Journal of Artificial Intelligence Research 71 (2021) 925-951 Submitted 12/2020; published 08/2021

Evaluating Strategic Structures in Multi-Agent Inverse Reinforcement Learning

Justin Fu JUSTINJFU@EECS.BERKELEY.EDU
University of California, Berkeley
Department of Electrical Engineering & Computer Science
Berkeley, CA, 94720, USA

Andrea Tacchetti ATACCHET@GOOGLE.COM
Julien Perolat PEROLAT@GOOGLE.COM
Yoram Bachrach YORAMBACH@GOOGLE.COM
DeepMind
6 Pancras Square
London, N1C 4AG, United Kingdom

Abstract

A core question in multi-agent systems is understanding the motivations for an agent's actions based on their behavior. Inverse reinforcement learning provides a framework for extracting utility functions from observed agent behavior, casting the problem as finding domain parameters which induce such a behavior from rational decision makers. We show how to efficiently and scalably extend inverse reinforcement learning to multi-agent settings, by reducing the multi-agent problem to N single-agent problems while still satisfying rationality conditions such as strong rationality. However, we observe that rewards learned naively tend to lack insightful structure, which causes them to produce undesirable behavior when optimized in games with different players from those encountered during training. We further investigate conditions under which rewards or utility functions can be precisely identified, on problem domains such as normal-form and Markov games, as well as auctions, where we show we can learn reward functions that properly generalize to new settings.

1. Introduction

Game theory provides a general framework for predicting the behaviors of agents who interact with each other. It has found application as a powerful tool in analyzing the behavior of agents in contexts ranging from economics, robotics and human-robot interaction (Huang et al., 2019), to biology and social interaction. As game theory primarily concerns itself with predicting behavior when the agent's objectives are known, a natural question to ask is the inverse problem: how can we infer the motivations of an agent by observing its behavior? This problem setting is known as multi-agent inverse reinforcement learning (MAIRL) (Yu et al., 2019), inverse game theory (IGT) (Kuleshov & Schrijvers, 2015), or the inverse equilibrium problem (Waugh et al., 2011).

There are numerous reasons why one may wish to solve the inverse problem. The first and foremost is simply given by the problem statement itself - we wish to build models of other agents for the sake of understanding their motivations. This is an especially important

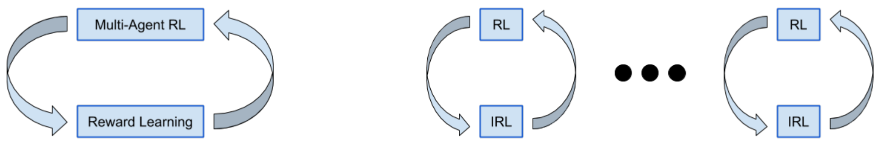


Figure 1: A standard multi-agent inverse reinforcement learning approach (left) interleaves a game-theoretic multi-agent solver with reward learning. In contrast, our approach (right) breaks down the problem into N simpler single-agent problems, which still results in an accurate approximation to solving the original problem.

problem to consider when one must interact with other agents, either adversarially (such as when competing over constrained resources) or cooperatively. A second reason is for the purpose of designing objectives for games such that desired behavior emerges when agents behave rationally. In multi-agent systems, unexpected and complex behavior may arise even when the rules of a game are very simple. Thus, if one has a desired outcome in mind, inverse game theory can be used to construct a game automatically via data-driven means, rather than through extensive human engineering. An example of such an application in the field of economics is mechanism design, which focuses on engineering incentive structures such that rational play results in the desired behavior.

Inverse game theory promises to recover utility functions that explain an observed set of behaviors. The typical assumption made is that agents are *rational*, and play to a solution concept or equilibrium where all players do not wish to deviate unilaterally from their current strategy. Constructing a general-purpose algorithm for the inverse game theory is difficult, due to the non-uniqueness of equilibria and the difficulty of computing them. A common formulation is to assume one is given behavior from a single equilibrium (Waugh et al., 2011; Song et al., 2018), but in general, an infinite number of games (including those that are not trivially equivalent) are compatible with a single equilibrium. From the algorithmic side, without enumerating all possible equilibria resulting from a utility or reward function, it is difficult to even determine if a correct solution has been found. Thus, much work has been concentrated in specific problem settings where the equilibrium is unique (Athey & Nekipelov, 2010; Song et al., 2018).

Our contributions are two-fold. First, we propose a simple approach for the inverse Nash equilibrium concept by showing a reduction of the problem to single-agent inverse reinforcement learning for Markov and Normal form games. This reduction approach can provide significant computational benefits since we do not need to run potentially expensive algorithms to solve for equilibria in N-player games. Our result also does not require restrictive assumptions such as the game admitting to a unique equilibrium, but our strongest result imposes additional constraints on the dynamics of the game. While the solutions found satisfy technical criteria for inverse equilibrium problems, we argue that due to ambiguity, rewards recovered in this way do not (in some particular sense) capture the full understanding of the game. We introduce a formal metric for game understanding and discuss special cases of unambiguous games where our approach can be used to recover reward functions uniquely, which includes important domains such as auctions. We evaluate our machinery on a classic game theory domain, a physics-based adversarial game, and a

Algorithm	Game Setting	Solution Concept	Func. Approx.
IESAR (ours)	Markov	Nash	Yes
MA-AIRL (Yu et al., 2019)	Markov	LSBRE	Yes
(Lin et al., 2019)	Markov	Various	No
(Natarajan et al., 2010)	Markov	Cooperative Game	No
(Waugh et al., 2011)	Normal-Form	Correlated	Yes
(Bestick et al., 2013)	Normal-Form	Correlated	Yes
(Kuleshov & Schrijvers, 2015)	Normal-Form	Correlated	No

Table 1: A comparison of our method against prior works in multi-agent inverse reinforcement learning and inverse game theory. The final column, labeled "Func. Approx", refers to whether the method allows learning utility functions representable by function approximators.

larger-scale simulated auction experiment where we show our method can extract accurate valuations for several popular auction mechanisms.

Our paper is structured as follows: Section 3 discusses relevant background from game theory and reinforcement learning. Section 4 introduces and analyzes our algorithm for multi-agent inverse reinforcement learning based on reducing the multi-agent problem to N instances of single-agent inverse reinforcement learning. Section 5 introduces the notion of player-agnostic rationality and discusses the effects and potential solutions to ambiguity in the multi-agent inverse reinforcement learning problem. Section 6 discusses practical instantiations of the IESAR concept in three distinct domains: a classic normal-form game, a continuous state space adversarial game, and valuation inference for several auction mechanisms.

2. Related Work

Reinforcement learning (RL) relates to methods that allow intelligent agents to learn to take actions in an environment so as to maximize their cumulative reward based on past experience; as opposed to supervised learning, RL does not require labelled input/output pairs, but rather focuses on tailoring the agent policy based on the rewards obtained in past interactions in the environment (Sutton & Barto, 2018). RL has been extensively studied in multi-agent settings, where multiple learners interact in the same environment (Hu et al., 1998; Busoniu et al., 2008).

Multi-agent reinforcement learning and related techniques have been applied across a very large spectrum of environments, ranging from board games (Tesauro, 1994; Silver et al., 2016; Moravcik et al., 2017; Brown & Sandholm, 2018; Jaderberg et al., 2019; Anthony et al., 2020; Gray et al., 2020) and computer games (Mnih et al., 2015; Vinyals et al., 2019; Berner et al., 2019), simulations of economic settings and social dilemmas (Leibo et al., 2017; Lerer & Peysakhovich, 2017; Hughes et al., 2020; Zheng et al., 2020), negotiation (Lewis et al., 2017; Bachrach et al., 2020), and simulated or real-world multi-robot systems (Yang & Gu, 2004; Kober et al., 2013; Stone et al., 2005; Banarse et al., 2019; Liu et al., 2019; Baker et al., 2019).

Inverse reinforcement learning (IRL) concerns itself with the inference of the reward functions of a Markov decision process (MDP), typically based on observed behavior (Ng & Russell, 2000). [1]

A popular framework for IRL is maximum entropy IRL (Ziebart et al., 2008; Ziebart, 2010). Since MaxEnt IRL is probabilistic in nature, it is able to accommodate sub-optimal demonstrations, and the MaxEnt principle disambiguates between multiple optimal policies arising from the same reward function. This framework has been applied successfully in high-dimensional continuous control problems (Ho & Ermon, 2016; Fu et al., 2017). Additionally, we argue for the learning of "robust" rewards which preserve performance as the parameters of the problem change, a concept which is explored in IRL by Amin et al. (2017) and Fu et al. (2017).

Inverse game theory. Within the context of game-theoretic and multi-agent settings, inverse game theory (Kuleshov & Schrijvers, 2015) and the inverse equilibrium problem (Waugh et al., 2011; Bestick et al., 2013) present general formulations for inferring utilities in normal form games using linear programming. In particular, these methods assume the agents rationalize to a correlated equilibrium, and Waugh et al. (2011) further adopts the MaxEnt principle to disambiguate among correlated equilibria that rationalizes observed behavior.

Multi-agent IRL (MAIRL) generalizes IGT from normal form games to Markov games (Natarajan et al., 2010; Lin et al., 2017; Yu et al., 2019; Lin et al., 2019). Lin et al. (2019) studies MAIRL for a variety of solution concepts, including the popular Nash and correlated equilibrium solution concepts. Yu et al. (2019) propose gradient-based algorithms for the inverse LSBRE (logistic stochastic best-response equilibrium) problem with function approximation, but requires that the game admits to a unique equilibrium.

