Learning from a Learner

Alexis D. Jacq [1 2] **Matthieu Geist** [1] **Ana Paiva** [2] **Olivier Pietquin** [1]

Abstract

In this paper, we propose a novel setting for Inverse Reinforcement Learning (IRL), namely "Learning from a Learner" (LfL). As opposed to standard IRL, it does not consist in learning a reward by observing an optimal agent, but from observations of another learning (and thus sub-optimal) agent. To do so, we leverage the fact that the observed agent's policy is assumed to improve over time. The ultimate goal of this approach is to recover the actual environment's reward and to allow the *observer* to outperform the *learner*. To recover that reward in practice, we propose methods based on the entropy-regularized policy iteration framework. We discuss different approaches to learn solely from trajectories in the state-action space. We demonstrate the genericity of our method by observing agents implementing various reinforcement learning algorithms. Finally, we show that, on both discrete and continuous state/action tasks, the *observer*'s performance (that optimizes the recovered reward) can surpass those of the observed *learner*.

1. Introduction

Imagine two friends from different nationalities: Bob is French and Alice is Japanese. During holidays, Bob is visiting Alice's family and wishes to discover the Japanese culture. One day, Alice's grandfather decides to teach Alice and Bob a traditional board game. Neither Bob nor Alice know that game. Unfortunately, Alice and her grandfather only speak Japanese, while Bob only speaks French. However, Alice decides to learn the game by playing against her grandfather. She hence practices the game until she is able to defeat him. As the old man was not an expert, she needed just a few trials to reach that level. During that time, Bob was observing Alice's strategy improvements. Now, we ask the question: *is Bob able to deduce the rules of the game and to derive his own strategy that may outperform both Alice and her grandfather?*

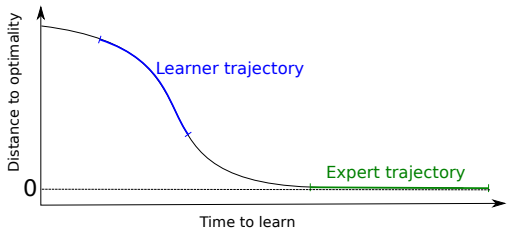

Figure 1. In standard IRL, the goal is to recover the reward from demonstrated trajectories that follow a stationary optimal policy (expert trajectory). In the LfL setting, we aim at recovering the reward from trajectories of a learning agent that is also discovering the problem. Such trajectories follow a sequence of sub-optimal policies, assumed to improve with time (learner trajectories).

Agents modelling is required in various fields of computational and social sciences in order to predict behaviours for better coordination. In the reinforcement learning (RL) paradigm, the behaviour of an agent is determined by a reward function. However, in many cases, it is impossible for agents to share their reward functions. This is especially the case in Human-Machine Interaction – or even Human-Human Interaction, because the complexity of human motivations hardly translates in terms of quantitative values. Inverse Reinforcement Learning (IRL) (Ng et al., 2000) addresses this problem by inferring a reward function so that it explains an agent's trajectories in its state-action space. In the standard approach, the observed agent (the expert) is thus supposed to follow an optimal policy according to some unknown reward function and the observing agent tries to infer that underlying reward function. The optimality assumption is essential in many scenarios, especially in training robots at complex tasks requiring help from a human expert. However, even if the expert's policy is given, an infinite number of solutions explains it, including the null reward function (for which any policy is optimal). Many different approaches aiming at addressing this issue can be found in the literature based either on game theory (Syed & Schapire, 2008), maximum entropy (Ziebart et al., 2008), relative entropy (Boularias et al., 2011) or supervised learning (Klein et al., 2013), among others.

[1]Google Brain, Paris, France [2]INESC-ID, IST, University of Lisbon. Correspondence to: Alexis D. Jacq <alexis.jacq@gmail.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Our first contribution is a new setting where an observed *learner* (Alice in our example) is assumed to be currently learning the task and improving its (sub-optimal) behaviour over time, while an *observer* (Bob in our example) is trying to infer the reward that the *learner* optimizes. Such situations are found in many multi-agent scenarios where agents have to mutually learn opponents goals in order to cooperate, and also in human-robot-based education, when a human learns a task with the help of a robot. In one hand, it is no longer possible to consider the observed agent as an expert (not even to consider stationarity). In the other hand, we may have more information than from an optimal behaviour. For example the *learner* will make (and hopefully correct) mistakes and will show, more than what must be done, what must be avoided. With this paper, we focus on this situation and we introduce the *Learning from a Learner* problem (LfL). It formalizes an IRL setting exploiting trajectories of a learning agent rather than optimal demonstrations of an expert agent (Fig. 1). In this setting, the *observer* can potentially learn the true reward provided by the environment and go beyond pure imitation, outperforming the learner.

Like in IRL, we make the assumption that the *learner* is motivated by a reward function encoding its task. LfL thus aims at inverting policy improvements: from a sequence of policies assumed to be improving w.r.t. some unknown reward function, the *observer* has to recover the reward function that better explains the successive improvements. Given the optimization algorithm assumed for the *learner*, different approaches and solutions may be investigated. In our work, we focus on the case where the *learner* improves a policy extracted from an underlying associated Q-function (see later for a formal definition). From this, our second contribution is an approach based on entropy-regularized RL, modelling the *learner* as performing soft policy improvements (Haarnoja et al., 2018). Under this assumption, we show that the reward function can be extracted from a single policy improvement step, up to a shaping that does not affect the optimal policy and which is specific to the improvement.

