Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

Multiagent Adversarial Inverse Reinforcement Learning

Ermo Wei ewei@gmu.edu
Drew Wicke drew.wicke@gmail.com
Sean Luke sean@cs.gmu.edu

Technical Report GMU-CS-TR-2019-2

Abstract

Learning to coordinate is a hard task for reinforcement learning due to a game-theoretic pathology known as *relative overgeneralization*. To help deal with this issue, we propose two methods which apply forms of imitation learning to the problem of learning coordinated behaviors. The proposed methods have a close connection to multiagent actor-critic models, and will avoid relative overgeneralization if the right demonstrations are given. We compare our algorithms with MADDPG, a state-of-the-art approach, and show that our methods achieve better coordination in multiagent cooperative tasks.

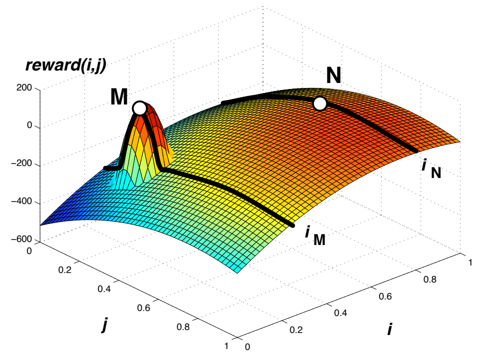


Figure 1: The relative overgeneralization pathology in continuous games.

1 Introduction

Multiagent Reinforcement Learning (or MARL) applies Reinforcement Learning (RL) to more than one agent. Like RL, the environment is some current *state*, which the agent can only sense through its *observations*; the agents each perform some *action* while in that state, the agents each receive some *reward*, the state *transitions* to some new state, and the process then repeats. The difference between MARL and RL is that in MARL both the transitions and the rewards are functions of the joint action of the agents while in that state. Each agent ultimately tries to learn a policy that maps its observation to its own optimal action in that state.

In this paper we focus on cooperative continuous stochastic games, that is, multiagent reinforcement learning scenarios with continuous actions and an identical reward signal for all the agents. Traditionally such games may be categorized based on how much information each agent knows. If the agents are learning concurrently but do not know what the other agents did, then it is an *independent learner* setting. On the other hand, if the agents are learning concurrently and each agent is told what the other agents did, then we have a *joint action learner* setting. If we have a central controller to direct the learning process of each agent, then we have a *centralized training with decentralized execution* setting [23].

It has been shown that the independent learner setting can suffer from a pathology called *relative overgeneralization* [39]. It has also been shown that centralized training can suffer from the same problem [38]. However, while various methods [37, 38] have been applied to deal with the problem in simple games with small state or action spaces, it has not been solved for high dimensional or continuous stochastic games typical of real problems. In this paper we will remedy this.

Relative overgeneralization occurs when a suboptimal Nash Equilibrium in the joint space of actions is preferred over an optimal Nash Equilibrium because each agent's action in the suboptimal equilibrium is a better choice when matched with arbitrary explorative actions from the collaborating agents. For instance, consider a continuous game in Figure 1. The axes i and j are the various actions that agents A_i and A_j may perform, and the axis rewards(i, j) is the joint reward received by the agents from a given joint action $\langle i, j \rangle$. Joint action M has a higher reward than joint action N. However, the average of all possible rewards for action i_M, of agent A_i is lower than the average of all possible rewards for action i_N. Thus, agent A_i's policy may converge to i_N when the other agent is trying arbitrary actions to explore the space rather than focusing on the optimum. This pathology generally occurs when we use an average-based learner [37], that is, for some agent, when the joint Q-values averaged over all the actions of other agents are used during learning.

To overcome this problem, we apply Generative Adversarial Networks (GANs) [12] to cooperative stochastic games. GANs have been very successful in multiple domains [28, 41]. Recently, it has been shown that GANs have a strong connection with Imitation Learning [6], and that its variants, Generative Adversarial Imitation Learning (GAIL) [14] and Adversarial Inverse Reinforcement Learning (AIRL) [11] can be used to train agents to achieve high performance in sequential decision-making tasks with demonstration examples. In this work, we extend these methods to the multiagent cooperative setting and show that they can help to better coordinate the behaviors of the agents. We also show that the proposed methods may be thought of as applying the Multiagent Actor-Critic model [20] in a Maximum Entropy Reinforcement Learning (MERL) [13] framework. Here the critic is learned from demonstration examples and, if the right examples are given, the learned reward function can help the learners avoid relative overgeneralization.

2 Background

In this section, we first introduce Markov Decision Processes (MDPs) and their multiagent generalization. Then we discuss policy gradient methods and Imitation Learning (IL).

2.1 Markov Decision Processes and Stochastic Games

A Markov Decision Process (or MDP) can be used to model the interaction an agent has with the task environment. An MDP is a tuple $\{S, A, T, R, \gamma, H\}$ where S is the set of states; A is the set of actions available to the agent; T is the transition function $T(s, a, s') = P(s'|s, a)$ defining the probability of transitioning to state $s' \in S$ when in state $s \in S$ and taking action $a \in A$; R is the reward function $R : S \times A \mapsto \mathbb{R}$; $0 < \gamma < 1$ is a discount factor; and H is the horizon time of the MDP, that is, the number of timesteps the MDP runs.[1] An agent selects its actions based on the policy $\pi_\phi(a|s)$, which is a distribution over all possible actions a in state s parameterized by $\phi \in \mathbb{R}^n$.

The concept of an MDP can be extended to partially observable (POMDP) settings, where agents do not directly sense the state s. Rather, they receive some observation o sampled from a distribution conditioned on s. MDPs can also be generalized to a cooperative multiagent setting, called a *Cooperative Stochastic Game* or CSG. This is a game with n agents (or players), defined by the tuple $\{S, \mathcal{A}, R, T, \gamma, H\}$, where S is the state space, $\mathcal{A} = A^1 \times ... \times A^n$ is the joint action space of n agents, $R : S \times \mathcal{A} \to \mathbb{R}$ is the reward function for each agent i, and $T(s, \vec{a}, s') = P(s'|s, \vec{a})$ is the transition function, where $\vec{a} = \langle a^1 \cdots a^n \rangle \in \mathcal{A}$ is the joint action of all agents. Thus the reward the agents receive and the state to which they transition depends on the current state and agents' joint action. Each agent i determines its action using a policy π^i. We will also use $-i$ to denote all agents except for agent i. A Partially Observable Cooperative Stochastic Game, or POCSG, combines the CSG with a partially observable setting.

In the multiagent setting, a rational agent will play its *best response* to the other agents' strategy. If all agents are following a policy that implements this strategy, they will arrive at a Nash equilibrium, defined as a solution where $\forall i \; R^i(s, \pi^{*1}, \ldots, \pi^{*i}, \ldots, \pi^{*n}) \geq R^i(s, \pi^{*1}, \ldots, \pi^{*i-1}, \pi^i, \pi^{*i+1}, \ldots, \pi^{*n})$ for all of the strategies π^i available to agent i. π^{*i} denotes the best response policy of agent i.

