Competitive Multi-agent Inverse Reinforcement Learning with Sub-optimal Demonstrations

Xingyu Wang [1] **Diego Klabjan** [1]

Abstract

This paper considers the problem of inverse reinforcement learning in zero-sum stochastic games when expert demonstrations are known to be sub-optimal. Compared to previous works that decouple agents in the game by assuming optimality in expert policies, we introduce a new objective function that directly pits experts against Nash Equilibrium policies, and we design an algorithm to solve for the reward function in the context of inverse reinforcement learning with deep neural networks as model approximations. To find Nash Equilibrium in large-scale games, we also propose an adversarial training algorithm for zero-sum stochastic games, and show the theoretical appeal of non-existence of local optima in its objective function. In numerical experiments, we demonstrate that our Nash Equilibrium and inverse reinforcement learning algorithms address games that are not amenable to existing benchmark algorithms. Moreover, our algorithm successfully recovers reward and policy functions regardless of the quality of the sub-optimal expert demonstration set.

1. Introduction

In the field of reinforcement learning, various algorithms have been proposed that guide an agent to make decisions, interact with the environment, and maximize the profit or return. As of late, multi-agent reinforcement learning, a generalization of single-agent reinforcement learning tasks, has been gaining momentum since it is aligned with the growing attention on multi-agent systems and the applications thereof. Either for solving a single-agent task or a multi-agent system, reinforcement learning entails the knowledge of the reward function, or at least observations of immediate reward. Some learning tasks, however, provide little or no knowledge of the reward function, but we do have access to demonstrations performed by "experts" in the task. For example, successful training of an auto-pilot system often takes full advantage of human driving behavior datasets. On the contrary, a naive, manually crafted artificial reward function may fail to capture the sophisticated balance between safety, speed, and simplicity in maneuvering during driving.

As one of the widely used approaches to these tasks, imitation learning concentrates on recovering policies from available demonstrations, which is justifiable as long as experts are known to be optimal (Ross et al., 2011). Another approach, termed as inverse reinforcement learning (IRL), formulates the task as an inverse problem of inferring the reward function from demonstrations of experts (Abbeel & Ng, 2004; Ziebart et al., 2008; Finn et al., 2016; Ho & Ermon, 2016; Choi et al., 2016). In general, the former approach is perceived as simpler, while the latter compresses task-related information into a succinct reward function that is more transferable to further applications.

To the best of our knowledge, existing IRL works concerning multi-agent competitive games make the same assumption that expert demonstrations are optimal. (In a multi-agent system, optimality is usually defined in the sense of Nash Equilibria.) IRL can thus be decoupled into two sub-problems in two-agent games. Reddy et al. (2012) base the algorithm on the optimality assumption on experts' policies, and decouple the IRL task in general-sum games when updating the reward function. By applying the Bayesian IRL framework (Ramachandran & Amir, 2007) to multi-agent zero-sum stochastic games, Lin et al. (2017) find reward functions that maximize the probability of the Nash Equilibrium policies with prescribed prior distribution of rewards and optimality constraints for both agent's demonstrations. Yet the more complicated the game is, the farther this optimality assumption may diverge from the truth. For instance, hardly would anyone believe that players in the Go game, as well as team games including most sport and video games, manage to find and employ a Nash Equilibrium policy.

To take sub-optimality in experts' demonstrations into account, we propose a completely different approach to per-

[1]Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL. Correspondence to: Diego Klabjan <d-klabjan@northwestern.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

form multi-agent IRL in zero-sum discounted stochastic games. We assume experts should be performing decently well but not necessarily optimally. Therefore, the margin between experts' performances and those of Nash Equilibria should be minimized by the reward function we yield, even though the performance gap is most likely above zero. Specifically, we take a game-theoretical perspective on competitive Markov Decision Processes (MDPs). The proposed IRL algorithm alternates between two major steps. In the policy step, we find a Nash Equilibrium policy for the game given the current reward function, while in the reward step we update the reward function so that the performance gap between expert demonstrations and Nash Equilibrium is minimized. Our framework significantly departs from previous works since the model and algorithm do not decouple the agents.

Naturally, our IRL algorithm entails an efficient subroutine of finding Nash Equilibrium in large-scale games before we could compare them against expert demonstrations. We use deep neural nets as model approximations for policies and value functions. On solving for Nash Equilibrium in a stochastic game, the past decade has seen novel algorithms (Akchurina, 2009; H.L. et al., 2015; H.L. & Bhatnagar, 2015). Unfortunately, most of these methods do not suit our case. For instance, the optimization-based ones rely on enumeration of state-action pairs of the game, which is infeasible for large games, and the algorithm for general-sum games (H.L. et al., 2015) provides a zero gradient for policy models in zero-sum games. Therefore, we propose an adversarial training algorithm for Nash Equilibrium in zero-sum discounted stochastic games. The algorithm maintains a best possible opponent model for the agent, and improves the agent's performance against its best opponent. We show the theoretical appeal of guarantee on global convergence to an optimal solution under this adversarial training formulation, as well as its superior performance in a large-scale game when using deep neural networks as model approximations for policies and an actor-critic style Proximal Policy Optimization algorithm (Schulman et al., 2017) as the policy gradient method.

Our major contributions are as follows. First, to the best of our knowledge, our IRL algorithm is drastically different from all previous competitive multi-agent IRL algorithms, as it is able to yield the reward function in zero-sum discounted stochastic games after abandoning the optimality assumption of expert demonstrations and it does not decouple the agents. Second, we propose an adversarial training algorithm to find a Nash Equilibrium policy in zero-sum stochastic games, and show the non-existence of local maxima in its objective function. Lastly, by utilizing deep neural networks and stating the algorithm accordingly, our multi-agent IRL approach addresses larger-scale games or tasks that are not amenable to previous methods, which rely on tabular representation and linear or quadratic programming. Our numerical experiments show that, compared with existing competitive multi-agent IRL methodologies, our approach manages to solve a large-scale problem and recover both the reward and policy functions robustly regardless of variation in the quality of expert demonstrations.