We then switch to a more realistic case of study where only trajectories in the state-action space are observed and the successively improved policies must be inferred. Our third contribution is an algorithm that directly learns the reward from sampled trajectories. To demonstrate the genericity of our approach under controlled conditions, we study the case of a *learner* in a discrete grid world, and that does not necessarily improve its policy with soft improvements. Experiments on various continuous control tasks show that our algorithm enables the *observer* to surpass the performance the *learner* obtained while it was observed, without access to the true reward function. This confirms that the learned reward is strongly correlated with the one provided by the environment and can lead to better policies than imitation.

2. Problem setting

The LfL problem involves two agents: a *learner* (instead of the expert in IRL) and an *observer* (instead of the apprentice in IRL). The *observer* perceives a sequence of states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$ performed by the *learner*, and makes two assumptions:

- The *learner*'s behaviour is motivated by a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.
- The *learner* is improving its behaviour according to r while being observed.

Formally, the *learner* is assumed to be improving its policy over time because it learns to solve a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ a set of actions, $\mathcal{P}(s'|s,a)$ a transition distribution, $r(a,s)$ a reward function and γ a discount factor. An observed policy $\pi(a|s)$ models the probability that the *learner* applies an action a while being in a state s. In that context, the presumed goal of the *learner* is the maximization of its expected cumulative discounted reward:

$$\mathcal{J}(\pi) = \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t r(a_t, s_t) \right].$$

Based on this objective, we say that a policy π_2 is an improvement of a policy π_1 if and only if $\mathcal{J}(\pi_2) > \mathcal{J}(\pi_1)$. Then, the goal of the *observer* is to recover the reward function r from the observed (supposedly) improving sequence of policies $\{\pi_1 \dots \pi_N\}$ of the *learner*.

3. Greedy improvements

Under the dynamics $\mathcal{P}$ of the MDP and a policy π, the expected cumulative reward for choosing an action a in state s is given by the Q-function:

$$Q^\pi(s,a) = \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t r(s_t, a_t) \middle| s_0 = s, a_0 = a \right].$$

The assumption that improvements are based on a Q-function makes sense for two reasons: i) many RL algorithms are based on the estimation of such a function, and ii) it brings the notion of greedy improvement. Given a policy π_1, we define the space $\mathcal{G}(\pi_1)$ of greedily-improved policies as follows:

$$\pi_2 \in \mathcal{G}(\pi_1) \Leftrightarrow \forall s \, \pi_2(.|s) = \underset{\pi'(.|s)}{\operatorname{argmax}} \, \mathbb{E}_{a \sim \pi'(.|s)} \left[Q^{\pi_1}(s,a) \right]. \tag{1}$$

By construction, such a pair of policies π_1 and π_2 meets the condition of the policy improvement theorem (Sutton et al., 1998), which guarantees that $\mathcal{J}(\pi_2) > \mathcal{J}(\pi_1)$.

Note that $\mathcal{G}(\pi_1)$ may only contain the deterministic policy $\pi_2(a|s) = \mathbb{1}\{\text{argmax}_a\, Q^{\pi_1}(s,a)\}$. In general, RL agents are exploring with non-deterministic policies, which makes the assumption that an observed improvement is a greedy improvement incompatible with observing an exploring behaviour. To address that issue, we place ourselves in the framework of entropy-regularized reinforcement learning.

4. Recovering rewards from soft improvements

Entropy-regularized RL prohibits the emergence of deterministic policies (eg., see Neu et al. (2017)). A wide range of recent deep-RL algorithms use this principle, e.g. (Mnih et al., 2016; Nachum et al., 2017; Haarnoja et al., 2017; 2018). We thus model the *learner* under this framework. Formally, the entropy-regularized objective is:

$$\mathcal{J}_{\text{soft}}(\pi) = \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t \left(r(s_t, a_t) + \alpha \mathcal{H}(\pi(.|s_t)) \right) \right], \quad (2)$$

where $\mathcal{H}$ refers to the Shannon entropy,

$$\mathcal{H}(\pi(.|s)) = -\mathbb{E}_{a \sim \pi(.|s)} \left[\ln \pi(a|s) \right],$$

and α is a trade-off factor that controls the degree of regularization. Based on this new objective and following a policy π, the value of a state-action couple (s,a) is given by the soft Q-function:

$$Q^\pi_{\text{soft}}(s_t, a_t) = \\ r(s_t, a_t) + \mathbb{E}_\pi \left[\sum_{l > t} \gamma^{l-t} \left(r(s_l, a_l) + \alpha \mathcal{H}(\pi(.|s_l)) \right) \right].$$

It is the unique fixed point of the associated Bellman evaluation equation:

$$Q^\pi_{\text{soft}}(s,a) = r(s,a) + \gamma \mathbb{E}_{s',a'} \left[Q^\pi_{\text{soft}}(s',a') - \alpha \ln \pi(a'|s') \right].$$

It can be shown that the space $\mathcal{G}_{\text{soft}}(\pi_1)$ of greedily-improved policies defined by $Q^{\pi_1}_{\text{soft}}$ as in Eq. (1) is reduced to the unique stochastic policy defined by:

$$\pi_2(a|s) \propto \exp \left\{ \frac{Q^{\pi_1}_{\text{soft}}(s,a)}{\alpha} \right\}. \quad (3)$$

Such greedy improvements, known as soft policy improvements, serve as the theoretical foundations of the Soft Actor Critic (SAC) algorithm (Haarnoja et al., 2018). In the next section, we will assume that an observed improvement is explained by Eq. (3) and will note it as an operator $\text{SPI}_r : \Pi \to \Pi$ that depends on the reward function (and the dynamics) of the MDP:

$$\pi_2 = \text{SPI}_r\{\pi_1\}.$$

In section 4.2, we will show how to retrieve the reward function from two consecutive policies, up to an unknown shaping. But first, we study what kind of shaping will induce the same optimal policy.

