Learning to Teach in Cooperative Multiagent Reinforcement Learning

Shayegan Omidshafiei[1,2]
shayegan@mit.edu

Dong-Ki Kim[1,2]
dkkim93@mit.edu

Miao Liu[2,3]
miao.liu1@ibm.com

Gerald Tesauro[2,3]
gtesauro@us.ibm.com

Matthew Riemer[2,3]
mdriemer@us.ibm.com

Christopher Amato[4]
camato@ccs.neu.edu

Murray Campbell[2,3]
mcam@us.ibm.com

Jonathan P. How[1,2]
jhow@mit.edu

[1]LIDS, MIT   [2]MIT-IBM Watson AI Lab   [3]IBM Research   [4]CCIS, Northeastern University

Abstract

Collective human knowledge has clearly benefited from the fact that innovations by individuals are taught to others through communication. Similar to human social groups, agents in distributed learning systems would likely benefit from communication to share knowledge and teach skills. The problem of teaching to improve agent learning has been investigated by prior works, but these approaches make assumptions that prevent application of teaching to general multiagent problems, or require domain expertise for problems they can apply to. This learning to teach problem has inherent complexities related to measuring long-term impacts of teaching that compound the standard multiagent coordination challenges. In contrast to existing works, this paper presents the first general framework and algorithm for intelligent agents to learn to teach in a multiagent environment. Our algorithm, Learning to Coordinate and Teach Reinforcement (LeCTR), addresses peer-to-peer teaching in cooperative multiagent reinforcement learning. Each agent in our approach learns both when and what to advise, then uses the received advice to improve local learning. Importantly, these roles are not fixed; these agents learn to assume the role of student and/or teacher at the appropriate moments, requesting and providing advice in order to improve teamwide performance and learning. Empirical comparisons against state-of-the-art teaching methods show that our teaching agents not only learn significantly faster, but also learn to coordinate in tasks where existing methods fail.

Introduction

In social settings, innovations by individuals are taught to others in the population through communication channels (Rogers 2010), which not only improves final performance, but also the effectiveness of the entire learning process (i.e., rate of learning). There exist analogous settings where learning agents adapt behaviors while interacting in a shared environment (e.g., autonomous cars and assistive robots). While any given agent may not be an expert during learning, it may have local knowledge that teammates may be unaware of. Similar to human social groups, these learning agents would likely benefit from communication to share knowledge and teach skills, thereby improving the effectiveness of system-wide learning. It is also desirable for agents in such systems to learn to teach one another, rather than rely on hand-crafted

teaching heuristics created by domain experts. The benefit of learned peer-to-peer teaching is that it can accelerate learning even without relying on the existence of "all-knowing" teachers. Despite these potential advantages, no algorithms exist for learning to teach in multiagent systems.

This paper targets the learning to teach problem in the context of cooperative Multiagent Reinforcement Learning (MARL). Cooperative MARL is a standard framework for settings where agents learn to coordinate in a shared environment. Recent works in cooperative MARL have shown final task performance can be improved by introducing inter-agent communication mechanisms (Sukhbaatar, Fergus, and others 2016; Foerster et al. 2016; Lowe et al. 2017). Agents in these works, however, merely communicate to coordinate in the given task, not to improve overall learning by teaching one another. By contrast, this paper targets a new multiagent paradigm in which agents learn to teach by communicating action advice, thereby improving final performance and accelerating teamwide learning.

The learning to teach in MARL problem has unique inherent complexities that compound the delayed reward, credit assignment, and partial observability issues found in general multiagent problems (Oliehoek and Amato 2016). As such, there are several key issues that must be addressed. First, agents must learn when to teach, what to teach, and how to learn from what is being taught. Second, despite coordinating in a shared environment, agents may be independent/decentralized learners with privacy constraints (e.g., robots from distinct corporations that cannot share full policies), and so must learn how to teach under these constraints. A third issue is that agents must estimate the impact of each piece of advice on their teammate's learning progress. Delays in the accumulation of knowledge make this credit assignment problem difficult, even in supervised/unsupervised learning (Graves et al. 2017). Nonstationarities due to agent interactions and the temporally-extended nature of MARL compound these difficulties in our setting. These issues are unique to our learning to teach setting and remain largely unaddressed in the literature, despite being of practical importance for future decision-making systems. One of the main reasons for the lack of progress addressing these inherent challenges is the significant increase in the computational complexity of this new teaching/learning paradigm compared to multiagent problems that have previously been considered.

Our paper targets the problem of learning to teach in a multiagent team, which has not been considered before. Each agent in our approach learns both when and what to advise, then uses the received advice to improve local learning. Importantly, these roles are not fixed (see Fig. 1); these agents learn to assume the role of student and/or teacher at appropriate moments, requesting and providing advice to improve teamwide performance and learning. In contrast to prior works, our algorithm supports teaching of heterogeneous teammates and applies to settings where advice exchange incurs a communication cost. Comparisons conducted against state-of-the-art teaching methods show that our teaching agents not only learn significantly faster, but also learn to coordinate in tasks where existing methods fail.

Background: Cooperative MARL

Our work targets cooperative MARL, where agents execute actions that jointly affect the environment, then receive feedback via local observations and a shared reward. This setting is formalized as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP), defined as $\langle \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathbf{\Omega}, \mathcal{O}, \gamma \rangle$ (Oliehoek and Amato 2016); $\mathcal{I}$ is the set of n agents, $\mathcal{S}$ is the state space, $\mathcal{A} = \times_i \mathcal{A}^i$ is the joint action space, and $\mathbf{\Omega} = \times_i \Omega^i$ is the joint observation space.[1] Joint action $\mathbf{a} = \langle a^1, \ldots, a^n \rangle$ causes state $s \in \mathcal{S}$ to transition to $s' \in \mathcal{S}$ with probability $P(s'|s, \mathbf{a}) = \mathcal{T}(s, \mathbf{a}, s')$. At each timestep t, joint observation $\mathbf{o} = \langle o^1, \ldots, o^n \rangle$ is observed with probability $P(\mathbf{o}|s', \mathbf{a}) = \mathcal{O}(\mathbf{o}, s', \mathbf{a})$. Given its observation history, $h_t^i = (o_1^i, \ldots, o_t^i)$, agent i executes actions dictated by its policy $a^i = \pi^i(h_t^i)$. The joint policy is denoted by $\boldsymbol{\pi} = \langle \pi^1, \ldots, \pi^n \rangle$ and parameterized by $\boldsymbol{\theta}$. It may sometimes be desirable to use a recurrent policy representation (e.g., recurrent neural network) to compute an internal state h_t that compresses the observation history, or to explicitly compute a belief state (probability distribution over states); with abuse of notation, we use h_t to refer to all such variations of internal states/observation histories. At each timestep, the team receives reward $r_t = \mathcal{R}(s_t, \mathbf{a}_t)$, with the objective being to maximize value, $V(s; \boldsymbol{\theta}) = \mathbb{E}[\sum_t \gamma^t r_t | s_0 = s]$, given discount factor $\gamma \in [0, 1)$. Let action-value $Q^i(o^i, a^i; h^i)$ denote agent i's expected value for executing action a^i given a new local observation o^i and internal state h^i, and using its policy thereafter. We denote by $\vec{Q}(o^i; h^i)$ the vector of action-values (for all actions) given new observation o^i.