2. Competitive Multi-agent Inverse Reinforcement Learning with Sub-Optimal Demonstrations

2.1. Inverse Reinforcement Learning Problem for Zero-Sum Discounted Stochastic Games

From the field of competitive MDPs, we adopt the zero-sum discounted stochastic game setting as the underlying framework for the current study. Formally, a zero-sum stochastic game is defined by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}^f, \mathcal{A}^g, R, P, \gamma \rangle$, where the quantities are defined as follows. **Agents:** $\mathcal{N} = \{f, g\}$ denotes the two agents (players) in this game. **States:** $\mathcal{S}$ is a finite set of $N_{\mathcal{S}} = |\mathcal{S}|$ distinct states (each being a real vector) that can be visited in the game. **Actions:** $\mathcal{A}^i = \times_{s \in \mathcal{S}} \mathcal{A}^i(s), i = f, g$ defines the action space for each agent. For each state $s \in \mathcal{S}$, $\mathcal{A}^i(s)$ is a finite, discrete set containing all the available actions (each being a real vector) for agent i at state s. If the set of candidate actions is the same at every state (which we assume for ease of exposition), then we write $\mathcal{A}^i = \{a^i(1), a^i(2), ..., a^i(|\mathcal{A}^i|)\}, i = f, g$ to refer to the fixed available action set for agent i. **Reward:** $R : \mathcal{S} \to \mathbb{R}$ is reward function for the game. In accordance with the zero-sum nature of the game, $R(s)$ denotes reward received by agent f while $-R(s)$ for agent g. **State Transition Function:** $P : \mathcal{S} \times \mathcal{A}^f \times \mathcal{A}^g \times \mathcal{S} \to \mathbb{R}$ is the state transition probability function of the game. If agents f, g take actions $a^f \in \mathcal{A}^f, a^g \in \mathcal{A}^g$ respectively at state $s \in \mathcal{S}$, then $s' \in \mathcal{S}$ is the next visited state with probability $P(s'|s, a^f, a^g)$. Note that in an MDP, the transition probability depends only on the current state and agents' actions. **Discount Factor:** $\gamma \in [0, 1)$ is the discount factor on cumulative reward.

We slightly abuse the notation and also use f, g to denote the **policies** the two agents used in the game. Note that the first order Markovian Property holds for all the policies involved in competitive MDPs mentioned in this study. Formally, $f \in \mathcal{F} = \underset{s \in \mathcal{S}}{\times} \Delta\big(\mathcal{A}^f(s)\big), g \in \mathcal{G} = \underset{s \in \mathcal{S}}{\times} \Delta\big(\mathcal{A}^g(s)\big)$ where $\Delta(A)$ denotes the set of all the possible probability distributions on a non-empty set A. Given policies f and g, the **state value function** (for f) at state s_0 is the expected discounted return

$$v^{f,g}(s_0; R) = \underset{\substack{a_t^f \sim f(a^f|s_t) \\ a_t^g \sim g(a^g|s_t) \\ s_{t+1} \sim P(s'|s_t, a_t^f, a_t^g)}}{\mathbb{E}} \sum_{t=0}^{\infty} \gamma^t R(s_t). \quad (1)$$

For a zero-sum discounted stochastic game, there exists at least one **Nash Equilibrium** $(f^*(R), g^*(R))$ such that, for any $s \in S$ and any policy pair (f, g) we have (Filar & Vrieze, 2012)

$$v^{f^*(R),g}(s;R) \geq v^{f^*(R),g^*(R)}(s;R) \geq v^{f,g^*(R)}(s;R), \tag{2}$$

and the **value vector** of the game is $\boldsymbol{v}^* = (v^*(s))_{s\in\mathcal{S}}$ where $v^*(s) = \max_f \min_g v^{f,g}(s;R)$. Videlicet, in zero-sum discounted stochastic games any Nash Equilibrium leads to the same state value functions and (2) always holds.

The purpose of our work is to address **inverse reinforcement learning** (IRL) problems in zero-sum stochastic games. As defined by Russel (1998), an IRL algorithm relies on the knowledge of (a) a model of the environment, i.e., state transition function $P(s'|s, a^f, a^g)$; and (b) an observation set $\mathcal{D} = \{(s_i, a_i^{E,f}, a_i^{E,g}) \mid i = 1, 2, 3, ..., N_\mathcal{D}\}$[1] is available, and shows how experts performed in the task. Here s_i is the visited state, and $a_i^{E,f}, a_i^{E,g}$ stand for actions taken by the two expert players where we use E to denote "experts". Note that rewards are not recorded in the demonstration set.

2.2. Objective Function

The formulation of an IRL problem hinges on the choice of the objective function. As stressed before, we do not assume the optimality of experts' demonstrations, but we still assume experts to have demonstrated performances that are comparable with the best possible policies in this game. More specifically, based on inequality (2) we have for all $s \in \mathcal{S}$,

$$v^{f^*(R),g^E|_\mathcal{D}}(s;R) \geq v^{f^*(R),g^*(R)}(s;R) \geq v^{f^E|_\mathcal{D},g^*(R)}(s;R).$$

Here R is the reward function we aim to solve for, and we stress that policies and value functions depend on it. The policy $f^E|_\mathcal{D}$ is a function that concurs with experts on the demonstration set $\mathcal{D}$. More formally speaking, $f^E|_\mathcal{D}(a|s) = \dfrac{\#\{(s,a,\_)\}}{\#\{(s,\_,\_)\}}$ when the denominator is larger than 0, where $\#$ denotes the count of occurrence in $\mathcal{D}$. For states s with $(s,\_,\_) \notin \mathcal{D}$, we remain agnostic about $f^E|_\mathcal{D}$. We similarly define $g^E|_\mathcal{D}(a|s) = \dfrac{\#\{(s,\_,a)\}}{\#\{(s,\_,\_)\}}$ when the denominator is positive. Meanwhile, we would like margins defined below to be reasonably tight:

$$\mathbb{E}_s\Big[v^{f^*(R),g^E|_\mathcal{D}}(s;R) - v^{f^*(R),g^*(R)}(s;R)\Big], \tag{3}$$

$$\mathbb{E}_s\Big[v^{f^*(R),g^*(R)}(s;R) - v^{f^E|_\mathcal{D},g^*(R)}(s;R)\Big] \tag{4}$$

[1] In $\mathcal{D}$, the next visited state is not required since we already have access to state transition probabilities. Subscript i does not necessarily stand for time, and each observation $(s_i, a_i^{E,f}, a_i^{E,g})$ can be drawn from different rounds of games.