4.1. SPI invariance under reward transformation

Soft policy improvements remain identical under transformations of the reward function of the form $\bar{r}(a,s) = r(a,s) + f(s) - \gamma \mathbb{E}_{s'|s,a}[f(s')]$. In other words, reward shaping (Ng et al., 1999) can be extended to entropy-regularized RL.

Lemma 1 (Shaping). *Let $\pi \in \Pi$ be any policy, $r_1 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $r_2 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be two reward functions, and Q^{π,r_1}_{soft} and Q^{π,r_2}_{soft} be the associated soft Q-functions. Then, for any function $g : \mathcal{S} \to \mathbb{R}$, the two following assertions are equivalent:*

- *(A) For all state-action couples (s,a):*
$$r_1(s,a) = r_2(s,a) + g(s) - \gamma \mathbb{E}_{s'|s,a}\left[g(s')\right],$$
- *(B) For all state-action couples (s,a):*
$$Q^{\pi,r_1}_{soft}(s,a) = Q^{\pi,r_2}_{soft}(s,a) + g(s).$$

Proof. The proof is provided in the appendix. □

An immediate consequence of this result is that shaping the reward this way will not change greedy policies, and will induce the same (unique, in this regularized framework) optimal policy.

Theorem 1 (SPI invariance under reward shaping). *Let $r_1 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $r_2 : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and $g : \mathcal{S} \to \mathbb{R}$ be such that*

$$r_1(a,s) = r_2(a,s) + g(s) - \gamma \mathbb{E}_{s'|s,a}\left[g(s')\right].$$

Greedy policies are invariant under this reward transform:

$$SPI_{r_1}\{\pi\} = SPI_{r_2}\{\pi\}.$$

Moreover, both rewards lead to the same optimal policy. Write $\pi_{*,j}$ the optimal policy for reward r_j, $j = 1, 2$, we have that $\pi_{*,1} = \pi_{*,2}$.

4.2. Inverting soft policy improvements

Given two consecutive policies $\hat{\pi}_1$ and $\hat{\pi}_2$ and under the assumption of soft policy improvement, there exists an underlying (unknown) reward function r such that $\hat{\pi}_2 = \text{SPI}_r\{\hat{\pi}_1\}$. The LfL *observer*'s objective is to extract such a reward function that would explain the whole sequence of observed policy changes $\{\hat{\pi}_1, ...\hat{\pi}_N\}$. In the ideal case of a real soft policy improvement the reward function r can be deduced from two consecutive policies, up to a shaping that is specific to the improvement.

Theorem 2 (Soft policy improvement inversion)**.** *Let π_1 and π_2 be two consecutive policies given by soft policy iterations ($\pi_2 = SPI_r\{\pi_1\}$). Then a reward $\bar{r}_{1\to2}(s,a)$ explaining the soft improvement is given by*

$$\bar{r}_{1\to2}(s,a) = \\ \alpha \ln \pi_2(a|s) + \alpha\gamma\mathbb{E}_{s'}\left[\mathrm{KL}(\pi_1(.|s')\|\pi_2(.|s'))\right],$$

with $\mathrm{KL}(\pi_1(.|s)||\pi_2(.|s)) = \mathbb{E}_{a\sim\pi_1(.|s)}[\ln \frac{\pi_1(.|s)}{\pi_2(.|s)}]$.

Indeed, there exists a function $f_{1\to2} : \mathcal{S} \to \mathbb{R}$ such that

$$\bar{r}_{1\to2}(s,a) = r(s,a) + f_{1\to2}(s) - \gamma\mathbb{E}_{s'}\left[f_{1\to2}(s')\right],$$

and $\bar{r}_{1\to2}$ has the same unique optimal policy as r.

Proof. The proof is provided in the appendix. □

4.3. Recovering state-only reward functions

If a shaping does not affect the optimal policy of the entropy-regularized problem, it depends on the dynamics and may not be robust to dynamic changes (Fu et al., 2017). In the case of a state-only ground-truth reward function, one simple solution consists in searching for a state-only reward $\bar{r} : S \to \mathbb{R}$ and a shaping $f : S \to \mathbb{R}$ such that:

$$\begin{aligned}\bar{r}_{1\to2}(s,a) &= \bar{r}(s) + f(s) - \gamma\mathbb{E}_{s'\sim\mathcal{P}(.|a,s)}\left[f(s')\right] \\ &= \bar{r}(s) + \mathrm{sh}(s,a).\end{aligned} \tag{4}$$

If Eq. (4) holds everywhere, then $\bar{r}$ equals $\bar{r}_{1\to2}$ up to a shaping, and so equals the ground truth r up to a shaping. For instance, $\bar{r}$ and sh can be obtained by minimizing:

$$\mathcal{L}(\bar{r},\mathrm{sh}) = \sum_{s,a}\left(\bar{r}_{1\to2}(s,a) - \bar{r}(s) - \mathrm{sh}(s,a)\right)^2.$$

This loss is convex in the case of linear parameterisations of $\bar{r}$ and sh and particularly in tabular discrete MDPs. Once Eq. (4) holds, $\bar{r}$ is known to recover the ground truth reward function up to a constant under deterministic environments (Fu et al., 2017). However, in our general approach, we do not focus on state-only reward function and, except in the empirical verification of this statement in our result section 6.1, we aim at recovering a state-action reward function $\bar{r}(s,a)$.

