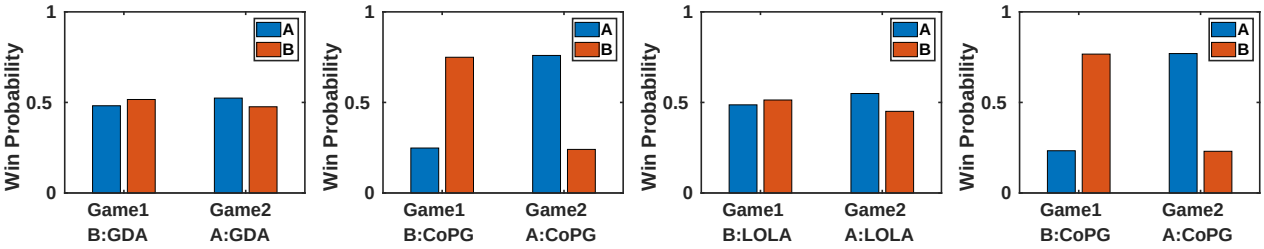
To formulate the racing game between two cars, we first define the state of the game as the concatenated state of the two players, $s = [z^1, z^2]$. Second we convert the objective of finishing the race first without an accident into a zero sum game. Therefore, we define the following reward function using reward shaping. First to model our no accident constraints we use a freezing mechanism: (i) If the car is leaving the track we freeze it until the episode ends, (ii) if the cars collide, the rear car (car with the lower progress ρ) is stopped for 0.1s, which corresponds to a penalty of about two car lengths. Note that this gives an advantage to the car ahead, but in car racing the following car has to avoid collisions. Furthermore, an episode ends if both cars are either frozen due to leaving the track or the first car finished the lap. Finally, to reward the players receive at every time step is $r(s_k, a^1_{k+1}, a^2_{k+1}) = \Delta\rho_{car_1} - \Delta\rho_{car_2}$, where $\Delta\rho = \rho_{k+1} - \rho_k$. This reward encourages the player to get as far ahead of the of the opponent player as possible. Note that the reward is zero sum, and the collision constraints do only indirectly influence the reward.

For the training we started the players on the start finish line ($\rho = 0$) and randomly assigned $d \in \{0.1, -0.1\}$. We also limited one episode to 700 time steps, which is about twice as long as it takes an expert player to finish a lap. For each policy gradient step we generated a batch of 8 game trajectories, and we run the training for roughly 20000 epochs until the player consistently finish the lap. This takes roughly 62 hours of training for a single core CPU. To increase the performance and

robustness of learning we adapted the learning rate with RMS prop for both GDA and CoPG and used a slow learning rate of 5.10^{-5} for both players. For our experiments we run 8 different random seeds and report the numbers of the best seed. As a policy we use a multi-layer perceptron with two hidden layers, each with 128 neurons, we used ReLU activation functions and a Tanh output layer to enforce the input constraints. GAE is used for advantage estimation with λ of 0.95 and γ of 0.99. The same setting was also used to compare performance of TRCoPO and TRGDA. The maximum KL divergence δ was set between 1e-4 to 1e-5 and the training was performed for 8 different random seeds. The statistics of the experiments are presented in the main part of the paper Sec. 4.

15.7 COMPARISON WITH MADDPG AND LOLA

Finally, we give more details on our comparison of CoPG with MADDPG and LOLA. We performed this comparison on the Markov soccer and the racing game. We start with the comparison for the Markov soccer game. To compare the performance of MADDPG on discrete action spaces (such as in the Markov soccer game), we used policies that can produce differentiable samples through a Gumbel-Softmax distribution, similar to [Iqbal and Sha, 2019]. The rest of the experimental setup was kept as similar as explained in 15.5, and we trained both MADDPG and CoPG. Fig. 15a shows that a policy learned by MADDPG performs similar to that of GDA. But when the MADDPG player competes against CoPG, the CoPG agent wins 80% of the games. Second we studied LOLA, we observed that LOLA learns better policies than GDA potentially due to the second level reasoning, but LOLA still loses 72% of the games when playing against CoPG. Fig. 15 show the number of games won by CoPG when competing against MADDPG and LOLA.