By summing up the two margins, we yield the optimization problem for our IRL model

$$\min_R \min_{f^E|_\mathcal{D}, g^E|_\mathcal{D}} \mathbb{E}_s\Big[v^{f^*(R),g^E|_\mathcal{D}}(s;R) - v^{f^E|_\mathcal{D},g^*(R)}(s;R)\Big] \tag{5}$$

$$\text{where } f^*(R) \in \operatorname*{argmax}_{f\in\mathcal{F}} \min_{g\in\mathcal{G}} \frac{1}{N_s}\sum_{s\in S} v^{f,g}(s;R), \tag{6}$$

$$\text{and } g^*(R) \in \operatorname*{argmin}_{g\in\mathcal{G}} \max_{f\in\mathcal{F}} \frac{1}{N_s}\sum_{s\in S} v^{f,g}(s;R). \tag{7}$$

Note that the minimization on $f^E|_\mathcal{D}, g^E|_\mathcal{D}$ in (5) is required since $f^E|_\mathcal{D}, g^E|_\mathcal{D}$ are uniquely defined only on $\mathcal{D}$. The model encourages $f^E|_\mathcal{D}$ and $g^E|_\mathcal{D}$ outside of $\mathcal{D}$ to follow the actions in $f^*(R)$ and $g^*(R)$.

2.3. Algorithm

We approach (5) by an iterative algorithm. In each iteration, we update the Nash Equilibrium policies $f^*(R), g^*(R)$ given current reward function R, then R is updated based on incumbent $f^*(R), g^*(R)$, which requires the estimation of (3) and (4).

As for solving Nash Equilibrium in zero-sum discounted games, the proposed adversarial training algorithm is detailed in Algorithm 2 in Section 3. Next we focus on the update step of the reward function and the minimization operation on $f^E|_\mathcal{D}, g^E|_\mathcal{D}$ in (5).

Note that $(f^E|_\mathcal{D}, g^E|_\mathcal{D})$ poses inconveniences since the expert policy is unknown when a state s outside of $\mathcal{D}$ is visited. Our workaround is to let experts act only at the very first step, and use (f^*, g^*) for the following states. Such a treatment follows the logic that, by bounding the performance gap at the first step (equivalent to bounding advantage of actions), we also bound the gap of discounted cumulative reward for the infinitely many steps.

Specifically, $v^{f^E|_\mathcal{D},g^*(R)}(s;R)$ is estimated by sampling trajectories $(s_1^{E,g^*(R)}, \ldots, s_T^{E,g^*(R)})$ of a fixed length T in the following manner (note that E stands for experts):

$$\begin{aligned}
(s_0^{E,g^*(R)}, a_0^f, _) &\sim \mathcal{D},\\
a_t^g &\sim g^*(R)(a|s_t^{E,g^*(R)}) \text{ for } 0 \leq t \leq T,\\
a_t^f &\sim f^*(R)(a|s_t^{E,g^*(R)}) \text{ for } 1 \leq t \leq T,\\
\text{and } s_t^{E,g^*(R)} &\sim P(s'|s_{t-1}^{E,g^*(R)}, a_{t-1}^f, a_{t-1}^g) \text{ for } 1 \leq t \leq T.
\end{aligned} \tag{8}$$

For the estimation of $v^{f^*(R),g^E|_\mathcal{D}}(s;R)$, trajectories $(s_1^{f^*(R),E}, s_2^{f^*(R),E}, \ldots, s_T^{f^*(R),E})$ are sampled in a symmetric fashion, where the initial state $s_0^{f^*(R),E}$ and initial action for g a_0^g is sampled from $\mathcal{D}$, while the other actions and states are generated by $f^*(R), g^*(R)$.

Algorithm 1 Inverse Reinforcement Learning in Zero-Sum Discounted Stochastic Games

1: **Require:** Observed experts demonstrations $\mathcal{D} = \{(s_i, a_i^f, a_i^g) \mid i = 1, 2, \ldots, N_{\mathcal{D}}\}$; Positive integers K_R, I_R; Nash Equilibrium threshold τ; learning rate λ.
2: **Initialize:** Parameters θ_R for the reward function, θ_f, θ_g to parametrize $f_{\theta_f}(a|s), g_{\theta_g}(a|s)$ for Nash Equilibrium policies
3: **for** i $= 1, 2, 3, \ldots$ **do**
4: Update θ_f to find Nash Equilibrium policy for f under current R_{θ_R}, return also $\hat{v}^{f,g^{\text{best}}}$
5: Update θ_g to find Nash Equilibrium policy for g under current R_{θ_R}, return also $\hat{v}^{f^{\text{best}},g}$
6: **if** $i \,\%\, K_R = 0$ and $\hat{v}^{f^{\text{best}},g} - \hat{v}^{f,g^{\text{best}}} < \tau$ **then**
7: **for** j $= 1, 2, 3, \ldots, I_R$ **do**
8: Sample one observation from $\mathcal{D}$: $(s, a^{E,f}, a^{E,g})$
9: Use (8) to get $\{s_1^{f^*(R_{\theta_R}),E}, s_2^{f^*(R_{\theta_R}),E}, \ldots, s_T^{f^*(R_{\theta_R}),E}\}$, $\{s_1^{E,g^*(R_{\theta_R})}, s_2^{E,g^*(R_{\theta_R})}, \ldots, s_T^{E,g^*(R_{\theta_R})}\}$
10: $\hat{v}^f(\bar{\theta_R}) \leftarrow R_{\bar{\theta_R}}(s) + \sum_{t=1}^{T} \gamma^{t-1} R_{\bar{\theta_R}}\left(s_t^{f^*(R_{\theta_R}),E}\right)$
11: $\hat{v}^g(\bar{\theta_R}) \leftarrow R_{\bar{\theta_R}}(s) + \sum_{t=1}^{T} \gamma^{t-1} R_{\bar{\theta_R}}\left(s_t^{E,g^*(R_{\theta_R})}\right)$
12: $\theta_R \leftarrow \theta_R - \lambda \nabla_{\bar{\theta_R}}\left(\hat{v}^f(\bar{\theta_R}) - \hat{v}^g(\bar{\theta_R}) + \phi(\bar{\theta_R})\right)|_{\bar{\theta_R}=\theta_R}$