Therefore, knowing exactly two consecutive policies and the whole model (the dynamics $\mathcal{P}$, the discount factors γ and the trade-off α) we can recover the reward function up to a shaping, and even up to a constant if the reward is known to be a state-dependent function.

5. Learning from improving trajectories

In practice, the *observer* has no access to the *learner*'s sequence of policies $\{\pi_1, ...\pi_K\}$, but can only see trajectories of states and actions explored by the *learner*. Let's assume that the *observer* is given a set of trajectories $\{\mathcal{D}_1, ...\mathcal{D}_K\}$, following a set of unknown improving policies:

$$\begin{aligned}\mathcal{D}_1 &= \{(a_1^1, s_1^1), \dots, (a_1^T, s_1^T)\} \sim \pi_1 \\ &\vdots \\ \mathcal{D}_K &= \{(a_K^1, s_K^1), \dots, (a_K^T, s_K^T)\} \sim \pi_K\end{aligned}$$

Also in practice, the learner may follow a different learning approach than soft policy iterations.

5.1. Trajectory-consistent reward function

The immediate solution is to infer the sequence of policies $\{\hat{\pi}_1, ...\hat{\pi}_K\}$, for example by likelihood maximization, and then to learn a consistent reward function that explains all policy improvements. Following Theorem 2, at each improvement, a first step is to recover the sequence of improvement-specific shaped reward functions $\{\bar{r}_{1\to2}, \dots, \bar{r}_{K-1\to K}\}$.

5.2. Learning the target rewards

In practice, we found that training the targets $\bar{r}_{k\to k+1}(s,a)$ with separated networks for the policy terms $\pi_{k+1}(a|s)$ and the divergence terms $\mathrm{KL}(\pi_k(.|s')\|\pi_{k+1}(.|s'))$ reduces the variance of the targets and improves the quality of the learned rewards.

Policies are learned by maximizing the likelihood of trajectories with parameterized distributions $\hat{\pi}_{\theta_k}$, with an entropic regularizer that prevents the learned policy from being too deterministic,

$$\mathcal{J}(\{\theta_k\}) = \sum_{k=1}^{K-1}\sum_{s,a\in\mathcal{D}_k} \ln \hat{\pi}_{\theta_k}(a|s) - \lambda\mathcal{H}(\hat{\pi}_{\theta_k}(.|s)).$$

Note that this regularization is not linked to the entropy used to soften the reinforcement learning objective of Eq. (2). Divergences are learned afterward by training a parameterized function $\rho_{\omega_k}(s)$ to minimize the loss:

$$\mathcal{L}(\{\omega_k\}) = \sum_{k=1}^{K-1}\sum_{s,a\in\mathcal{D}_k}\left(\rho_{\omega_k}(s) - \ln\frac{\hat{\pi}_{\theta_k}(a|s)}{\hat{\pi}_{\theta_{k+1}}(a|s)}\right)^2.$$

5.3. Consistency loss

Then, we would like to have Eq (4) holding at each improvement $k \to k+1$ with one consistent function $\bar{r}_\phi$. This can be obtained by minimizing over ϕ and a set of parameters $\{\psi_k\}$ the following loss:

$$\mathcal{L}(\phi,\{\psi_k\}) = \\ \sum_{k=1}^{K-1}\sum_{s,a,s'\in\mathcal{D}_k}\left(\bar{r}_k(s,a,s') - r_\phi(s,a) + \mathrm{sh}_{\psi_k}(s,s')\right)^2, \tag{5}$$

Algorithm 1 Recovering trajectory-consistent reward

input trajectories $\{\mathcal{D}_1, \dots, \mathcal{D}_N\}$
for $i = 1$ **to** N_θ **do** {train target policies $\hat{\pi}_{\theta_k}$}
  $\forall k, \theta_k \leftarrow \theta_k + \eta_\theta \nabla_{\theta_k} \mathcal{J}(\{\theta_k\})$
end for
for $i = 1$ **to** N_ω **do** {train target divergences ρ_{ω_k}}
  $\forall k, \omega_k \leftarrow \omega_k - \eta_\omega \nabla_{\omega_k} \mathcal{L}(\{\omega_k\})$
end for
for $i = 1$ **to** N_{ϕ_0} **do** {initialize reward $r_\phi = \ln \pi_\phi$}
  $\phi \leftarrow \phi + \eta_\phi \nabla_\phi \sum_{a,s \sim \mathcal{D}_K} \ln \pi_\phi(a|s)$
end for
for $i = 1$ **to** $N_{\phi;\psi}$ **do** {train reward and shaping}
  $\phi \leftarrow \phi - \eta_\phi \nabla_\phi \mathcal{L}(\phi, \{\psi_k\})$
  $\forall k, \psi_k \leftarrow \psi_k - \eta_\psi \nabla_{\psi_k} \mathcal{L}(\phi, \{\psi_k\})$
end for

where $\bar{r}_k(s, a, s') = \alpha \pi_{\theta_{k+1}}(a|s) + \alpha\gamma\rho_{\omega_k}(s')$ and $\text{sh}_{\psi_k}(s, s') = f_{\psi_k}(s) - \gamma f_{\psi_k}(s')$. Notice that contrary to Section 4.3, we consider a reward function that depends on state-action pairs. This makes initialization easier (see Section 5.4) and allows separating shapings that are improvement-dependant from the core common reward. This can also give better empirical results, if the dynamics does not change (Fu et al., 2017).