Teaching in Cooperative MARL

This work explores multiagent teaching in a setting where no agent is necessarily an all-knowing expert. This section provides a high-level overview of the motivating scenario. Consider the cooperative MARL setting in Fig. 1, where agents i and j learn a joint task (i.e., a Dec-POMDP). In each learning iteration, these agents interact with the environment and collect data used by their respective learning algorithms, $\mathbb{L}^i$ and $\mathbb{L}^j$, to update their policy parameters, θ^i and θ^j. This is the standard cooperative MARL problem, which we hereafter refer to as $\mathcal{P}_{\text{Task}}$: **the task-level learning problem**. For

---

[1] Superscript i denotes parameters for the i-th agent. Refer to the supplementary material for a notation list.
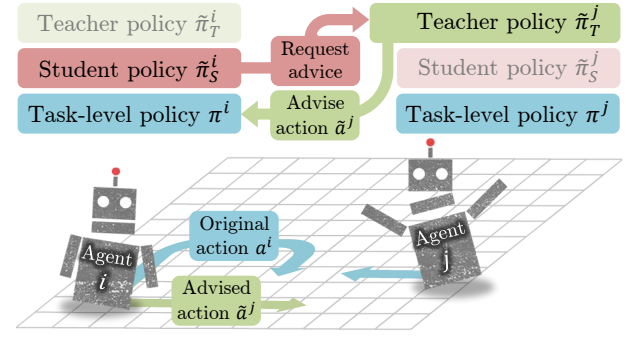

Figure 1: Overview of teaching via action advising in MARL. Each agent learns to execute the task using task-level policy π, to request advice using learned student policy $\widetilde{\pi}_S$, and to respond with action advice using learned teacher policy $\widetilde{\pi}_T$. Each agent can assume a student and/or teacher role at any time. In this example, agent i uses its student policy to request help, agent j advises action $\widetilde{a}^j$, which the student executes instead of its originally-intended action a^i. By learning to transform the local knowledge captured in task-level policies into action advice, the agents can help one another learn.

example, task-level policy π^i is the policy agent i learns and uses to execute actions in the task. Thus, task-level policies summarize each agent's learned behavioral knowledge.

During task-level learning, it is unlikely for any agent to be an expert. However, each agent may have unique experiences, skill sets, or local knowledge of how to learn effectively in the task. Throughout the learning process, it would likely be useful for agents to advise one another using this local knowledge, in order to improve final performance and accelerate teamwide learning. Moreover, it would be desirable for agents to learn when and what to advise, rather than rely on hand-crafted and domain-specific advising heuristics. Finally, following advising, agents should ideally have learned effective task-level policies that no longer rely on teammate advice at every timestep. We refer to this new problem, which involves agents learning to advise one another to improve joint task-level learning, as $\widetilde{\mathcal{P}}_{\text{Advise}}$: **the advising-level problem**.

The advising mechanism used in this paper is *action advising*, where agents suggest actions to one another. By learning to appropriately transform local knowledge (i.e., task-level policies) into action advice, teachers can affect students' experiences and their resulting task-level policy updates. Action advising makes few assumptions, in that learners need only use task-level algorithms $\langle \mathbb{L}^i, \mathbb{L}^j \rangle$ that support off-policy exploration (enabling execution of action advice for policy updates), and that they receive advising-level observations summarizing teammates' learning progress (enabling learning of when/what to advise). Action advising has a good empirical track record (Torrey and Taylor 2013; Taylor et al. 2014; Fachantidis, Taylor, and Vlahavas 2017; da Silva, Glatt, and Costa 2017). However, existing frameworks have key limitations: the majority are designed for single-agent RL and do not consider multiagent learning; their teachers always advise optimal actions to students, mak-

ing decisions about when (not what) to teach; they also use heuristics for advising, rather than training teachers by measuring student learning progress. By contrast, agents in our paper learn to interchangeably assume the role of a student (advice requester) and/or teacher (advice responder), denoted S and T, respectively. Each agent learns task-level policy π used to actually perform the task, student policy $\widetilde{\pi}_S$ used to request advice during task-level learning, and teacher policy $\widetilde{\pi}_T$ used to advise a teammate during task-level learning.[2]

Before detailing the algorithm, let us first illustrate the multiagent interactions in this action advising scenario. Consider again Fig. 1, where agents are learning to execute a task (i.e., solving $\mathcal{P}_{\text{Task}}$) while advising one another. While each agent in our framework can assume a student and/or teacher role at any time, Fig. 1 visualizes the case where agent i is the student and agent j the teacher. At a given task-level learning timestep, agent i's task-level policy π^i outputs an action ('original action a^i' in Fig. 1). However, as the agents are still learning to solve $\mathcal{P}_{\text{Task}}$, agent i may prefer to execute an action that maximizes local learning. Thus, agent i uses its student policy $\widetilde{\pi}_S^i$ to decide whether to ask teammate j for advice. If this advice request is made, teammate j checks its teacher policy $\widetilde{\pi}_T^j$ and task-level policy π^j to decide whether to respond with action advice. Given a response, agent i then executes the advised action ($\widetilde{a}^j$ in Fig. 1) as opposed to its originally-intended action (a^i in Fig. 1). This results in a local experience that agent i uses to update its task-level policy. A reciprocal process occurs when the agents' roles are reversed. The benefit of advising is that agents can learn to use local knowledge to improve teamwide learning.

Similar to recent works that model the multiagent learning process (Hadfield-Menell et al. 2016; Foerster et al. 2018), we focus on the pairwise (two agent) case, targeting the issues of when/what to advise, then detail extensions to n agents. Even in the pairwise case, there exist issues unique to our learning to teach paradigm. First, note that the objectives of $\mathcal{P}_{\text{Task}}$ and $\widetilde{\mathcal{P}}_{\text{Advise}}$ are distinct. Task-level problem, $\mathcal{P}_{\text{Task}}$, has a standard MARL objective of agents learning to coordinate to maximize final performance in the task. Learning to advise ($\widetilde{\mathcal{P}}_{\text{Advise}}$), however, is a higher-level problem, where agents learn to influence teammates' task-level learning by advising them. However, $\mathcal{P}_{\text{Task}}$ and $\widetilde{\mathcal{P}}_{\text{Advise}}$ are also coupled, as advising influences the task-level policies learned. Agents in our problem must learn to advise despite the nonstationarities due to changing task-level policies, which are also a function of algorithms $\langle \mathbb{L}^i, \mathbb{L}^j \rangle$ and policy parameterizations $\langle \theta^i, \theta^j \rangle$.