One focus of our applications is within the field of auctions. Recent work has proposed using deep reinforcement learning for optimizing bidding strategies in sponsored search problems (Zhao et al., 2018). Additionally, the inverse game theory problem has been studied extensively in auctions. Athey and Nekipelov (2010) study a model of sponsored search auctions where the equilibrium is unique, and propose an estimator for the valuations of individual agents. Nekipelov et al. (2015) studies the case of no-regret learning solution concept as an alternative to Nash equilibrium formulations. Mechanism design (Myerson, 1983) concerns itself designing the incentives of an auction such that desired behavior, such as truthfulness, arises. Similar approaches, where game design is cast as an optimization problem, have been discussed in other contexts such as for voting games (De Keijzer et al., 2014), combinatorial auctions (Fotakis et al., 2018), and general trading games (Larsen & Zhang, 2018). Automated mechanism design (Sandholm, 2003) casts mechanism design as an optimization problem, where payoff structures can be found via constrained optimization or inferred from data (Vorobeychik et al., 2007; Tacchetti et al., 2019).

3. Preliminaries

In this section, we introduce our notation and prerequisite concepts in game theory and reinforcement learning.

1. IRL focuses on understanding or inferring agent motive and goals based on their observed behavior, which is a crucial component of building cooperative intelligent systems (Hadfield-Menell et al., 2016; Kretzschmar et al., 2016; Dafoe et al., 2020).

3.1 Reinforcement Learning

An infinite-horizon *Markov decision process* (MDP) is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, R, \gamma, \rho_0)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, $\mathcal{T}(s'|s,a)$ denotes the transition distribution,$R \in \mathcal{R}$ denotes a reward function, $\gamma \in (0,1)$ denotes a discount factor, and $\rho_0(s)$ denotes the initial state distribution.

The goal of forward reinforcement learning (RL) is typically to find a policy $\pi(a|s)$ that maximizes the expected future returns. We overload notation, and use $R(\pi)$ to denote the returns of a policy π under reward function R:

$$R(\pi) = E_{\rho_0,\pi,\mathcal{T}}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)] \tag{1}$$

A policy π^* is *optimal* if it achieves the maximum returns, $R(\pi^*) = \max_\pi R(\pi)$.

We use the notation $\mathcal{M}\backslash R = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \rho_0)$ for an MDP without a reward function. We can view solving the RL problem on an MDP $\mathcal{M}\backslash R$ as mapping from a reward function R to a set of optimal policies, $RL : \mathcal{R} \to \mathcal{P}(\Pi)$, where $\mathcal{P}(\Pi)$ denotes the power set over policies. Therefore, for any optimal policy π^* for a reward function R, we can write $\pi^* \in RL(R)$.

3.2 Inverse Reinforcement Learning

The inverse reinforcement learning (IRL) problem selects rewards for which demonstrated behavior is optimal. Formally, we denote an IRL algorithm as $IRL(\pi)$, which is a function that maps policies π back to a set of reward functions $\{R \in \mathcal{R} : R(\pi) = \max_{\pi'} R(\pi')\}$. If π is optimal under reward R^*, we use the notation $R^* \in IRL(\pi)$.

A common use-case of IRL, known as apprenticeship learning (Abbeel & Ng, 2004), aims to imitate a policy by optimizing a reward function obtained via IRL. We can use the notion of utility-matching to enforce that an inverse RL algorithm leads to good apprenticeship learning under the forward RL problem:

Definition 3.1. *Let π^* denote the behavior policy. An RL and IRL algorithm are ϵ-utility matching with respect to an MDP $\mathcal{M}\backslash R$ and reward class $\mathcal{R}$ if for all $\pi \in RL(\hat{R})$, $\hat{R} \in IRL(\pi^*)$,*

$$\max_{R \in \mathcal{R}} |R(\pi) - R(\pi^*)| \leq \epsilon$$

Utility matching eliminates many trivial solutions which technically solve the IRL problem - for example, the constant zero reward would be optimal for any behavior policy, but an algorithm that outputs the zero reward would not satisfy ϵ-utility matching in general. In fact, many IRL algorithms are constructed to ensure utility matching. A popular approach is feature matching (Abbeel & Ng, 2004), which provides a sufficient condition under which the learned policy will achieve similar returns to the behavior policy under any linear reward function. MaxEnt (Ziebart, 2010) approaches have been shown to be equivalent to a form of divergence minimization between the occupancy measures of the behavior and learned policies (Neu & Szepesvari, 2007; Ho & Ermon, 2016), which also guarantees utility matching under a broader class of reward functions.

3.3 Markov Games

A *Markov game* is an extension of an MDP to the multi-agent scenario. An N-player Markov game, like an MDP, is defined by the tuple $(\mathcal{S}, \mathcal{A}_{1:N}, \mathcal{T}, R_{1:N}, \gamma, \rho_0)$. Here, each player or agent has its own action space $\mathcal{A}_i$ and reward function $R_i(s, a_{1:N})$, and the transition function $\mathcal{T}(s'|s, a_{1:n})$ depends on the actions taken by all agents. As a notational shorthand, we use $\mathbf{a}_{-i}$ or $\boldsymbol{\pi}_{-i}$ to denote the set of actions or policies of the $N-1$ players other than player i. A *normal-form game* can be viewed as a single-step Markov game without state, or a multi-agent, multi-armed bandit problem - that is, it is defined as a tuple of action sets and rewards for each player, $(\mathcal{A}_{1:N}, R_{1:N})$, and policies (or strategies) $\pi_i(a_i)$ are distributions over the action set.

As each agent has its own reward function, the concept of optimality is not well-defined. Instead, the typical aim in game-theoretic scenarios is to find equilibria defined by a *solution concept*, which represent joint strategies at which agents cannot unilaterally improve their performance. A popular solution concept is the Nash equilibrium, defined as a set of joint policies $\boldsymbol{\pi}_{1:N}$ such that for all π_i:

$$E_{\pi_i, \boldsymbol{\pi}_{-i}}[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_{t,1:N})] \geq \max_{\pi_i} E_{\pi_i, \boldsymbol{\pi}_{-i}}[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_{t,1:N})] - \epsilon \tag{2}$$

The amount of slack, ϵ, is also referred to as the *regret* of the strategy profile $\boldsymbol{\pi}_{1:N}$ with respect to $R_{1:N}$, denoted as $\text{Reg}(\boldsymbol{\pi}_{1:N}|R_{1:N})$ (Waugh et al., 2011). The regret quantifies how far away players are from a perfect equilibrium. The traditional Nash equilibrium is obtained when $\epsilon = 0$, and when $\epsilon > 0$, the equilibrium is often referred to as an approximate ϵ-Nash equilibrium.

3.4 Inverse Equilibria and Strong Rationality

The goal of the inverse equilibrium problem is to find a set of utility or reward functions $\hat{R}_{1:N}$ for which observed behavior is at an equilibrium. We denote the expert or behavior strategies that produce the observed behavior as $\boldsymbol{\pi}^E$. The analogous concept to utility-matching in the inverse equilibrium problem is known as strong rationality, which likewise eliminates spurious solutions.

First, we assume that demonstration behavior comes from *rational players*, which are defined as players who prefer joint strategies with lower regret, meaning that they will try to play to perfect equilibria with $\epsilon = 0$, if possible.

We define the external regret of an agent, with respect to other players and a reward function, as:

$$\text{Reg}(\pi_i|\boldsymbol{\pi}_{-i}, R_i) = R_i(\pi_i, \boldsymbol{\pi}_{-i}) - \max_{\pi_i} R_i(\pi_i, \boldsymbol{\pi}_{-i}) \tag{3}$$

We define the "regret" of a joint strategy as the maximum external regret over all players:

$$\text{Reg}(\boldsymbol{\pi}_{1:N}, R_{1:N}) = \max_i \left[R_i(\pi_i, \boldsymbol{\pi}_{-i}) - \max_{\pi_i} R_i(\pi_i, \boldsymbol{\pi}_{-i}) \right] \tag{4}$$

If the regret of the joint strategy is ϵ, then the players are at an ϵ-Nash equilibrium.

Strong rationality can then be defined as:

Algorithm 1 Inverse Equilibrium Single-Agent Reduction (IESAR) Method

```
**Input:** Demonstration samples $\hat{\sigma} \sim \boldsymbol{\pi}^{E}{}_{1:N}$
(Markov Games) Estimate $\boldsymbol{\pi}^{E}{}_{1:N}$ from $\hat{\sigma}$.
**for** player $i = 1$ **to** $N$ **do**
    Solve the single agent IRL problem with a utility-matching IRL method
    $\hat{R}_i = IRL(\pi_i^E | \boldsymbol{\pi}_{-i}^E)$
**end for**
**return** $\hat{R}_{1:N}$
```

Definition 3.2 (Strong Rationality (Waugh et al., 2011)). *For a joint strategy $\pi_{1:N}$ to be ϵ-strongly rational, its regret must be no greater than that of the expert policies $\boldsymbol{\pi}^{E}{}_{1:N}$ under any set of reward functions within the reward class $\mathcal{R}$ by an ϵ amount:*

$$\max_{R_{1:N} \in \mathcal{R}^N} \mathit{Reg}(\boldsymbol{\pi}_{1:N}, R_{1:N}) - \mathit{Reg}(\boldsymbol{\pi}^{E}{}_{1:N}, R_{1:N}) \leq \epsilon$$

4. From Inverse Equilibria to Inverse Reinforcement Learning

The formal problem setting we are concerned with is the following: given a distribution over joint strategies $\boldsymbol{\pi}^E$ (the *expert behavior*), we wish to find a set of reward or utility functions $R_{1:N}$ that induces an equilibrium $\hat{\pi}$ that matches or is close to $\boldsymbol{\pi}^E$. Our approach, which we refer to as the **inverse equilibrium single-agent reduction** (IESAR) method, is to construct inverse equilibria for $\boldsymbol{\pi}^E$ by breaking down the problem into n separate, single-agent IRL problems, one for each agent, while holding the other agents fixed to the demonstrations. In this section, we introduce several theoretical results justifying why this approach is valid. First, we establish that the demonstrations form an equilibrium under the learned reward for any Markov game. Second, we show a stronger condition that for normal form games, rational behavior under the learned rewards is *strongly rational*, and that it is an equilibrium under the original rewards that produced the demonstrations.