2.2 Policy Gradient Methods

Policy Gradient methods are a key approach to learning in MDPs with continuous actions. Here, agents are trying to optimize the following objective function:

$$J(\phi) = \mathbb{E}_{P_\phi(\tau)}\left[R(\tau)\right] = \sum_\tau P_\phi(\tau) R(\tau), \tag{1}$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$ denotes a sequence of states and actions induced by policy and transition function. Here, we also overload notation R to represent the summation of all the reward along the trajectory τ. Various methods have been proposed to optimize this function: for example, likelihood-ratio trick [40], and the *Stochastic* (SPG) and *Deterministic Policy Gradient* (DPG) Theorems [35, 30]. These two policy gradient theorems are shown below respectively:

$$\begin{aligned}
\nabla_\phi J(\phi) &= \int_S \rho^{\pi_\phi}(s) \int_A \nabla_\phi \pi_\phi(a|s) Q^{\pi_\phi}(s, a) \; da \; ds \\
&= \mathbb{E}_{s \sim \rho^{\pi_\phi}, a \sim \pi_\phi}\left[\nabla_\phi \ln \pi_\phi(a|s) Q^{\pi_\phi}(s, a)\right] \\
\nabla_\phi J(\phi) &= \int_S \rho^{\pi_\phi}(s) \nabla_\phi \pi_\phi(s) \nabla_a Q^{\pi_\phi}(s, a)|_{a=\pi_\phi(s)} \; ds \\
&= \mathbb{E}_{s \sim \rho^{\pi_\phi}}\left[\nabla_\phi \pi_\phi(s) \nabla_a Q^{\pi_\phi}(s, a)|_{a=\pi_\phi(s)}\right],
\end{aligned}$$

where $\rho^{\pi_\phi}(s') = \int_S \sum_{t=1}^\infty \gamma^{t-1} P(s) P(s \to s', t, \pi_\phi) \; ds$ is the discounted distribution over states induced by policy π_ϕ starting from some state $s \in S$. Specifically, $P(s \to s', t, \pi)$ is the probability of going t steps under policy π_ϕ from state s and ending up in state s'. These theorems introduced a class of algorithms called *actor-critic* methods. The *actor* is the policy π and the *critic* is the Q-function.

[1]Any infinite horizon MDP with discounted rewards can be ϵ-approximated by a finite horizon MDP using a horizon $H_\epsilon = \frac{\log_\gamma(\epsilon(1-\gamma))}{\max_{s,a}|R(s,a)|}$ [16].

2.3 Inverse Reinforcement Learning and Imitation Learning

Inverse Reinforcement Learning (IRL) focuses on tasks where the reward function is not accessible to agents, and only a set of expert demonstration examples is available. The goal of IRL is to recover the reward function from the demonstrations. IRL has very close connection with Imitation Learning [1, 36, 15], where the recovered reward signal can be used to teach agents the expert behaviors. In this work, we focus on the MERL framework (or Soft Q-Learning) [13, 42], where the objective of the RL algorithm is augmented with the entropy of the policy,

$$\pi^*_{\text{MaxEnt}} = \text{argmax}_\pi \sum_t \mathbb{E}_{\tau\sim\rho_\pi}\left[R(s_t,a_t)+\mathcal{H}(\pi(\cdot|s_t))\right]$$

Under this framework, the optimal policy is given by $\pi^*(a|s) = \exp\{Q_{soft}(s,a) - V_{soft}(s)\}$, where $Q_{soft}(s,a)$ and $V_{soft}(s)$ are the *soft* value functions defined as following.

$$Q^*_{soft}(s_t,a_t) = R(s_t,a_t) + \mathbb{E}_{(s_{t+1},\ldots)\sim\rho_\pi}\left[\sum_{t'=t}^{\infty} R(s_{t'},a_{t'}) + \mathcal{H}(\pi(\cdot|s_{t'}))\right]$$

$$V^*_{soft}(s_t) = \log\int_A \exp Q^*_{soft}(s_t,a_t)\, da$$

With MERL, we can assume the demonstrations $\mathcal{D} = \{\tau_i\}$ given by the expert are sampled from the optimal policy mentioned above. When combined with IRL (forming MEIRL), the problem can be interpreted as solving the maximum likelihood problem

$$\max_\theta \mathbb{E}_{\tau\sim\mathcal{D}}[\log p_\theta(\tau)]$$

where $p(\tau) = \frac{1}{Z}p(s_0)\Pi_{t=0}^T p(s_{t+1}|s_t,a_t)e^{r_\theta(s_t,a_t)}$, and $p(s_0)$ and Z are the distributions of start states and the partition function respectively. One of the core tasks in MEIRL is to estimate or compute the partition function [7].

It has been shown [6] that the prior problem can be reformulated as a GAN problem where the discriminator has the special form

$$D_\theta(\tau) = \frac{\exp\{f_\theta(\tau)\}}{\exp\{f_\theta(\tau)\} + \pi(\tau)}$$

where $f_\theta(\tau)$ parameterized by θ serves as the reward function and $\pi(\tau)$ is the precomputed value using the policy of the agent. The training of this model involves updating both the policy and discriminator. We start training with some arbitrary policy and update the policy based on the reward $R(\tau) = \log(1 - D(\tau)) - \log D(\tau)$. This step can be thought of as *Guided Cost Learning* (or GCL), where we are adapting a sampling distribution to gather low variance samples for estimating the partition function [7]. Then we update the discriminator using both sampled trajectories and the experts' trajectories, which can be viewed as updating the reward function. This method, called GAN-GCL, has been further extend to AIRL, where the discriminator takes only state-action pairs as training samples instead of the whole trajectory, which in turn can greatly reduce the variance in estimation [11]. In this case, the discriminator takes the form

$$D_\theta(s,a) = \frac{\exp\{f_\theta(s,a)\}}{\exp\{f_\theta(s,a)\} + \pi(a|s)}$$

It can be shown that when training at optimal, $f_\theta(s,a)$ recovers the expert's advantage function, which is the optimal soft advantage function, such that $f_\theta(s,a) = A^*_{soft}(s,a) = Q^*_{soft}(s,a) - V^*_{soft}(s)$.

Aside from the work above, GANs have been directly applied to IL [14] where the discriminator takes a general form and implicitly learns the advantage function [17].

3 Related Work

The idea of learning to cooperate through policy gradient methods has been around for a long time, but mainly for discrete action domains [2]. Peshkin et al. have applied the likelihood-ratio method to both CSG and POCSG tasks. Nair et al. proposed *Joint Equilibrium-Based Search for Policies* (JESP), applied to POCSGs. The main idea here is to perform policy search in one agent while fixing the policies of other agents. Although this method is guaranteed to converge to a local Nash Equilibrium, it is essentially a round-robin single agent algorithm.