In the case of discrete MDPs with tabular parameters for ϕ and $\{\psi_k\}$, this method relies on policy inference accuracy: the longer the trajectories, the closer the reward function to the ground truth. However, with larger environments, performing directly the minimization of the loss $\mathcal{L}(\phi, \{\psi_k\})$ results in local minima that fail at generalizing the rewards to unknown states.

5.4. Reward initialization

One simple and efficient trick to prevent this issue consists in initializing the reward function with any standard imitation learning process based on the last observed trajectory. For instance, assuming that the last two trajectories are optimal and by consequence identical, the result of Theorem 2 would give $\ln \pi_K(a|s) \propto \bar{r}_K(s, a)$, so an initialization of the reward function can be obtained under the form $r_\phi(s, a) = \ln \pi_\phi(a|s)$ by looking for the parameter ϕ that maximizes the log-likelihood of the last trajectory. The resulting reward function is then improved by searching for the set of parameters ϕ and $\{\psi_k\}$ that minimize the loss given by Eq. (5) over all observed trajectories, as shown in Algorithm 1.

6. Experiments

The quality of a recovered reward function $\bar{r}$ is measured by the maximal score of an agent trained in the same environment but rewarded by $\bar{r}$ instead of the true rewards. While standard IRL recovers a reward function that ideally leads an apprentice to the observed expert's policy, we expect a reward recovered from LfL to lead an *observer* to outperform the observed *learner*, which was stopped before reaching maximal performance.

-1 Start	-1	-1	-1	-12
-1	-1	-1	-1	-1
-1	-1	-1 Reset	-1	-1
-1	-1	-1	-1	-1
0	-1	-1	-1	+10 Reset

Figure 2. Grid world. The middle point is avoided because of the dynamics rather than the associated reward. At the down-left corner, a small reward attracts the path that leads to the objective, situated at the down-right corner.

Table 1. Comparison of score $\mathcal{J}(\pi)$ between the *learner*'s two policies and an *observer* using an optimal policy based on the recovered state-action reward $\bar{r}_{1\to2}$ or the state-only reward $\bar{r}_\phi$ after regression described in section 4.3. Regrets are computed with respect to the maximal entropy-regularizeed return.

agent	used reward	policy	score	regret
optimal	r	π^*_{soft}	5.68	0
learner	r	π_1	-19.7	25.4
	r	π_2	0.72	4.95
observer	$\bar{r}_{1\to2}$	π^*_{soft}	5.68	.e-13
	$\bar{r}_\phi$ (state-only)	π^*_{soft}	5.68	.e-10

6.1. Grid world

Fig. 2 displays the discrete and deterministic grid MDP we consider for illustrating our theoretical results. We use a discount factor $\gamma = 0.96$ and a trade-off factor $\alpha = 0.3$. Our first verification involves two policies exactly known, one being uniform over the action space and the other being the immediate soft policy improvement:

$$\pi_1(.|s) = \mathcal{U}(\mathcal{A}) \text{ and } \pi_2 = \text{SPI}_r\{\pi_1\}.$$

We apply Theorem 2 to recover a reward function $\bar{r}_{1\to2}$ and we verify that:

- the score of an agent trained with $\bar{r}_{1\to2}$ is maximal (Table 1);
- a regression searching for a state-only reward function $\bar{r}_\phi$ recovers the ground truth (Fig. 3).

We also use this discrete environment to show the genericity of our model w.r.t. the *learner*'s RL algorithm, compar-

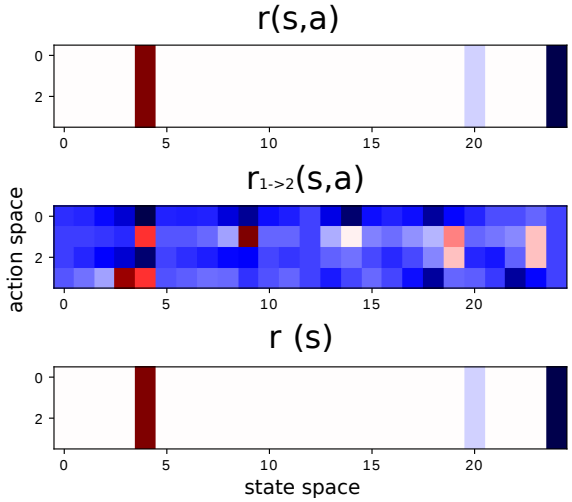

Figure 3. Ground truth reward (up), state-action function $\bar{r}_{1\to 2}$ from Theorem 2 (middle) and state function $\bar{r}_\phi$ after regression described in section 4.3 (down). $\bar{r}_\phi$ recovers the ground truth up to a small constant (not visible on the color scale).