The algorithm is shown in Algorithm 1. We use deep neural networks as model approximations for policies and reward functions involved in the IRL task. To model the reward function, a deep neural network $R_{\theta_R}(s)$ is used and parametrized by θ_R. Similarly, two deep neural nets $f_{\theta_f}(a|s)$, $g_{\theta_g}(a|s)$ are maintained and updated in the algorithm to approximate $f^*(R_{\theta_R})$, $g^*(R_{\theta_R})$. A regularization term $\phi(\theta_R)$ is added to prevent trivial solutions ($R(s) \equiv 0$ for instance) and normalize the scale of $R(s)$.

As shown in Algorithm 1, in the policy step at steps 3 and 4, we update θ_f, θ_g under the current R_{θ_R} and try to find a Nash Equilibrium. Every K_R iterations, we check the performance of Nash Equilibrium policies in step 6. (The estimation of $\hat{v}^{f^{\text{best}},g}, \hat{v}^{f,g^{\text{best}}}$ is detailed in Section 3.) To ensure a good approximation of optimal policies under the incumbent reward function, the policy step is performed much more frequently than the reward step, and we update θ_R only when we believe f_{θ_f} and g_{θ_g} approximate $f^*(R_{\theta_R}), g^*(R_{\theta_R})$ well enough. When the current $f_{\theta_f}, g_{\theta_g}$ are accurate enough compared against a threshold τ, we perform I_R times a reward step that minimizes the performance gap between $f^*(R), g^E|_{\mathcal{D}}$ and $f^E|_{\mathcal{D}}, g^*(R)$ with regularization term $\phi(\theta_R)$ based on trajectories generated as described in (8). Note that the policy step is performed at each iteration, and the only difference between each policy step is that different trajectories are sampled and used to update the Nash Equilibrium policy under current R_{θ_R}.

3. Solving Nash Equilibrium in Zero-Sum Discounted Stochastic Games

To find Nash Equilibrium in zero-sum stochastic games, we adopt the framework of generative adversarial networks in (Goodfellow et al., 2014) to propose an algorithm for the policy step in Algorithm 1 at steps 4 and 5. Note that Algorithm 2 finds a Nash equilibrium policy for only one agent. We present the algorithm for solving $f^*(R)$, and $g^*(R)$ can be solved in a similar fashion. In the remainder of this section, we omit the dependency on R, which is fixed in the policy step.

Algorithm 2 Adversarial Training Algorithm for Solving $f^*(R)$ in Zero-Sum Games (Sketch)

1: **Require:** Positive integers K_g, K_{cycle}.
2: **Initialize:** Parameters θ_f, θ_g for policy models
3: **for** $i = 1, 2, 3, \ldots$. **do**
4: **if** $i \,\%\, K_{\text{cycle}} \leq K_g$ **then**
5: Update θ_g to optimize return for g based on PPO.
6: **else**
7: Update θ_f to optimize return for f based on PPO.

The definition of Nash Equilibrium in (6) and (7) in zero-sum discounted games naturally suggests adversarial training as a solution methodology. Informally speaking, we first identify the best response g to current f, and then update f marginally to compete against the best response g. We repeat these two steps until f^* has been reached. Two deep neural networks $f_{\theta_f}(s)$ and $g_{\theta_g}(s)$ are used, parametrized by θ_f and θ_g, respectively. Both networks take state vector $s \in \mathcal{S}$ as input, which is completely public to both sides. Each network outputs a probability distribution over the action space. The agents then sample actions from the probability distribution and act accordingly.

We give only the sketch of the algorithm in Algorithm 2 since the detailed implementation relies on a policy gradient method. In our experiments we choose the actor-critic style Proximal Policy Optimization algorithm (PPO) from (Schulman et al., 2017) because of its superior per-