Deep MARL algorithms have recently been proposed to tackle more complex problems [24, 25]. One of the main thrust has been the centralized training with decentralized execution. Foerster et al. proposed a method to learn communication protocols between the agents, using inter-agent backpropagation and parameter sharing. Foerster et al. studied how to stabilize the training of multiagent deep reinforcement learning using importance sampling. Two actor-critic algorithms have been proposed in [10, 20]. They argue that by using a central critic one can ease the training of multiple agents, and that by keeping a separate policy the agent can execute with only its local information, which makes it possible to learn in POCSGs. Among these two algorithms, MADDPG [20] is most relevant to us. It uses the learning rule from DDPG [19] to learn a central critic, and uses the following gradient estimator to learn the policies for each agent i:

$$\nabla_{\phi^i} J(\phi^i) = \mathbb{E}_{s,a^{-i}\sim D}\left[\nabla_{\phi^i}\pi^i_{\phi^i}(a^i|o^i)\nabla_{a^i}Q(s,\vec{a})|_{a^i=\pi^i_{\phi^i}(o^i)}\right],$$

where ϕ^i is the agent i's policy parameters, D is the replay buffer, and o^i is the local observation of agent i. During centralized training the critic has access to the true state $s = [o^1,\ldots,o^n]$, which is the concatenation of all observations. But at execution time, each agent only has access to o^i.

There has been some work applying IL to cooperative tasks as well. Sullivan and Luke trained swarms, albeit in a supervised fashion, using learned Hierarchical Finite State Automata for foraging and patrolling tasks. Later, they extend the method to heterogeneous agents for box-pushing task [34]. Another work on applying IL to multiple agents is [4, 5], where the agents are exploring the environment and constantly consult a human coach for advice. A similar work to ours is [31], where GAIL is used for multiagent learning. The primary differences between our work and theirs is that we use only one discriminator while they use multiple ones, and we establish a clear connection between the multiagent actor-critic model and multiagent imitation learning. Other work worth mentioned here is [32] and [22], where IRL is used to train a large number of agents for coordination tasks.

4 Multiagent Adversarial Inverse Reinforcement Learning

We now consider how to apply AIRL or GAIL in multiagent scenarios with cooperative games. First consider the situation where a coach wants to teach a coordination strategy to players. He may start with some demonstration, then let the players practice following the demonstration while continuing to give advice to the agents during practice. This is much like the training process of AIRL or GAIL with multiple agents, where we can think of a discriminator as a coach, and all the other agents as players on the team. During practice, although each player can only sense the environment through his own local observations, the coach usually has much more information either through his expertise or due to a global view of the game state during practice. Thus, if we want to apply AIRL and GAIL to cooperative games, we can make a simple modification to both algorithms by letting each agent have only a local observation o^i, and give the discriminator access to the full state information s, the same as the critic in MADDPG. We call these modified algorithms Multiagent AIRL (MAIRL) and Multiagent GAIL (MGAIL).

Next, we demonstrate the connection between multiagent actor-critic model with our proposed methods. We are not the first to connect actor-critic model with GAN methods [27]. But to our knowledge, we are the first to connect the actor-critic model with AIRL and GAIL using an analytical method. Recall that in AIRL we have the discriminator taking the following form

$$D_\theta(s,a) = \frac{\exp\{f_\theta(s,a)\}}{\exp\{f_\theta(s,a)\} + \pi(a|s)}$$

On the other hand, in GAIL, the form of the discriminator is not specified. However, it has been shown that the discriminator is learning a function of the following form (see Equation (6) in [3])

$$D_\theta(s,a) = \frac{p_E(s,a)}{p_E(s,a) + \pi(s,a)}$$

where $p_E(s,a)$ and $\pi(s,a)$ are the joint distribution of state and action under expert's policy and learner's policy respectively. When we assume the state marginal distribution is same for both expert and learner, that is $p_E(s) = p_\pi(s)$, GAIL is doing the same works as AIRL. To see that,

$$\begin{aligned} D_\theta(s,a) = \frac{p_E(s,a)}{p_E(s,a) + \pi(s,a)} &= \frac{p_E(a|s)p_E(s)}{p_E(a|s)p_E(s) + \pi(a|s)p_\pi(s)} \\ = \frac{p_E(a|s)}{p_E(a|s) + \pi(a|s)} &= \frac{\exp\{f_\theta(s,a)\}}{\exp\{f_\theta(s,a)\} + \pi(a|s)} \end{aligned}$$

A justification of this transformation can be found in Appendix (A.2) in [11]. In this paper, we focus on the discussion of MAIRL. However, when the marginal distribution of state is the same for expert and learners, the theoretical conclusion of MAIRL also applies to MGAIL. This usually happens in goal-achieving tasks, where the games will not terminate until certain goals are achieved.

In AIRL, $\pi(a|s)$ is the sampling distribution which we constantly update to acquire better samples for estimating Z. To update the sampling policy $\pi(a|s)$, we can apply policy gradient methods, optimizing with respect to the reward function

$$R(s,a) = \log(D_\theta(s,a)) - \log(1 - D_\theta(s,a)) = f_\theta(s,a) - \log\pi(a|s)$$

To do a similar thing in the multiagent case we have each agent update its own policy $\pi^i(a^i|o^i)$ to get better estimation samples. The policy for each agent takes its local observation and outputs its own action. Thus, we have the following MAIRL discriminator:

$$D_\theta(s,\vec{a}) = \frac{\exp\{f_\theta(s,\vec{a})\}}{\exp\{f_\theta(s,\vec{a})\} + \pi^1(a^1|o^1)\dots\pi^n(a^n|o^n)}$$

We follow MADDPG [20] and assume that the function f has access to the true state $s = [o^1, \dots, o^n]$. With this revised discriminator, we now have the reward function

$$\begin{aligned} R(s,\vec{a}) &= \log(D_\theta(s,\vec{a})) - \log(1 - D_\theta(s,\vec{a})) \\ &= \log\frac{\exp\{f_\theta(s,\vec{a})\}}{\exp\{f_\theta(s,\vec{a})\} + \pi^i(a^i|o^i)\pi^{-i}(a^{-i}|o^{-i})} \\ &\quad - \log\frac{\pi^i(a^i|o^i)\pi^{-i}(a^{-i}|o^{-i})}{\exp\{f_\theta(s,\vec{a})\} + \pi^i(a^i|o^i)\pi^{-i}(a^{-i}|o^{-i})} \\ &= f_\theta(s,\vec{a}) - \log\pi^i(a^i|o^i) - \log\pi^{-i}(a^{-i}|o^{-i}) \end{aligned}$$

Based on this new reward function, we show the connection between MAIRL and MADDPG, as both methods are composed of a central critic that has access to the underlying state and joint actions of all the agents and multiple actors for which only local observation is available. The difference is that MAIRL does not have access to the true reward signals, but only to a set of expert demonstrations, and that our algorithm works in a MERL framework.