Table 2. Comparison of score $\mathcal{J}(\pi)$ between the *learner*'s best policy and an *observer* using an optimal policy based on the recovered reward function $\bar{r}_\phi$ from observed trajectories of 1000 state-action couples at each improvement. Scores are averaged over ten runs. The second column reports the number of observed improvements (K) performed by the *learner* for each algorithm.

learner	K	**learner score**	**observer score**
SPI	3	4.18	**4.68 ± 0.24**
SVI	20	3.59	**4.73 ± 0.61**
Q-learning	50	**3.99 ± 0.88**	**3.65 ± 0.78**
rand. impro.	10	1.76 ± 2.64	**3.95 ± 0.49**

ing the results from different RL algorithms used by the *learner*: soft policy iterations (SPI) (as expected by the model), soft value iterations (Haarnoja et al., 2017) (SVI), Q-learning (Watkins, 1989) and random improvements, generated by randomly interpolating between the uniform policy and the optimal policy. In all cases, the *observer* models the *learner* as performing soft policy iterations with $\alpha_{\text{model}} = 0.7$, while the true parameter used for soft value and policy iterations as well as for the score evaluations is $\alpha = 0.3$. The policy associated to our Q-learning implementation is a softmax distribution $\exp\{\frac{Q}{\alpha}\}$. Unlike the previous experiment, here the *observer* has no access to the exact policies. Instead, at each *learner*'s policy update, the *observer* is provided with a trajectory of 1000 new sampled state-action couples and we use Algorithm 1 to recover a state-action reward $\bar{r}_\phi(s,a)$. Results are reported in Table 2. It shows that LfL is rather agnostic to the actual *learner*'s RL algorithm and the *observer* outperforms or equals the *learner*, whatever the original RL algorithm is.

6.2. Continuous control

To evaluate how our approach holds when dealing with large dimensions, we use the same experimental setting on continuous control tasks taken from the OpenAI gym benchmark suite (Brockman et al., 2016). The *learner*'s trajectories are obtained using Proximal Policy Optimization (PPO) (Schulman et al., 2017). Using PPO is motivated by two reasons: the learned policy is stochastic (as expected in our entropy-regularization model) and it performs rollouts of exploration using fixed static policies, which helps an *observer* to infer the sequence of policies (the problem is harder when the observed trajectories are continuously updated after each action, for example as with SAC). In order to accelerate the *learner*'s improvements, we parallel 32 environment explorations at each step. However, the trajectories given to the *observer* only contain 1 of these 32 explorations, resulting in observations containing 2048 state-action pairs for each improvement.

Once the *observer* has recovered a reward function using Algorithm 1, it is also trained using PPO and paralleling 32 explorations at each step. The *observer* starts with a policy that clones the *learner*'s last observed rollout by maximizing the likelihood of the trajectory. In Fig. 4 we compare the evolution of the *learner*'s score during its observed improvements, and the evolution of the *observer*'s score when trained on the same environment and using the recovered reward function (comparison is done on the original environment reward). We also compare in Table 3 the maximal observed score of the *learner* with the final score of the *observer*, and the score that would be obtained using standard IRL based on the last observed policy of the *learner*. IRL scores are taken from figures in (Kostrikov et al., 2018) (Discriminator Actor Critic, or DAC) and tables from (Fu et al., 2017) (Adversarial Inverse Reinforcement Learning, or AIRL).

We normalize scores by setting to 1 the score of the last observed policy and to 0 the score of the initial one, in order to measure improvements. Yet, it is worth noting that the corresponding absolute scores are different for IRL and LfL, as we tend to stop earlier the learning agent. However, it is quite plausible that the expert trajectories used in these IRL papers are not optimal, and could be improved. Anyway, the goal of these IRL methods is to imitate a behavior, they are not designed to do better than the observed agent, and the result of Table 3 are thus quite expectable.

On most of the environments, LfL learns a reward that leads to better performance for the *observer* than for the last observed policy from the *learner*. LfL only fails at recovering a reward function for the Hopper environment. This failure could come from the fact that this simulated robot often falls on the ground during the first steps of training, resulting in strongly absorbing states perceived as

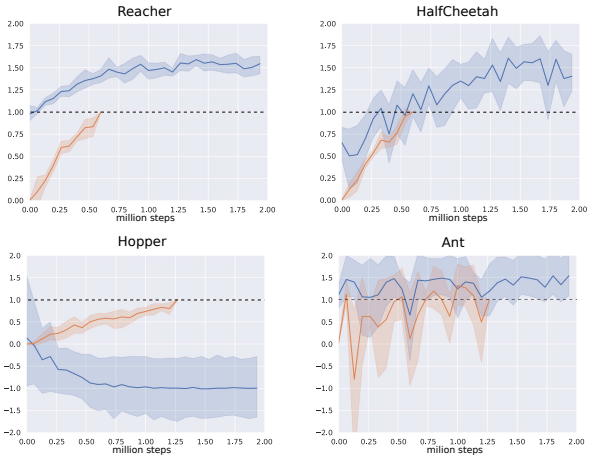

Figure 4. (Red) Evolution of the *learner*'s score during its observed improvements and (Blue) evolution of the *observer*'s score when training on the same environment and using the recovered reward function. Scores are normalized with respect to the rewards associated with the first and last observed behaviour (0 corresponds to the first observed policy while 1 corresponds to the last observed policy). The *observer* starts with a policy that clones the *learner*'s last observed policy by maximizing the likelihood of the last trajectory (in that way, the *observer* has already used the number of steps performed by the *learner* to train itself and does not start from scratch).

rewarding by the *observer*. Assessing this possible issue is left for future work.