(a) GDA vs MADDPG (b) CoPG vs MADDPG (c) LOLA vs GDA (d) CoPG vs LOLA

Figure 15: Comparison with LOLA and MADDPG on the Markov soccer game

Next we evaluated MADDPG and LOLA in the game of Car Racing, where we kept all the setting identical to 15.6. Fig. 5e show the progress of an MADDPG agent during learning, and we can see that it performs similar to GDA. The car learns to drive up to 30% of the track, but after that the agents show oscillatory behaviour in training. We also performed this experiment with LOLA, but due to the variance in estimating the bilinear term and more importantly the non adaptive learning rate of LOLA it does not learn to drive around the track.

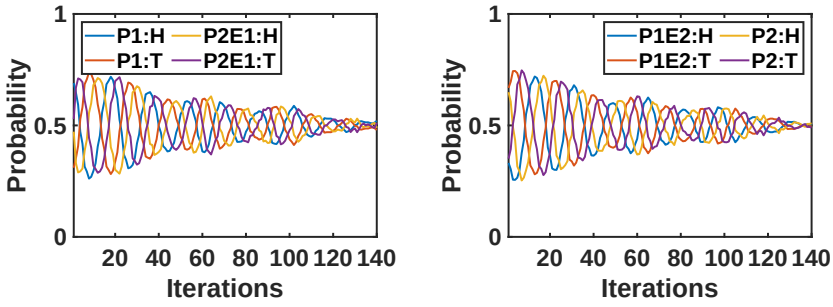
16 OPPONENT PARAMETER ESTIMATION

In the following section we conducted experiments comparing CoPG, with its opponent parameter estimation counterpart CoPG-OP as explained in Sec. 3.2.

Matching pennies The game setting is explained in Apx. 15.3. The opponents' parameters are estimated by maximizing the likelihood using Eq. 9. The Fig. 16a shows the policy trajectory of player 1 and the estimation of the trajectory of player 2 by player 1 and vice versa in Fig. 16b for player 2. We observe that both the trajectories of player 1 and player 2 converge to the unique Nash equilibrium of the game which is p(H,T) = $(\frac{1}{2}, \frac{1}{2})$.

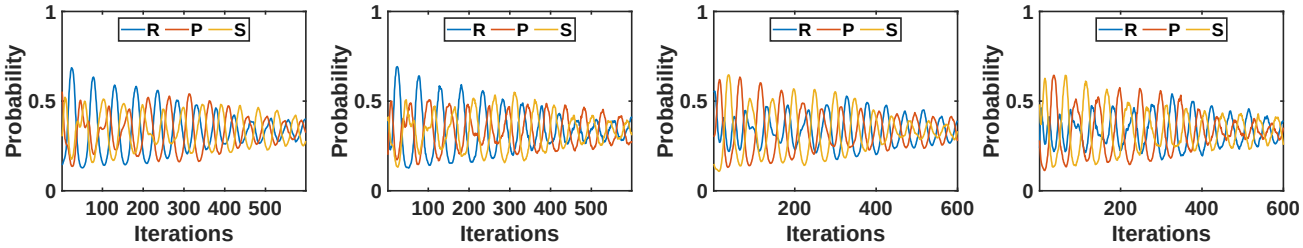
Rock paper scissors The game setting is explained in Apx. 15.4. The opponents' parameters are estimated by maximizing likelihood using Eq. 9. The Fig. 17a, 17b shows the policy trajectory of player 1 and the estimation of the trajectory of player 2 by player 1 and vice versa in Fig. 17c, Fig. 17d for player 2. We observe that both the trajectory of player 1 and player 2 converge to the unique Nash equilibrium of the game which is p(R,P, S) = $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$.

Markov Soccer We next tested the opponent modelling approach on the Markov soccer game, which is discussed in more detail in Sec. 15.5. We focus in this comparison on the Markov soccer game since it is strategic enough while being representative. The game has two player A and B. Player A maintains a record of policy of player B (B Est A)