Learning to teach is also distinct from prior works that involve agents learning to communicate (Sukhbaatar, Fergus, and others 2016; Foerster et al. 2016; Lowe et al. 2017). These works focus on agents communicating in order to coordinate in a task. By contrast, our problem focuses on agents learning how advising affects the underlying task-level learning process, then using this knowledge to accelerate learning even when agents are non-experts. Thus, the objectives of communication-based multiagent papers are disparate from ours, and the two approaches may even be combined.

---

[2]Tilde accents (e.g., $\widetilde{\pi}$) denote advising-level properties.

Phase I:
Train task-level policies from scratch, using latest student/teacher policies

Phase II:
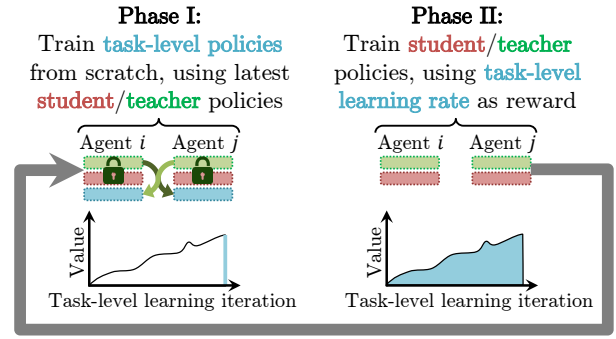Train student/teacher policies, using task-level learning rate as reward

Figure 2: LeCTR consists of two iterated phases: task-level learning (Phase I), and advising-level learning (Phase II). In Phase II, advising policies are trained using rewards correlated to task-level learning (see Table 1). Task-level, student, and teacher policy colors above follows convention of Fig. 1.

Applications   There are a number of practical applications for the proposed learning to teach setting, even in the two agent case. Examples include: commercial applications such as industrial robots learning to coordinate to move an object together (e.g., advising appropriate actions to increase movement speed); agents learning communication policies to transfer local information to one another (e.g., advising certain communication messages to quickly learn a communication protocol); and online or lifelong learning applications where agents must learn to coordinate with new teammates from scratch (i.e., where rate of learning is critical).

LeCTR: Algorithm for Learning to Coordinate and Teach Reinforcement

This section introduces our learning to teach approach, details how issues specific to our problem setting are resolved, and summarizes the overall training protocol. Pseudocode is presented in Algorithm 2.

Overview   Our algorithm, Learning to Coordinate and Teach Reinforcement (LeCTR), solves advising-level problem $\widetilde{\mathcal{P}}_{\text{Advise}}$. The objective is to learn advising policies that augment agents' task-level algorithms $\langle \mathbb{L}^i, \mathbb{L}^j \rangle$ to accelerate solving of $\mathcal{P}_{\text{Task}}$. Our approach involves 2 phases (see Fig. 2):

- Phase I: agents learn $\mathcal{P}_{\text{Task}}$ from scratch using blackbox learning algorithms $\langle \mathbb{L}^i, \mathbb{L}^j \rangle$ and latest advising policies.

- Phase II: advising policies are updated using advising-level rewards correlated to teammates' task-level learning.

No restrictions are placed on agents' task-level algorithms (i.e., they can be heterogeneous). Iteration of Phases I and II enables training of increasingly capable advising policies.

Advising Policy Inputs & Outputs   LeCTR learns student policies $\langle \widetilde{\pi}_S^i, \widetilde{\pi}_S^j \rangle$ and teacher policies $\langle \widetilde{\pi}_T^i, \widetilde{\pi}_T^j \rangle$ for agents i and j, constituting a jointly-initiated advising approach that learns when to request advice and when/what to advise. It is often infeasible to learn high-level policies that directly map task-level policy parameters $\langle \theta^i, \theta^j \rangle$ (i.e., local knowledge) to advising decisions: the agents may be independent/decentralized learners and the cost of communicating

Algorithm 1 Get advising-level observations

1: **function** GETADVISEOBS($\boldsymbol{o}, \boldsymbol{\theta}$)
2:     **for** agents $\alpha \in \{i, j\}$ **do**
3:         Let $-\alpha$ denote α's teammate.
4:         $\widetilde{o}_S^\alpha = \langle o^\alpha, Q^\alpha(o^\alpha; h^\alpha)\rangle$
5:         $\widetilde{o}_T^\alpha = \langle o^{-\alpha}, Q^{-\alpha}(o^{-\alpha}; h^{-\alpha}), Q^\alpha(o^{-\alpha}; h^{-\alpha})\rangle$
6:     **end for**
7:     **return** $\widetilde{\boldsymbol{o}} = \langle \widetilde{o}_S^i, \widetilde{o}_S^j, \widetilde{o}_T^i, \widetilde{o}_T^j \rangle$
8: **end function**

---

task-level policy parameters may be high; sharing policy parameters may be undesirable due to privacy concerns; and learning advising policies over the task-level policy parameter space may be infeasible (e.g., if the latter policies involve millions of parameters). Instead, each LeCTR agent learns advising policies over advising-level observations $\widetilde{o}$. As detailed below, these observations are selected to provide information about agents' task-level state and knowledge in a more compact manner than full policy parameters $\langle \theta^i, \theta^j \rangle$.

Each LeCTR agent can be a student, teacher, or both simultaneously (i.e., request advice for its own state, while advising a teammate in a different state). For clarity, we detail advising protocols when agents i and j are student and teacher, respectively (see Fig. 1). LeCTR uses distinct advising-level observations for student and teacher policies. Student policy $\widetilde{\pi}_S^i$ for agent i decides when to request advice using advising-level observation $\widetilde{o}_S^i = \langle o^i, \vec{Q}^i(o^i; h^i)\rangle$, where o^i and $\vec{Q}^i(o^i; h^i)$ are the agent's task-level observation and action-value vectors, respectively. Through $\widetilde{o}_S^i$, agent i observes a measure of its local task-level observation and policy state. Thus, agent i's student-perspective action is $\widetilde{a}_S^i = \widetilde{\pi}_S^i(\widetilde{o}_S^i) \in \{\text{request advice, do not request advice}\}$.

Similarly, agent j's teacher policy $\widetilde{\pi}_T^j$ uses advising-level observation $\widetilde{o}_T^j = \langle o^i, \vec{Q}^i(o^i; h^i), \vec{Q}^j(o^i; h^i)\rangle$ to decide when/what to advise. $\widetilde{o}_T^j$ provides teacher agent j with a measure of student i's task-level state/knowledge (via o^i and $\vec{Q}^i(o^i; h^i)$) and of its own task-level knowledge given the student's context (via $\vec{Q}^j(o^i; h^i)$). Using $\widetilde{\pi}_T^j$, teacher j decides what to advise: either an action from student i's action space, $\mathcal{A}^i$, or a special no-advice action $\widetilde{a}_\emptyset$. Thus, the teacher-perspective action for agent j is $\widetilde{a}_T^j = \widetilde{\pi}_T^j(\widetilde{o}_T^j) \in \mathcal{A}^i \cup \{\widetilde{a}_\emptyset\}$.