6.3. Implementation details

In the grid-world experiments, we use tabular representations for ϕ and ψ_k. In that simple case, KL divergence terms are explicitly computed from estimated policies and dynamics instead of using a third set of parameters, and the reward initialization step is not necessary. Policy estimation is performed by maximum likelihood with tabular parameters θ_k as described in Algorithm 1. We use 10 gradient steps containing the full observed set of transitions for each trajectory $\mathcal{D}_k$. For the reward consistency regression, we use 200 gradient steps, each one summing the losses across all observed improvements. In both policy and reward regressions, we use Adam gradient descent (Kingma & Ba, 2014) with learning rate $1e^{-3}$. The random improvements are generated by randomly interpolating 15 points between the uniform and the optimal policies, and the 10 improvements in Table 2 mean that we provide the *observer* with sampled trajectories from the 10 first policies.

In the continuous control experiments, we use a neural network with one hidden layer for parameters ψ_k and ρ_k, both sharing across all k the latent layer containing 128 units with hyperbolic tangent activation. We use the actor parameters as described in PPO's original implementation (Schulman et al., 2017) for reward parameters ϕ as well as for policies parameters θ_k. Our PPO implementation conserves the set of hyperparameters described in the original paper, at the exception that we parallel 32 environment explorations at each step. All gradient descents of Algorithm 1 are performed across batches containing the whole 2048 state-action pairs for each improvement, using Adam descent with learning rate $1e^{-3}$. Like in the discrete case, the algorithm is run by modelling SPI with $\alpha_{\text{model}} = 0.7$. We use 1000 steps for the policy regressions, 100 steps for the KL divergence regressions, 3000 steps for the reward initialization and 1000 steps for the reward consistency regression. Depending on the environment, we provide the *observer* with different sets of *learner* trajectories. For Reacher that converges quickly we select early PPO updates from 10 to 20 while for HalfCheetah we rather select updates from 30 to 40. For both Hopper and Ant which give more noisy trajectories, we select updates from 10 to 30 with an increment of 5 updates. The *observer* is trained across 30 updates of PPO, summing a total of 2 million environment steps.

Table 3. Comparison between standard IRL based on the best rollouts and our LfL solution based on the whole *learner*'s observed improvements. To obtain AIRL results, the *observer* is given 50 trajectories and to obtain the reported DAC results, the *observer* needs at least 4 trajectories. AIRL and DAC values are manually reported from the respective paper results and are obtained with near-optimal experts trajectories (corresponding to 1). In our LfL setting, the *learner* has access to only one rollout of 2048 state-action couples at each improvement (the last improved policy corresponds to 1).

Environment	**AIRL**	**DAC**	**LfL** (inverted SPI)
Reacher	/	0.99	**1.54 ± 0.11**
Hopper	/	**0.99**	-0.99 ± 0.78
HalfCheetah	1.01	1.15	**1.40 ± 0.25**
Ant	0.80	1.12	**1.53 ± 0.60**

7. Related work

To the best of our knowledge, observing a sequence of policies assumed to improve in order to recover the reward function is a new setting. Here the goal is not to imitate the observed agent as in standard imitation learning or IRL, since it is not supposed to follow an optimal behaviour (even at the end of the observation). However, we discuss links to these two fields and especially to IRL (Ng et al., 2000), since these methods are sharing the aim of learning a reward function from observations of an other agent's behaviour. im In this work, we place ourselves in the framework of entropy-regularized RL and model the observed policies as following a softmax distribution weighted by a state-action value function. This model alleviates the ill-posed nature of IRL. It is actually induced by the hy-

pothesis of maximum entropy (Ziebart et al., 2008). Recent approaches, based on generative adversarial networks (GANs) (Goodfellow et al., 2014) also use the entropy-regularization framework to solve the imitation learning problem (explicitly mentioning the learning of a reward or not). Generally speaking, these methods train an apprentice with a discriminator-based reward function optimized to induce policies that match an observed behavior. This is the basis of Generative Adversarial Imitation Learning (Ho & Ermon, 2016) (GAIL) and GAN-based Guided Cost Learning (Finn et al., 2016) (GAN-GCL). GAN-GCL has the advantage to propose a structured discriminator $D(\tau)$ for an observed trajectory τ, that directly translates the reward function $R(\tau) = \ln(1 - D(\tau)) - \ln D(\tau)$. Adversarial Inverse Reinforcement Learning (Fu et al., 2017) improves this reward by learning, with the discriminator, both the reward function and the possible shaping as a separated state-function. Our work shares similarities with this last approach as we also learn separately the reward from the shaping. Discriminator Actor critic (Kostrikov et al., 2018) (DAC) suggests a correction to the bias created by absorbing states (that we mentioned in section 6.2), and combines Twin Delayed Deep Deterministic policy gradient (Fujimoto et al., 2018) (TD3) with AIRL, resulting in a improvement in sample-efficiency. Another variant of AIRL, Empowerment-based Adversarial Inverse Reinforcement Learning (Qureshi et al., 2019) (EAIRL) uses a structure for the shaping term based on a quantification of the observed agent's empowerment, defined by its ability to influence its future. This modification allows to learn disentangled state-action reward functions that significantly improve transfer learning results.