First, we notice that $f_\theta(s, \vec{a})$ can be treated as a central advantage function in a Maximum Entropy framework. Specifically, as we mentioned in the previous section, $f_\theta(s, a)$ recovers the optimal advantage function $A^*_{soft}(s, a)$ when the discriminator reaches the global minimum. Thus, we can view $f_\theta(s, a)$ as the approximation of $A^*_{soft}(s, a)$ during the training process of discriminator. Now the reward function for each agent to optimize can be written as

$$R^i(s, \vec{a}) = A_\theta(s, \vec{a}) - \log \pi^i(a^i|o^i) - \log \pi^{-i}(a^{-i}|o^{-i})$$

If we sum this over entire trajectories, then we get policy objective for agent i

$$\begin{aligned}&\mathbb{E}_{\tau\sim p(\tau)}\Big[\sum_{t=0}^{T} R^i(s_t, \vec{a}_t)\Big]\\ =&\mathbb{E}_{\tau\sim p(\tau)}\Big[\sum_{t=0}^{T} A_\theta(s_t, \vec{a}_t) - \log \pi^i(a^i_t|o^i_t) - \log \pi^{-i}(a^{-i}_t|o^{-i}_t)\Big]\end{aligned}$$

Let $p_t(s_t, \vec{a}_t) = \int_{s_{t'\neq t}, \vec{a}_{t'\neq t}} p(\tau)$ denote the state-action marginal distribution at time t. Rewriting the above equation, we have

$$\begin{aligned}&\sum_{t=0}^{T}\mathbb{E}_{s_t, \vec{a}_t\sim p_t(s_t, \vec{a}_t)}\Big[A_\theta(s_t, \vec{a}_t) - \log \pi^i(a^i_t|o^i_t) - \log \pi^{-i}(a^{-i}_t|o^{-i}_t)\Big]\\
=&\sum_{t=0}^{T}\mathbb{E}_{s_t\sim p_t(s_t), a^i_t\sim\pi^i(a^i_t|o^i_t), a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}\Big[A_\theta(s_t, \vec{a}_t)\\
&-\log \pi^i(a^i_t|o^i_t) - \log \pi^{-i}(a^{-i}_t|o^{-i}_t)\Big]\\
=&\sum_{t=0}^{T}\mathbb{E}_{s_t\sim p_t(s_t), a^i_t\sim\pi^i(a^i_t|o^i_t)}\Big[-\log \pi^i(a^i_t|o^i_t)\\
&+\mathbb{E}_{a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}\big[A_\theta(s_t, \vec{a}_t) - \log \pi^{-i}(a^{-i}_t|o^{-i}_t)\big]\Big]\\
=&\sum_{t=0}^{T}\mathbb{E}_{s_t\sim p_t(s_t), a^i_t\sim\pi^i(a^i_t|o^i_t)}\Big[-\log \pi^i(a^i_t|o^i_t) + \overline{A}_\theta(s_t, a^i_t)\\
&+\mathcal{H}(\pi^{-i}(\cdot|o^{-i}_t))\Big]\end{aligned}$$

where we denote $\overline{A}_\theta(s_t, a^i_t)$ as the central advantage function averaged over samples from other agents' policies. We further notice that the term $\mathcal{H}(\pi^{-i}(\cdot|o^{-i}_t))$ does not depend on agent i's policy, and thus the objective becomes

$$\begin{aligned}&\sum_{t=0}^{T}\mathbb{E}_{s_t\sim p_t(s_t), a^i_t\sim\pi^i(a^i_t|o^i_t)}\Big[-\log \pi^i(a^i_t|o^i_t) + \overline{A}_\theta(s_t, a^i_t)\Big]\\
=&\sum_{t=0}^{T}\mathbb{E}_{s_t\sim p_t(s_t)}\Big[-\mathbb{E}_{a^i_t\sim\pi^i(a^i_t|o^i_t)}\big[\log \pi^i(a^i_t|o^i_t) - \log\exp\{\overline{A}_\theta(s_t, a^i_t)\}\big]\Big]\\
=&\sum_{t=0}^{T}\mathbb{E}_{s_t\sim p_t(s_t)}\Big[-\mathbb{E}_{a^i_t\sim\pi^i(a^i_t|o^i_t)}\big[\log\frac{\pi^i(a^i_t|o^i_t)}{\exp\{\overline{A}_\theta(s_t, a^i_t)\}}\big]\Big]\end{aligned}$$

By minimizing the objective instead of maximizing, we then get:

$$\sum_{t=0}^{T}\mathbb{E}_{s_t\sim p_t(s_t)}\Big[\mathrm{D_{KL}}\big[\pi^i(\cdot|o^i_t)||\exp\{\overline{A}_\theta(s_t, a^i_t)\}\big]\Big] \qquad (2)$$

Next, we show this objective is the same objective of applying MADDPG style actor-critic architecture to MERL, specifically, Soft Q-Learning. We first write down agent's policy objective of Soft Q-Learning (see Equation (12) in [13]), which is curiously similar:

$$J(\phi) = \sum_{t=0}^{T}\mathbb{E}_{s_t\sim p_t(s_t)}\Big[\mathrm{D_{KL}}\big[\pi_\phi(\cdot|s_t)||\exp\{Q_{soft}(s_t, \cdot) - V_{soft}(s_t)\}\big]\Big] \qquad (3)$$

Then we consider how to apply the idea of MADDPG to MERL. In MADDPG, for each agent i, its policy gradient estimator is

$$\nabla_{\phi^i} J(\phi^i) = \mathbb{E}_{s, a^{-i}\sim D}\Big[\nabla_{\phi^i}\pi^i_{\phi^i}(o^i)\nabla_{a^i}Q(s, \vec{a})|_{a^i=\pi^i_{\phi_i}(o^i)}\Big].$$

If we are dealing with a stochastic policy, then

$$\begin{aligned}\nabla_{\phi^i} J(\phi^i) =&\mathbb{E}_{s, a^{-i}\sim D}\Big[\mathbb{E}_{a^i\sim D}\big[\nabla_{\phi^i}\log\pi^i_{\phi^i}(a^i|o^i)Q(s, \vec{a})\big]\Big]\\
=&\mathbb{E}_{s, \vec{a}\sim D}\Big[\nabla_{\phi^i}\log\pi^i_{\phi^i}(a^i|o^i)Q(s, \vec{a})\Big].\end{aligned}$$

The difference is that for stochastic policies, we need to sum over all the actions a^i sampled from the policies, and the likelihood-ratio trick must be used. Then we can rewrite the above estimator as

$$\begin{aligned}\nabla_{\phi^i} J(\phi^i) &= \mathbb{E}_{s, \vec{a}\sim D}\Big[\nabla_{\phi^i}\log\pi^i_{\phi^i}(a^i|o^i)Q(s, \vec{a})\Big]\\
&= \mathbb{E}_{s, a^i\sim D}\Big[\nabla_{\phi^i}\log\pi^i_{\phi^i}(a^i|o^i)\mathbb{E}_{a^{-i}\sim D}\big[Q(s, \vec{a})\big]\Big]\\
&= \mathbb{E}_{s, a^i\sim D}\Big[\nabla_{\phi^i}\log\pi^i_{\phi^i}(a^i|o^i)\overline{Q}(s, a^i)\Big]\end{aligned}$$

where $\overline{Q}(s, a^i)$ is the central Q-function averaged over the samples of the other agents' actions from the replay buffer (a proof of this estimator can be found in [38]). If we want to combine the idea of MADDPG with Equation (3), a simple way is to replace the $Q_{soft}(s, \cdot)$ with $\overline{Q}_{soft}(s, a^i) = \mathbb{E}_{a^{-i}\sim D}\big[Q_{soft}(s, \vec{a})\big]$ in Equation (3) for agent i and change the state s in the policy to the agents' local observation o^i, then we have the following objective for agent i' s policy,