Pseudocode for the method is shown in Alg 1. For Markov games, as one cannot simulate rollouts or trajectories through the reduced single-agent game (typically necessary to solve the IRL problem) without knowledge of the other player's policies, an additional step is needed to estimate each players' policies through a procedure such as behavioral cloning.

4.1 Analysis

We analyze our method by showing a utility-matching result, similar to Definition 3.1, for the multi-agent case.

First, we begin by defining the notion of an induced MDP. From the perspective of a single player, if all other players' strategies are held fixed, then the Markov game reduces to an MDP.

Definition 4.1 (Induced MDP). *Consider a Markov game $(\mathcal{S}, \mathcal{A}_{1:N}, \mathcal{T}, R_{1:N}, \gamma, \rho_0)$ with N players. Let $\mathcal{M}^{\boldsymbol{\pi}_{-i}}$ denote the MDP for player i holding the other player's strategies, $\boldsymbol{\pi}_{-i}$,* *fixed. Then, $\mathcal{M}^{\boldsymbol{\pi}_{-i}}$ is an MDP defined by the tuple $(\mathcal{S}, \mathcal{A}_i, \mathcal{T}^{\boldsymbol{\pi}_{-i}}, R^{\boldsymbol{\pi}_{-i}}, \gamma, \rho_0)$, where*

$$\mathcal{T}^{\boldsymbol{\pi}_{-i}}(s'|s, a_i) = \sum_{\boldsymbol{a}_{-i}} \mathcal{T}(s'|s, a_{1:N}) \prod_{j \neq i} \pi_j(a_j|s)$$

$$R^{\boldsymbol{\pi}_{-i}}(s, a_i) = E_{\boldsymbol{\pi}_{-i}}[R_i(s, a_{1:N})]$$

Using the notion of induced MDPs, we can rewrite the notion of an ϵ-Nash equilibrium as follows.

Definition 4.2. *Let RL and IRL problems under an induced MDP be denoted as $RL(R_i|\boldsymbol{\pi}_{-i})$ and $IRL(\pi_i|\boldsymbol{\pi}_{-i})$, respectively. Then, a set of policies $\boldsymbol{\pi}_{1:N}$ are in ϵ-Nash equilibrium if*

$$\forall_i, \pi_i \in RL^\epsilon(R_i|\boldsymbol{\pi}_{-i})$$

where RL^ϵ denotes the set of policies that are ϵ-suboptimal (achieving a return of at least $R(\pi_i^*) - \epsilon$) for the MDP induced by R_i and $\boldsymbol{\pi}_{-i}$.

Before stating our results, we lay out our assumptions.

- First, we assume that the observed equilibrium originates from *rational players*, as defined in Sec. 3.4.
- Second, we assume that the RL and IRL algorithms used satisfy the utility matching property outlined in Def. 3.1 for some ϵ.

We can now state our reduction result, which states that we can rationalize an equilibrium by solving a single-agent IRL problem for each player, as long as we have access to an RL and IRL algorithm that guarantees utility matching:

Theorem 4.1. *Let $\boldsymbol{\pi}^E$ denote a joint strategy (typically assumed to be a Nash equilibrium), and let RL and IRL be ϵ-utility matching algorithms for the induced MDPs $\mathcal{M}^{\boldsymbol{\pi}^E_{-i}}$. Then the observed equilibrium $\boldsymbol{\pi}^E$ is also an ϵ-equilibrium of the learned rewards $\hat{R}_{1:N}$, where each reward is the output of a single-agent IRL algorithm holding other players at the observed equilibrium, $\hat{R}_i \in IRL(\pi_i^E|\boldsymbol{\pi}^E_{-i})$.*

Proof. From the utility matching property, we have that for all i and R, $|R(\pi_i, \boldsymbol{\pi}^E_{-i}) - R(\pi_i^E, \boldsymbol{\pi}^E_{-i})| \leq \epsilon$, where π_i is the RL solution for the MDP induced by $\boldsymbol{\pi}^E_{-i}$ and $\hat{R}_i$. However, since π_i is optimal for $\hat{R}_i(\pi, \boldsymbol{\pi}^E_{-i})$, the utility matching statement implies that π_i^E is at most ϵ-suboptimal with respect to $\hat{R}_i(\pi, \boldsymbol{\pi}^E_{-i})$:

$$|\hat{R}(\pi_i^E, \boldsymbol{\pi}^E_{-i}) - \max_\pi \hat{R}(\pi, \boldsymbol{\pi}^E_{-i})| \leq \epsilon$$

Since this holds true for all players, $\boldsymbol{\pi}^E$ must be an ϵ-equilibrium of $\hat{R}_{1:N}$. □

To strengthen this result of Theorem 4.1, we can further establish that the equilibrium found is *strongly rational* (Waugh et al., 2011), meaning that the regret of the induced behavior $\hat{\pi}$ has regret no greater than the demonstration $\boldsymbol{\pi}^E$ under any reward function $R \in \mathcal{R}$. This condition is similar to the utility matching criteria extended to the multi-agent case, but requires a stronger condition in that players cannot influence each other's dynamics. This is rarely true in a Markov game, but it is always true in a normal-form game.

To show this result, we require one additional assumption on the expressivity of the reward class, which is that the class of reward functions $\mathcal{R}$ is inclusive of rewards in all possible induced MDPs. Formally, let $\mathcal{U}$ denote the set of utility functions in the Markov game, and Π_{-i} denote the class of all possible opponent's strategies. Then, we require the reward class to be at least as big as $\mathcal{R} \supseteq \{R : R(s_i, a_i) = E_{\boldsymbol{\pi}_{-i}}[U(s_i, a_i, \boldsymbol{s}_{-i}, \boldsymbol{a}_{-i})] \; \forall \boldsymbol{\pi}_{-i} \in \Pi_{-i}, U \in \mathcal{U}\}$. While this may seem like a technical and restrictive assumption, in practice this may not be difficult to satisfy:

- In games with linear utility functions, U is a linear function of the states and actions of the players. Since expectations are also linear, the reward class $\mathcal{R}$ is therefore simply linear functions of (s_i, a_i). Therefore, this assumption would equate to linear single-agent rewards $\mathcal{R}$ with the underlying payoff matrices $\mathcal{U}$ being linear.
- One can also simply use a highly expressive function class to represent rewards, such as neural networks, which will likely encompass the rewards defined by $\mathcal{R}$. In general, the simpler the reward class used by the IRL method, the easier it will be express $\mathcal{R}$ and satisfy the assumption. While this is not a precise statement, in our experiments in Sec. 6 we found neural network reward functions to work well empirically for recovering low-regret equilibria.

Theorem 4.2. *Let RL and IRL ϵ-utility matching. Furthermore, assume the game is a normal-form game, and the reward class includes all rewards representable as $\mathcal{R} \supseteq \{R : R(s_i, a_i) = E_{\boldsymbol{\pi}_{-i}}[U(s_i, a_i, \boldsymbol{s}_{-i}, \boldsymbol{a}_{-i})]\}$. Then, $\hat{\pi}_{1:N}$ is $2N\epsilon$-strongly rational with respect to $\boldsymbol{\pi}^E$, where $\hat{\pi}_i \in RL(IRL(\pi_i^E | \boldsymbol{\pi}_{-i}^E))$.*

Proof. Unlike Markov games, in normal form games there are no transition dynamics. By leveraging the reward class assumption, for any $\boldsymbol{\pi}_{-i}$,

$$\max_{R' \in \mathcal{R}} |R'(\hat{\pi}, \boldsymbol{\pi}_{-i}) - R'(\boldsymbol{\pi}^E, \boldsymbol{\pi}_{-i})| \leq \epsilon$$

This is because we can interpret the other player's strategies as a parameter of the reward function in the bandit setting.

We first bound the difference in regret of the equilibrium as follows:

$$\begin{aligned}
&\max_{R_{1:N}} \mathrm{Reg}(\hat{\boldsymbol{\pi}}_{1:N}, R_{1:N}) - \mathrm{Reg}({\boldsymbol{\pi}^E}_{1:N}, R_{1:N}) \\
&= \max_{R_{1:N}} \max_i \left[R_i(\hat{\pi}_i, \hat{\boldsymbol{\pi}}_{-i}) - \max_\pi R_i(\pi, \hat{\boldsymbol{\pi}_{-i}}) \right] - \max_i \left[R_i({\boldsymbol{\pi}^E}_i, \boldsymbol{\pi}^E_{-i}) - \max_\pi R_i(\pi, \boldsymbol{\pi}^E_{-i}) \right] \\
&\leq \max_{R_{1:N}} \max_i |R_i(\hat{\pi}_i, \hat{\boldsymbol{\pi}}_{-i}) - \max_\pi R_i(\pi, \hat{\boldsymbol{\pi}}_{-i}) - R_i({\boldsymbol{\pi}^E}_i, \boldsymbol{\pi}^E_{-i}) + \max_\pi R_i(\pi, \boldsymbol{\pi}^E_{-i})| \\
&\leq \max_{R_{1:N}} \max_i \underbrace{|R_i(\hat{\pi}_i, \hat{\boldsymbol{\pi}}_{-i}) - R_i({\boldsymbol{\pi}^E}_i, \boldsymbol{\pi}^E_{-i})|}_{L_1} + \underbrace{|\max_\pi R_i(\pi, \boldsymbol{\pi}^E_{-i}) - \max_\pi R_i(\pi, \hat{\boldsymbol{\pi}}_{-i})|}_{L_2}
\end{aligned}$$

Next, we bound L_1 and L_2 separately.

