Multi-Agent Generative Adversarial Imitation Learning

Jiaming Song
Stanford University
tsong@cs.stanford.edu

Hongyu Ren
Stanford University
hyren@cs.stanford.edu

Dorsa Sadigh
Stanford University
dorsa@cs.stanford.edu

Stefano Ermon
Stanford University
ermon@cs.stanford.edu

Abstract

Imitation learning algorithms can be used to learn a policy from expert demonstrations without access to a reward signal. However, most existing approaches are not applicable in multi-agent settings due to the existence of multiple (Nash) equilibria and non-stationary environments. We propose a new framework for multi-agent imitation learning for general Markov games, where we build upon a generalized notion of inverse reinforcement learning. We further introduce a practical multi-agent actor-critic algorithm with good empirical performance. Our method can be used to imitate complex behaviors in high-dimensional environments with multiple cooperative or competing agents.

1 Introduction

Reinforcement learning (RL) methods are becoming increasingly successful at optimizing reward signals in complex, high dimensional environments [1]. A key limitation of RL, however, is the difficulty of designing suitable reward functions for complex and not well-specified tasks [2, 3]. If the reward function does not cover all important aspects of the task, the agent could easily learn undesirable behaviors [4]. This problem is further exacerbated in multi-agent scenarios, such as multiplayer games [5], multi-robot control [6] and social interactions [7]; in these cases, agents do not even necessarily share the same reward function and might even have conflicting rewards.

Imitation learning methods address these problems via expert demonstrations [8–11]; the agent directly learns desirable behaviors by imitating an expert. Notably, inverse reinforcement learning (IRL) frameworks assume that the expert is (approximately) optimizing an underlying reward function, and attempt to recover a reward function that rationalizes the demonstrations; an agent policy is subsequently learned through RL [12, 13]. Unfortunately, this paradigm is not suitable for general multi-agent settings due to the environment being non-stationary to individual agents [14] and the existence of multiple equilibrium solutions [15]. The optimal policy of one agent could depend on the policies of other agents, and vice versa, so there could exist multiple solutions in which each agent's policy is the optimal response to others.

In this paper, we propose a new framework for multi-agent imitation learning – provided with demonstrations of a set of experts interacting with each other in the same environment, we aim to learn multiple parametrized policies that imitate the behavior of each expert respectively. Using the framework of Markov games, we integrate multi-agent RL with a suitable extension of multi-agent inverse RL. The resulting procedure strictly generalizes Generative Adversarial Imitation Learning (GAIL, [16]) in the single agent case. Imitation learning in our setting corresponds to a two-player

32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada.

game between a generator and a discriminator. The generator controls the policies of all the agents in a distributed way, and the discriminator contains a classifier for each agent that is trained to distinguish that agent's behavior from that of the corresponding expert. Upon training, the behaviors produced by the policies should be indistinguishable from the training data. We can incorporate prior knowledge into the discriminators, including the presence of cooperative or competitive agents. In addition, we propose a novel multi-agent natural policy gradient algorithm that addresses the issue of high variance gradient estimates commonly observed in reinforcement learning [14, 17]. Empirical results demonstrate that our method can imitate complex behaviors in high-dimensional environments, such as particle environments and cooperative robotic control tasks, with multiple cooperative or competitive agents; the imitated behaviors are close to the expert behaviors with respect to "true" reward functions which the agents do not have access to during training.

2 Preliminaries

2.1 Markov games

We consider an extension of Markov decision processes (MDPs) called Markov games [18]. A Markov game (MG) for N agents is defined via a set of states $\mathcal{S}$, and N sets of actions $\{\mathcal{A}_i\}_{i=1}^N$. The function $P : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathcal{P}(\mathcal{S})$ describes the (stochastic) transition process between states, where $\mathcal{P}(\mathcal{S})$ denotes the set of probability distributions over the set $\mathcal{S}$. Given that we are in state s_t at time t, the agents take actions $(a_1, \ldots, a_N)$ and the state transitions to s_{t+1} with probability $P(s_{t+1}|s_t, a_1, \ldots, a_N)$. Each agent i obtains a (bounded) reward given by a function $r_i : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$. Each agent i aims to maximize its own total expected return $R_i = \sum_{t=0}^{\infty} \gamma^t r_{i,t}$, where γ is the discount factor, by selecting actions through a (stationary and Markovian) stochastic policy $\pi_i : \mathcal{S} \times \mathcal{A}_i \to [0, 1]$. The initial states are determined by a distribution $\eta : \mathcal{S} \to [0, 1]$. The joint policy is defined as $\boldsymbol{\pi}(\mathbf{a}|s) = \prod_{i=1}^N \pi_i(a_i|s)$, where we use bold variables without subscript i to denote the concatenation of all variables for all agents (e.g., $\boldsymbol{\pi}$ denotes the joint policy $\prod_{i=1}^N \pi_i$ in a multi-agent setting, $\mathbf{r}$ denotes all rewards, $\mathbf{a}$ denotes actions of all agents). We use expectation with respect to a policy π to denote an expectation with respect to the trajectories it generates, and use subscript $-i$ to denote *all agents except for* i. For example, (a_i, a_{-i}) represents $(a_1, \ldots, a_N)$, the actions of all N agents.

2.2 Reinforcement learning and Nash equilibrium

In reinforcement learning (RL), the goal of each agent is to maximize total expected return $\mathbb{E}_\pi[r(s, a)]$ given access to the reward signal r. In single agent RL, an optimal Markovian policy exists but the optimal policy might not be unique (e.g., all policies are optimal for an identically zero reward; see [19], Chapter 3.8). An entropy regularizer can be introduced to resolve this ambiguity. The optimal policy is found via the following RL procedure:

$$\mathrm{RL}(r) = \arg\max_{\pi \in \Pi} H(\pi) + \mathbb{E}_\pi[r(s, a)], \tag{1}$$

where $H(\pi)$ is the γ-discounted causal entropy [20] of policy $\pi \in \Pi$.

Definition 1 (γ-discounted Causal Entropy)**.** *The γ-discounted causal entropy for a policy π is defined as follows:*

$$H(\pi) \triangleq \mathbb{E}_\pi[-\log \pi(a|s)] = \mathbb{E}_{s_t, a_t \sim \pi}\left[-\sum_{t=0}^{\infty} \gamma^t \log \pi(a_t|s_t)\right].$$

The addition of $H(\pi)$ in (1) resolves this ambiguity – the policy with both the highest reward and the highest entropy[1] is unique because the entropy function is strictly concave with respect to π.

In Markov games, however, the optimal policy of an agent depends on other agents' policies. One approach is to use an equilibrium solution concept, such as Nash equilibrium [15]. Informally, a set of policies $\{\pi_i\}_{i=1}^N$ is a Nash equilibrium if no agent can achieve higher reward by unilaterally

[1] We use the term "entropy" to denote the γ-discounted causal entropy for policies in the rest of the paper.

changing its policy, i.e., $\forall i \in [1, N], \forall \hat{\pi}_i \neq \pi_i, \mathbb{E}_{\pi_i, \pi_{-i}}[r_i] \geq \mathbb{E}_{\hat{\pi}_i, \pi_{-i}}[r_i]$. The process of finding a Nash equilibrium can be defined as a constrained optimization problem ([21], Theorem 3.7.2):

$$\min_{\boldsymbol{\pi} \in \Pi, \mathbf{v} \in \mathbb{R}^{\mathcal{S} \times N}} f_r(\boldsymbol{\pi}, \mathbf{v}) = \sum_{i=1}^{N} \left(\sum_{s \in \mathcal{S}} v_i(s) - \mathbb{E}_{a_i \sim \pi_i(\cdot|s)} q_i(s, a_i) \right) \tag{2}$$

$$v_i(s) \geq q_i(s, a_i) \triangleq \mathbb{E}_{\pi_{-i}} \left[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a}) v_i(s') \right] \quad \forall i \in [N], s \in \mathcal{S}, a_i \in \mathcal{A}_i \tag{3}$$

$$\mathbf{a} \triangleq (a_i, a_{-i}) \triangleq (a_1, \ldots, a_N) \qquad \mathbf{v} \triangleq [v_1; \ldots; v_N]$$

where the joint action $\mathbf{a}$ includes actions a_{-i} sampled from π_{-i} and a_i. Intuitively, $\mathbf{v}$ can be thought of as a value function and $\mathbf{q}$ represents the Q-function that corresponds to $\mathbf{v}$. The constraints enforce the Nash equilibrium condition – when the constraints are satisfied, $(v_i(s) - q_i(s, a_i))$ is non-negative for every $i \in [N]$. Hence $f_r(\boldsymbol{\pi}, \mathbf{v})$ is always non-negative for a feasible $(\boldsymbol{\pi}, \mathbf{v})$. Moreover, this objective has a global minimum of zero if a Nash equilibrium exists, and $\boldsymbol{\pi}$ forms a Nash equilibrium if and only if $f_r(\boldsymbol{\pi}, \mathbf{v})$ reaches zero while being a feasible solution ([22], Theorem 2.4).