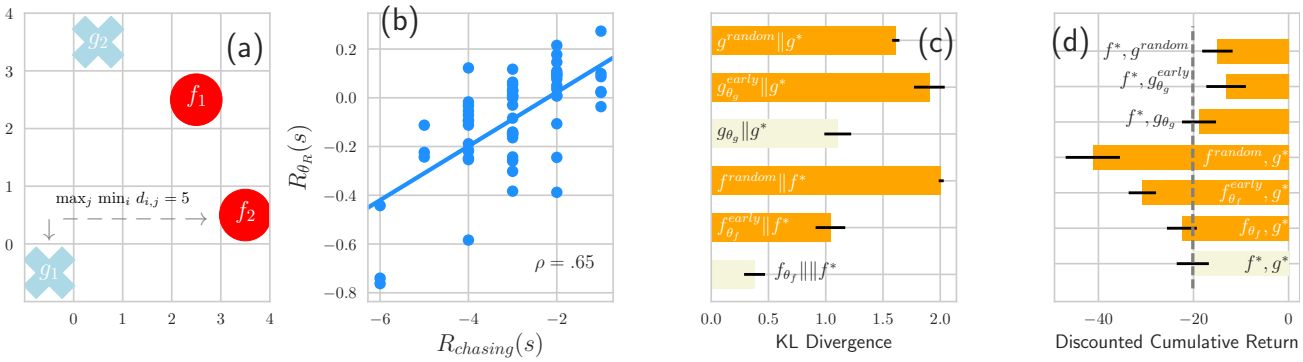

Figure 1. IRL training and results. (a) The chasing game on a 5×5 grid. In this example, the immediate reward, defined by the min-max distance, is -5 for f and 5 for g at the current state. (b) The recovered reward function $R_{\theta_R}(s)$ demonstrates a strong correlation to $R_{\text{chasing}}(s)$ ($p < .001$). (c) KL divergence between f_{θ_f}(or g_{θ_g}) and $f^*(R_{\text{chasing}})$(or $g^*(R_{\text{chasing}})$). "Early" denotes the models at the 20,000-th iteration, while "random" model follows a uniform distribution on all available actions. The final results of IRL training are as expected most similar to Nash Equilibrium policies. Error bars indicate the standard errors estimated on a batch of 64 samples. (d) Performance of policy models under R_{chasing}. The dashed reference line represents the performance of the Nash Equilibria models. Policies recovered by IRL training play similarly well when compared with the Nash Equilibria policies, while "early" and "random" models exhibit much more significant performance gaps. Error bars indicate the standard deviation estimated on a batch of 64 samples.

formances, and both policy and state-value models are used during training. The full algorithm is in Section A2 of the Appendix. Lastly, as requested in step 6 in Algorithm 1, periodically we need to estimate the performance of f_{θ_f} when competing against its best adversarial g_{θ_g}[2]. Each time, we run policies f, g to sample a batch of 64 T-step trajectories $\mathcal{T}$, and calculate the average of the discounted cumulative return on these trajectories to yield $\hat{v}^{f,g^{\text{best}}} = \frac{1}{|\mathcal{T}|} \sum_{(s_0,s_1,\ldots,s_T)\in\mathcal{T}} \sum_{t=0}^{T} \gamma^t R(s_t)$.

For solving Nash Equilibrium in zero-sum discounted stochastic games, we train an agent to always compete against its best possible opponent. The process can be thought of as performing gradient ascent for

$$F(f) = \min_{g\in\mathcal{G}} \frac{1}{N_s} \sum_{s\in\mathcal{S}} v^{f,g}(s).$$

Considering the case as if we use a tabular approach[3] for each agent's policy instead of using model approximations, we show that using this objective function to solve for f^* (or g^*) is theoretically a sound choice, because there is no local maximum in $F(f)$. We formally state the claim as the proposition below. For the details of the roof please refer to Section A3 of the Appendix.

Proposition 1: *Function $F(f)$ has no local maxima. Namely, if at a certain $\tilde{f}$ there exists no feasible strictly ascent direction for $F(\tilde{f})$, then $F(\tilde{f}) = \max_{f\in\mathcal{F}} F(f)$.*

[2]We reiterate that Algorithm 2 is meant for finding $f^*(R)$. The training for g^* is conducted separately in a similar fashion, and we estimate $\hat{v}^{f^{\text{best}},g}$ similarly.

[3]The tabular approach means that for any s and any a^f, a^g, $f(a^f|s)$ or $g(a^g|s)$ would be variables, and the only constraints are standard probability requirements.

4. Experimental Study

In this section, we illustrate the proposed IRL and Nash Equilibrium algorithms on a zero-sum stochastic game and present their performances. First, we introduce the zero-sum stochastic game used in our experiments and discuss its complexity and scale. Next, we demonstrate the quality of the reward functions and Nash Equilibrium policies solved by our IRL algorithm given only expert demonstrations, after which we show the performance of our Nash Equilibrium algorithm when the reward function is available.

4.1. The Chasing Game on Gridworld

Games on a grid have been widely used in reinforcement learning and IRL research works (Abbeel & Ng, 2004; Reddy et al., 2012; Lin et al., 2017; H.L. et al., 2015). In this type of games, each agent occupies one of the cells and is allowed to move to one of the neighboring cells at each step. The goal of each agent is to navigate itself to its own advantageous states, and the optimal policy depends on the relationship between the reward and location of all the agents. In our "chasing game" we simply flip the sign of the reward function in the stick-together game used by Prasad et al. (2015) to translate the purely cooperative game into its purely competitive counterpart, and extend the 1 vs 1 game to a 2 vs 2 version, which significantly increases the complexity of the game.

As shown in Fig. 1(a), the chasing game is played on a 5×5 grid. One team of predators (agent f) and another

team of preys (agent g) participate in the game with two players (denoted as f_1, f_2 and g_1, g_2 respectively) in each team. For the remainder of the paper, we set the discount factor γ as 0.9. At each state, each player is allowed to move upward, downward, leftward, rightward, or stay, and each action is deterministically executed. At the boundary the player must stay put. When choosing the actions, the agent f or g simultaneously controls the two predators or preys on each team. Thus, the game is considered to have $N_s = 25^4 = 390,625$ states and about $2 \cdot 10^7$ state-action pairs, which renders the existing nonlinear-optimization-based or iterative multi-agent IRL algorithms (Abbeel & Ng, 2004; Lin et al., 2017) inefficient.

As suggested by the name of the game, the immediate return is dictated by the distances between the predators and the preys, driving the predators to pursue the preys and the preys to stay away from the predators. The distance between any predator/prey pair (f_i,g_j with $1 \le i, j \le 2$) is $d_{i,j} = |x_{f_i} - x_{g_j}| + |y_{f_i} - y_{g_j}|$, namely the L1-norm distance where state $s = (x_{f_1}, y_{f_1}, x_{f_2}, y_{f_2}, x_{g_1}, y_{g_1}, x_{g_2}, y_{g_2})$ is the coordinate vector of both agents. Based on this pairwise distance, the immediate reward for the predators (f) is

$$R_{\text{chasing}}(s) = -D(s), \tag{9}$$

$$D(s) = \max_{j=1,2} \min_{i=1,2} d_{i,j}, \tag{10}$$

and for the preys it is $-R_{\text{chasing}}(s) = D(s)$. In other words, the immediate reward is determined by the prey that stays the farthest from all predators. This reward function encourages cooperation and accurate allocation of tasks within a team and adds an extra layer of complexity to the game. For example, two predators chasing the same prey would result in a low return as the other prey could conveniently run away, while the two preys hiding at the same corner make themselves approachable simultaneously. Lastly, there is no terminal state or "capture" action in the game. Therefore, even if a predator and a prey encounter each other, neither of them will be removed from the grid.