For L_1, we have

$$\begin{aligned}
&|R_i(\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_N) - R_i(\boldsymbol{\pi}^E{}_1, \ldots, \boldsymbol{\pi}^E{}_N)| \\
&\leq |R_i(\hat{\pi}_1, \hat{\pi}_2, \ldots, \hat{\pi}_N) - R_i(\boldsymbol{\pi}^E{}_1, \hat{\pi}_2, \ldots, \hat{\pi}_N) + R_i(\boldsymbol{\pi}^E{}_1, \hat{\pi}_2, \ldots, \hat{\pi}_N) - R_i(\boldsymbol{\pi}^E{}_1, \ldots, \boldsymbol{\pi}^E{}_N)| \\
&\leq \epsilon + |R_i(\boldsymbol{\pi}^E{}_1, \hat{\pi}_2, \ldots, \hat{\pi}_N) - R_i(\boldsymbol{\pi}^E{}_1, \ldots, \boldsymbol{\pi}^E{}_N)| \\
&\leq \epsilon + |R_i(\boldsymbol{\pi}^E{}_1, \hat{\pi}_2, \ldots, \hat{\pi}_N) - R_i(\boldsymbol{\pi}^E{}_1, \boldsymbol{\pi}^E{}_2, \ldots, \hat{\pi}_N) \\
&\quad + R_i(\boldsymbol{\pi}^E{}_1, \boldsymbol{\pi}^E{}_2, \ldots, \hat{\pi}_N) - R_i(\boldsymbol{\pi}^E{}_1, \ldots, \boldsymbol{\pi}^E{}_N)| \\
&\leq 2\epsilon + |R_i(\boldsymbol{\pi}^E{}_1, \boldsymbol{\pi}^E{}_2, \ldots, \hat{\pi}_N) - R_i(\boldsymbol{\pi}^E{}_1, \ldots, \boldsymbol{\pi}^E{}_N)| \\
&\ldots \\
&\leq N\epsilon
\end{aligned}$$

The inequalities follow because if two reward terms differ in only one agent's policy, we can apply the utility-matching assumption for that agent to bound the difference by ϵ. Initially, the two rewards $R(\hat{\pi}_1, \ldots, \hat{\pi}_N)$ and $R(\boldsymbol{\pi}^E{}_1, \ldots \boldsymbol{\pi}^E{}_N)$ differ in all N policies, but we can incrementally change each learned policy $\hat{\pi}_i$ to the demonstration policy $\boldsymbol{\pi}^E{}_i$ one-by-one by adding and subtracting the same quantity.

For L_2, we use the same sequence of inequalities to obtain:

$$\begin{aligned}
&|\max_{\pi} R_i(\pi, \boldsymbol{\pi}^E_{-i}) - \max_{\pi} R_i(\pi, \hat{\boldsymbol{\pi}}_{-i})| \\
&\leq \max_{\pi} |R_i(\pi, \boldsymbol{\pi}^E_{-i}) - R_i(\pi, \hat{\boldsymbol{\pi}}_{-i})| \\
&\leq (N-1)\epsilon
\end{aligned}$$

Adding L_1 and L_2 yields:

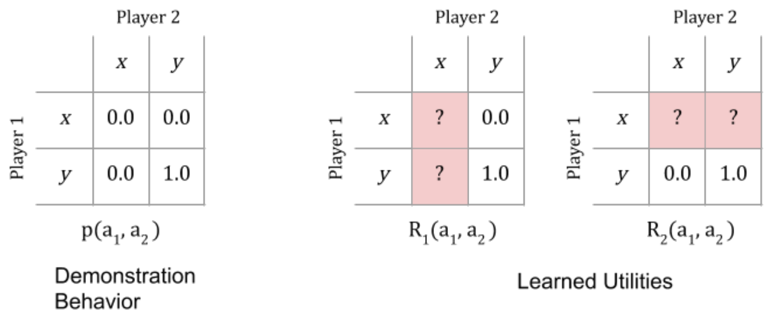
$$\max_{R_{1:N}} \mathrm{Reg}(\hat{\boldsymbol{\pi}}_{1:N}, R_{1:N}) - \mathrm{Reg}(\boldsymbol{\pi}^E{}_{1:N}, R_{1:N}) \leq (2N-1)\epsilon \leq 2N\epsilon$$

By the definition of strong-rationality, we can conclude that $\hat{\pi}_{1:N}$ is $2N\epsilon$-strongly rational with respect to $\boldsymbol{\pi}^E$ □

Strong rationality allows us to state that the learned behavior $\hat{\pi}_{1:N}$ is an equilibrium under the rewards of the expert $R_{1:N}$.

Corollary 4.2.1. *Let $\boldsymbol{\pi}^E$ be an δ-equilibrium of $R_{1:N}$. Then, $\hat{\pi}_{1:N}$ is a $(2N\epsilon+\delta)$-equilibrium of $R_{1:N}$.*

With a guarantee of strong rationality, we can expect the learned behavior to well-approximate the demonstrations, but the rewards extracted by the algorithm may not be representative of the ground truth rewards from which the demonstrations were trained, due to ambiguity. For example, this particular reduction procedure learns a reward that is constant with respect to the actions of other players, since the rewards learned by a single-agent IRL algorithm cannot depend on other agents. Thus, in the context of normal-form games, one can view the algorithm as extracting a single row (or linear combination of rows) of the entire payoff matrix, corresponding to the strategy played by the opponents at equilibrium (Fig. 2). However, Thm. 4.1 and Thm. 4.2 show that the learned rewards are nonetheless non-trivial, in that they are predictive of the expert behavior.


Player 2
Player 1
x
y
0.0
1.0
?
$p(a_1, a_2)$
$R_1(a_1, a_2)$
$R_2(a_1, a_2)$
Demonstration Behavior
Learned Utilities

	x	y
x	0.0	0.0
y	0.0	1.0

	x	y
x	?	0.0
y	?	1.0

	x	y
x	?	?
y	0.0	1.0

Figure 2: An example of ambiguity in inverse game theory, and demonstrating the "row"-identifying nature of single-agent IRL. The joint demonstration distribution is shown on the left ($p(a_1, a_2)$), for a game with 2 players and 2 actions x, y. Hypothetical learned rewards inferred for each player are shown on the right (R_1 and R_2), which infers rewards with the other player's strategies held to the expert distribution (in this case, a pure strategy of playing y). However, no matter what values are selected for the unknown reward values, marked by ?'s, the expert distribution will always be an equilibrium under R_1 and R_2.

A final analysis to consider is in the case of Markov games, where our method requires the ability to simulate the other player's policies in order to run RL or IRL. As mentioned previously, one way to address this issue in practice is to run behavioral cloning or another imitation learning method on the expert data. An important question to ask is how the error, measured in our case as the total variation distance $D_{TV}(\pi_1(\cdot|s)||\pi_2(\cdot|s))$, in the imitation learning process affects the final performance of the algorithm. To analyze this error, the following bound shows how running IRL on imitated policies $\hat{\boldsymbol{\pi}}^E$ affects the utility matching assumption when compared to running IRL on the original behavior policies $\boldsymbol{\pi}^E$:

Theorem 4.3. *Let RL and IRL be ϵ-utility matching for the induced MDPs $\mathcal{M}^{\hat{\boldsymbol{\pi}}^E_{-i}}$, the imitation learning error for each player be bounded as $\max_s D_{TV}(\hat{\pi}^E_i||\pi^E_i) \leq c$, and the reward function be bounded as $\max_{s,a_{1:N}} R(s, a_{1:N}) \leq r_{max}$. Then, for all $\pi \in RL(IRL(\pi^E_i|\hat{\boldsymbol{\pi}}^E_{-i}))$, the degree of utility matching can be bounded as*

$$\max_{R\in\mathcal{R}} |R(\pi, \hat{\boldsymbol{\pi}}^E_{-i}) - R(\pi^E_i, \boldsymbol{\pi}^E_{-i})| \leq \epsilon + \frac{2r_{max}Nc}{(1-\gamma)^2}$$

.

Proof. We begin by splitting the error into two terms as follows:

$$\max_R |R(\pi, \hat{\boldsymbol{\pi}}^E_{-i}) - R(\pi^E_i, \boldsymbol{\pi}^E_{-i})| \leq \max_R |R(\pi, \hat{\boldsymbol{\pi}}^E_{-i}) - R(\pi^E_i, \hat{\boldsymbol{\pi}}^E_{-i})| + \max_R |R(\pi^E_i, \hat{\boldsymbol{\pi}}^E_{-i}) - R(\pi^E_i, \boldsymbol{\pi}^E_{-i})|$$

The first term is controlled by the utility matching property of RL and IRL and is bounded by ϵ. Therefore, we only need to analyze the second term, which is concerned with the difference in returns from executing π^E_i under the MDP induced by the imitated policies $\hat{\boldsymbol{\pi}}^E_{-i}$

versus the original behavior policies $\boldsymbol{\pi}^E_{-i}$. We can leverage proofs from Janner et al. (2019) to simplify the analysis. We begin by first bounding the difference in the state distribution visited by the policies, and then bounding the returns.