Given no advice, student i executes originally-intended action a^i. However, given advice $\widetilde{a}_T^j$, student i executes action $\beta^i(\widetilde{a}_T^j)$, where $\beta^i(\cdot)$ is a local behavioral policy not known by j. The assumption of local behavioral policies increases the generality of LeCTR, as students may locally transform advised actions before execution.

Following advice execution, agents collect task-level experiences and update their respective task-level policies. A key feature is that LeCTR agents learn what to advise by training $\langle \widetilde{\pi}_T^i, \widetilde{\pi}_T^j \rangle$, rather than always advising actions they would have taken in students' states. These agents may learn to advise exploratory actions or even decline to advise if they estimate that such advice will improve teammate learning.

Rewarding Advising Policies Recall in Phase II of

Algorithm 2 LeCTR Algorithm

1: **for** Phase II episode $\widetilde{e} = 1$ to $\widetilde{E}$ **do**
2:     Initialize task-level policy parameters $\boldsymbol{\theta}$
3:     **for** Phase I episode $e = 1$ to E **do**
4:         $\boldsymbol{o} \leftarrow$ initial task-level observation
5:         **for** task-level timestep $t = 1$ to t_{end} **do**
6:             $\widetilde{\boldsymbol{o}} \leftarrow$ GETADVISEOBS($\boldsymbol{o}, \boldsymbol{\theta}$)
7:             **for** agents $\alpha \in \{i, j\}$ **do**
8:                 Exchange advice $\widetilde{a}^\alpha$ via advising policies
9:                 **if** No advising occurred **then**
10:                     Select action a^α via local policy π^α
11:                 **end if**
12:             **end for**
13:             $\boldsymbol{a} \leftarrow \langle a^i, a^j \rangle, \widetilde{\boldsymbol{a}} \leftarrow \langle \widetilde{a}^i, \widetilde{a}^j \rangle$
14:             $r, \boldsymbol{o}' \leftarrow$ Execute action $\boldsymbol{a}$ in task
15:             $\theta'^i \leftarrow \mathbb{L}^i(\theta'^i, \langle o^i, a^i, r, o'^i \rangle)$
16:             $\theta'^j \leftarrow \mathbb{L}^j(\theta'^j, \langle o^j, a^j, r, o'^j \rangle)$
17:             $\widetilde{\boldsymbol{o}}' \leftarrow$ GETADVISEOBS($\boldsymbol{o}', \boldsymbol{\theta}'$)
18:             $\widetilde{r}_T^i, \widetilde{r}_T^j \leftarrow$ Compute advising-level rewards
19:             Store $\langle \widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}, \widetilde{r} = \widetilde{r}_T^i + \widetilde{r}_T^j, \widetilde{\boldsymbol{o}}' \rangle$ in buffer $\widetilde{\mathcal{M}}$
20:         **end for**
21:     **end for**
22:     Update advising-level critic by minimizing loss,

$$\mathcal{L}(\widetilde{\boldsymbol{\theta}}) = \mathbb{E}_{\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}, \widetilde{r}, \widetilde{\boldsymbol{o}}' \sim \widetilde{\mathcal{M}}} [(\widetilde{r} + \gamma \widetilde{Q}(\widetilde{\boldsymbol{o}}', \widetilde{\boldsymbol{a}}'; \widetilde{\boldsymbol{\theta}}) - \widetilde{Q}(\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}; \widetilde{\boldsymbol{\theta}}))^2] \Big|_{\widetilde{\boldsymbol{a}}' = \widetilde{\boldsymbol{\pi}}(\widetilde{\boldsymbol{o}}')}$$

23:     **for** agents $\alpha \in \{i, j\}$ **do**
24:         **for** roles $\rho \in \{S, T\}$ **do**
25:             Update advising policy parameters $\widetilde{\theta}_\rho^\alpha$ via,

$$\nabla_{\widetilde{\boldsymbol{\theta}}} J(\widetilde{\boldsymbol{\theta}}) = \mathbb{E}_{\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}} \sim \widetilde{\mathcal{M}}} \Big[\sum_{\substack{\alpha \in \{i,j\} \\ \rho \in \{S,T\}}} \nabla_{\widetilde{\theta}_\rho^\alpha} \log \widetilde{\pi}_\rho^\alpha(\widetilde{a}_\rho^\alpha | \widetilde{o}_\rho^\alpha) \nabla_{\widetilde{a}_\rho^\alpha} \widetilde{Q}(\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}; \widetilde{\boldsymbol{\theta}}) \Big]$$

26:         **end for**
27:     **end for**
28: **end for**

LeCTR, advising policies are trained to maximize advising-level rewards that should, ideally, reflect the objective of accelerating task-level learning. Without loss of generality, we focus again on the case where agents i and j assume student and teacher roles, respectively, to detail these rewards. Since student policy $\widetilde{\pi}_S^i$ and teacher policy $\widetilde{\pi}_T^j$ must coordinate to help student i learn, they receive identical advising-level rewards, $\widetilde{r}_S^i = \widetilde{r}_T^j$. The remaining issue is to identify advising-level rewards that reflect learning progress.

Remark 1. *Earning task-level rewards by executing advised actions may not imply actual learning. Thus, rewarding advising-level policies with the task-level reward, r, received after advice execution can lead to poor advising policies. We demonstrate this in our later experiments.*

We evaluate many choices of advising-level rewards, which are summarized and described in Table 1. The unifying intuition is that each reward type corresponds to a different measure of the advised agent's task-level learning,

Table 1: Summary of rewards used to train advising policies. Rewards shown are for the case where agent i is student and agent j teacher (i.e., flip the indices for the reverse case). Each reward corresponds to a different measure of task-level learning after the student executes an action advice and uses it to update its task-level policy. Refer to the supplementary material for more details.

Advising Reward Name	Description	Reward Value $\widetilde{r}_T^j = \widetilde{r}_S^i$
JVG: Joint Value Gain	Task-level value $V(s; \boldsymbol{\theta})$ improvement after learning	$V(s; \boldsymbol{\theta}_{t+1}) - V(s; \boldsymbol{\theta}_t)$
QTR: Q-Teaching Reward	Teacher's estimate of best vs. intended student action	$\max_a Q_T(o^i, a; h^i) - Q_T(o^i, a^i; h^i)$
LG: Loss Gain	Student's task-level loss $\mathcal{L}(\theta^i)$ reduction	$\mathcal{L}(\theta_t^i) - \mathcal{L}(\theta_{t+1}^i)$
LGG: Loss Gradient Gain	Student's task-level policy gradient magnitude	$\|\nabla_{\theta^i} \mathcal{L}(\theta^i)\|_2^2$
TDG: TD Gain	Student's temporal difference (TD) error δ^i reduction	$\|\delta_t^i\| - \|\delta_{t+1}^i\|$
VEG: Value Estimation Gain	Student's value estimate $\hat{V}(\theta^i)$ gain above threshold τ	$\mathbb{1}(\hat{V}(\theta^i) > \tau)$

which occurs after executing an advised action. Readers are referred to the supplementary material for more discussion of advising-level rewards.