4.2. IRL Algorithm

In order to test our IRL algorithm using the chasing game where the immediate reward is unknown, we generate the sub-optimal demonstration set $\mathcal{D}$ as follows. First, under $R_{\text{chasing}}(s)$ we use Algorithm 2 to yield Nash Equilibrium policies $f^*(R_{\text{chasing}}), g^*(R_{\text{chasing}})$ approximated by deep neural nets. (Details of Nash Equilibrium training are covered in Section 4.3.) Then the policies act in a "deflected" ϵ-greedy fashion: we set $\epsilon = .1$ by default so that for each of the 4 players, there is 10% chance that it would (1) deflect the action sampled from $f^*(R_{\text{chasing}}), g^*(R_{\text{chasing}})$ by 90° or -90° if the action is not "stay," or (2) randomly sample one of the 4 remaining actions if the action is "stay", while 90% chance it would take the original action. We thus denote this set of expert demonstrations as $\mathcal{D}_{\epsilon=.1}$. The set consists of $64 \times 500 = 32,000$ trajectories with the length of 10 steps. Similarly, we generate sets $\mathcal{D}_{\epsilon=.05}, \mathcal{D}_{\epsilon=.2}$ to compare the performance of the IRL algorithm under demonstration sets of various quality. Note that when $\epsilon = .2$, the chance that none of the 4 players would take a deflected action at a certain step is only $(1 - 0.2)^4 = 0.4096$.

Next we describe the specifications of our models and the algorithm. For both policy models and state value function models required in actor-critic style PPO in Algorithm 2, we use deep neural nets with a 2-layer 256-neuron structure with rectified linear (Nair & Hinton, 2010) activation functions, after which a softmax layer outputs a probability distribution on the action space in policy models (or a linear transformation layer outputs an estimated $\hat{v}(s)$ in state value models). Moreover, with the natural state vector s (coordinates of players) we augment another vector s' which contains $x_{f_i} - x_{g_j}$ and $y_{f_i} - y_{g_j}$ for any pair (i, j) with $1 \le i, j \le 2$. Empirically, we find that with this tailored input vector (s, s') the models perform better without significantly increasing network complexities. For $R_{\theta_R}(s)$ we use a 2-layer 256-neuron structure with rectified linear activation functions, which is followed by a linear transformation layer that outputs a scalar as the immediate reward for state s, and the augmented input is also used. In terms of the training procedure, we set $K_R = 1000$, $I_R = 20$, and $\tau = 3$. The learning rate parameter for the reward function is $2.5 \cdot 10^{-5}$, T is set as 50, and Adam (Kingma & Ba, 2014) is used as optimizer. At each iteration[4], a batch of 64 observations are sampled simultaneously from $\mathcal{D}$, and each of the observation provides a gradient calculated in step 13 in Algorithm 1, after which the batch of gradients are averaged to update θ_R.

For the regularization function ϕ we adapt to our task the empirically useful concept of the "strong covariance prior" from (Lin et al., 2017). We assume the prior knowledge that rewards are related to distances between the predators and the preys. However, without knowing the max-min distance $D(s)$ defined in (10), we simply assume a negative covariance between $R_{\theta_R}(s)$ and the averaged distance $\bar{D}(s) = \frac{1}{4}\sum_i \sum_j d_{i,j}$. Besides, we control the absolute value and variance of $R(s)$ to prevent the scale of rewards from collapsing or explosion. These assumptions lead to

$$\phi(\theta_R) = c \Big(\mathbb{E}_{s\in\mathcal{D}} cov(R_{\theta_R}(s), \bar{D}(s)) + \big|\mathbb{E}_{s\in\mathcal{D}} R_{\theta_R}(s)\big| + \big|\mathbb{E}_{s\in\mathcal{D}} var[R_{\theta_R}(s)] - \rho\big| \Big),$$

where we set $c = 0.25$ and $\rho = 5$ in our experiments. During training, $\phi(\theta_R)$ is calculated on a batch of 64 demonstrations. Though not shown in the figures, we mention that before IRL training we also initialize the reward function by

[4] In this section, the word "iteration" refers to i in Algorithm 1.