Lemma B.1 of Janner et al. (2019) allows us to bound difference in the joint distribution of states and actions at a time step t by the sum of the individual errors. Let $\hat{p}(a_{1:N,t}|s_t)$ denote the action joint distribution under the imitated policies, and $p(a_{1:N,t}|s_t)$ denote the action joint distribution under the original behavior policies. Then, we have:

$$D_{TV}(\hat{p}(a_{1:N,t}|s_t)||p(a_{1:N,t}|s_t)) \leq c(N-1)$$

We can then apply Lemma B.2 of Janner et al. (2019) to bound the overall state distribution at time step t as:

$$D_{TV}(\hat{p}(s_t, a_{1:N,t})||p(s_t, a_{1:N,t})) \leq tc(N-1)$$

Next, we let $p(s, a_{1:N}) = (1-\gamma)\sum_{t=0}^{T} \gamma^t p(s_t, a_{1:N,t})$ denote the discounted state-action marginal distribution.

$$\begin{aligned}
|R(\pi_i^E, \hat{\boldsymbol{\pi}}^E_{-i}) - R(\pi_i^E, \boldsymbol{\pi}^E_{-i})| &= |\sum_{s,a_{1:N}} (\hat{p}(s, a_{1:N}) - p(s, a_{1:N}))R(s, a_{1:N})| \\
&= |\sum_{s,a_{1:N}} (\sum_t \gamma^t \hat{p}(s_t, a_{1:N,t}) - p(s_t, a_{1:N,t}))R(s_t, a_{1:N,t})| \\
&= |\sum_t \sum_{s,a_{1:N}} \gamma^t (\hat{p}(s_t, a_{1:N,t}) - p(s_t, a_{1:N,t}))R(s, a_{1:N,t})| \\
&\leq \sum_t \sum_{s,a_{1:N}} \gamma^t |\hat{p}(s_t, a_{1:N,t}) - p(s_t, a_{1:N,t})| R(s, a_{1:N,t}) \\
&\leq r_{\max} \sum_t \sum_{s,a_{1:N}} \gamma^t |\hat{p}(s_t, a_{1:N,t}) - p(s_t, a_{1:N,t})| \\
&\leq 2r_{\max} \sum_t \sum_{s,a_{1:N}} \gamma^t tc(N-1) \\
&\leq \frac{2r_{\max}c(N-1)}{(1-\gamma)^2}
\end{aligned}$$

Substituting this back into the original error, we have:

$$\max_R |R(\pi, \hat{\boldsymbol{\pi}}^E_{-i}) - R(\pi_i^E, \boldsymbol{\pi}^E_{-i})| \leq \epsilon + \frac{2r_{\max}c(N-1)}{(1-\gamma)^2}$$

□

This bound shows that the gap in utility matching introduced by imitation learning errors is bounded linearly by the number of players in the game, and quadratically in the horizon. This bound also can be substituted into Thm. 4.1 to characterize the relationship between the observed equilibrium $\boldsymbol{\pi}^E$ and the learned rewards.

4.2 On the Choice of RL and IRL Algorithms

IESAR describes a generic procedure that can be used with any RL/IRL algorithm pair that satisfies the utility matching property (Def. 3.1). There are many potential choices for IRL algorithms, which we will now discuss.

A major class of IRL algorithms that satisfy utility matching are the so-called feature-matching approaches (Abbeel & Ng, 2004; Syed & Schapire, 2008). These approaches allow one to use linear function approximation for problems when the state space is large and complex enough that a tabular representation of the reward is intractable. Some popular variants of feature-matching approaches are based around the MaxEnt framework (Ziebart, 2010). Ho and Ermon (2016) show that MaxEnt IRL can be interpreted as matching occupancy measures (which in turn satisfies the utility-matching property), and (Fu et al., 2017) propose an extension that allows one to learn rewards represented by differentiable functions, such as neural networks. These MaxEnt approaches have been demonstrated to work effectively on continuous, high-dimensional tasks.

5. Predictive Power of Rewards in Novel Settings

Inverse methods are commonly used as “imitation learning” methods where the aim is to train a policy that replicates some empirical behavior. A question one may ask is: why not directly imitate the behavior policy, rather than perform an indirect procedure where one learns rewards and trains a policy on those rewards? One scenario in which one may wish to learn rewards instead of policies is when one must learn a model of behavior in one environment, and test the model in a novel environment. In such a scenario, one could potentially train a policy using a reward in the novel environment, rather than relying on the policy to generalize to a new task. However, most inverse methods will not produce desirable results in this setting due to reward ambiguity.

Thm. 4.1 provides a simple procedure that satisfies several conditions of inverse game theory (such as strong rationality), but also highlights that the problem is under-constrained, and the algorithm will return one (of potentially infinitely many) rewards which satisfy the problem conditions. In some sense, the standard criteria for inverse game theory are ”too loose”, and do not constrain the problem enough to always capture strategic understanding of an agent’s intentions. We can easily learn utilities that do not reveal the nature of a game. For example, we can recover a cooperative game when the original game is an anti-coordination game (see Sec. 6.3), or we can learn a reward function equal to an advantage function which would then recover the behavior policy when performing RL. Reward ambiguity can be a problem when one wishes to inspect the learned rewards for the sake of understanding behavior, but also more importantly, predicting the behavior of an agent when certain problem parameters change, such as when paired with new agents with different utilities, such as an agent playing an auction against players different than those seen during training.

We can quantify this concept by extending the notion of rationality to opponents with changing utilities. This notion of rationality is invariant to transformations of the reward function which leave behaviors unchanged - for example, values of the reward on dominated actions.

Definition 5.1 (Player-Agnostic Rationality). *Let $R = (R_i)_{i \in \{1,\ldots,N\}}$ be the estimated reward and $R^* = (R_i^*)_{i \in \{1,\ldots,N\}}$ represent the ground truth reward. The estimated reward is ϵ-player agnostic if $\forall i, \forall \boldsymbol{\pi}_{-i}$ we have $|\max_{\pi_i} R_i(\pi_i, \boldsymbol{\pi}_{-i}) - \max_{\pi_i} R_i^*(\pi_i, \boldsymbol{\pi}_{-i})| < \epsilon$*

It is easy to see that a reward that is consistent with strongly rational behavior does not necessarily satisfy player-agnostic rationality. Take for example the game in Fig. 2, which satisfies strong rationality no matter what the unknown values are. If we change player 2 to prefer playing action x over y, then there are no constraints on whether player 1 prefers action x or y.

How can we ensure that a learned reward is player-agnostic? As implied by the definition, a natural solution is to learn from a dataset where the same agent plays with the same reward function against different agents. This style of concept has been explored in the field of IRL in the context of different dynamics (Amin et al., 2017). For the purposes of this paper, we explore a simpler solution, which is to narrow the reward function class until it is *identifiable* from the behavior. In other words, we assume there is an injective mapping from equilibrium behaviors to the parameters of the reward function. Therefore, in the limit of infinite data, one will recover the ground-truth reward function and satisfy player-agnostic rationality. An important class of identifiable games is auctions, which are identifiable from equilibrium behavior under mild assumptions (i.e. first price and second price auctions (Athey & Haile, 2002), and generalized second price auctions (Athey & Nekipelov, 2010)).

6. Applications

We evaluate our approach on three domains, with the goal of testing whether the single-agent reduction method is effective in rationalizing expert behavior, and whether in identifiable games, we can successfully predict behavior. These domains are:

1. A stateful Markov game, to see if learned rewards rationalize the given demonstrations. We selected the game Keep-Away (Lowe et al., 2017), a continuous space and action game where an agent navigates to a target and the adversary tries to block the agent.

2. A classic normal-form game, Chicken, to check for strong rationality of learned behaviors and demonstrate problems with reward ambiguity.

3. First-price, second-price, and generalized second-price (GSP) auctions, to see if we can accurately infer the parameters of the player's reward functions (in this case, valuations), and to study player-agnostic rationality (Section 5) in situations when an agent must generalize their play against agents that are different from those encountered during training.

6.1 Practical Algorithm

The single-agent reduction procedure infers utilities by running single-agent IRL for each agent, while fixing the other agents' policies to the demonstrations. Therefore in order to evaluate our method, we need to select an appropriate single-agent IRL procedure to run. For all of our experiments, we adopt a game-theoretic formulation inspired by Abbeel and Ng

(2004), Syed and Schapire (2008). We use the following saddle-point optimization problem:

$$IRL(\boldsymbol{\pi}^E) = \text{argmin}_{\theta, \|\theta\| \leq c} \max_{\pi} R_\theta(\pi) - R_\theta(\boldsymbol{\pi}^E)$$

The output of this procedure is a reward function parameter θ^*. As regularization, we constrain the 1-norm of the parameter vector for our Chicken, Keep-Away, and neural network Auction utility experiments, but many other choices are reasonable (Ho & Ermon, 2016).

The game-theoretic formulation describes a saddle-point problem. We implement this procedure as a coordinate descent algorithm, where we optimize the inner loop (policy optimization) for N gradient descent steps for every outer loop (reward optimization) gradient step. For the choice of first-order optimizer, we use Adam (Kingma & Ba, 2015), which is a common choice for IRL methods (Ho & Ermon, 2016; Fu et al., 2017). The gradients are straightforward to compute for normal-form games. However, for Markov games, we assume access to a transition model that it is amenable to the reparameterization trick (the model is differentiable with respect to the states, actions, and noise that is drawn i.i.d.). We perform a form of differentiable planning by directly computing the reparameterized gradient of the returns with respect to the policy. Practically, this is implemented by encoding the dynamics via an automatic differentiation library. We use a differentiable policy optimizer to simplify the experimental setup, but we note that our method can accommodate sample-based policy optimizers, such as model-free and model-based reinforcement learning algorithms if one chooses to do so.

6.2 Keep-Away

Keep-Away (Lowe et al., 2017; Song et al., 2018) is a continuous state-action game that involves an agent attempting to reach a landmark, and an adversary trying to push the agent away. The agent's and adversary's rewards are inverse to each other - the agent is rewarded based on the Euclidean distance to the landmark, and the adversary is penalized on the same distance (the adversary also has an extra shaping term which encourages it to be closer to the landmark). A diagram of this game is shown in Fig. 3.

As an evaluation metric, we compute an estimate of the regret of the learned behaviors, as measured by the ground-truth rewards from which demonstration behavior was generated from. The regret quantifies how close a behavior is to a Nash equilibrium - having at most ϵ regret for all players implies that the behavior is an ϵ-Nash equilibrium. We estimate regret by optimizing the policy under the given reward function using gradient descent, and taking the difference between the returns of the resulting policy and the original policy.

We choose regret as an evaluation metric because in continuous state space problems such as Keep-Away, it is not obvious how to measure how close a learned reward function is to the ground truth. Reward functions can be ambiguous, such as under linear and reward shaping transformations, which makes direct comparison difficult.