Note that at any time, task-level action a^i executed by agent i may either be selected by its local task-level policy, or by a teammate j via advising. In Phase II, pair $\langle \widetilde{\pi}_S^i, \widetilde{\pi}_T^j \rangle$ is rewarded only if advising occurs (with zero advising reward otherwise). Analogous advising-level rewards apply for the reverse student-teacher pairing j-i, where $\widetilde{r}_S^j = \widetilde{r}_T^i$. During Phase II, we train all advising-level policies using a joint advising-level reward $\widetilde{r} = \widetilde{r}_T^i + \widetilde{r}_T^j$ to induce cooperation.

Advising-level rewards are only used during advising-level training, and are computed using either information already available to agents or only require exchange of scalar values (rather than full policy parameters). It is sometimes desirable to consider advising under communication constraints, which can be done by deducting a communication cost c from these advising-level rewards for each piece of advice exchanged.

Training Protocol Recall LeCTR's two phases are iterated to enable training of increasingly capable advising policies. In Phase I, task-level learning is conducted using agents' blackbox learning algorithms and latest advising policies. At the task-level, agents may be independent learners with distinct algorithms. Advising policies are executed in a decentralized fashion, but their training in Phase II is centralized. Our advising policies are trained using the multiagent actor-critic approach of Lowe et al. (2017). Let joint advising-level observations, advising-level actions, and advising-level policies (i.e., 'actors') be, respectively, denoted by $\widetilde{\boldsymbol{o}} = \langle \widetilde{o}_S^i, \widetilde{o}_S^j, \widetilde{o}_T^i, \widetilde{o}_T^j \rangle$, $\widetilde{\boldsymbol{a}} = \langle \widetilde{a}_S^i, \widetilde{a}_S^j, \widetilde{a}_T^i, \widetilde{a}_T^j \rangle$, and $\widetilde{\boldsymbol{\pi}} = \langle \widetilde{\pi}_S^i, \widetilde{\pi}_S^j, \widetilde{\pi}_T^i, \widetilde{\pi}_T^j \rangle$, with $\widetilde{\boldsymbol{\theta}}$ parameterizing $\widetilde{\boldsymbol{\pi}}$. To induce $\widetilde{\boldsymbol{\pi}}$ to learn to teach both agents i and j, we use a centralized action-value function (i.e., 'critic') with advising-level reward $\widetilde{r} = \widetilde{r}_T^i + \widetilde{r}_T^j$. Critic $\widetilde{Q}(\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}; \widetilde{\boldsymbol{\theta}})$ is trained by minimizing loss,

$$\mathcal{L}(\widetilde{\boldsymbol{\theta}}) \qquad (1)$$
$$= \mathbb{E}_{\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}, \widetilde{r}, \widetilde{\boldsymbol{o}}' \sim \widetilde{\mathcal{M}}} [(\widetilde{r} + \gamma \widetilde{Q}(\widetilde{\boldsymbol{o}}', \widetilde{\boldsymbol{a}}'; \widetilde{\boldsymbol{\theta}}) - \widetilde{Q}(\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}; \widetilde{\boldsymbol{\theta}}))^2]\big|_{\widetilde{\boldsymbol{a}}' = \widetilde{\boldsymbol{\pi}}(\widetilde{\boldsymbol{o}}')},$$

where $\widetilde{\boldsymbol{a}}' = \widetilde{\boldsymbol{\pi}}(\widetilde{\boldsymbol{o}}')$ are next advising actions computed using the advising policies, and $\widetilde{\mathcal{M}}$ denotes advising-level replay buffer (Mnih et al. 2015). To update advising policies (actors), the policy gradient theorem (Sutton et al. 2000) is invoked on objective $J(\widetilde{\boldsymbol{\theta}}) = \mathbb{E}[\sum_{t=k}^T \widetilde{\gamma}^{t-k} \widetilde{r}_t]$, yielding gradient,

$$\nabla_{\widetilde{\boldsymbol{\theta}}} J(\widetilde{\boldsymbol{\theta}}) \qquad (2)$$
$$= \mathbb{E}_{\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}} \sim \widetilde{\mathcal{M}}} [\nabla_{\widetilde{\boldsymbol{\theta}}} \log \widetilde{\boldsymbol{\pi}}(\widetilde{\boldsymbol{a}}|\widetilde{\boldsymbol{o}}) \widetilde{Q}(\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}; \widetilde{\boldsymbol{\theta}})]$$
$$= \mathbb{E}_{\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}} \sim \widetilde{\mathcal{M}}} \big[\sum_{\substack{\alpha \in \{i,j\} \\ \rho \in \{S,T\}}} \nabla_{\widetilde{\theta}_\rho^\alpha} \log \widetilde{\pi}_\rho^\alpha(\widetilde{a}_\rho^\alpha | \widetilde{o}_\rho^\alpha) \nabla_{\widetilde{a}_\rho^\alpha} \widetilde{Q}(\widetilde{\boldsymbol{o}}, \widetilde{\boldsymbol{a}}; \widetilde{\boldsymbol{\theta}}) \big],$$

where $\widetilde{\pi}_\rho^\alpha$ is agent α's policy in role ρ.

During training, the advising feedback nonstationarities mentioned earlier are handled as follows: in Phase I, task-level policies are trained online (i.e., no replay memory is used so impact of advice on task-level policies is immediately observed by agents); in Phase II, centralized advising-level learning reduces nonstationarities due to teammate learning, and reservoir sampling is used to further reduce advising reward nonstationarities (see supplementary material for details). Our overall approach stabilizes advising-level learning.

Advising n Agents In the n agent case, students must also decide how to fuse advice from multiple teachers. This is a complex problem requiring full investigation in future work; feasible ideas include using majority voting for advice fusion (as in da Silva, Glatt, and Costa (2017)), or asking a specific agent for advice by learning a 'teacher score' modulated using teacher knowledge or previous teaching experiences.

Evaluation

We conduct empirical evaluations on a sequence of increasingly challenging domains involving two agents. In the 'Repeated' game domain, agents coordinate to maximize the payoffs in Fig. 3a over 5 timesteps. In 'Hallway' (see Fig. 4a), agents only observe their own positions and receive $+1$ reward if they reach opposite goal states; task-level actions are 'move left/right', states are agents' joint grid positions. The higher-dimensional 'Room' game (see Fig. 5a) has the same state/observation/reward structure, but 4 actions ('move up/right/down/left'). Recall student-perspective advising-level actions are to 'ask' or 'not ask' for advice. Teacher-perspective actions are to advise an action from the teammate's task-level action space, or to decline advising.