Our method to solve LfL is split in two steps of supervised classification: one estimates the policies, the other learns the rewards based on the policy discriminating losses (the log probabilities). This structure is sharing close similarities with Cascaded Supervised IRL and Structured Classification for IRL (SCIRL) (Klein et al., 2012; 2013) but fundamentally differs by the fact that LfL doesn't assume the Bellman optimality but soft policy improvements.

Policy improvements is also somehow used in preference-based IRL (Christiano et al., 2017; Ibarz et al., 2018) where a learning agent frequently asks a human to chose the best between two policies, and improves its knowledge about the reward function from this preference. Our solution for LfL could certainly be used for human preference-based learning and *vice-versa*. Yet this work differs from LfL in to ways: i) the agent inferring the reward function needs information about its own policies, and ii) the learned reward function has no intent to approach the ground truth even up to a shaping. Similarly, score-based IRL (El Asri et al., 2016) that learns a reward from rated trajectories requires human intervention to annotate trajectories and doesn't guarantee to recover the actual environment reward.

Goal recognition in agent modelling problems is another field related to this work. In multi-agent learning problems, modelling the opponents goal helps at finding rich and adaptive social behaviour like cooperation or negotiation. It is worth mentioning Learning with Opponent Learning Awarness (Foerster et al., 2018) (LOLA) where agents assume opponents are following a policy-gradient algorithm to predict and shape their own learning steps. LOLA supposes that opponents are motivated by symmetric goals and does not directly address the goal recognition problem. A similar opponent modelling approach, Modeling Others using Oneself (Raileanu et al., 2018), suggests to learn the goal of the opponent into a latent and arbitrary representation, that would explain the observed updates as if this goal representation was given as input of the observing agent. Like in LfL, the goal is inferred from the observed agent's sub-optimal learning behaviour. However, this approach models qualitative goals and requires to experience "oneself" rewards in order to model others' goals.

8. Conclusion

In this paper, we introduced the "Learning from a Learner" (LfL) problem, a new setting that aims at recovering a reward function from the observation of an agent that improves its policy over time. Unlike standard Inverse Reinforcement Learning approaches, LfL does not intend to imitate the observed behaviour, but to learn a reward function that leads to actually solve the (unknown) task and hence to potentially outperform the observed behaviour.

We propose a first approach to address this problem, based on entropy-regularized reinforcement learning. For this purpose, we model the observed agent (the *learner*) as performing soft policy improvements and we show that under this assumption, it is possible to recover the actual reward function up to a shaping. We propose an algorithm that alleviates this shaping by learning a reward function which explains consistently a set of observed trajectories generated by improving policies. Our experiments show the rightness of our theoretical assertions as well as the genericity of the method when facing different types of RL agents and in the case of continuous state-action spaces.

Although we do not claim we solved the general LfL problem, we consider the results presented in this paper as inspirational for further works. They indeed show that observation of a learning agent may lead to enhanced agents that outperform their tutor. To go beyond our findings, we think that our method can be significantly improved by addressing common IRL issues such as absorbing states bias or using learner's empowerment. Also, different models than soft policy improvement could be worth investigating.

Acknowledgements

Alexis Jacq would like to thank the Portugal FCT who funded him through grant SFRH/BD/105782/2014 before this work.

References

Boularias, A., Kober, J., and Peters, J. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 182–189, 2011.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017.

El Asri, L., Piot, B., Geist, M., Laroche, R., and Pietquin, O. Score-based inverse reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 457–465, 2016.

Finn, C., Christiano, P., Abbeel, P., and Levine, S. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Fujimoto, S., van Hoof, H., and Meger, D. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

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

Haarnoja, T., Tang, H., Abbeel, P., and Levine, S. Reinforcement learning with deep energy-based policies. *arXiv preprint arXiv:1702.08165*, 2017.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

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

Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., and Amodei, D. Reward learning from human preferences and demonstrations in atari. In *Advances in Neural Information Processing Systems*, pp. 8022–8034, 2018.

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

Klein, E., Geist, M., Piot, B., and Pietquin, O. Inverse reinforcement learning through structured classification. In *Advances in Neural Information Processing Systems*, pp. 1007–1015, 2012.

Klein, E., Piot, B., Geist, M., and Pietquin, O. A cascaded supervised learning approach to inverse reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 1–16. Springer, 2013.

Kostrikov, I., Agrawal, K. K., Dwibedi, D., Levine, S., and Tompson, J. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *International Conference on Representation Learning*, 2018.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Nachum, O., Norouzi, M., Xu, K., and Schuurmans, D. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2775–2785, 2017.

Neu, G., Jonsson, A., and Gómez, V. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

Ng, A. Y., Harada, D., and Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *Icml*, pp. 663–670, 2000.

Qureshi, A. H., Boots, B., and Yip, M. C. Adversarial imitation via variational inverse reinforcement learning.

In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJlmHoR5tQ.

Raileanu, R., Denton, E., Szlam, A., and Fergus, R. Modeling others using oneself in multi-agent reinforcement learning. *arXiv preprint arXiv:1802.09640*, 2018.

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

Sutton, R. S., Barto, A. G., et al. *Reinforcement learning: An introduction*. MIT press, 1998.

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

Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

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