$$\begin{aligned}J(\phi^i) &= \mathrm{D_{KL}}\big(\pi(\cdot|o^i)||\exp\{\overline{Q}_{soft}(s, a^i) - V_{soft}(s)\}\big)\\
&= \mathrm{D_{KL}}\big(\pi(\cdot|o^i)||\exp\{\mathbb{E}_{s, a^{-i}\sim D}[Q_{soft}(s, \vec{a})] - V_{soft}(s)\}\big)\\
&= \mathrm{D_{KL}}\big(\pi(\cdot|o^i)||\exp\{\mathbb{E}_{s, a^{-i}\sim D}[Q_{soft}(s, \vec{a}) - V_{soft}(s)]\}\big)\\
&= \mathrm{D_{KL}}\big(\pi(\cdot|o^i)||\exp\{\mathbb{E}_{s, a^{-i}\sim D}[A_{soft}(s, \vec{a})]\}\big)\\
&= \mathrm{D_{KL}}\big(\pi(\cdot|o^i)||\exp\{\overline{A}_{soft}(s, a^i)\}\big)\end{aligned}$$

where $\overline{A}_{soft}(s, a^i)$ can be thought of as the joint soft advantage function averaged by the samples of the other agents' policies from the replay buffer. This objective matches the inner KL-divergence term in Equation (2). The only difference is that in Equation (2) the estimator is using the on-policy samples and MADDPG uses the off-policy samples. This is because MADDPG uses off-policy learning for optimal advantage functions, while MAIRL learns the optimal advantage function through demonstration and thus can avoid using off-policy samples and a replay buffer.

5 Learning with coordination examples

In this section, we explore whether our proposed method can potentially suffer from the relative overgeneralization problem and if so, is there a way to avoid it? To answer the first question, we first consider the inner KL-divergence term in Equation (2) as only this term involves the actions from other agents.

$$\begin{aligned}
&\nabla_\phi \mathrm{D}_{\mathrm{KL}}\left[\pi^i_\phi(\cdot|o^i_t)||\exp\{\overline{A}_\theta(s_t,a^i_t)\}\right]\\
=&\nabla_\phi \mathbb{E}_{a^i_t\sim\pi^i_\phi(a^i_t|o^i_t)}\left[\log\pi^i_\phi(a^i_t|o^i_t)-\overline{A}_\theta(s_t,a^i_t)\right]\\
=&\nabla_\phi \mathbb{E}_{a^i_t\sim\pi^i_\phi(a^i_t|o^i_t)}\left[\log\pi^i_\phi(a^i_t|o^i_t)\right]-\mathbb{E}_{a^i_t\sim\pi^i_\phi(a^i_t|o^i_t)}\left[\overline{A}_\theta(s_t,a^i_t)\right]\\
=&\nabla_\phi \mathcal{H}(\pi^i_\phi(\cdot|o^i_t))-\nabla_\phi\int_{a^i}\pi^i_\phi(a^i_t|o^i_t)\overline{A}_\theta(s_t,a^i_t)da^i
\end{aligned}$$

By using the likelihood-ratio trick and using the definition of the advantage function, the second part becomes

$$\begin{aligned}
&\nabla_\phi\int_{a^i}\pi^i_\phi(a^i_t|o^i_t)\overline{A}_\theta(s_t,a^i_t)da^i\\
=&\nabla_\phi \mathbb{E}_{a^i_t\sim\pi^i(a^i_t|o^i_t)}\left[\mathbb{E}_{a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}\left[Q_\theta(s_t,\vec{a}_t)-V_\theta(s_t)\right]\right]\\
=&\nabla_\phi \mathbb{E}_{a^i_t\sim\pi^i(a^i_t|o^i_t)}\left[\mathbb{E}_{a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}\left[Q_\theta(s_t,\vec{a}_t)\right]\right]-\nabla_\phi V_\theta(s_t)\\
=&\mathbb{E}_{a^i_t\sim\pi^i(a^i_t|o^i_t)}\left[\nabla_\phi\log\pi^i_\phi(a^i_t|o^i_t)\mathbb{E}_{a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}\left[Q_\theta(s_t,\vec{a}_t)\right]\right]\\
=&\mathbb{E}_{a^i_t\sim\pi^i(a^i_t|o^i_t)}\left[\nabla_\phi\log\pi^i_\phi(a^i_t|o^i_t)\overline{Q}_\theta(s_t,a^i_t)\right]
\end{aligned}$$

This is the regular policy gradient [35] with averaged $Q_{soft}(s,a)$. Since an average-based learner can suffer from relative overgeneralization, MAIRL will have the same issue. To fix this, we notice that the difficulty comes from the Q-function. Suppose agent i has two actions, a and b, in state s. When the other agents are playing their best response policies π^{*-i}, and $Q(s,a,\pi^{*-i}) > Q(s,b,\pi^{*-i})$, then agent i ought to prefer a over b. However, in an average-based learner setting, it is possible that b is preferred over a when $\overline{Q}(s,b) > \overline{Q}(s,a)$ where $\overline{Q}$ is some averaged Q-function. Thus, a simple way to avoid relative overgeneralization is to make sure the rank ordering between the actions is always maintained.

Definition 1. *A Q-function is relative overgeneralization free if for any agent i and any of its two actions a^i and b^i, we always have $\mathbb{E}_{a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}[Q(s,a^i,a^{-i})] > \mathbb{E}_{a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}[Q(s,b^i,a^{-i})]$ given that $Q(s,a^i,a^{*-i}) > Q(s,b^i,a^{*-i})$, where $\pi^{-i}(a^{-i}_t|o^{-i}_t)$ is some arbitrary policy of all the other agents, and a^{*-i} is the best response action of all the other agents.*

Note that a Q-function can always be decomposed into an advantage function and a state value function, and the state value function does not have the action involved. Thus, we are really looking for an advantage function that is relative overgeneralization free. A natural idea would be a quadratic function with a block diagonal matrix.

Proposition 1. *Suppose $A(s,\vec{a}) = -\frac{1}{2}(\vec{a}-\mu(s))^T M(s)(\vec{a}-\mu(s))$ is the advantage function for some state s and joint actions $\vec{a}$ of all n agents, where M(s) is a positive definite block diagonal matrix with n blocks with each block of size d_i by d_i, namely, the action dimension of agent i. Then the corresponding state-action value function $Q(s,\vec{a})$ is relative overgeneralization free.*

Proof. We partition the joint action $\vec{a}$ into two disjoint subvectors. Without loss of generality, we take a^i to be the first d_i components of $\vec{a}$ and a^{-i} be the rest of the components of $\vec{a}$, that is,

$$\vec{a}=\begin{pmatrix}a^i\\a^{-i}\end{pmatrix}$$

Correspondingly, we can partition the vector $\mu(s)$ and matrix $M(s)$. For simplicity, we refer to them as μ and M. Then,

$$\mu=\begin{pmatrix}\mu^i\\\mu^{-i}\end{pmatrix}, M=\begin{pmatrix}M_{11}&M_{12}\\M_{21}&M_{22}\end{pmatrix},$$