For the Repeated, Hallway, and Room games, respectively, each iteration of LeCTR Phase I consists of 50, 100, and 150 task-level learning iterations. Our task-level agents are independent Q-learners with tabular policies for the Repeated

	Agent j	
	a_1	a_2
Agent i a_1	0	1
a_2	0.1	0

(a) Repeated game payoffs. Each agent has two actions (a_1, a_2).
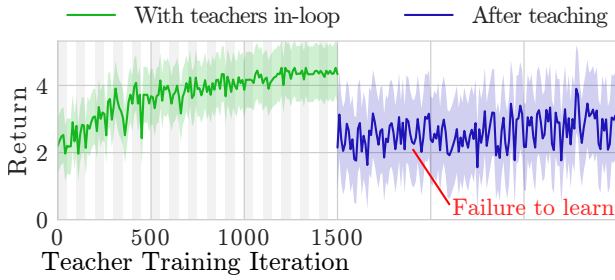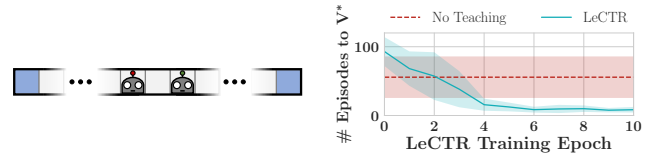

(b) Counterexample showing poor advising reward choice $\widetilde{r}_T = r$. LeCTR Phase I and II iterations are shown as background bands.

Figure 3: Repeated game. (b) shows a counterexample where using $\widetilde{r}_T = r$ yields poor advising, as teachers learn to advise actions that maximize reward (left half, green), but do not actually improve student task-level learning (right half, blue).

(a) Hallway domain overview. (b) Simult. learning & teaching

Figure 4: Hallway game. (a) Agents receive $+1$ reward by navigating to opposite states in 17-grid hallway. (b) LeCTR accelerates learning & teaching compared to no-teaching.

(a) Room domain overview. (b) Teaching heterogeneous agents.

Figure 5: Room game. (a) Agents receive $+1$ reward by navigating to opposite goals in 17×5 grid. (b) LeCTR outperforms prior approaches when agents are heterogeneous.

game and tile-coded policies (Sutton and Barto 1998) for the other games. Advising policies are neural networks with internal rectified linear unit activations. Refer to the supplementary material for hyperparameters. The advising-level learning nature of our problem makes these domains challenging, despite their visual simplicity; their complexity is comparable to domains tested in recent MARL works that learn over multiagent learning processes (Foerster et al. 2018), which also consider two agent repeated/gridworld games.

Counterexample demonstrating Remark 1 Fig. 3b shows results given a poor choice of advising-level reward, $\widetilde{r}_T = r$, in the Repeated game. The left plot (in green) shows task-level return received due to both local policy actions *and* advised actions, which increases as teachers learn. However, in the right plot (blue) we evaluate how well task-level policies perform by themselves, after they have been trained using the final advising-level policies. The poor performance of the resulting task-level policies indicates that advising policies learned to maximize their own rewards $\widetilde{r}_T = r$ by always advising optimal actions to students, thereby disregarding whether task-level policies actually learn. No exploratory actions are advised, causing poor task-level performance after advising. This counterexample demonstrates that advising-level rewards that reflect student learning progress, rather than task-level reward r, are critical for useful advising.

Comparisons to existing teaching approaches Table 2 shows extensive comparisons of existing heuristics-based teaching approaches, no teaching (independent Q-learning), and LeCTR with all of the advising rewards introduced in Table 1. We use the VEG advising-level reward in the final version of our LeCTR algorithm, but show all advising reward results for completeness. We report both final task-level performance after teaching, $\bar{V}$, and also area under the task-level learning curve (AUC) as a measure of rate of learning;
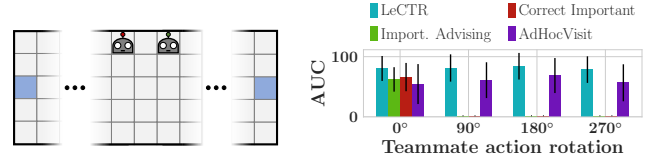
higher values are better for both. $\bar{V}$ is computed by running each algorithm until convergence of task-level policies $\boldsymbol{\pi} = \langle \pi^i, \pi^j \rangle$, and computing the mean value obtained by the final joint policy $\boldsymbol{\pi}$. AUC is computed by intermittently evaluating the resulting task-level policies $\boldsymbol{\pi}$ throughout learning; while teacher advice is used *during* learning, the AUC is computed by evaluating the resulting $\boldsymbol{\pi}$ *after* advising (i.e., in absence of teacher advice actions, such that AUC measures actual student learning progress). Comparisons in Table 2 were conducted with 50 independent trials, with the majority of our other experiments using at least 10 independent trials. In Table 2, best results are computed using a t-test with a significance level of $\alpha = 0.05$ and a multiple comparison Bonferroni correction (Weisstein 2004).

Single-agent approaches requiring an expert teacher are extended to the MARL setting by using teammates' policies (pre-trained to expert level) as each agent's teacher. In the Repeated game, LeCTR attains best performance in terms of final value and rate of learning (AUC), as agents learn to advise each other to take the appropriate high-value opposite actions. The majority of prior approaches always advise the teaching agent's optimal action to its teammate, resulting in suboptimal returns. The median-performance advising agents (e.g., AdhocVisit/AdhocTD) can advise opposite actions to one another, such that high return is sometimes attained. In the Hallway and Room games, approaches that tend to over-advise (e.g., Ask Uncertain, Early Advising, and Early Correcting) perform poorly. AdHocVisit and AdHocTD fare better, as their probabilistic nature permits agents to take exploratory actions and sometimes learn optimal policies. Importance Advising and Correct Important heuristics lead agents to suboptimal (distant) goals in Hallway and Room, yet attain positive value due to domain symmetries. Low-performing methods (e.g., Early Advising) did not learn to solve the tasks primarily due to over-advising, leading agents

Table 2: $\bar{V}$ and Area under the Curve (AUC) for teaching algorithms. Best results in bold (computed via a t-test with $p < 0.05$). Independent Q-learning correspond to the no-teaching case. [†]Final version LeCTR uses the VEG advising-level reward.