2.3 Inverse reinforcement learning

Suppose we do not have access to the reward signal r, but have demonstrations $\mathcal{D}$ provided by an expert (N expert agents in Markov games). Imitation learning aims to learn policies that behave similarly to these demonstrations. In Markov games, we assume all experts/players operate in the same environment, and the demonstrations $\mathcal{D} = \{(s_j, \mathbf{a}_j)\}_{j=1}^{M}$ are collected by sampling $s_0 \sim \eta(s), \mathbf{a}_t = \pi_E(\mathbf{a}_t|s_t), s_{t+1} \sim P(s_{t+1}|s_t, \mathbf{a}_t)$; we assume knowledge of N, γ, $\mathcal{S}$, $\mathcal{A}$, as well as access to T and η as black boxes. We further assume that once we obtain $\mathcal{D}$, we cannot ask for additional expert interactions with the environment (unlike in DAgger [23] or CIRL [24]).

Let us first consider imitation in Markov decision processes (as a special case to Markov games) and the framework of single-agent Maximum Entropy IRL [8, 16] where the goal is to recover a reward function r that rationalizes the expert behavior π_E:

$$\mathrm{IRL}(\pi_E) = \operatorname*{arg\,max}_{r \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\pi_E}[r(s, a)] - \left(\max_{\pi \in \Pi} H(\pi) + \mathbb{E}_{\pi}[r(s, a)] \right)$$

In practice, expectations with respect to π_E are evaluated using samples from $\mathcal{D}$.

The IRL objective is ill-defined [12, 10] and there are often multiple valid solutions to the problem when we consider all $r \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$. To resolve this ambiguity, [16] introduce a convex reward function regularizer $\psi : \mathbb{R}^{\mathcal{S} \times \mathcal{A}} \to \mathbb{R}$, which can be used for example to restrict rewards to be linear in a pre-determined set of features [16]:

$$\mathrm{IRL}_\psi(\pi_E) = \operatorname*{arg\,max}_{r \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}} -\psi(r) + \mathbb{E}_{\pi_E}[r(s, a)] - \left(\max_{\pi \in \Pi} H(\pi) + \mathbb{E}_{\pi}[r(s, a)] \right) \tag{4}$$

2.4 Imitation by matching occupancy measures

[16] interprets the imitation learning problem as matching two occupancy measures, i.e., the distribution over states and actions encountered when navigating the environment with a policy. Formally, for a policy π, it is defined as $\rho_\pi(s, a) = \pi(a|s) \sum_{t=0}^{\infty} \gamma^t P(s_t = s|\pi)$. [16] draws a connection between IRL and occupancy measure matching, showing that the former is a dual of the latter:

Proposition 1 (Proposition 3.1 in [16])**.**

$$\mathrm{RL} \circ \mathrm{IRL}_\psi(\pi_E) = \operatorname*{arg\,min}_{\pi \in \Pi} -H(\pi) + \psi^\star(\rho_\pi - \rho_{\pi_E})$$

Here $\psi^\star(x) = \sup_y x^\top y - \psi(y)$ is the convex conjugate of ψ, which could be interpreted as a measure of similarity between the occupancy measures of expert policy and agent's policy. One instance of $\psi = \psi_{\mathrm{GA}}$ gives rise to the Generative Adversarial Imitation Learning (GAIL) method:

$$\psi^\star_{\mathrm{GA}}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in (0,1)^{\mathcal{S} \times \mathcal{A}}} \mathbb{E}_{\pi_E}[\log(D(s, a))] + \mathbb{E}_{\pi}[\log(1 - D(s, a))] \tag{5}$$

The resulting imitation learning method from Proposition 1 involves a *discriminator* (a classifier D) competing with a *generator* (a policy π). The discriminator attempts to distinguish real vs. synthetic trajectories (produced by π) by optimizing (5). The generator, on the other hand, aims to perform optimally under the reward function defined by the discriminator, thus "fooling" the discriminator with synthetic trajectories that are difficult to distinguish from the expert ones.

3 Generalizing IRL to Markov games

Extending imitation learning to multi-agent settings is difficult because there are multiple rewards (one for each agent) and the notion of optimality is complicated by the need to consider an equilibrium solution [15]. We use $\mathrm{MARL}(\mathbf{r})$ to denote the set of (stationary and Markovian) policies that form a Nash equilibrium under $\mathbf{r}$ and have the maximum γ-discounted causal entropy (among all equilibria):

$$\begin{aligned} \mathrm{MARL}(\mathbf{r}) = \arg\min_{\boldsymbol{\pi}\in\Pi} \min_{\mathbf{v}\in\mathbb{R}^{\mathcal{S}\times N}} f_r(\boldsymbol{\pi}, \mathbf{v}) - H(\boldsymbol{\pi}) \\ v_i(s) \geq q_i(s, a_i) \quad \forall i \in [N], s \in \mathcal{S}, a_i \in \mathcal{A}_i \end{aligned} \tag{6}$$

where q is defined as in Equation (3). Our goal is to define a suitable inverse operator MAIRL, in analogy to IRL in Equation (4), which chooses a reward that creates a *margin* between the expert and every other policy. However, the *constraints* in the Nash equilibrium optimization (Equation (6)) can make this challenging. To that end, we derive an equivalent Lagrangian formulation of (6), where we "move" the constraints into the objective function, so that we can define a margin between the expected reward of two sets of policies that captures their "difference".

3.1 Equivalent constraints via temporal difference learning

Intuitively, the Nash equilibrium constraints imply that any agent i cannot improve π_i via 1-step temporal difference learning; if the condition for Equation (3) is not satisfied for some v_i, q_i, and (s, a_i), this would suggest that we can update the policy for agent i and its value function. Based on this notion, we can derive equivalent versions of the constraints corresponding to t-step temporal difference (TD) learning.

Theorem 1. *For a certain policy $\boldsymbol{\pi}$ and reward $\mathbf{r}$, let $\hat{v}_i(s; \boldsymbol{\pi}, \mathbf{r})$ be the unique solution to the Bellman equation:*

$$\hat{v}_i(s; \boldsymbol{\pi}, \mathbf{r}) = \mathbb{E}_{\mathbf{a}\sim\boldsymbol{\pi}} \left[r_i(s, \mathbf{a}) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s, \mathbf{a}) \hat{v}_i(s'; \boldsymbol{\pi}, \mathbf{r}) \right] \qquad \forall s \in \mathcal{S}.$$

Denote $\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)}; \boldsymbol{\pi}, \mathbf{r})$ as the discounted expected return for the i-th agent conditioned on visiting the trajectory $\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}$ in the first $(t-1)$ steps and choosing action $a_i^{(t)}$ at the t step, when other agents use policy π_{-i}:

$$\begin{aligned} &\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)}; \boldsymbol{\pi}, \mathbf{r}) \\ = &\sum_{j=0}^{t-1} \gamma^j r_i(s^{(j)}, \mathbf{a}^{(j)}) + \gamma^t \mathbb{E}_{a_{-i}\sim\pi_{-i}} \left[r_i(s^{(t)}, \mathbf{a}^{(t)}) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s, \mathbf{a}^{(t)}) \hat{v}_i(s'; \boldsymbol{\pi}, \mathbf{r}) \right]. \end{aligned}$$

Then $\boldsymbol{\pi}$ is a Nash equilibrium if and only if for all $t \in \mathbb{N}^+, i \in [N], j \in [t], s^{(j)} \in \mathcal{S}, \mathbf{a}^{(j)} \in \mathcal{A}$

$$\hat{v}_i(s^{(0)}; \boldsymbol{\pi}, \mathbf{r}) \geq \mathbb{E}_{a_{-i}\sim\pi_{-i}} \left[\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)}; \boldsymbol{\pi}, \mathbf{r}) \right] \triangleq Q_i^{(t)}(\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}; \boldsymbol{\pi}, \mathbf{r}). \tag{7}$$

Intuitively, Theorem 1 states that if we replace the 1-step constraints with $(t+1)$-step constraints, we obtain the same solution as $\mathrm{MARL}(\mathbf{r})$, since $(t+1)$-step TD updates (over one agent at a time) are still stationary with respect to a Nash equilibrium solution. So the constraints can be unrolled for t steps and rewritten as $\hat{v}_i(s^{(0)}) \geq Q_i^{(t)}(\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}; \boldsymbol{\pi}, \mathbf{r})$ (corresponding to Equation (7)).