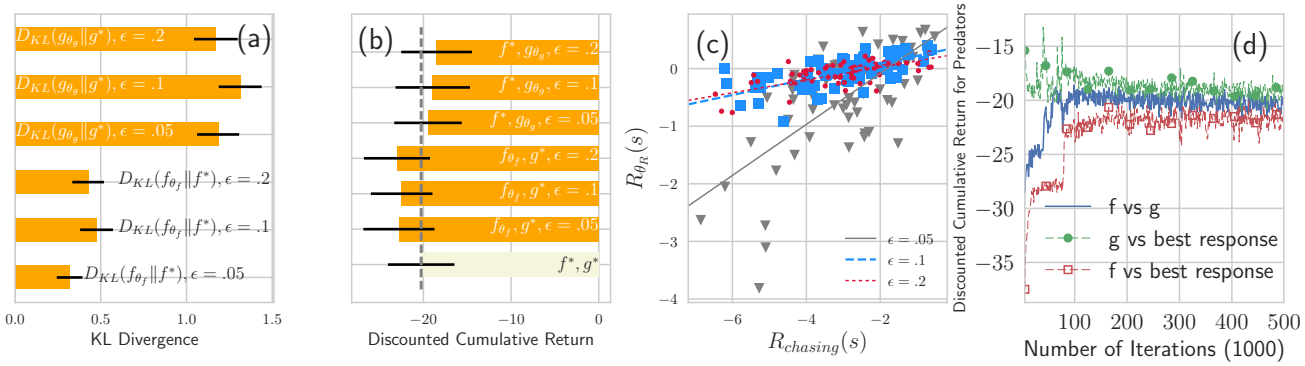

Figure 2. Comparison of IRL performances under $\mathcal{D}_\epsilon$ with different ϵ values. (a) KL divergence between the IRL and Nash Equilibrium policies are similar under different $\mathcal{D}_\epsilon$. Error bars indicate standard errors estimated on a batch of 64 samples. (b) Using different $\mathcal{D}_\epsilon$ we recover policies that perform similarly under the original $R_{\text{chasing}}(s)$. Performances are measured in the same way as for Fig. 1(c). Error bars indicate standard deviations estimated on a batch of 64 rounds of games. (c) Range of the recovered $R_{\theta_R}(s)$ under different $\mathcal{D}_\epsilon$. The larger the ϵ is, the smaller the range of $R_{\theta_R}(s)$. For readability of the figure a jitter is added to the x-coordinate of each point. (d) Performance of the proposed Nash Equilibrium algorithm in the chasing game.

training $R_{\theta_R}(s)$ for 5,000 iterations using only $\phi(\theta_R)$ above as the loss function.[5]

Performances of the algorithm using $\mathcal{D}_{\epsilon=.1}$ are shown in Fig. 1. First of all, regarding the quality of the obtained reward function, Fig. 1(b) reveals a strong correlation ($\rho = 0.65, p < .001$) between $R_{\text{chasing}}(s)$ and the $R_{\theta_R}(s)$ we recovered after 500,000 iterations of training, indicating that the model learns that the reward of each state should be highly dependent on $D(s)$ and behaves similarly as $R_{\text{chasing}}(s)$. As for quality of the recovered policies, we compare the divergence between the IRL and Nash Equilibrium policies. As shown in Fig. 1(c), when compared against a model that acts randomly or the "early" policy models obtained after 20,000 iterations, the IRL policies demonstrate behaviors that are most similar to those of the Nash Equilibrium policies.

A more direct measurement is to plug IRL policies back into the chasing game and evaluate their performances when competing against the Nash Equilibrium policies. In Fig. 1(d) we depict the performances of the IRL and Nash Equilibrium polices estimated in 64 rounds of games. The policies $f_{\theta_f}, g_{\theta_g}$ obtained after 500,000 iterations of IRL training demonstrate performances that are relatively close to those of Nash Equilibrium policies. For demonstrations of the policies please refer to Section A4 in the Appendix.

A comparison of model performances when using $\mathcal{D}_{\epsilon=.05}$, $\mathcal{D}_{\epsilon=.1}$, and $\mathcal{D}_{\epsilon=.2}$ illustrates the robustness of our algorithm

[5]Under the initialized R, we observe that the "early" policy models obtained after 20,000 iterations aim to minimize/maximize the averaged distance $\bar{D}(s)$ and act significantly differently from the final models, thus refuting the concern that prior knowledge in ϕ is too strong or the initialized $R(s)$ is close enough to $R_{\text{chasing}}(s)$. Section A4 in the Appendix provides details.

Table 1. Correlations between recovered $R(s)$ and $R_{\text{chasing}}(s)$

IRL Algorithm	$\epsilon = .05$	$\epsilon = .1$	$\epsilon = .2$
Algorithm 1	**0.65**	**0.68**	**0.66**
BIRL	0.28	-0.02	0.12
DIRL	-0.31	-0.15	0.11

despite the variation in sub-optimality of expert demonstrations. Fig. 2(a) and (b) show that IRL policies produced under different $\mathcal{D}_\epsilon$ behave similarly when compared against Nash Equilibrium policies, and demonstrate nearly the same performances in the original game. Interestingly, a similarly strong correlation ($r \approx .65$) is produced for each $\mathcal{D}_\epsilon$, while Fig. 2(c) shows that the scale of $R_{\theta_R}(s)$ decreases when ϵ increases. Intuitively, the error rate ϵ would not necessarily affect the order of preferences in states or actions, but could directly control the confidence in the absolute gap between different actions or states. This also explains why different policy functions perform similarly in Fig. 2(a) and (b), since by simply rescaling the reward function the optimal policies remain largely unchanged. This is ideal as the success of IRL training should not be critically influenced by the quality of the available demonstration set, and from sub-optimal demonstrations of distinct qualities the algorithm should always recover policies that perform very well.

To further illustrate the superior performance of our algorithm in large-scale games, we select the Bayesian-IRL (Lin et al., 2017) (BIRL) algorithm and Decentralized-IRL (Reddy et al., 2012) (DIRL) as benchmark IRL algorithms since to the best of our knowledge they are the only ones that solve competitive multi-agent IRL tasks. For the sake of a fair comparison, both algorithms need modifications since deep neural nets should be used as model approximations and the IRL training should proceed efficiently for large-

Table 2. Performance deterioration of recovered policies under R_{chasing} (A:Algorithm 1; B:BIRL; D:DIRL)

$\mathcal{D}_\epsilon$	f^{A}	f^{B}	f^{D}	g^{A}	g^{B}	g^{D}
.05	**11.8%**	24.1%	197.0%	**4.4%**	33.5%	38.9%
.1	**10.3%**	44.3%	100.0%	**6.9%**	33.5%	41.9%
.2	**13.3%**	68.5%	200.1%	**9.3%**	33.0%	40.9%

Table 3. Performances of solved Nash Equilibrium policies (A:Algorithm 2; B:Benchmark; R:Random)

Grid Size	$f^{\text{A}}, g^{\text{A}}$	$f^{\text{B}}, g^{\text{A}}$	$f^{\text{R}}, g^{\text{A}}$	$f^{\text{A}}, g^{\text{B}}$	$f^{\text{A}}, g^{\text{R}}$
5×5	-20.3	-21.2	-41.1	-20.0	-14.9
10×10	-44.7	-87.4	-94.2	-42.2	-31.8

scale games that cannot be solved by tabular approaches with enumeration of states or actions. For details of their implementation and training specifications please refer to Section A5 in the Appendix. From Tables 1 and 2 we see that benchmark algorithms fail to find a reward function that bears reasonably high correlation with $R_{\text{chasing}}(s)$ or recover well-performing policies especially for worse demonstration sets.[6] In conclusion, by dropping the optimality assumption in expert demonstrations and tailoring the subroutine Nash Equilibrium algorithm for large-scale games, our IRL algorithm outperforms benchmark algorithms significantly when they are implemented and utilized in the same setting.