| Algorithm	Repeated Game		Hallway Game		Room Game	
	$\bar{V}$	AUC	$\bar{V}$	AUC	$\bar{V}$	AUC
Independent Q-learning (i.e., no teaching)	2.75±2.12	272±210	0.56±0.35	36±24	0.42±0.33	22±25
Ask Important (Amir et al. 2016)	1.74±1.89	178±181	0.53±0.36	39±27	0.00±0.00	0±0
Ask Uncertain (Clouse 1996)	1.74±1.89	170±184	0.00±0.00	0±0	0.00±0.00	0±0
Early Advising (Torrey and Taylor 2013)	0.45±0.00	45±1	0.00±0.00	0±0	0.00±0.00	0±0
Import. Advising (Torrey and Taylor 2013)	0.45±0.00	45±1	0.67±0.07	39±17	0.57±0.03	48±8
Early Correcting (Amir et al. 2016)	0.45±0.00	45±1	0.00±0.00	0±0	0.00±0.00	0±0
Correct Important (Torrey and Taylor 2013)	0.45±0.00	45±1	0.67±0.07	39±16	0.56±0.00	51±7
AdHocVisit (da Silva, Glatt, and Costa 2017)	2.49±2.04	244±199	0.57±0.34	38±24	0.43±0.33	22±26
AdHocTD (da Silva, Glatt, and Costa 2017)	1.88±1.94	184±189	0.49±0.37	26±24	0.39±0.33	26±29
LeCTR (with JVG)	4.16±1.17	405±114	0.25±0.37	21±31	0.11±0.27	6±21
LeCTR (with QTR)	**4.52±0.00**	**443±3**	0.21±0.35	12±22	0.20±0.32	11±22
LeCTR (with TDG)	3.36±1.92	340±138	0.19±0.34	15±25	0.26±0.34	25±32
LeCTR (with LG)	**3.88±1.51**	375±132	0.13±0.29	13±22	0.37±0.34	30±32
LeCTR (with LGG)	**4.41±0.69**	**430±53**	0.22±0.35	27±29	0.56±0.27	56±23
LeCTR[†]	**4.52±0.00**	**443±3**	**0.77±0.00**	**71±3**	**0.68±0.07**	**79±16**

(a) Hallway domain.   (b) Room domain.
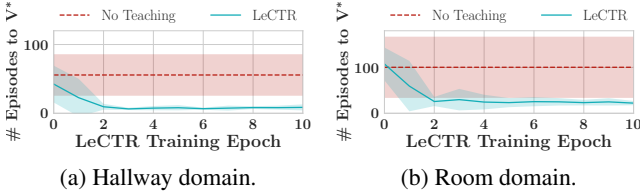
Figure 6: LeCTR accelerates multiagent transfer learning.

to take random actions, rarely or never reaching the goal states simultaneously.

LeCTR outperforms all approaches when using the VEG advising-level reward (Table 2). While the JVG advising-level reward seems an intuitive measure of learning progress due to directly measuring task-level performance, its high variance in situations where the task-level value is sensitive to policy initialization sometimes destabilizes training. JVG is also expensive to compute, requiring game rollouts after each advice exchange. LG and TDG perform poorly due to the high variance of task-level losses used to compute them. We hypothesize that VEG performs best as its thresholded binary advising-level reward filters the underlying noisy task-level losses for teachers. A similar result is reported in recent work on teaching of supervised learners, where threshold-based advising-level rewards have good empirical performance (Fan et al. 2018). Fig. 4b shows improvement of LeCTR's advising policies due to training, measured by the number of task-level episodes needed to converge to the max value reached, V^*. LeCTR outperforms the rate of learning for the no-teaching case, stabilizing after roughly 4-6 training epochs.

Teaching for transfer learning Learning to teach can also be applied to multiagent transfer learning. We first pretrain task-level policies in the Hallway/Room tasks (denote these $\mathbb{T}_1$), flip agents' initial positions, then train agents to

use teammates' $\mathbb{T}_1$ task-level policies to accelerate learning in flipped task $\mathbb{T}_2$. Results for Hallway and Room are shown in Figs. 6a and 6b, respectively, where advising accelerates rate of learning using prior task knowledge. Next, we test *transferability of advising policies* themselves (i.e., use advising policies trained for one task to accelerate learning in a brand new, but related, task). We fix (no longer train) advising policies from the above transfer learning test. We then consider 2 variants of Room: one with the domain (including initial agent positions) flipped vertically ($\mathbb{T}_3$), and one flipped vertically *and* horizontally ($\mathbb{T}_4$). We evaluate the fixed advising policies (trained to transfer from $\mathbb{T}_1 \to \mathbb{T}_2$) on transfer from $\mathbb{T}_3 \to \mathbb{T}_4$. Learning without advising on $\mathbb{T}_4$ yields AUC 24 ± 26, while using the fixed advising policy for transfer $\mathbb{T}_3 \to \mathbb{T}_4$ attains AUC 68 ± 17. Thus, learning is accelerated even when using pre-trained advising policies. While transfer learning typically involves more significant differences in tasks, these preliminary results motivate future work on applications of advising for MARL transfer learning.

Advising heterogeneous teammates We consider heterogeneous variants of the Room game where one agent, j, uses rotated versions of its teammate i's action space; e.g., for rotation $90°$, agent i's action indices correspond to (**up**/right/down/left), while j's to (left/**up**/right/down). Comparisons of LeCTR and the best-performing existing methods are shown in Fig. 5b for all rotations. Prior approaches (Importance Advising and Correct Important) work well for homogeneous actions ($0°$ rotation). However, they attain 0 AUC for heterogeneous cases, as agents always advise action indices corresponding to their local action spaces, leading teammates to no-reward regions. AdHocVisit works reasonably well for all rotations, by sometimes permitting agents to explore. LeCTR attains highest AUC for all rotations.

Effect of communication cost on advice exchange We evaluate impact of communication cost on advising by de-
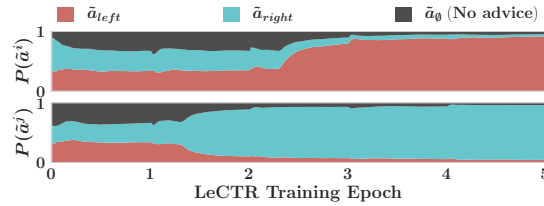
(a) No communication cost, $c = 0$, agents advise opposite actions.
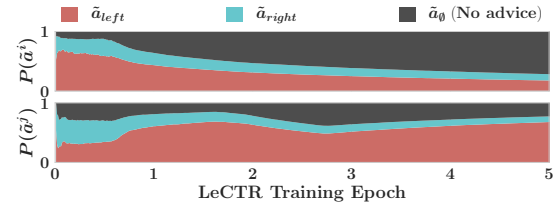
(b) With $c = 0.5$, one agent leads & advises actions opposite its own.

Figure 7: Hallway game, impact of communication cost c on advising policy behaviors. First and second rows show probabilities of action advice, $P(\widetilde{a}^i)$ and $P(\widetilde{a}^j)$, for agents i and j, respectively, as their advising policies are trained using LeCTR.

ducting cost c from advising rewards for each piece of advice exchanged. Fig. 7 shows a comparison of action advice probabilities for communication costs $c = 0$ and $c = 0.5$ in the Hallway game. With no cost ($c = 0$ in Fig. 7a), agents learn to advise each other opposite actions ($\widetilde{a}_{\text{left}}$ and $\widetilde{a}_{\text{right}}$, respectively) in addition to exploratory actions. As LeCTR's VEG advising-level rewards are binary (0 or 1), two-way advising with communication cost $c = 0.5$ nullifies positive advising-level rewards, penalizing excessive advising. Thus, when $c = 0.5$ (Fig. 7b), advising becomes unidirectional: one agent advises opposite exploratory actions of its own, while its teammate tends not to advise.