3.2 Multi-agent inverse reinforcement learning

We are now ready to construct the Lagrangian dual of the primal in Equation (6), using the equivalent formulation from Theorem 1. The first observation is that for any policy $\boldsymbol{\pi}$, $f(\boldsymbol{\pi}, \hat{v}) = 0$ when $\hat{v}$ is defined as in Theorem 1 (see Lemma 1 in appendix). Therefore, we only need to consider the "unrolled" constraints from Theorem 1, obtaining the following dual problem

$$\max_{\lambda \geq 0} \min_{\boldsymbol{\pi}} L_{\mathbf{r}}^{(t+1)}(\boldsymbol{\pi}, \lambda) \triangleq \sum_{i=1}^{N} \sum_{\tau_i \in \mathcal{T}_i^t} \lambda(\tau_i) \left(Q_i^{(t)}(\tau_i; \boldsymbol{\pi}, \mathbf{r}) - \hat{v}_i(s^{(0)}; \boldsymbol{\pi}, \mathbf{r}) \right) \tag{8}$$

where $\mathcal{T}_i(t)$ is the set of all length-t trajectories of the form $\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}$, with $s^{(0)}$ as initial state, λ is a vector of $N \cdot |\mathcal{T}_i(t)|$ Lagrange multipliers, and $\hat{v}$ is defined as in Theorem 1. This dual formulation is a sum over agents and trajectories, which uniquely corresponds to the constraints in Equation 7.

In the following theorem, we show that for a specific choice of λ we can recover the difference of the sum of expected rewards between two policies, a performance gap similar to the one used in single agent IRL in Equation (4). This amounts to "relaxing" the primal problem.

Theorem 2. *For any two policies $\boldsymbol{\pi}^\star$ and $\boldsymbol{\pi}$, let*

$$\lambda_{\boldsymbol{\pi}}^\star(\tau_i) = \eta(s^{(0)}) \pi_i(a_i^{(0)}|s^{(0)}) \prod_{j=1}^{t} \pi_i(a_i^{(j)}|s^{(j)}) \sum_{a_{-i}^{(j-1)}} P(s^{(j)}|s^{(j-1)}, a^{(j-1)}) \pi_{-i}^\star(a_{-i}^{(j)}|s^{(j)})$$

be the probability of generating the sequence τ_i using policy π_i and $\pi_{-i}^\star$. Then

$$\lim_{t \to \infty} L_r^{(t+1)}(\boldsymbol{\pi}^\star, \lambda_{\boldsymbol{\pi}}^\star) = \sum_{i=1}^{N} \mathbb{E}_{\pi_i, \pi_{-i}^\star}[r_i(s,a)] - \sum_{i=1}^{N} \mathbb{E}_{\pi_i^\star, \pi_{-i}^\star}[r_i(s,a)] \tag{9}$$

where $L_r^{(t+1)}(\boldsymbol{\pi}^\star, \lambda_{\boldsymbol{\pi}}^\star)$ corresponds to the dual function where the multipliers are the probability of generating their respective trajectories of length t.

We provide a proof in Appendix A.3. Intuitively, the $\lambda^\star(\tau_i)$ weights correspond to the probability of generating trajectory τ_i when the policy is π_i for agent i and $\pi_{-i}^\star$ for the other agents. As $t \to \infty$, the first term of left hand side in Equation (8), $\sum_{i=1}^{N} \sum_{\tau_i \in \mathcal{T}_i^t} \lambda(\tau_i) Q_i^{(t)}(\tau_i)$, converges to the expected total reward $\mathbb{E}_{\pi_i, \pi_{-i}^\star}[r_i]$, which is the first term of right hand side. The marginal of $\lambda^\star$ over the initial state is the initial state distribution, so the second term of left hand side, $\sum_s \hat{v}(s)\eta(s)$, converges to $\mathbb{E}_{\pi_i^\star, \pi_{-i}^\star}[r_i]$, which is the second term of right hand side. Thus, the left hand side and right hand side of Equation (8) are the same as $t \to \infty$. We could also view the right hand side of Equation (8) as the case where policies of $\pi_{-i}^\star$ are part of the environment.

Theorem 2 motivates the following definition of multi-agent IRL with regularizer ψ.

$$\text{MAIRL}_\psi(\boldsymbol{\pi}_E) = \arg\max_{\mathbf{r}} -\psi(\mathbf{r}) + \sum_{i=1}^{N} (\mathbb{E}_{\boldsymbol{\pi}_E}[r_i]) - \left(\max_{\boldsymbol{\pi}} \sum_{i=1}^{N} (\beta H_i(\pi_i) + \mathbb{E}_{\pi_i, \pi_{E_{-i}}}[r_i]) \right),$$

where $H_i(\pi_i) = \mathbb{E}_{\pi_i, \pi_{E_{-i}}}[-\log \pi_i(a_i|s)]$ is the discounted causal entropy for policy π_i when other agents follow $\pi_{E_{-i}}$, and β is a hyper-parameter controlling the strength of the entropy regularization term as in [16]. This formulation is a strict generalization to the single agent IRL in [16].

Corollary 2.1. *If $N = 1$, $\beta = 1$ then $\text{MAIRL}_\psi(\boldsymbol{\pi}_E) = \text{IRL}_\psi(\boldsymbol{\pi}_E)$.*

Furthermore, if the regularization ψ is additively separable, and for each agent i, π_{E_i} is the unique optimal response to other experts $\pi_{E_{-i}}$, we obtain the following:

Theorem 3. *Assume that $\psi(\mathbf{r}) = \sum_{i=1}^{N} \psi_i(r_i)$, ψ_i is convex for each $i \in [N]$, and that $\text{MARL}(\mathbf{r})$ has a unique solution[2] for all $r \in \text{MAIRL}_\psi(\boldsymbol{\pi}_E)$, then*

$$\text{MARL} \circ \text{MAIRL}_\psi(\boldsymbol{\pi}_E) = \arg\min_{\pi \in \Pi} \sum_{i=1}^{N} -\beta H_i(\pi_i) + \psi_i^\star(\rho_{\pi_{i,E_{-i}}} - \rho_{\pi_E}) \tag{10}$$

where $\pi_{i,E_{-i}}$ denotes π_i for agent i and $\pi_{E_{-i}}$ for other agents.

[2] The set of Nash equilibria is not always convex, so we have to assume $\text{MARL}(\mathbf{r})$ returns a unique solution.

The above theorem suggests that ψ-regularized multi-agent inverse reinforcement learning is seeking, for each agent i, a policy whose occupancy measure is close to one where we replace policy π_i with expert π_{E_i}, as measured by the convex function $\psi_i^\star$.

However, we do not assume access to the expert policy π_E during training, so it is not possible to obtain $\rho_{\pi_{i,E_{-i}}}$. Therefore, we consider an alternative approach where we match the occupancy measure between ρ_{π_E} and ρ_π. We can obtain our practical algorithm if we select an adversarial reward function regularizer and remove the effect from entropy regularizers.