Since M is block diagonal, $M_{12}=M_{21}=0$. Now suppose we take the exponential of the advantage function, such that

$$p(\vec{a})=\exp\left\{-\frac{1}{2}(\vec{a}-\mu)^T M(\vec{a}-\mu)\right\}.$$

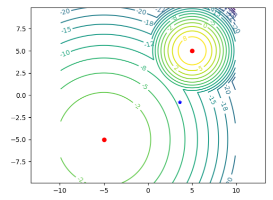
Then we can see that $p(\vec{a})$ is a multivariate Gaussian distribution. If we assume a^{-i} is given and M is a diagonal matrix as described above, by using the independence property of multivariate Gaussian distribution, the resulting distribution of a^i is also a Gaussian distribution with μ^i and M_{11} as the mean and covariance matrix. The inner quadratic form of this Gaussian is

$$A(s,a^i)=-\frac{1}{2}(a^i-\mu^i)^T M_{11}(a^i-\mu^i)$$

The resulting value function does not depend on a^{-i}, so if $\exists a^{-i}\ A(s,a^i,a^{-i}) > A(s,b^i,a^{-i})$, then we have $\forall a^{-i}\ A(s,a^i,a^{-i}) > A(s,b^i,a^{-i})$. Now, since

$$\begin{aligned}
\overline{Q}(s,a^i)-\overline{Q}(s,b^i)&=\mathbb{E}_{a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}\left[Q(s,a^i,a^{-i})-Q(s,b^i,a^{-i})\right]\\
&=\mathbb{E}_{a^{-i}_t\sim\pi^{-i}(a^{-i}_t|o^{-i}_t)}\left[A(s,a^i,a^{-i})-A(s,b^i,a^{-i})\right].
\end{aligned}$$

The inner term is always positive, so $\overline{Q}(s,a^i) > \overline{Q}(s,b^i)$ □


(a) The *Max of Two Quadratics* game. Red dots mark the two NEs in the joint action space. The blue star marks the joint action of the two agents. The contour shows the reward level.

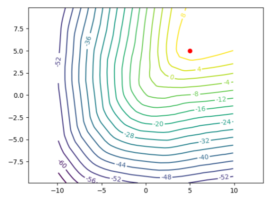

(b) Reward recovered by Multiagent AIRL. The optimal NE in the original game is marked with red dot. In the recovered reward function, only one NE is maintained.

Figure 2: Max of Two Quadratics Game and recovered reward

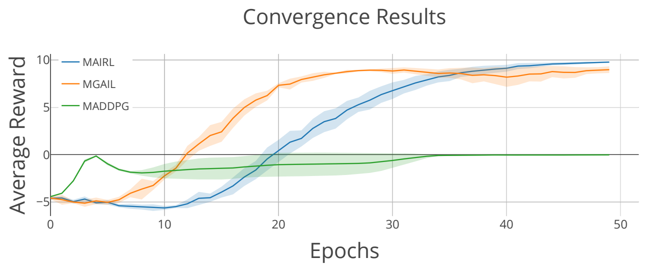


Figure 3: Average reward of three algorithms on Max of Two Quadratics domain. MADDPG converge to the suboptimal NE due to relative overgeneralization.

Recall that when trained at optimal, $f_\theta(s,a)$ in the discriminator recovers the advantage function of the expert. Thus, if we require the expert's advantage function to satisfy Proposition 1, then we just need to guarantee that the expert's policy is a Gaussian policy on the joint-action space with a block diagonal matrix, which is not a hard requirement to fulfill. A closely related work employs a reward function that is learned using Laplacian approximation, thus the transition model has to be known [18].

6 Experiments

In this section, we consider three domains to test our algorithms. The first domain is a repeated game from [38], while the other two domains are sequential decision making tasks which are carefully designed to demonstrate the challenge of cooperation in simple games. With these experiments, we show that by using IL we can greatly reduce the difficulty of learning cooperative tasks, and especially the ones caused by relative overgeneralization.

For all our experiments, we used a two-layer ReLU network with 64 units for the discriminator and a two-layer Tanh network with 64 units for the Gaussian policy. The batch size of Trust Region Policy Optimization (TRPO) algorithm [29] and the number of steps for every epoch of MADDPG was set to 10000. The entropy regularizer for MAIRL in the first two domains (Max of Two Quadratics and Marching) was 0.01 and was 0.1 for the last domain (Narrow). The entropy regularizer for MGAIL was always 0.0. For the final results in Table 1, we verified statistical significance ANOVA with a Tukey post-hoc test at $p = 0.05$.

6.1 Repeated Game Experiment

The first game was *Max of Two Quadratics*. This is a simple single state continuous game for two agents with one action dimension per agent. Each agent has a bounded action space. The reward for a joint action is given by the equation

$$f_1 = h_1 \times \left[-((a_1 - x_1)/s_1)^2 - ((a_2 - y_1)/s_1)^2\right]$$
$$f_2 = h_2 \times \left[-((a_1 - x_2)/s_2)^2 - ((a_2 - y_2)/s_2)^2\right] + c$$
$$r(a_1, a_2) = \max(f_1, f_2)$$

where a_1 and a_2 are the actions of agent 1 and agent 2 respectively. In the above equation, $h_1 = 0.8, h_2 = 1, s_1 = 3, s_2 = 1, x_1 = 5, x_2 = 5, y_1 = -5, y_2 = -5, c = 10$ are the coefficients to specify the reward surface (see Figure 2a). Although the formulation of the game is rather simple, it poses a major difficulty to gradient-based algorithms as, over almost all the joint-action space, the gradient points towards the sub-optimal solution at (-5, -5). We applied MAIRL, MGAIL, and MADDPG in this setting. The demonstration sample was collected using a Gaussian distribution centered at the optimal Nash Equilibrium. The convergence results are shown in Figure 3 and Table 1. As can be seen, MADDPG generally converged to a suboptimal Nash Equilibrium, while MAIRL and MGAIL made use of the demonstration and converged to the optimal Nash Equilibrium.

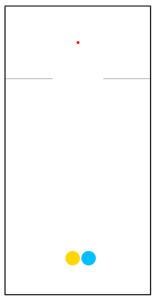
6.2 Stochastic Game Experiments

To test the performance of our algorithms on stochastic games we designed the games *Marching* and *Narrow*, shown in Figure 4a and Figure 4b. In each game there are two agents, each with a radius of 0.05. Both games terminate after 200 steps. In Marching the agents must march towards the red dot. A shaped reward is provided for the two agents based on the distance between the agent's center point and the red dot. The agents receive a large penalty (-10) if they collide or are too far from each other (if the distances between the centers is $\geq$0.11). They will receive a large reward (10) if they manage to reach the red dot. The purpose of this game is to test whether the agents can coordinate their moving speeds:

	MAIRL		MGAIL		MADDPG		TRPO	
Max of Two Quadratics	**9.78**	**±0.07**	**8.98**	**±0.35**	-0.03	±0.02		
Marching	**-152.88**	**±129.76**	**-80.88**	**±52.08**	**-178.36**	**±89.46**	-324.58	±111.88
Narrow (u, 0.2)	**5.99**	**±0.84**	**5.49**	**±1.79**	-2.00	±0.00	-2.00	±0.00
Narrow (u, 0.205)	**4.53**	**±2.51**	**5.54**	**±2.15**	-2.00	±0.00	-2.08	±0.16
Narrow (sh, 0.2)	**-4.40**	**±0.85**	**-4.94**	**±1.80**	-23.46	±6.10	-79.21	±6.96
Narrow (sh, 0.205)	**-6.76**	**±3.78**	**-5.23**	**±2.13**	**-39.98**	**±37.06**	-79.34	±4.54
Narrow (s, 0.2)	**5.95**	**±0.83**	**5.44**	**±1.78**	-2.80	±1.32	-2.63	±0.14
Narrow (s, 0.205)	**4.50**	**±2.51**	**5.50**	**±2.15**	-3.68	±1.04	-2.66	±0.17

u=*unshaped reward* sh=*shaped reward* s=*scaled shaped reward*
Number in the parentheses marks the width of the field.