(a) CoPG-OP Player 1 (b) CoPG-OP Player 2

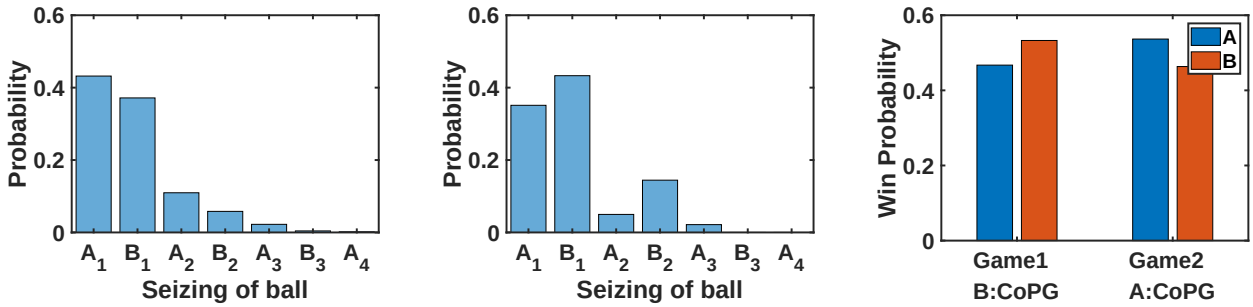
Figure 16: CoPG-OP policy trajectory of player P1 (P2) and its opponents' estimation P2E1 (P1E2) in matching pennies



(a) Player 1 (b) Player 2 estimated by player 1 (c) Player 2 (d) Player 1 estimated player 2

Figure 17: CoPG-OP policy trajectory of players and opponents' estimation in rock paper scissors

which is estimated online by observing state-action pairs of the player B. The state vector in perspective of player A is $s^1 = [s^A, s^{BestA}]$ and in player B $s^2 = [s^{AestB}, s^B]$, in accordance with the game setting explained in Apx. 15.5. In contrast to the CoPG experiment, in the case of CoPG-OP the batch size is increased to 40 to reliably estimate the opponent's parameters. The player estimates it's opponent's parameters by maximizing likelihood Eq. 9 with observed state-action pairs of the opponent. The player is trained while competing with the estimated opponent using Alg. 6. Fig. 18a below show the interaction plot for CoPG-OP agents with it's estimated opponent. We see that the CoPG-OP player learns to seize ball/interact(A_3, A_4) with the estimated opponent. Fig. 18b shows the interaction plot of CoPG-OP with CoPG, where we observe that CoPG-OP can also learns a policy similar to CoPG, where it learns to defend, escape and score the goal (A_3). When directly competing with CoPG, we observed that CoPG-OP can win 46.5% of the games against CoPG (Fig. 18c). CoPG has slight edge in the number of game wins, which is probably due to variance in the estimation of the opponent's parameter.



(a) CoPG-OP (A) vs CoPG-OP (B est A) (b) CoPG-OP vs CoPG (c) CoPG-OP vs CoPG

Figure 18: Interaction plots evaluated by playing 5000 games. Matches played between a) CoPG-OP player A and player B estimated by A b) CoPG-OP and CoPG c) Probability of games won between CoPG vs CoPG-OP.

17 GAIL CASE STUDY

The GAIL case study is conducted on the Car Racing game Apx. 15.6, with a single car, where $s = [z]$. The aim is to learn to drive around the track Fig. 6 by imitating experts. For expert trajectories, we collected a batch of 80 trajectories using four pure pursuit controllers each having a different set of tuning parameters. This ensures that the batch of expert trajectory is collected across various speeds and with a variation in the action set. The set of parameters are $pp_1 : \{k_p = 10, v_{tar} = 1.45, l_d = 26\}$, $pp_2 : \{k_p = 10, v_{tar} = 1.3, l_d = 15\}$, $pp_3 : \{k_p = 10, v_{tar} = 1.2, l_d = 20\}$ and $pp_4 : \{k_p = 8, v_{tar} = 1.35, l_d = 15\}$. Here, k_p is the proportional gain of the velocity control, l_d is the look ahead distance and v_{tar} is the target velocity.

In the GAIL training, one episode was limited to 700 time steps, which is about twice as long as it takes an expert player to finish a lap. For each competitive policy gradient step, we generated a batch of 8 game trajectories, and we run the training for roughly 4000 epochs until the player consistently finishes the lap. This takes roughly 2 hours of training for a single-core CPU. For our experiments, we run 8 different random seeds and report the numbers of the best seed. As a policy we use a multi-layer perceptron with two hidden layers, each with 128 neurons, we used ReLU activation functions and a Tanh output layer to enforce the input constraints. The agent receives a reward according to the GAIL framework Alg.2 in [Ho and Ermon, 2016]. GAE is used for advantage estimation with a λ of 0.95 and a γ of 0.99. The agent's policy and discriminator are updated simultaneously using CoPO, and the critic to CoPO update ration is 1:1. We use a slow learning rate of 3e-4 for CoPO and 8e-3 for the critic update.