Proposition 2. *If $\beta = 0$, and $\psi(\mathbf{r}) = \sum_{i=1}^N \psi_i(r_i)$ where $\psi_i(r_i) = \mathbb{E}_{\boldsymbol{\pi}_E}[g(r_i)]$ if $r_i > 0$; $+\infty$ otherwise, and*

$$g(x) = \begin{cases} -x - \log(1 - e^x) & \text{if} \quad r_i > 0 \\ +\infty & \text{otherwise} \end{cases}$$

then

$$\arg\min_{\pi} \sum_{i=1}^N \psi_i^\star(\rho_{\pi_i, \pi_{E_{-i}}} - \rho_{\boldsymbol{\pi}_E}) = \arg\min_{\pi} \sum_{i=1}^N \psi_i^\star(\rho_{\pi_i, \pi_{-i}} - \rho_{\boldsymbol{\pi}_E}) \tag{11}$$

and both are equal to π_E.

Theorem 3 and Proposition 2 discuss the differences from the single agent scenario. In Theorem 3 we make the assumption that MARL($\mathbf{r}$) has a unique solution, which is always true in the single agent case due to convexity of the space of the optimal policies. In Proposition 2 we remove the entropy regularizer because here the causal entropy for π_i may depend on the policies of the other agents. Specifically, the entropy for the left hand side of Equation (11) conditions on $\pi_{E_{-i}}$ and the entropy for the right hand side conditions on π_{-i} (both would disappear in the single-agent case).

4 Practical multi-agent imitation learning

Despite the recent successes in deep RL, it is notoriously hard to train policies with RL algorithms-because of high variance gradient estimates. This is further exacerbated in Markov games since an agent's optimal policy depends on other agents [14, 17]. In this section, we address these problems and propose practical algorithms for multi-agent imitation.

4.1 Multi-agent generative adversarial imitation learning

We select ψ_i to be our reward function regularizer in Proposition 2; this corresponds to the two-player game introduced in Generative Adversarial Imitation Learning (GAIL, [16]). For each agent i, we have a discriminator (denoted as D_{ω_i}) mapping state action-pairs to *scores* optimized to discriminate expert demonstrations from behaviors produced by π_i. Implicitly, D_{ω_i} plays the role of a reward function for the generator, which in turn attempts to train the agent to maximize its reward thus fooling the discriminator. We optimize the following objective:

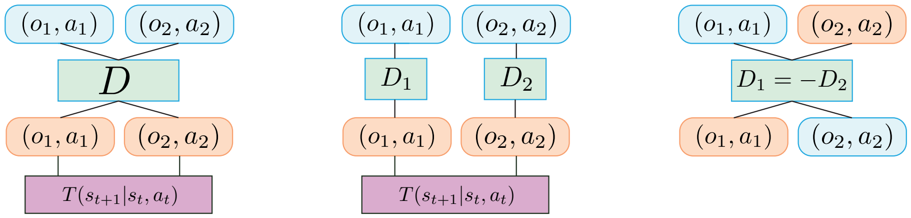
$$\min_\theta \max_\omega \mathbb{E}_{\boldsymbol{\pi}_\theta}\left[\sum_{i=1}^N \log D_{\omega_i}(s, a_i)\right] + \mathbb{E}_{\boldsymbol{\pi}_E}\left[\sum_{i=1}^N \log(1 - D_{\omega_i}(s, a_i))\right] \tag{12}$$

We update π_θ through reinforcement learning, where we also use a baseline V_ϕ to reduce variance. We outline the algorithm – Multi-Agent GAIL (MAGAIL) – in Appendix B.

We can augment the reward regularizer $\psi(\mathbf{r})$ using an indicator $y(r)$ denoting whether r fits our prior knowledge; the augmented reward regularizer $\hat{\psi} : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R} \cup \{\infty\}$ is then: $\psi(\mathbf{r})$ if $y(\mathbf{r}) = 1$, and ∞ if $y(\mathbf{r}) = 0$. We introduce three types of $y(\mathbf{r})$ for common settings.

Centralized The easiest case is to assume that the agents are fully cooperative, i.e. they share the same reward function. Here $y(\mathbf{r}) = \mathbb{I}(r_1 = r_2 = \ldots r_n)$ and $\psi(\mathbf{r}) = \psi_{\text{GA}}(\mathbf{r})$. One could argue this corresponds to the GAIL case, where the RL procedure operates on multiple agents (a joint policy).

Decentralized We make no prior assumptions over the correlation between the rewards. Here $y(\mathbf{r}) = \mathbb{I}(r_i \in \mathbb{R}^{\mathcal{O}_i \times \mathcal{A}_i})$ and $\psi_i(r_i) = \psi_{\text{GA}}(r_i)$. This corresponds to one discriminator for each agent which discriminates the trajectories as observed by agent i. However, these discriminators are not learned independently as they interact indirectly via the environment.



(a) Centralized (Cooperative) (b) Decentralized (Mixed) (c) Zero-sum (Competitive)

Figure 1: Different MAGAIL algorithms obtained with different priors on the reward structure. The discriminator tries to assign higher rewards to top row and low rewards to bottom row. In centralized and decentralized, the policy operates with the environment to match the expert rewards. In zero-sum, the policy do not interact with the environment; expert and policy trajectories are paired together as input to the discriminator.

Zero Sum Assume there are two agents that receive opposite rewards, so $r_1 = -r_2$. As such, ψ is no longer additively separable. Nevertheless, an adversarial training procedure can be designed using the following fact:

$$v(\pi_{E_1}, \pi_2) \geq v(\pi_{E_1}, \pi_{E_2}) \geq v(\pi_1, \pi_{E_2}) \tag{13}$$

where $v(\pi_1, \pi_2) = \mathbb{E}_{\pi_1, \pi_2}[r_1(s, a)]$ is the expected outcome for agent 1, and is modeled by the discriminator. The discriminator could then try to maximize v for trajectories from (π_{E_1}, π_2) and minimize v for trajectories from (π_2, π_{E_1}) according to Equation (13).

These three settings are in summarized in Figure 1.

4.2 Multi-agent actor-critic with Kronecker factors

To optimize over the generator parameters θ in Eq. (12) we wish to use an algorithm for multi-agent RL that has good sample efficiency in practice. Our algorithm, which we refer to as Multi-agent Actor-Critic with Kronecker-factors (MACK), is based on Actor-Critic with Kronecker-factored Trust Region (ACKTR, [25–27]), a state-of-the-art natural policy gradient [28, 29] method in deep RL. MACK uses the framework of centralized training with decentralized execution [17]; policies are trained with additional information to reduce variance but such information is not used during execution time. We let the advantage function of every agent agent be a function of all agents' observations and actions:

$$A_{\phi_i}^{\pi_i}(s, \mathbf{a}_t) = \sum_{j=0}^{k-1}(\gamma^j r_i(s_{t+j}, \mathbf{a}_{t+j}) + \gamma^k V_{\phi_i}^{\pi_i}(s_{t+k}, a_{-i,t+k})) - V_{\phi_i}^{\pi_i}(s_t, a_{-i,t}) \tag{14}$$

where $V_{\phi_i}^{\pi_i}(s_k, a_{-i})$ is the baseline for i, utilizing the additional information (a_{-i}) for variance reduction. We use (approximated) natural policy gradients to update both θ and ϕ but without trust regions to schedule the learning rate, using a linear decay learning rate schedule instead.

MACK has some notable differences from Multi-Agent Deep Deterministic Policy Gradient [14]. On the one hand, MACK does not assume knowledge of other agent's policies nor tries to infer them; the value estimator merely collects experience from other agents (and treats them as black boxes). On the other hand, MACK does not require gradient estimators such as Gumbel-softmax [30, 31] to optimize over discrete actions, which is necessary for DDPG [32].

5 Experiments

We evaluate the performance of (centralized, decentralized, and zero-sum versions) of MAGAIL under two types of environments. One is a particle environment which allows for complex interactions and behaviors; the other is a control task, where multiple agents try to cooperate and move a plank forward. We collect results by averaging over 5 random seeds. Our implementation is based on OpenAI baselines [33]; please refer to Appendix C for implementation details[3].

[3] Code for reproducing the experiments are in `https://github.com/ermongroup/multiagent-gail`.