Table 1: Convergence results of algorithms by domain, showing the mean ± standard error over 5 trials. The best-performing algorithms on each task are shown in boldface.


(a) Marching domain. Agents need to march through the exit.

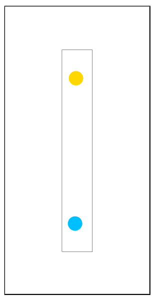

(b) Narrow domain. The valid moving area is within the inside black box.

Figure 4: Sequential Decision Making Tasks

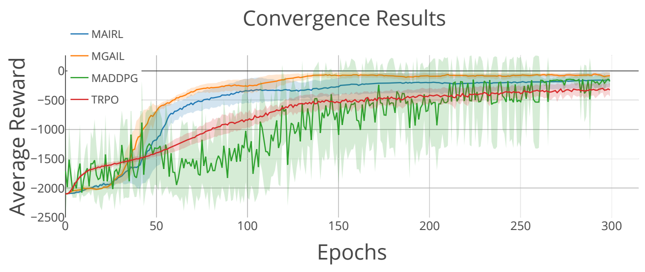


Figure 5: Average reward of four algorithms on Marching domain. MGAIL and MAIRL converge faster and have smaller variance compared to MADDPG.

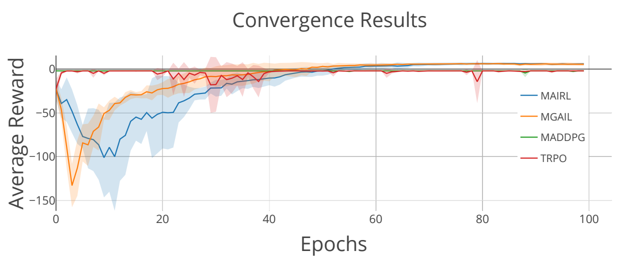
at each step, they need to try to get closer to the target while maintaining formation. In this game, if one of the agents moves more aggressively during the learning, it could then cause a penalty for both agents. As a consequence, if both agents want to seek the safe option — to stay still — then we have relative overgeneralization.

The other game, Narrow, requires the two agents switch their positions. They start at two different ends in an aisle, and each must pass the other to reach the start position of the other agent. The aisle is very narrow and requires careful coordination between two agents, because when they collide they receive a large penalty (-10). If both agents manage to reach their goal positions, they both receive a large reward (10). We considered several variations of this game.

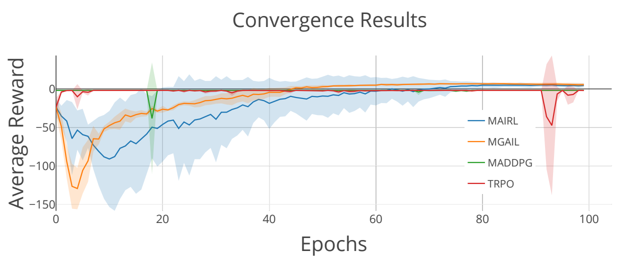
For both tasks we also included an independent learner where the two agents ran TRPO in parallel. The demonstrations for MGAIL and MAIRL were collected using a joint action DDPG learner.

Figure 5 and Table 1 show the results of the four algorithms in the Marching domain. Among them, MGAIL performed the best, while the convergence result of MAIRL was similar to MADDPG. However, during learning, we can see that MADDPG had a much higher variance than MGAIL and MAIRL and converged slower. We think this is because the reward function given to MADDPG and also to TRPO lacked useful coordination information. Thus, although MADDPG and TRPO learned to march to the exit and to avoid relative overgeneralization, they had to make many collisions during movement. On the contrary, MGAIL and MAIRL used the demonstration and learned how to avoid collisions while moving.

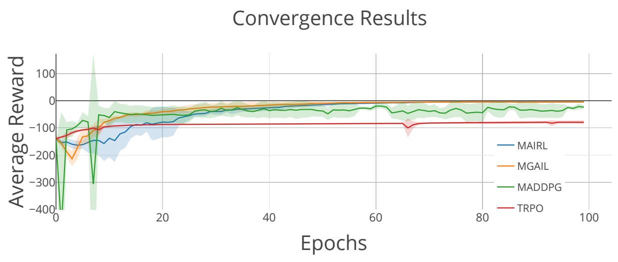
In the Narrow domain, we tested the four algorithms on three different reward functions, namely: an *unshaped reward* where agents received -0.01 for every step until the episode ended or all reached the target (and so the accumulated reward for simply wandering around was -2, which was better than the collision penalty); a *shaped reward* that was the same as in the Marching domain (to encourage each agent to reach the other end); and



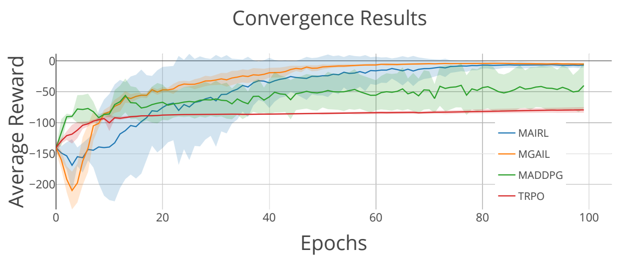
(a) Unshaped reward with width = **0.200**



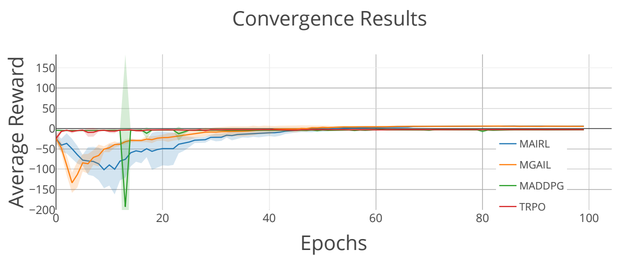
(b) Unshaped reward with width = **0.205**



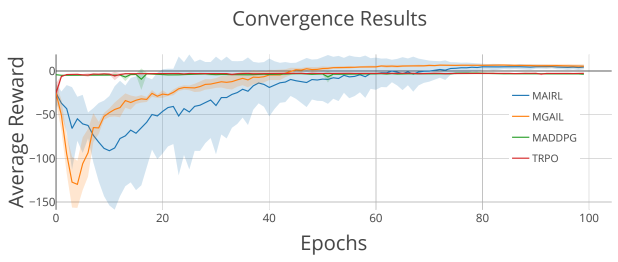
(c) Shaped reward with width = **0.200**



(d) Shaped reward with width = **0.205**



(e) Scaled reward with width = **0.200**



(f) Scaled reward with width = **0.205**

Figure 6: Convergence results of Four algorithms on Narrow domains.

a *scaled shaped reward* to provide dense reward signals but to also make sure the accumulated reward was better than the collision penalty. Here, if agents got close to each other they could collide, and as a consequence, the agents might choose to simply stay at where they started, resulting in relative overgeneralization. Furthermore, the final large reward was only provided if both agents reached the target. To pass each other in the first stage, the agents might try too hard then overshoot into a corner of the environment, and thus fail to discover the final large reward.