4.3. Nash Equilibrium Algorithm

The Nash Equilibrium algorithm assumes that the reward function is available. We herein showcase the algorithm's performance under $R_{\text{chasing}}(s)$. As noted before, Algorithm 2 is meant for finding $f^*(R)$, and a separate training procedure is carried out to solve for $g^*(R)$. We only discuss in detail training of $f^*(R)$ because the same hyper-parameters and model structures are used for training $g^*(R)$.

The following hyper parameters are used in Algorithm 2 for the experiments in this section together with those already presented in Section 4.2. For the PPO style training, we set horizon length T as 10, and refresh frequency K_{refresh} as 10. Parameter λ for eligibility traces is set as 0.9. Regarding the adversarial training, we set K_{cycle} as 100 and K_g as 90. The learning rate parameter for best response models is set as $3 \cdot 10^{-4}$, while for the Nash Equilibrium policies $f_{\theta_f}, g_{\theta_g}$ it is 10^{-4}. Adam is used as optimizer. For each iteration[7], a batch of 64 trajectories is generated and used together for the stochastic gradient descent steps in the g or f step in Algorithm 2. For the first 5,000 iterations of every 50,000 iterations we perform the g step only. This is to ensure the quality of g_{θ_g}, which is expected to be the best possible response to current f_{θ_f} during training.

Fig. 2(d) shows the evolution of the policy models. The plot depicts $v^{f_{\theta_f}, g_{\theta_g}}(s_0; R_{\text{chasing}})$, $\min_g v^{f_{\theta_f}, g}(s_0; R_{\text{chasing}})$, and $\max_f v^{f, g_{\theta_g}}(s_0; R_{\text{chasing}})$. Recall that if f_{θ_f} and g_{θ_g} manage to reach the Nash Equilibrium, the three series should converge perfectly. As the figure shows, the adversarial training on f_{θ_f} and g_{θ_g} succeed in decreasing the gap to a pretty marginal level, suggesting that f_{θ_f} and g_{θ_g} are close enough to the Nash Equilibrium policies and there is little room to further improve their performances. The reader should check Section A6 in the Appendix for demonstrations of the solved Nash Equilibria policies.

To further demonstrate the superiority of the proposed Nash Equilibrium algorithm particularly for large games, we reformulate the quadratic programming problem proposed on page 125 in (Filar & Vrieze, 2012), which inspires the gradient descent algorithm to solve for Nash Equilibrium proposed in (H.L. & Bhatnagar, 2015). Section A7 in the Appendix provides details of algorithm implementation and training specification. In Table 3 we summarize their performances in the chasing game on both the original 5×5 grid and a 10×10 one. The values are the average of a batch of 64 trajectories with randomly initialized starting states. Performances under random policies are used as references. For the game on the 5×5 grid, we see that our algorithm yields a better policy for predators, while preys of both algorithms perform similarly. For the game on the 10×10 grid, the benchmark algorithm learned a much worse policy for predators (the improvement of our algorithm is $\frac{-87.4-(-44.7)}{-87.4} = 48.6\%$ from the benchmark algorithm), and the policies for preys learned by our algorithm are $\frac{-44.7-(-42.2)}{-42.2} = 5.9\%$ better than that of the benchmark algorithm. Overall, our adversarial training algorithm for Nash Equilibria in zero-sum stochastic games shows a significantly improved performance against the benchmark algorithm, especially for larger cases.

[6] Performance deterioration is measured by the change in discounted cumulative return when replacing f^*(or g^*) with the policy f(or g) recovered by IRL algorithms.

[7] In this section, the word "iteration" refers to i in Algorithm 2.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 1–8. ACM, 2004.

Akchurina, N. Multiagent reinforcement learning: algorithm converging to nash equilibrium in general-sum discounted stochastic games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 725–732, 2009.

Choi, S., Lee, K., Park, A., and Oh, S. Density matching reward learning. *arXiv preprint arXiv:1608.03694*, 2016.

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

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

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

H.L., P. and Bhatnagar, S. A study of gradient descent schemes for general-sum stochastic games. *arXiv preprint arXiv:1507.00093*, 2015.

H.L., P., L.A., P., and Bhatnagar, S. Two-timescale algorithms for learning nash equilibria in general-sum stochastic games. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pp. 1371–1379, 2015.

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

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Lin, X., Beling, P. A., and Cogill, R. Multi-agent inverse reinforcement learning for two-person zero-sum games. *IEEE Transactions on Computational Intelligence and AI in Games*, 2017.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning*, pp. 807–814, 2010.

Ramachandran, D. and Amir, E. Bayesian inverse reinforcement learning. *International Joint Conference on Artificial Intelligence*, pp. 2586–2591, 2007.

Reddy, T., Gopikrishna, V., Zaruba, G., and Huber, M. Inverse reinforcement learning for decentralized non-cooperative multiagent systems. In *Systems, Man, and Cybernetics*, pp. 1930–1935, 2012.

Ross, S., Gordon, G. J., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

Russell, S. Learning agents for uncertain environments. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, pp. 101–103, 1998.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ziebart, B. D., Maas, A. L., Bagnell, A., and Dey, A. K. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.