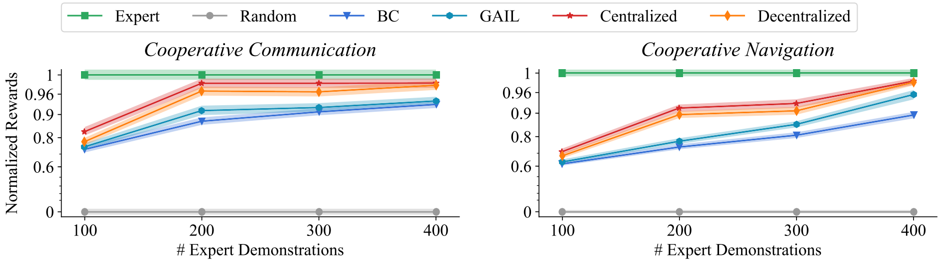


Figure 2: Average true reward from cooperative tasks. Performance of experts and random policies are normalized to one and zero respectively. We use inverse log scale for better comparison.

We compare our methods (centralized, decentralized, zero-sum MAGAIL) with two baselines. The first is behavior cloning (BC), which learns a maximum likelihood estimate for a_i given each state s and does not require actions from other agents. The second baseline is a GAIL IRL baseline that operates on each agent separately – for each agent we first pretrain the other agents with BC, and then train the agent with GAIL; we then gather the trained GAIL policies from all the agents and evaluate their performance.

5.1 Particle environments

We first consider the particle environment proposed in [14], which consists of several agents and landmarks. We consider two cooperative environments and two competitive ones. All environments have an underlying true reward function that allows us to evaluate the performance of learned agents.

The environments include: **Cooperative Communication** – two agents must cooperate to reach one of three colored landmarks. One agent ("speaker") knows the goal but cannot move, so it must convey the message to the other agent ("listener") that moves but does not observe the goal. **Cooperative Navigation** – three agents must cooperate through physical actions to reach three landmarks; ideally, each agent should cover a single landmark. **Keep-Away** – two agents have contradictory goals, where agent 1 tries to reach one of the two targeted landmarks, while agent 2 (the adversary) tries to keep agent 1 from reaching its target. The adversary does not observe the target, so it must act based on agent 1's actions. **Predator-Prey** – three slower cooperating adversaries must chase the faster agent in a randomly generated environment with obstacles; the adversaries are rewarded by touching the agent while the agent is penalized.

For the cooperative tasks, we use an analytic expression defining the expert policy; for the competitive tasks, we use MACK to train expert policies based on the true underlying rewards (using larger policy and value networks than the ones that we use for imitation). We then use the expert policies to simulate trajectories $\mathcal{D}$, and then do imitation learning on $\mathcal{D}$ as demonstrations, where we assume the underlying rewards are unknown. Following [34], we pretrain our Multi-Agent GAIL methods and the GAIL baseline using behavior cloning as initialization to reduce sample complexity for exploration. We consider 100 to 400 episodes of expert demonstrations, each with 50 timesteps, which is close to the number of timesteps used for the control tasks in [16]. Moreover, we randomly sample the starting position of agents and landmarks each episode, so our policies have to learn to generalize when they encounter new settings.

5.1.1 Cooperative tasks

We evaluate performance in cooperative tasks via the average expected reward obtained by all the agents in an episode. In this environment, the starting state is randomly initialized, so generalization is crucial. We do not consider the zero-sum case, since it violates the cooperative nature of the task. We display the performance of centralized, decentralized, GAIL and BC in Figure 2.

Naturally, the performance of BC and MAGAIL increases with more expert demonstrations. MAGAIL performs consistently better than BC in all the settings; interestingly, in the cooperative communication task, centralized MAGAIL is able to achieve expert-level performance with only 200 demonstrations, but BC fails to come close even with 400 demonstrations. Moreover, the cen-

Table 1: Average agent rewards in competitive tasks. We compare behavior cloning (BC), GAIL (G), Centralized (C), Decentralized (D), and Zero-Sum (ZS) methods. Best marked in bold (high vs. low rewards is preferable depending on the agent vs. adversary role).

Task	Predator-Prey								
Agent	Behavior Cloning					G	C	D	ZS
Adversary	BC	G	C	D	ZS	Behavior Cloning			
Rewards	-93.20	-93.71	-93.75	-95.22	**-95.48**	-90.55	-91.36	**-85.00**	-89.4
Task	Keep-Away								
Agent	Behavior Cloning					G	C	D	ZS
Adversary	BC	G	C	D	ZS	Behavior Cloning			
Rewards	24.22	24.04	23.28	23.56	**23.19**	26.22	26.61	**28.73**	27.80

tralized MAGAIL performs slightly better than decentralized MAGAIL due to the better prior, but decentralized MAGAIL still learns a highly correlated reward between two agents.

5.1.2 Competitive tasks

We consider all three types of Multi-Agent GAIL (centralized, decentralized, zero-sum) and BC in both competitive tasks. Since there are two opposing sides, it is hard to measure performance directly. Therefore, we compare by letting (agents trained by) BC play against (adversaries trained by) other methods, and vice versa. From Table 1, decentralized and zero-sum MAGAIL often perform better than centralized MAGAIL and BC, which suggests that the selection of the suitable prior $\hat{\psi}$ is important for good empirical performance.

5.2 Cooperative control

In some cases we are presented with sub-optimal expert demonstrations because the environment has changed; we consider this case in a cooperative control task [35], where N bipedal walkers cooperate to move a long plank forward; the agents have incentive to collaborate since the plank is much longer than any of the agents. The expert demonstrates its policy on an environment with no bumps on the ground and heavy weights, while we perform imitation in an new environment with bumps and lighter weights (so one is likely to use too much force). Agents trained with BC tend to act more aggressively and fail, whereas agents trained with centralized MAGAIL can adapt to the new environment. With 10 (imperfect) expert demonstrations, BC agents have a chance of failure of 39.8$\%$ (with a reward of 1.26), while centralized MAGAIL agents fail only 26.2$\%$ of the time (with a reward of 26.57). We show videos of respective policies in this link.

6 Discussion

There is a vast literature on single-agent imitation learning [36]. Behavior Cloning (BC) learns the policy through supervised learning [37]. Inverse Reinforcement Learning (IRL) assumes the expert policy optimizes over some unknown reward, recovers the reward, and learns the policy through reinforcement learning (RL). BC does not require knowledge of transition probabilities or access to the environment, but suffers from compounding errors and covariate shift [38, 23].

Most existing work in multi-agent imitation learning assumes the agents have very specific reward structures. The most common case is fully cooperative agents [39], where the challenges mainly lie in other factors, such as unknown role assignments [40], scalability to swarm systems [41] and agents with partial observations [42]. In non-cooperative settings, [43] considers the case of IRL for two-player zero-sum games and cast the IRL problem as Bayesian inference, while [44] assumes agents are non-cooperative but the reward function is a linear combination of pre-specified features.

Our work is the first to propose a general multi-agent IRL framework that combines state-of-the art multi-agent reinforcement learning methods [14, 17] and implicit generative models such as generative adversarial networks [45]. Experimental results demonstrate that it is able to imitate complex behaviors in high-dimensional environments with both cooperative and adversarial interactions. An interesting future direction is to explore new paradigms for learning from experts, such as allowing the expert to participate in the agent's learning process [24].

Acknowledgements

This work was supported by Toyota Research Institute and Future of Life Institute. The authors would like to thank Lantao Yu for discussions over implementation, and Andy Shih for identifying several typos.

References

[1] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *arXiv preprint arXiv:1802.01561*, 2018.

[2] D. Hadfield-Menell, S. Milli, P. Abbeel, S. J. Russell, and A. Dragan, "Inverse reward design," in *Advances in Neural Information Processing Systems*, pp. 6768–6777, 2017.

[3] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," *arXiv preprint arXiv:1606.06565*, 2016.

[4] D. Amodei and J. Clark, "Faulty reward functions in the wild," 2016.

[5] P. Peng, Q. Yuan, Y. Wen, Y. Yang, Z. Tang, H. Long, and J. Wang, "Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games," *arXiv preprint arXiv:1703.10069*, 2017.

[6] L. Matignon, L. Jeanpierre, A.-I. Mouaddib, *et al.*, "Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes.," in *AAAI*, 2012.

[7] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel, "Multi-agent reinforcement learning in sequential social dilemmas," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 464–473, International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[8] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *AAAI*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.

[9] P. Englert and M. Toussaint, "Inverse kkt–learning cost functions of manipulation tasks from demonstrations," in *Proceedings of the International Symposium of Robotics Research*, 2015.