Related Work

Effective diffusion of knowledge has been studied in many fields, including inverse reinforcement learning (Ng and Russell 2000), apprenticeship learning (Abbeel and Ng 2004), and learning from demonstration (Argall et al. 2009), wherein students discern and emulate key demonstrated behaviors. Works on curriculum learning (Bengio et al. 2009) are also related, particularly automated curriculum learning (Graves et al. 2017). Though Graves et al. focus on single student supervised/unsupervised learning, they highlight interesting measures of learning progress also used here. Several works meta-learn active learning policies for supervised learning (Bachman, Sordoni, and Trischler 2017; Fang, Li, and Cohn 2017; Pang, Dong, and Hospedales 2018; Fan et al. 2018). Our work also uses advising-level meta-learning, but in the regime of MARL, where agents must learn to advise teammates without destabilizing coordination.

In action advising, a student executes actions suggested by a teacher, who is typically an expert always advising the optimal action (Torrey and Taylor 2013). These works typically use state importance value $I(s, \hat{a}) = \max_a Q(s, a) - Q(s, \hat{a})$ to decide when to advise, estimating the performance difference between the student's best action versus intended/worst-case action $\hat{a}$. In student-initiated approaches such as Ask Uncertain (Clouse 1996) and Ask Important (Amir et al. 2016), the student decides when to request advice using heuristics based on $I(s, \hat{a})$. In teacher-initiated approaches such as Importance Advising (Torrey and Taylor 2013), Early Correcting (Amir et al. 2016), and Correct Important (Torrey and Taylor 2013), the teacher decides when to advise by comparing student policy π_S to expert policy π_T. Q-Teaching (Fachantidis, Taylor, and Vlahavas 2017) learns when to ad-

vise by rewarding the teacher $I(s, \hat{a})$ when it advises. See the supplementary material for details of these approaches.

While most works on information transfer target single-agent settings, several exist for MARL. These include imitation learning of expert demonstrations (Le et al. 2017), cooperative inverse reinforcement learning with a human and robot (Hadfield-Menell et al. 2016), and transfer to parallel learners in tasks with similar value functions (Taylor et al. 2013). To our knowledge, AdHocVisit and AdHocTD (da Silva, Glatt, and Costa 2017) are the only action advising methods that do not assume expert teachers; teaching agents always advise the action they would have locally taken in the student's state, using state visit counts as a heuristic to decide when to exchange advise. Wang et al. (2018) uses da Silva, Glatt, and Costa's teaching algorithm with minor changes.

Contribution

This work introduced a new paradigm for learning to teach in cooperative MARL settings. Our algorithm, LeCTR, uses agents' task-level learning progress as advising policy feedback, training advisors that improve the rate of learning without harming final performance. Unlike prior works (Torrey and Taylor 2013; Taylor et al. 2014; Zimmer, Viappiani, and Weng 2014), our approach avoids hand-crafted advising policies and does not assume expert teachers. Due to the many complexities involved, we focused on the pairwise problem, targeting the issues of *when* and *what* to teach. A natural avenue for future work is to investigate the n-agent setting, extending the ideas presented here where appropriate.

Acknowledgements

References

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

Amir, O.; Kamar, E.; Kolobov, A.; and Grosz, B. J. 2016. Interactive teaching strategies for agent training. International Joint Conferences on Artificial Intelligence.

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

Bachman, P.; Sordoni, A.; and Trischler, A. 2017. Learning algorithms for active learning. In *International Conference on Machine Learning*, 301–310.

Bengio, Y.; Louradour, J.; Collobert, R.; and Weston, J. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, 41–48. ACM.

Clouse, J. A. 1996. On integrating apprentice learning and reinforcement learning.

da Silva, F. L.; Glatt, R.; and Costa, A. H. R. 2017. Simultaneously learning and advising in multiagent reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, 1100–1108. International Foundation for Autonomous Agents and Multiagent Systems.

Fachantidis, A.; Taylor, M. E.; and Vlahavas, I. 2017. Learning to teach reinforcement learning agents. *Machine Learning and Knowledge Extraction* 1(1):2.

Fan, Y.; Tian, F.; Qin, T.; Li, X.-Y.; and Liu, T.-Y. 2018. Learning to teach. In *International Conference on Learning Representations*.

Fang, M.; Li, Y.; and Cohn, T. 2017. Learning how to active learn: A deep reinforcement learning approach. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 595–605.

Foerster, J.; Assael, I. A.; de Freitas, N.; and Whiteson, S. 2016. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, 2137–2145.

Foerster, J. N.; Chen, R. Y.; Al-Shedivat, M.; Whiteson, S.; Abbeel, P.; and Mordatch, I. 2018. Learning with opponent-learning awareness. In *Proceedings of the 17th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems.

Graves, A.; Bellemare, M. G.; Menick, J.; Munos, R.; and Kavukcuoglu, K. 2017. Automated curriculum learning for neural networks. In *International Conference on Machine Learning*, 1311–1320.

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

Le, H. M.; Yue, Y.; Carr, P.; and Lucey, P. 2017. Coordinated multi-agent imitation learning. In *International Conference on Machine Learning*, 1995–2003.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O. P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, 6382–6393.

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.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 663–670. Morgan Kaufmann Publishers Inc.

Oliehoek, F. A., and Amato, C. 2016. *A concise introduction to decentralized POMDPs*, volume 1. Springer.

Pang, K.; Dong, M.; and Hospedales, T. 2018. Meta-learning transferable active learning policies by deep reinforcement learning.

Rogers, E. M. 2010. *Diffusion of innovations*. Simon and Schuster.

Sukhbaatar, S.; Fergus, R.; et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, 2244–2252.

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

Sutton, R. S.; McAllester, D. A.; Singh, S. P.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, 1057–1063.

Taylor, A.; Dusparic, I.; Galván-López, E.; Clarke, S.; and Cahill, V. 2013. Transfer learning in multi-agent systems through parallel transfer. In *Workshop on Theoretically Grounded Transfer Learning at the 30th International Conf. on Machine Learning (Poster)*, volume 28, 28. Omnipress.

Taylor, M. E.; Carboni, N.; Fachantidis, A.; Vlahavas, I.; and Torrey, L. 2014. Reinforcement learning agents providing advice in complex video games. *Connection Science* 26(1):45–63.

Torrey, L., and Taylor, M. 2013. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems.

Wang, Y.; Lu, W.; Hao, J.; Wei, J.; and Leung, H.-F. 2018. Efficient convention emergence through decoupled reinforcement social learning with teacher-student mechanism. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, 795–803. International Foundation for Autonomous Agents and Multiagent Systems.

Weisstein, E. W. 2004. Bonferroni correction.

Zimmer, M.; Viappiani, P.; and Weng, P. 2014. Teacher-student framework: a reinforcement learning approach. In *AAMAS Workshop Autonomous Robots and Multirobot Systems*.