6.2.1 Experiment Parameters

We represent each agent's policy as a 2-layer ReLU neural network that outputs the parameters of a Gaussian distribution, and each reward as a 2-layer ReLU network which takes as input the state and both players' actions.

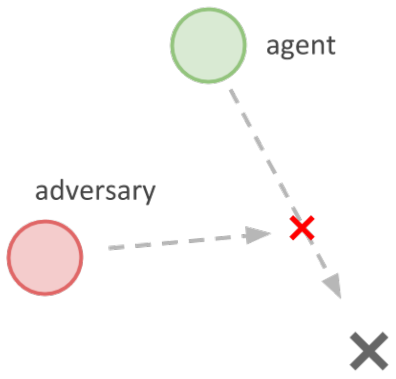


Figure 3: A diagram of the 2-player Keep-Away game. The agent (green) tries to reach the landmark (black X), while the adversary (red) tries to keep the agent away.

We obtain demonstrations by running simultaneous gradient descent on each player's utility function, and then sampling the behavior from the policies once an equilibrium is found. While theoretically there is a possibility of orbiting behavior, we did not observe this to a large extent in practice. The regret of the equilibrium used to collect the demonstrations is recorded as the "Expert" column in Table 2.

For all methods, we use a learning rate of 10^{-3} (selected via grid search between 10^{-4} and 10^{-1}) for both the policy and utility functions. We constrain the norm of the utility function to $c = 100$ (selected via grid search between 1 and 1000). In our coordinate descent procedure, we optimize the inner loop (policy optimization) for 10 steps for each outer loop (utility optimization) step. We found that increasing the number of inner loop steps to 5 or 10 improved the stability of the algorithm, at the cost of more computation.

For the environment, we used a horizon length of 25 steps to mitigate the cost of back-propagation through the environment dynamics. The goal location is located at the origin, $(0, 0)$, and the agents positions are initialized randomly in a uniform box between $[(-1, -1), (1, 1)]$.

6.2.2 Results and Discussion

Our results are shown in Table 2. We compare our method to two baselines. Cloning refers to behavioral cloning (Argall et al., 2009), which simply uses least-squares regression to predict each agent's actions from the state. We select MA-GAIL as a baseline for a multi-agent inverse reinforcement learning algorithm. MA-GAIL (Song et al., 2018) is an extension of the GAIL (Ho & Ermon, 2016) to the multi-agent setting, and can be viewed as a GAN, where the policy serves as the generator and the reward function is the logits of a discriminator. We use MA-GAIL rather than MA-AIRL (Yu et al., 2019) as for the purposes of apprenticeship learning (learning a policy via a learned reward function), the two methods differ primarily in the architecture of the discriminator. Additionally, the other methods discussed in Table 1 are not amenable to neural network function approximation. MA-GAIL requires an equilibrium solver, which we implement as simultaneous gradient

	Expert	Random	Cloning	MA-GAIL	IESAR (ours)
Max. Regret	0.011 ± 0.013	41.3 ± 9.07	0.032 ± 0.038	0.033 ± 0.026	0.067 ± 0.063
P1 Returns	-3.84 ± 0.049	-37.5 ± 13.0	-3.87 ± 0.143	-3.94 ± 0.022	-6.50 ± 1.90
P2 Returns	0.045 ± 0.035	-4.13 ± 15.9	-0.030 ± 0.142	0.155 ± 0.132	-0.122 ± 1.66

Table 2: Experimental results for the 2-player Keep-Away game. The regret of the equilibrium found (lower is better), and the returns for each player are reported (P1 returns and P2 returns), with mean and standard deviations over 10 random seeds. The regret of the expert demonstrations is reported to provide an oracle lower-bound - we would not expect an agent to perform better than this even if they perfectly imitated the demonstrations.

ascent on the agent's individual policies. We use the same differentiable planner as IESAR to optimize the policy to keep the comparison fair. While this simultaneous gradient ascent lacks theoretical guarantees compared to an algorithm such as fictitious play, we found little empirical difference on this particular problem. We note that both MA-GAIL and IESAR suffer from theoretical deficiencies in this domain - MA-GAIL requires that the game admits to a unique equilibrium, and IESAR requires that the game has non-interactive dynamics.

We evaluate on two metrics: regret and agent returns. We measure regret by running additional policy optimization steps after an algorithm has finished training. This suffers from a potential drawback in that it depends on the performance of the policy optimizer, so the reported value is approximate. We also include the returns of the agent at the equilibrium found during training.

We find that all three methods (cloning, MA-GAIL, and IESAR) are able to produce behavior that achieves regret that is very close to the expert demonstrations themselves. Thus, we view the final performance of all three methods as roughly equivalent. This is fairly remarkable in that while IESAR theoretically requires non-interactive games to produce good behavior, it is still able to produce expert-like behavior in a more general Markov game.

6.3 Chicken

As an initial experiment to highlight the pitfalls of ambiguity, we consider the game of chicken, which describes a simple 2-player anti-coordination game. The game models two drivers approaching a single intersection - each driver may either "swerve" to yield to the other driver, or go "straight". Going straight while the other player swerves yields a small positive reward. However, if both players go straight, then they crash and receive a large negative penalty. The payoff matrix for Chicken is described as follows:

	Swerve	Straight
Swerve	0, 0	+1, -1
Straight	-1, +1	-10, -10

From this game, we select the (Swerve, Straight) Nash equilibrium to use as a demonstration, represented as a joint probability table:

	Swerve	Straight
Swerve	0	1
Straight	0	0

We run IESAR with a reward that is constant with respect to the other player's action, and is restricted to have an L1 norm of 1. We obtain the following learned reward (top) and learned behavior (bottom):

	Swerve	Straight
Swerve	0.255, -0.233	0.255, 0.267
Straight	-0.245, -0.233	-0.245, 0.267

	Swerve	Straight
Swerve	0	1
Straight	0	0

As we can see, the solution found has zero regret. However, an obvious ambiguity problem arises. The inferred game is a cooperative game that does not resemble the original anti-coordination game, but nevertheless shares the same Nash equilibrium as the demonstrated behavior. Without placing further assumptions on the game, or providing additional data (such as specifying a second or third Nash equilibrium), it is difficult to disambiguate the problem.

6.4 Payoff Inference in Auctions

Our final experiment relates to the problem of inferring valuations in three commonly used auction mechanisms: sealed-bid first-price, second-price, and generalized second-price auctions with independently drawn private valuations. In first and second-price auctions, players bid for a single item, and the winner pays either the highest bid (first-price) or the second highest bid (second-price). The generalized second-price auction, commonly used for search engine advertising, has players bid for multiple items. The highest bidding player wins the most valuable item, and pays the bid of the second-highest bidder. The second highest bidder wins the second most valuable item, and pays the bid of the third-highest bidder, and so on until all items are exhausted. In the context of advertising, the "items" are advertisement slots on a search result page - the slots higher up in a search page are typically considered more valuable. These auctions can be cast as an extension of normal-form games known as a *Bayesian game*.

The second-price auction is a truthful auction, and bidding one's true valuation is always a dominant strategy. Therefore, the problem of inferring valuations is somewhat less interesting. However, first-price and generalized second price (Edelman et al., 2007) are non-truthful and rational players may find it advantageous to underbid their valuations in an attempt to pay lower costs. In these auctions, knowing other players' underlying valuations may give one a competitive advantage in bidding.

In our experimental setup, we simulate a 4-player auction, where each player's valuation is drawn from a Gaussian distribution with a fixed standard deviation of 1.0 and a mean that varies between 1.0 and 2.0. Each player has a linear policy, and bids $b_i = \alpha_i v_i$, where v_i is the player's valuation and α_i is the learned policy parameter. We do not believe this is a restrictive assumption, since for both first-price and second-price auctions, the strategy at the Bayesian Nash equilibrium is a linear function of the valuation. To obtain training demonstration data, we perform simultaneous gradient updates for each player against the aggregate historical play (analogous to fictitious play) until convergence, and then draw samples from the resulting trained policies. Once presented with the demonstration samples, we learn a reward function parameterized by a scalar valuation.

6.4.1 Bayesian Game Representation

For our auctions, we use the independent private values (IPV) model, meaning that each player has a valuation that is independently drawn and unknown to the other agents. Such an auction is typically formalized as a Bayesian game, where the private valuations are implemented as an additional *type* variable τ_i that is drawn for each player at the start of each game. The reward function then depends on this parameter τ_i.

In our experiments, each type variable τ_i is independently drawn from a unit normal distribution. Each player's learnable valuations and policies/strategies are a function of the type, i.e. $v_i(\tau_i)$ and $\pi(\tau_i)$.

To incorporate types τ in a Bayesian game, we use a slightly modified IRL objective of the form:

$$IRL(\boldsymbol{\pi}^E) = \text{argmin}_{\theta, \|\theta\|=c} \max_{\pi} \mathbb{E}_\tau[R_\theta(\pi, \tau) - R_\theta(\boldsymbol{\pi}^E, \tau^E)]$$

The issue with this formulation is that the types of the demonstrations, τ^E, are assumed to be unobserved. To alleviate this problem, we used monotonically increasing policies to estimate types from demonstrations by inverting the observed bids, $\tau = \pi^{-1}(b)$. All policies we used are monotonically increasing and linear policies in the bid parameterized as $\pi(\tau) = \tau e^w + b$, where w and b are the learnable scale and bias parameters. We found that using invertible policies to estimate types yielded comparable performance to directly using the true types of the demonstrations.

6.4.2 Soft Payoff Functions

To ensure differentiablity of payoff functions, we use "soft" variants of first-price, second-price, and generalized second-price auctions by interpreting the bids as logits to a softmax or ranking distribution that models the probability an agent is selected as a winner. The original auctions can be recovered by taking the temperature parameter to the limit, $\beta \to 0$. For our experiments, we use a temperature of $\beta = 1$.