[10] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, pp. 49–58, 2016.

[11] B. Stadie, P. Abbeel, and I. Sutskever, "Third person imitation learning," in *ICLR*, 2017.

[12] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *Icml*, pp. 663–670, 2000.

[13] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 1, ACM, 2004.

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

[15] J. Hu, M. P. Wellman, *et al.*, "Multiagent reinforcement learning: theoretical framework and an algorithm.," in *ICML*, vol. 98, pp. 242–250, Citeseer, 1998.

[16] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

[17] J. Foerster, Y. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems*, pp. 2137–2145, 2016.

[18] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proceedings of the eleventh international conference on machine learning*, vol. 157, pp. 157–163, 1994.

[19] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, vol. 1. MIT press Cambridge, 1998.

[20] M. Bloem and N. Bambos, "Infinite time horizon maximum causal entropy inverse reinforcement learning," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pp. 4911–4916, IEEE, 2014.

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

[22] H. Prasad and S. Bhatnagar, "A study of gradient descent schemes for general-sum stochastic games," *arXiv preprint arXiv:1507.00093*, 2015.

[23] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning.," in *AISTATS*, p. 6, 2011.

[24] D. Hadfield-Menell, S. J. Russell, P. Abbeel, and A. Dragan, "Cooperative inverse reinforcement learning," in *Advances in neural information processing systems*, pp. 3909–3917, 2016.

[25] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba, "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," in *Advances in neural information processing systems*, pp. 5285–5294, 2017.

[26] Y. Song, J. Song, and S. Ermon, "Accelerating natural gradient with higher-order invariance," in *International Conference on Machine Learning (ICML)*, 2018.

[27] Y. Song, R. Shu, N. Kushman, and S. Ermon, "Constructing unrestricted adversarial examples with generative models," *arXiv preprint arXiv:1805.07894*, 2018.

[28] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[29] S. M. Kakade, "A natural policy gradient," in *Advances in neural information processing systems*, pp. 1531–1538, 2002.

[30] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

[31] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712*, 2016.

[32] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[33] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Openai baselines." `https://github.com/openai/baselines`, 2017.

[34] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," *arXiv preprint arXiv:1703.08840*, 2017.

[35] J. K. Gupta and M. Egorov, "Multi-agent deep reinforcement learning environment." `https://github.com/sisl/madrl`, 2017.

[36] J. A. Bagnell, "An invitation to imitation," tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2015.

[37] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[38] S. Ross and D. Bagnell, "Efficient reductions for imitation learning.," in *AISTATS*, pp. 3–5, 2010.

[39] S. Barrett, A. Rosenfeld, S. Kraus, and P. Stone, "Making friends on the fly: Cooperating with new teammates," *Artificial Intelligence*, vol. 242, pp. 132–171, 2017.

[40] H. M. Le, Y. Yue, and P. Carr, “Coordinated multi-agent imitation learning,” *arXiv preprint arXiv:1703.03121*, 2017.

[41] A. Šošic, W. R. KhudaBukhsh, A. M. Zoubir, and H. Koeppl, “Inverse reinforcement learning in swarm systems,” *stat*, vol. 1050, p. 17, 2016.

[42] K. Bogert and P. Doshi, “Multi-robot inverse reinforcement learning under occlusion with interactions,” in *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 173–180, International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[43] X. Lin, P. A. Beling, and R. Cogill, “Multi-agent inverse reinforcement learning for zero-sum games,” *arXiv preprint arXiv:1403.6508*, 2014.

[44] T. S. Reddy, V. Gopikrishna, G. Zaruba, and M. Huber, “Inverse reinforcement learning for decentralized non-cooperative multiagent systems,” in *Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on*, pp. 1930–1935, IEEE, 2012.

[45] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[46] J. Martens and R. Grosse, “Optimizing neural networks with kronecker-factored approximate curvature,” in *International Conference on Machine Learning*, pp. 2408–2417, 2015.

[47] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” *arXiv preprint arXiv:1506.02438*, 2015.

A Proofs

We use $\hat{v}_i(s)$, $\hat{q}_i(s, a_i)$ and $Q(\tau)$ to represent $\hat{v}_i(s; \boldsymbol{\pi}, \mathbf{r})$, $\hat{q}_i(s, a_i; \boldsymbol{\pi}, \mathbf{r})$ and $Q(\tau; \boldsymbol{\pi}, \mathbf{r})$, where we implicitly assume dependency over $\boldsymbol{\pi}$ and $\mathbf{r}$.

A.1 Proof of Lemma 1

For any policy $\boldsymbol{\pi}$, $f_{\mathbf{r}}(\boldsymbol{\pi}, \hat{v}) = 0$ when $\hat{v}$ is the value function of $\boldsymbol{\pi}$ (due to Bellman equations). However, only policies that form a Nash equilibrium satisfies the constraints in Eq. 2; we formalize this in the following Lemma.

Lemma 1. *Let $\hat{v}_i(s; \boldsymbol{\pi}, \mathbf{r})$ be the solution to the Bellman equation*

$$\hat{v}_i(s) = \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}}[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a}) \hat{v}_i(s')]$$

and $\hat{q}_i(s, a_i) = \mathbb{E}_{a_{-i} \sim \pi_{-i}}[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a}) \hat{v}_i(s')]$. Then for any π,

$$f_{\mathbf{r}}(\boldsymbol{\pi}, \hat{v}(\boldsymbol{\pi})) = 0$$

Furthermore, $\boldsymbol{\pi}$ is Nash equilibrium under r if and only if $\hat{v}_i(s) \geq \hat{q}_i(s, a_i)$ for all $i \in [N], s \in \mathcal{S}, a_i \in \mathcal{A}_i$.

Proof. By definition of $\hat{v}_i(s)$ we have:

$$\begin{aligned}
\hat{v}_i(s) &= \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}}[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a}) \hat{v}_i(s')] \\
&= \mathbb{E}_{a_i \sim \pi_i} \mathbb{E}_{a_{-i} \sim \pi_{-i}}[r_i(s, \mathbf{a}) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \mathbf{a}) \hat{v}_i(s')] \\
&= \mathbb{E}_{a_{-i} \sim \pi_i}[\hat{q}_i(s, a_i)]
\end{aligned}$$

which uses the fact that a_i and a_{-i} are independent conditioned on s. Hence $f_{\mathbf{r}}(\boldsymbol{\pi}, \hat{v}) = 0$ immediately follows.

If $\boldsymbol{\pi}$ is a Nash equilibrium, and at least one of the constraints does not hold, i.e. there exists some i and s, a_i such that $\hat{v}(s) < \hat{q}(s, a_i)$, then agent i can achieve a strictly higher expected return if it chooses to take actions a_i whenever it encounters state s_i and follow π_i for the rest of the states, which violates the Nash equilibrium assumption.

If the constraints hold, i.e. for all i and (s, a_i), $\hat{v}_i(s) \geq \hat{q}_i(s, a_i)$ then

$$\hat{v}_i(s) \geq \mathbb{E}_{\pi_i}[\hat{q}_i(s, a_i)] = \hat{v}_i(s)$$

so value iteration over $\hat{v}_i(s)$ converges. If we can find another policy $\boldsymbol{\pi}'$ such that $\hat{v}_i(s) < \mathbb{E}_{\pi'_i}[\hat{q}_i(s, a_i)]$, then there should be at least one violation in the constraints since π'_i must be a convex combination (expectation) over actions a_i. Therefore, for any policy π'_i and action a_i for any agent i, $\mathbb{E}_{\pi_i}[\hat{q}_i(s, a_i)] \geq \mathbb{E}_{\pi'_i}[\hat{q}_i(s, a_i)]$ always hold, so π_i is the optimal response to π_{-i}, and π constitutes a Nash equilibrium when we repeat this argument for all agents.