In addition to changing the reward function, we also varied the width of the Narrow domain. In the basic setting, the width was 0.2, just wide enough to allow the two agents to pass each other. We then relaxed this to 0.205. The results of the Narrow task experiments are shown Figure 6.

We found that, in all cases, MGAIL and MAIRL agents successfully passed each other and reached the target. However, in the unshaped reward setting, both MADDPG and TRPO agents failed to learn to pass one another. This is not a surprising result, as the coordination is extremely difficult and both algorithms fail to find it. However, our imitation learning agents used an advantage function as reward, and $A_{soft}(s,a) = \mathbb{E}_{s'}[R(s,a) + V_{soft}(s') - V_{soft}(s)]$, which can be thought of as an original reward function shaped by a soft value function. Thus, our imitation learners were receiving a better reward signal for exploration.

Then in the shaped reward setting we found that MADDPG agents managed to pass one another but failed to reach the target jointly, and TRPO agents could not pass each other at all. This suggests that shaped reward helped the exploration of these agents and reduced the effect of relative overgeneralization for MADDPG. The performance difference was significant with a width of 0.2, but not 0.205, which further suggested that relative overgeneralization was the cause of this difference.

Finally, we gave MADDPG and TRPO agents a scaled shaped reward. Since the reward for collision was worse, the algorithms were caught by relative overgeneralization and failed to learn the passing behavior. The difference between our algorithms and MADDPG was significant for both width settings.

7 Conclusion and Future Work

In this paper we proposed two methods to achieve better coordination in cooperative continuous games based on imitation learning. We showed that these methods have a close connection with multiagent actor-critic model. We also investigated the reasons why these methods avoided relative overgeneralization in cooperative games. A drawback of our approach is that the input space of our discriminator can grow linearly with the number of agents. A possible solution is to use hierarchal methods to decompose the task and learning spaces. We will investigate this direction in future work.

References

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 1.

[2] Bikramjit Banerjee and Jing Peng. 2003. Adaptive Policy Gradient in Multiagent Learning. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*. 686–692.

[3] Nir Baram, Oron Anschel, Itai Caspi, and Shie Mannor. 2017. End-to-End Differentiable Adversarial Imitation Learning. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. 390–399.

[4] Sonia Chernova and Manuela Veloso. 2008. Teaching collaborative multi-robot tasks through demonstration. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*. IEEE, 385–390.

[5] Sonia Chernova and Manuela Veloso. 2010. Confidence-based multi-robot learning from demonstration. *International Journal of Social Robotics* 2, 2 (2010), 195–215.

[6] Chelsea Finn, Paul F. Christiano, Pieter Abbeel, and Sergey Levine. 2016. A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models. (2016). arXiv:1611.03852

[7] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*. 49–58.

[8] Jakob Foerster, Yannis M Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*. 2137–2145.

[9] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip H. S. Torr, Pushmeet Kohli, and Shimon Whiteson. 2017. Stabilising Experience Replay for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. 1146–1155.

[10] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *AAAI*. 2974–2982.

[11] Justin Fu, Katie Luo, and Sergey Levine. 2018. Learning Robust Rewards with Adverserial Inverse Reinforcement Learning. In *International Conference on Learning Representations*.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[13] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. 2017. Reinforcement Learning with Deep Energy-Based Policies. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. 1352–1361.

[14] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 4565–4573.

[15] Jonathan Ho, Jayesh Gupta, and Stefano Ermon. 2016. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*. 2760–2769.

[16] Tang Jie and Pieter Abbeel. 2010. On a connection between importance sampling and the likelihood ratio policy gradient. In *Advances in Neural Information Processing Systems*. 1000–1008.

[17] Sergey Levine. 2018. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. *CoRR* abs/1805.00909 (2018).

[18] Sergey Levine and Vladlen Koltun. 2012. Continuous Inverse Optimal Control with Locally Optimal Examples. In *ICML '12: Proceedings of the 29th International Conference on Machine Learning*.

[19] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2015).

[20] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*. 6382–6393.

[21] Ranjit Nair, Milind Tambe, Makoto Yokoo, David Pynadath, and Stacy Marsella. 2003. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*. 705–711.

[22] Sriraam Natarajan, Gautam Kunapuli, Kshitij Judah, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. 2010. Multi-agent inverse reinforcement learning. In *Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on*. IEEE, 395–400.

[23] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32 (2008), 289–353.

[24] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. 2017. Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, International Convention Centre, Sydney, Australia, 2681–2690.

[25] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani. 2018. Lenient Multi-Agent Deep Reinforcement Learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 443–451.

[26] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. 2000. Learning to cooperate via policy search. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 489–496.

[27] David Pfau and Oriol Vinyals. 2016. Connecting Generative Adversarial Networks and Actor-Critic Methods. *CoRR* abs/1610.01945 (2016).

[28] Alec Radford, Luke Metz, and Soumith Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *CoRR* abs/1511.06434 (2015).

[29] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

[30] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *ICML*.

[31] Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. 2018. Multi-Agent Generative Adversarial Imitation Learning. In *ICLR Workshop*.

[32] Adrian Šošić, Wasiur R KhudaBukhsh, Abdelhak M Zoubir, and Heinz Koeppl. 2017. Inverse reinforcement learning in swarm systems. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1413–1421.

[33] Keith Sullivan and Sean Luke. 2012. Learning from Demonstration with Swarm Hierarchies. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[34] Keith Sullivan, Ermo Wei, Bill Squires, Drew Wicke, and Sean Luke. 2015. Training Heterogeneous Teams of Robots. In *Autonomous Robots and Multirobot Systems (ARMS)*.

[35] Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[36] Umar Syed and Robert E Schapire. 2008. A game-theoretic approach to apprenticeship learning. In *Advances in neural information processing systems*. 1449–1456.

[37] Ermo Wei and Sean Luke. 2016. Lenient Learning in Independent-Learner Stochastic Cooperative Games. *Journal of Machine Learning Research* 17, 84 (2016), 1–42.

[38] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. 2018. Multiagent Soft Q-Learning. In *AAAI Fall Symposium on Data Efficient Reinforcement Learning*.

[39] Rudolph Paul Wiegand. 2004. *An Analysis of Cooperative Coevolutionary Algorithms*. Ph.D. Dissertation. Department of Computer Science, George Mason University.

[40] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[41] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*.

[42] Brian D Ziebart. 2010. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University.