Let $\Pr(\text{rank}_i = 1) = \frac{e^{b_i/\beta}}{\sum_j e^{b_j/\beta}}$ denote the probability of player i of achieving the highest rank, or winning, the auction with bid b_i.

For **soft first-price auctions**, we use the payoff function:

$$R_i(b_{1:N}, \tau_i) = (v_i(\tau_i) - b_{(1)})\Pr(\text{rank}_i = 1)$$

Where $b_{(1)}$ denotes the highest bid.

Soft second-price auctions are identical to soft first-price auctions except the cost paid is that of the second-highest bid $b_{(2)}$ in the auction:

$$R_i(b_{1:N}, \tau_i) = (v_i(\tau_i) - b_{(2)})\Pr(\text{rank}_i = 1)$$

Soft generalized second-price auctions require modeling a distribution over rankings of players, rather than simply the winner. We use the Plackett-Luce model for computing rank probabilities, which can be viewed as a recursive softmax where if a player "wins" a slot, the player's bid/logit is removed and the remaining players repeat for the next slot. If there are J slots in the auction, then the payoff is:

$$R_i(b_{1:N}, \tau_i) = \sum_{j=1}^{J} \alpha_j (v_i(\tau_i) - b_{(j+1)}) \Pr(\text{rank}_i = j)$$

where α_j is the CTR value of slot j. Generally, higher ranked slots have higher CTR values, to indicate higher desirability.

6.4.3 Results and Discussion

Learning curve results for our valuation inference experiments are shown in Fig. 4, plotted across coordinate descent steps.

We can see that the learned valuations are fairly accurate across all 3 auction domains, with the estimated mean valuations for all players approaching the true mean valuations. We see that in the second-price and GSP domains, the estimated valuations for some players are overestimated, then underestimated, and eventually converge to the true valuations. This behavior may be due to our optimization procedure, which runs coordinate descent steps alternating between the reward function and the player's policy that may create orbiting behavior.

Overall, this experiment demonstrates that valuation inference is possible even for non-truthful auctions such as first-price and GSP auctions.

6.5 Generalization Capabilities of Learned Utilities

Our next experiment evaluates the performance of inverse game theory agents in the context of playing against new agents. As discussed in Section 5, we believe one of the primary benefits of using inverse game-theoretic frameworks for modeling behavior (over arguably simpler methods such as behavioral cloning) is that if the parameters of the game change (the opponents change their objectives, or the mechanics of the game change), then one can still hope to recover optimal play in the new game. In this sense, inverse game-theory allows one to recover a "deeper" understanding of a user's behavior than simple behavioral cloning, and has more predictive power in novel settings. To avoid notational confusion, we make the distinction between the *payoff* or *utility* function, versus the *valuation*, which are often used interchangeably in auctions literature. The utility function in our terminology refers to (with minor variations depending on the specific mechanism of the auction) the valuation of the agent, minus the cost the agent pays, times an indicator function if the agent wins the auction. In this experiment, we study how we can achieve player-agnostic rationality in an auction setting by restricting the function class of the payoff function.

We investigate this problem as an extension of our study in auctions, as playing auctions against different players with different valuations is a common occurrence in practice. We set up the experiment as follows - the "train" auction is played by 4 players with the same valuations as our previous experiment (a standard deviation of 1.0, with mean valuations of

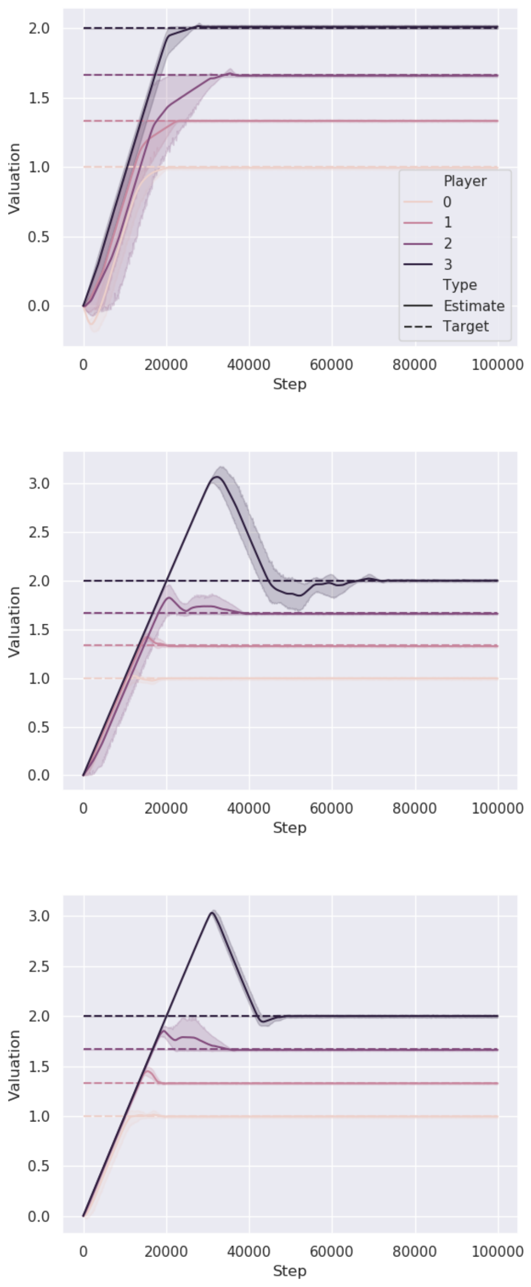


Figure 4: Learning curves for inferring valuations in first-price (top), second-price (middle), and generalized second-price (bottom) auctions. Means and standard deviations (shaded regions) are plotted over 10 seeds. The dotted horizontal line represents the player's ground truth valuation, and the solid line of the same color represents the algorithm's guess of the player's valuation, varying over algorithm iterations.

	Oracle	Cloning	IESAR-Neural	IESAR-Valuation
Train	0.257 ± 0.005	0.208 ± 0.055	0.223 ± 0.026	0.252 ± 0.010
Transfer	0.464 ± 0.006	0.386 ± 0.117	0.366 ± 0.050	0.458 ± 0.003

Table 3: Transfer learning results on the auction domain, with mean and standard deviations of payoffs over 10 random seeds. We observe that a neural utility architecture, while performing similarly to the valuation architecture during training, performs poorly when evaluated against new agents. In contrast, the identifiable valuation architecture preserves performance when playing against new agents.

1.0, 1.33, 1.66, and 2.0). From the perspective of player 1, we then lower the mean valuations of other players in the "transfer" auction by 1, resulting in mean valuations of 1.0, 0.33, 0.66, and 1.0.

We evaluate several strategies under this setup. "Oracle" refers to the payoff received by an agent at the Bayesian Nash equilibrium of the game. "Cloning" refers to the performance of the oracle strategy from the "Train" auction, and directly executing it in the "transfer" auction. Here, the strategy of the agent is fixed, and we let the opponents play to an equilibrium. The payoff reported is the payoff of the agent at this equilibrium. Finally, we report 2 methods that use utility learning. "Neural" refers to the payoff of a strategy using a learned neural network utility function in the "Train" auction, and "Valuation" refers to the payoff of a strategy using a learned valuation from the "Train" auction. The "Valuation" strategy can be viewed as a special case of "Neural", where the architecture of the utility is restricted to have a single parameter (the valuation mean). In order to evaluate the method, we substitute the learned utility for the utility of the agent, and then report the payoff of the agent at the Bayesian Nash equilibrium against the opponents. Our results (measured in average payoff, over 5 random seeds) for a first-price auction are shown in Table 3:

From these results, we can see that all methods achieve good performance on the "Train" task. However, the performance of both the "Cloning" and "Neural" strategies suffer when evaluated in the "Transfer" auction, whereas the "Valuation" strategy better preserves performance. The sub-par performance of "Cloning" is expected - as the valuations of other players are lowered in the transfer domain, the agent should aggressively bid higher to maximize returns, and therefore keeping the same strategy that performed well in the "Train" auction is suboptimal. The sub-par performance of the "Neural" strategy likely comes down to identifiability - inverse game theory learned a utility function which performed well in the "Train" auction, but did not preserve the semantics of the game outside of the equilibrium. As a concrete example of this, in our previous experiment in the Chicken domain, IGT recovered a cooperative game rather than an anti-coordination game. Only in the "Valuation" strategy is performance preserved, likely because the restricted architecture of the utility function maintains identifiability of the model, and therefore is still predictive of agent behavior when the parameters of the game change.

7. Conclusions and Future Work

In this paper, we have proposed IESAR - a simple, efficient method for extracting utility functions from demonstrated behavior by effectively reducing the problem to single-agent inverse reinforcement learning. The reduction produces rewards that rationalize observed equilibria in Markov games, and additionally satisfies strong rationality in Normal-form games. IESAR has significant computational advantages in that it does not require solving an N-player game in the forward part of the algorithm, and only requires running RL. Moreover, it can operate on games with non-unique equilibria. Our empirical analysis shows that IESAR produces comparable performance compared to alternative methods that do require full game-theoretic solvers.

Second, we highlight the importance of player-agnostic rationality in extracting salient reward functions. We note that utility functions can be ambiguous even under the strong rationality assumption. While this has little effect on behavior when an agent plays against the same players as seen during training, if the agent were to optimize the learned utility against new agents, the behavior could deviate. We show that one way to guarantee that learned rewards satisfy player-agnostic rationality is to restrict the reward class, and we demonstrate in auction settings that rewards satisfying strong rationality alone can produce poor performance when playing against new agents.