Notably, Theorem 3.8.2 in [21] discusses the equivalence by assuming $f_{\mathbf{r}}(\boldsymbol{\pi}, v) = 0$ for some v; if v satisfies the assumptions, then $v = \hat{v}'$. □

A.2 Proof of Theorem 1

Proof. If $\boldsymbol{\pi}$ is a Nash equilibrium, and at least one of the constraints does not hold, i.e. there exists some i and $\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}$, such that

$$\hat{v}_i(s^{(0)}) < \mathbb{E}_{\pi_{-i}}[\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)})]$$

Then agent i can achieve a strictly higher expected return on its own if it chooses a particular sequence of actions by taking $a_i^{(j)}$ whenever it encounters state $s^{(j)}$, and follow π_i for the remaining states. We note that this is in expectation over the policy of other agents. Hence, we construct a policy for agent

i that has strictly higher value than π_i without modifying π_{-i}, which contradicts the definition of Nash equilibrium.

If the constraints hold, i.e for all i and $\{s^{(j)}, a_i^{(j)}\}_{j=0}^{t}$,

$$\hat{v}_i(s^{(0)}) \geq \mathbb{E}_{\pi_{-i}}[\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)})]$$

then we can construct any $\hat{q}_i(s^{(0)}, a_i^{(0)})$ via a convex combination by taking the expectation over π_i:

$$\hat{q}_i(s^{(0)}, a_i^{(0)}) = \mathbb{E}_{\pi_i}[\mathbb{E}_{\pi_{-i}}[\hat{q}_i^{(t)}(\{s^{(j)}, \mathbf{a}^{(j)}\}_{j=0}^{t-1}, s^{(t)}, a_i^{(t)})]]$$

where the expectation over π_i is taken over actions $\{a_i^{(j)}\}_{j=0}^{t}$ (the expectation over states are contained in the inner expectation over π_{-i}). Therefore, $\forall i \in [N], s \in \mathcal{S}, a_i \in \mathcal{A}_i$,

$$\hat{v}_i(s) \geq \hat{q}_i(s, a_i)$$

and we recover the constraints in Eq. 2. By Lemma 1, π is a Nash equilibrium. □

A.3 Proof of Theorem 2

Proof. We use $Q^\star$, $\hat{q}^\star$, $\hat{v}^\star$ to denote the Q, $\hat{q}$ and $\hat{v}$ quantities defined for policy $\boldsymbol{\pi}^\star$. For the two terms in $L_{\mathbf{r}}^{(t+1)}(\boldsymbol{\pi}^\star, \lambda_{\boldsymbol{\pi}}^\star)$ we have:

$$L_{\mathbf{r}}^{(t+1)}(\boldsymbol{\pi}^\star, \lambda_{\boldsymbol{\pi}}^\star) = \sum_{i=1}^{N} \sum_{\tau_i \in \mathcal{T}_i} \lambda^\star(\tau_i)(Q_i^\star(\tau_i) - \hat{v}_i^\star(s^{(0)})) \tag{15}$$

For any agent i, we note that

$$\sum_{\tau_i \in \mathcal{T}_i} \lambda^\star(\tau_i) Q_i^\star(\tau_i) = \mathbb{E}_{\pi_i} \mathbb{E}_{\pi_{-i}^\star}[\sum_{j=0}^{t-1} \gamma^j r_i(s^{(j)}, a^{(j)}) + \gamma^t \hat{q}_i^\star(s^t, a_i^{(t)})]$$

which amounts to using π_i for agent i for the first t steps and using $\pi_i^\star$ for the remaining steps, whereas other agents follow $\pi_{-i}^\star$. As $t \to \infty$, this converges to $\mathbb{E}_{\pi_i, \pi_{-i}^\star}[r_i]$ since $\gamma^t \to 0$ and $q_i^\star(s^{(t)}, a_i^{(t)})$ is bounded. Moreover, for $\hat{v}_i^\star(s^{(0)})$, we have

$$\sum_{\tau_i \in \mathcal{T}_i} \lambda^\star(\tau_i) \hat{v}_i^\star(s^{(0)}) = \mathbb{E}_{s^{(0)} \sim \eta}[\hat{v}_i^\star(s^{(0)})] = \mathbb{E}_{\boldsymbol{\pi}^\star}[r_i]$$

Combining the two we have

$$L_{\mathbf{r}}^{(t+1)}(\boldsymbol{\pi}^\star, \lambda_{\boldsymbol{\pi}}^\star) = \sum_{i=1}^{N} \mathbb{E}_{\pi_i, \pi_{-i}^\star}[r_i] - \sum_{i=1}^{N} \mathbb{E}_{\boldsymbol{\pi}^\star}[r_i]$$

which describes the differences in expected rewards. □

A.4 Proof of Theorem 3

Proof. Define the "MARL" objective for a single agent i where other agents have policy π_{E_i}:

$$\text{MARL}_i(r_i) = \max_{\pi_i} H_i(\pi_i) + \mathbb{E}_{\pi_i, \pi_{E_{-i}}}[r_i]$$

Define the "MAIRL" objective for a single agent i where other agents have policy $\boldsymbol{\pi}_E$:

$$\text{MAIRL}_{i,\psi}(\boldsymbol{\pi}^\star) = \arg\max_{r_i} \psi_i(r_i) + \mathbb{E}_{\pi_E}[r_i] - (\max_{\pi_i} H_i(\pi_i) + \mathbb{E}_{\pi_i, \pi_{E_{-i}}}[r_i])$$

Since r_i and π_i's are independent in the MAIRL objective, the solution to MAIRL_ψ can be represented by the solutions of $\text{MAIRL}_{i,\psi}$ for each i:

$$\text{MAIRL}_\psi = [\text{MAIRL}_{1,\psi}, \ldots, \text{MAIRL}_{N,\psi}]$$

Moreover, the single agent "MARL" objective $\text{MARL}_i(r_i)$ has a unique solution π_{E_i}, which also composes the (unique) solution to MARL (which we assumed in Section 3. Therefore,

$$\text{MARL}(\mathbf{r}) = [\text{MARL}_1(r_1), \ldots, \text{MARL}_N(r_N)]$$

So we can use Proposition 3.1 in [16] for each agent i with $\text{MARL}_i(r_i)$ and $\text{MAIRL}_{i,\psi}(\boldsymbol{\pi}^\star)$ and achieve the same solution as $\text{MARL} \circ \text{MAIRL}_\psi$. □

A.5 Proof of Proposition 2

Proof. From Corollary A.1.1 in [16], we have

$$\psi^\star_{GA}(\rho_{\boldsymbol{\pi}} - \rho_{\boldsymbol{\pi}_E}) = \max_{D \in (0,1)^{\mathcal{S}\times\mathcal{A}}} \mathbb{E}_{\boldsymbol{\pi}}[\log D(s,a)] + \mathbb{E}_{\boldsymbol{\pi}_E}[\log(1 - D(s,a))] \equiv D_{JS}(\rho_{\boldsymbol{\pi}}, \rho_{\boldsymbol{\pi}_E})$$

where D_{JS} denotes Jensen-Shannon divergence (which is a squared metric), and $\equiv$ denotes equivalence up to shift and scaling.

Taking the min over this we obtain

$$\arg\min_{\pi} \sum_{i=1}^{N} \psi^\star_{GA}(\rho_{\boldsymbol{\pi}} - \rho_{\boldsymbol{\pi}_E}) = \boldsymbol{\pi}_E$$

Similarly,

$$\arg\min_{\pi} \sum_{i=1}^{N} \psi^\star_{GA}(\rho_{\pi_i, \pi_{E_{-i}}} - \rho_{\boldsymbol{\pi}_E}) = \boldsymbol{\pi}_E$$

So these two quantities are equal. □

B MAGAIL Algorithm

We include the MAGAIL algorithm as follows:

Algorithm 1 Multi-Agent GAIL (MAGAIL)

```
Input: Initial parameters of policies, discriminators and value (baseline) estimators $\theta_0, \omega_0$,
$\phi_0$; expert trajectories $\mathcal{D} = \{(s_j, a_j)\}_{j=0}^{M}$; batch size $B$; Markov game as a black box
$(N, \mathcal{S}, \mathcal{A}, \eta, T, r, \gamma)$; initial policy $\boldsymbol{\pi}$.
Output: Learned policies $\pi_\theta$ and reward functions $D_\omega$.

for $u = 0, 1, 2, \ldots$ do
  Obtain trajectories of size $B$ from $\boldsymbol{\pi}$ by the process
      $s_0 \sim \eta(s), \mathbf{a}_t \sim \pi_{\theta_u}(\mathbf{a}_t|s_t), s_{t+1} \sim P(\cdot, |s_t, \mathbf{a}_t)$
  Sample state-action pairs from $\mathcal{D}$ with batch size $B$.
  Denote state-action pairs from $\pi$ and $\mathcal{D}$ as $\chi$ and $\chi_E$.
  for $i = 1, \ldots, n$ do
    Update $\omega_i$ to increase the objective
        $\mathbb{E}_\chi[\log D_{\omega_i}(s, a_i)] + \mathbb{E}_{\chi_E}[\log(1 - D_{\omega_i}(s, a_i))]$
  end for
  for $i = 1, \ldots, n$ do
    Compute value estimate $V^\star$ and advantage estimate $A_i$ for $(s, \mathbf{a}) \in \chi$.
    Update $\phi_i$ to decrease the objective
        $\mathbb{E}_\chi[(V_\phi(s, a_{-i}) - V^\star(s, a_{-i}))^2]$
    Update $\theta_i$ by policy gradient with small step sizes:
        $\mathbb{E}_\chi[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|s_t) A_i(s, \mathbf{a})]$
  end for
end for
```

Table 2: Performance in cooperative navigation.

# Expert Episodes	100	200	300	400
Expert	-13.50 ± 6.3			
Random	-128.13 ± 32.1			
Behavior Cloning	-56.82 ± 18.9	-43.10 ± 16.0	-35.66 ± 15.2	-25.83 ± 12.7
Centralized	**-46.66** ± 20.8	**-23.10** ± 12.4	**-21.53** ± 12.9	**-15.30** ± 7.0
Decentralized	-50.00 ± 18.6	-25.61 ± 12.3	-24.10 ± 13.3	-15.55 ± 6.5
GAIL	-55.01 ± 17.7	-39.21 ± 16.5	-29.89 ± 13.5	-18.76 ± 12.1

Table 3: Performance in cooperative communication.

# Expert Episodes	100	200	300	400
Expert	-6.22 ± 4.5			
Random	-62.49 ± 28.7			
Behavior Cloning	-21.25 ± 10.6	-13.25 ± 7.4	-11.37 ± 5.9	-10.00 ± 5.36
Centralized	**-15.65** ± 10.0	**-7.11** ± 4.8	**-7.11** ± 4.8	**-7.09** ± 4.8
Decentralized	-18.68 ± 10.4	-8.06 ± 5.3	-8.16 ± 5.5	-7.34 ± 4.9
GAIL	-20.28 ± 10.1	-11.06 ± 7.8	-10.51 ± 6.6	-9.44 ± 5.7

C Experiment Details

C.1 Hyperparameters

For the particle environment, we use two layer MLPs with 128 cells in each layer, for the policy generator network, value network and the discriminator. We use a batch size of 1000. The policy is trained using K-FAC optimizer [46] with learning rate of 0.1. All other parameters for K-FAC optimizer are the same in [25].

For the cooperative control task, we use two layer MLPs with 64 cells in each layer for all the networks. We use a batch size of 2048, and learning rate of 0.03. We obtain expert trajectories by training the expert with MACK and sampling demonstrations from the same environment. Hence, the expert's demonstrations are imperfect (or even flawed) in the environment that we test on.

The particle environments are set up exactly as in OpenAI MultiAgent Particle Environment, except for two minor differences. One, we set the environment to have maximum episode length of 50. Two, we end the episode once the agents have reached their targets in the cooperative environments.

C.2 Detailed Results

We use the particle environment introduced in [14] and the multi-agent control environment [35] for experiments. We list the exact performance in Tables 2, 3 for cooperative tasks, and Table 4 and competitive tasks. The means and standard deviations are computed over 100 episodes. The policies in the cooperative tasks are trained with varying number of expert demonstrations. The policies in the competitive tasks are trained with on a dataset with 100 expert trajectories.

The environment for each episode is drastically different (e.g. location of landmarks are randomly sampled), which leads to the seemingly high standrad deviation across episodes.

C.3 Video Demonstrations

We show certain trajectories generated by our methods. The vidoes are here: videos[4].

For the particle case:

Navigation-BC-Agents.gif Agents trained by behavior cloning in the navigation task.

Navigation-GAIL-Agents.gif Agents trained by proposed framework in the navigation task.

Predator-Prey-BC-Agent-BC-Adversary.gif Agent (green) trained by behavior cloning play against adversaries (red) trained by behavior cloning.

[4] https://drive.google.com/open?id=1Oz4ezMaKiIsPUKtCEOb6YoHJ9jLk6zbj

Table 4: Performance in competitive tasks.

Task	Agent Policy	Adversary Policy	Agent Reward
Predator-Prey	Behavior Cloning	Behavior Cloning	-93.20 ± 63.7
		GAIL	-93.71 ± 64.2
		Centralized	-93.75 ± 61.9
		Decentralized	-95.22 ± 49.7
		Zero-Sum	**-95.48** ± 50.4
	GAIL	Behavior Cloning	-90.55 ± 63.7
	Centralized		-91.36 ± 58.7
	Decentralized		**-85.00** ± 42.3
	Zero-Sum		-89.4 ± 48.2
Keep-Away	Behavior Cloning	Behavior Cloning	24.22 ± 21.1
		GAIL	24.04 ± 18.2
		Centralized	23.28 ± 20.6
		Decentralized	23.56 ± 19.9
		Zero-Sum	**23.19** ± 19.9
	GAIL	Behavior Cloning	26.22 ± 19.1
	Centralized		26.61 ± 20.0
	Decentralized		**28.73** ± 18.3
	Zero-Sum		27.80 ± 19.2

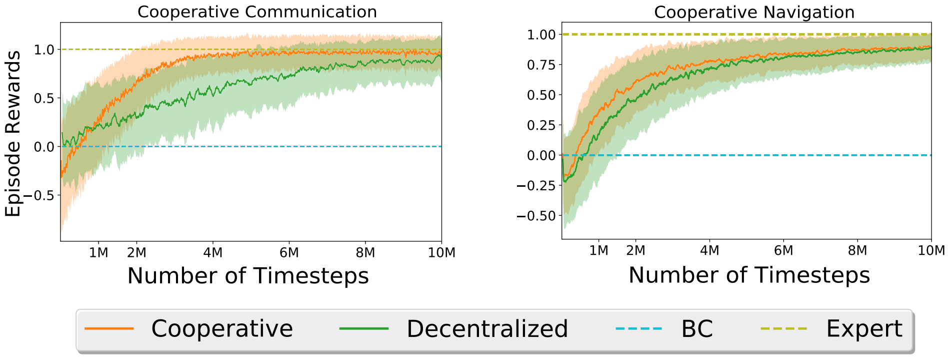


Figure 3: Sample complexity of multi-agent GAIL methods under cooperative tasks. Performance of experts is normalized to one, and performance of behavior cloning is normalized to zero. The standard deviation is computed with respect to episodes, and is noisy due to randomness in the environment.

Predator-Prey-GAIL-Agent-BC-Adversary.gif Agent (green) trained by proposed framework play against adversaries (red) trained by behavior cloning.

For the cooperative control case:

Multi-Walker-Expert.mp4 Expert demonstrations in the “easy” environment.

Multi-Walker-GAIL.mp4 Centralized GAIL trained on the “hard” environment.

Multi-Walker-BC.mp4 BC trained on the “hard” environment.

Interestingly, the failure modes for the agents in “hard” environment is mostly having the plank fall off or bounce off, since by decreasing the weight of the plank will decrease its friction force and increase its acceleration.

C.4 Potential Alternatives for RL Algorithms

There have been a range of deep reinforcement learning algorithms proposed recently; in this paper that all our MAGAIL / GAIL [16] algorithms in our experiments are using MACK / ACKTR [25] as the underlying RL algorithm, so the performance gain for MAGAIL over GAIL is caused by the multi-agent formulation, instead of the specific RL algorithm used.

In fact, our MAGAIL formulation (similar to GAIL) does not restrict the choice of RL algorithm. We choose to use ACKTR as the RL algorithm because its scalability (as opposed to TRPO [47], which requires some form of inverse Fisher) and its ability to deal with discrete action spaces directly (as opposed to DDPG [14], which requires continuous relaxation of the discrete actions).