There are several limitations to our work that would be interesting directions to study in future work. First, our method only has strong-rationality guarantees for Normal-form games. However, our empirical results indicate that IESAR can perform well even in Markov games. One potential question to study is to analyze what properties of games make IESAR still an effective algorithm in Markov game scenarios. Developing a general-purpose method that provides strong rationality guarantees for arbitrary Markov games is still an important open problem. Additionally, we have been primarily concerned with Nash equilibria in this work, and other solution concepts could be more relevant for modeling human behavior. For example, using extensive-form correlated equilibria (Von Stengel & Forges, 2008) could provide a more computationally feasible problem formulation, as well as potentially be a more realistic model of human behavior on real demonstrations.

The analysis in our work does not explicitly take into account the sampling error due to using a finite number of demonstrations, but our analysis includes the study of approximate equilibria. An interesting theoretical direction to explore would be to study the sample complexity of IESAR and develop finite-sample bounds.

We also only studied player-agnostic rationality in the context of restricting the reward class to ensure that the optimal solution is identifiable from observed behavior. One promising direction for future work is to study how to learn agnostic rewards with more general function classes, by learning a shared utility across a wide variety of games. This idea is similar to the concept of repeated inverse reinforcement learning (Amin et al., 2017) in the single-agent case.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, p. 1.

ACM.

Amin, K., Jiang, N., & Singh, S. (2017). Repeated inverse reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 1815–1824.

Anthony, T., Eccles, T., Tacchetti, A., Kramar, J., Gemp, I., Hudson, T. C., Porcel, N., Lanctot, M., Perolat, J., Everett, R., et al. (2020). Learning to play no-press diplomacy with best response policy iteration. *CoRR*, *abs/2006.04635*.

Argall, B. D., Chernova, S., Veloso, M., & Browning, B. (2009). A survey of robot learning from demonstration. *Robotics and autonomous systems*, *57*(5), 469–483.

Athey, S., & Haile, P. A. (2002). Identification of standard auction models. *Econometrica*, *70*(6), 2107–2140.

Athey, S., & Nekipelov, D. (2010). A structural model of sponsored search advertising auctions. In *Sixth ad auctions workshop*.

Bachrach, Y., Everett, R., Hughes, E., Lazaridou, A., Leibo, J. Z., Lanctot, M., Johanson, M., Czarnecki, W. M., & Graepel, T. (2020). Negotiating team formation using deep reinforcement learning. *Artificial Intelligence*, *288*, 103356.

Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., & Mordatch, I. (2019). Emergent tool use from multi-agent autocurricula. In *International Conference on Learning Representations*.

Banarse, D., Bachrach, Y., Liu, S., Lever, G., Heess, N., Fernando, C., Kohli, P., & Graepel, T. (2019). The body is not a given: Joint agent policy learning and morphology evolution. In *AAMAS*, Vol. 18, pp. 1134–1142.

Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al. (2019). Dota 2 with large scale deep reinforcement learning. *CoRR*, *abs/1912.06680*.

Bestick, A., Ratliff, L. J., Yan, P., Bajcsy, R., & Sastry, S. S. (2013). An inverse correlated equilibrium framework for utility learning in multiplayer, noncooperative settings. In *Proceedings of the 2nd ACM international conference on High confidence networked systems*, pp. 9–16. ACM.

Brown, N., & Sandholm, T. (2018). Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, *359*(6374), 418–424.

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *38*(2), 156–172.

Dafoe, A., Hughes, E., Bachrach, Y., Collins, T., McKee, K. R., Leibo, J. Z., Larson, K., & Graepel, T. (2020). Open problems in cooperative ai. *CoRR*, *abs/2012.08630*.

De Keijzer, B., Klos, T. B., & Zhang, Y. (2014). Finding optimal solutions for voting game design problems. *Journal of Artificial Intelligence Research*, *50*, 105–140.

Edelman, B., Ostrovsky, M., & Schwarz, M. (2007). Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American economic review*, *97*(1), 242–259.

Fotakis, D., Krysta, P., & Ventre, C. (2018). The power of verification for greedy mechanism design. *Journal of Artificial Intelligence Research*, *62*, 459–488.

Fu, J., Luo, K., & Levine, S. (2017). Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations*.

Gray, J., Lerer, A., Bakhtin, A., & Brown, N. (2020). Human-level performance in no-press diplomacy via equilibrium search. In *International Conference on Learning Representations*.

Hadfield-Menell, D., Russell, S. J., Abbeel, P., & Dragan, A. (2016). Cooperative inverse reinforcement learning. *Advances in neural information processing systems*, *29*, 3909–3917.

Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in neural information processing systems*, pp. 4565–4573.

Hu, J., Wellman, M. P., et al. (1998). Multiagent reinforcement learning: theoretical framework and an algorithm.. In *International Conference on Machine Learning*, Vol. 98, pp. 242–250. Citeseer.

Huang, S. H., Held, D., Abbeel, P., & Dragan, A. D. (2019). Enabling robots to communicate their objectives. *Autonomous Robots*, *43*(2), 309–326.

Hughes, E., Anthony, T. W., Eccles, T., Leibo, J. Z., Balduzzi, D., & Bachrach, Y. (2020). Learning to resolve alliance dilemmas in many-player zero-sum games. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 538–547.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. (2019). Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, *364*(6443), 859–865.

Janner, M., Fu, J., Zhang, M., & Levine, S. (2019). When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pp. 12519–12530.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), 1238–1274.

Kretzschmar, H., Spies, M., Sprunk, C., & Burgard, W. (2016). Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, *35*(11), 1289–1307.

Kuleshov, V., & Schrijvers, O. (2015). Inverse game theory: Learning utilities in succinct games. In *International Conference on Web and Internet Economics*, pp. 413–427. Springer.

Larsen, B., & Zhang, A. L. (2018). A mechanism design approach to identification and estimation. Tech. rep., National Bureau of Economic Research.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., & Graepel, T. (2017). Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pp. 464–473.

Lerer, A., & Peysakhovich, A. (2017). Maintaining cooperation in complex social dilemmas using deep reinforcement learning. In *arXiv preprint arXiv:1707.01068*.

Lewis, M., Yarats, D., Dauphin, Y., Parikh, D., & Batra, D. (2017). Deal or no deal? end-to-end learning of negotiation dialogues. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2443–2453.

Lin, X., Adams, S. C., & Beling, P. A. (2019). Multi-agent inverse reinforcement learning for certain general-sum stochastic games. *Journal of Artificial Intelligence Research*, *66*, 473–502.

Lin, X., Beling, P. A., & Cogill, R. (2017). Multiagent inverse reinforcement learning for two-person zero-sum games. *IEEE Transactions on Games*, *10*(1), 56–68.

Liu, S., Lever, G., Merel, J., Tunyasuvunakool, S., Heess, N., & Graepel, T. (2019). Emergent coordination through competition. In *International Conference on Learning Representations*.

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

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Moravcik, M., Schmid, M., Burch, N., Lisy, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., & Bowling, M. (2017). Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, *356*(6337), 508–513.

Myerson, R. (1983). Mechanism design by an informed principal. *Econometrica*, *51*(6), 1767–97.

Natarajan, S., Kunapuli, G., Judah, K., Tadepalli, P., Kersting, K., & Shavlik, J. (2010). Multi-agent inverse reinforcement learning. In *2010 Ninth International Conference on Machine Learning and Applications*, pp. 395–400. IEEE.

Nekipelov, D., Syrgkanis, V., & Tardos, E. (2015). Econometrics for learning agents. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pp. 1–18.

Neu, G., & Szepesvari, C. (2007). Apprenticeship learning using inverse reinforcement learning and gradient methods. In *Proceedings of the Twenty-Third Conference on Uncertainty in Artificial Intelligence*, pp. 295–302. AUAI Press.

Ng, A. Y., & Russell, S. J. (2000). Algorithms for inverse reinforcement learning.. In *International Conference on Machine Learning*.

Sandholm, T. (2003). Automated mechanism design: A new application area for search algorithms. In *International Conference on Principles and Practice of Constraint Programming*, pp. 19–36. Springer.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Song, J., Ren, H., Sadigh, D., & Ermon, S. (2018). Multi-agent generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 7461–7472.

Stone, P., Sutton, R. S., & Kuhlmann, G. (2005). Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, *13*(3), 165–188.

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

Syed, U., & Schapire, R. E. (2008). A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*, pp. 1449–1456.

Tacchetti, A., Strouse, D., Garnelo, M., Graepel, T., & Bachrach, Y. (2019). A neural architecture for designing truthful and efficient auctions. *CoRR*, *abs/1907.05181*.

Tesauro, G. (1994). Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural computation*, *6*(2), 215–219.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, *575*(7782), 350–354.

Von Stengel, B., & Forges, F. (2008). Extensive-form correlated equilibrium: Definition and computational complexity. *Mathematics of Operations Research*, *33*(4), 1002–1022.

Vorobeychik, Y., Wellman, M. P., & Singh, S. (2007). Learning payoff functions in infinite games. *Machine Learning*, *67*(1-2), 145–168.

Waugh, K., Ziebart, B. D., & Bagnell, J. A. (2011). Computational rationalization: The inverse equilibrium problem. In *International Conference on Machine Learning.*

Yang, E., & Gu, D. (2004). Multiagent reinforcement learning for multi-robot systems: A survey. Tech. rep., tech. rep.

Yu, L., Song, J., & Ermon, S. (2019). Multi-agent adversarial inverse reinforcement learning. In *International Conference on Machine Learning*, pp. 7194–7201. PMLR.

Zhao, J., Qiu, G., Guan, Z., Zhao, W., & He, X. (2018). Deep reinforcement learning for sponsored search real-time bidding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1021–1030.

Zheng, S., Trott, A., Srinivasa, S., Naik, N., Gruesbeck, M., Parkes, D. C., & Socher, R. (2020). The ai economist: Improving equality and productivity with ai-driven tax policies. In *arXiv preprint arXiv:2004.13332.*

Ziebart, B. D. (2010). *Modeling purposeful adaptive behavior with the principle of maximum causal entropy.* Ph.D. thesis, Carnegie Mellon University.

Ziebart, B. D., Maas, A., Bagnell, J. A., & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence.*