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?

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.
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 of 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 $s \in S$ and actions $a \in A$ performed by the learner, and makes two assumptions:

- The learner’s behaviour is motivated by a reward function $r : S \times A \rightarrow \mathbb{R}$.
- 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) $M = (S, A, \mathcal{P}, r, \gamma)$ where $S$ is a set of states, $A$ a set of actions, $\mathcal{P}(s'|s, a)$ a transition distribution, $r(a, s)$ a reward function and $\gamma$ 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:

$$J(\pi) = \mathbb{E}_\pi \left[ \sum_{t\geq0} \gamma^t r(a_t, s_t) \right].$$

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

3. Greedy improvements

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

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[ \sum_{t\geq0} \gamma^t r(s_t, a_t) \mid s_0 = s, a_0 = a \right].$$

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 $\pi_1$, we define the space $G(\pi_1)$ of greedily-improved policies as follows:

$$\pi_2 \in G(\pi_1) \Leftrightarrow \forall s \pi_2(.|s) = \arg\max_{\pi(.|s)} \mathbb{E}_{a \sim \pi(.|s)} [Q^{\pi_1}(s, a)].$$

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

$$J(\pi_2) > J(\pi_1).$$
4. Recovering rewards from soft improvements

Entropy-regularized RL prohibits the emergence of deterministic policies (e.g., 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:

\[
J_{\text{soft}}(\pi) = \mathbb{E}_\pi \left[ \sum_{t \geq 0} \gamma^t (r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))) \right], \tag{2}
\]

where \(H\) refers to the Shannon entropy,

\[
H(\pi(\cdot|s)) = -\mathbb{E}_{a \sim \pi(\cdot|s)} [\ln \pi(a|s)],
\]

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

\[
Q_{\text{soft}}(s_1, a_t) = r(s_t, a_t) + \mathbb{E}_\pi \left[ \sum_{t \geq t} \gamma^{l-t} (r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))) \right].
\]

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

\[
Q_{\text{soft}}(s, a) = r(s, a) + \gamma \mathbb{E}_{s' \sim \alpha, a'} [Q_{\text{soft}}(s', a') - \alpha \ln \pi(a'|s')] \tag{3}.
\]

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

\[
\pi_2(a|s) \propto \exp \left\{ \frac{Q_{\text{soft}}(s, a)}{\alpha} \right\}. \tag{3}
\]

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 see 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 \(\hat{r}(a, s) = r(a, s) + f(s) - \gamma \mathbb{E}_{s'|s,a} [f(s')]\). 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, \(r_1 : S \times A \to \mathbb{R}\) and \(r_2 : S \times A \to \mathbb{R}\) be two reward functions, and \(Q_{\text{soft}}^{r_1}\) and \(Q_{\text{soft}}^{r_2}\) be the associated soft \(Q\)-functions. Then, for any function \(g : S \to \mathbb{R}\), the two following assertions are equivalent:

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

**Proof.** The proof is provided in the appendix. \(\square\)

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 \(r_1 : S \times A \to \mathbb{R}\), \(r_2 : S \times A \to \mathbb{R}\) and \(g : S \to \mathbb{R}\) be such that

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

Greedy policies are invariant under this reward transform:

\[
\text{SPI}_{r_1}\{\pi\} = \text{SPI}_{r_2}\{\pi\}.
\]

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 $\pi_1$ and $\pi_2$ be two consecutive policies given by soft policy iterations ($\pi_2 = \text{SPI}_1\{\pi_1\}$). Then a reward $\bar{r}_{1\rightarrow 2}(s,a)$ explaining the soft improvement is given by

$$\bar{r}_{1\rightarrow 2}(s,a) = \alpha \ln \pi_2(a|s) + \alpha \gamma E_{a'\sim \pi_1(s)}[\text{KL}(\pi_1(.|s')||\pi_2(.|s'))],$$

with $\text{KL}(\pi_1(.|s)||\pi_2(.|s)) = E_{a\sim \pi_1(s)}[\ln \frac{\pi_2(a|s)}{\pi_1(a|s)}].$

Indeed, there exists a function $f_{1\rightarrow 2} : S \rightarrow \mathbb{R}$ such that

$$\bar{r}_{1\rightarrow 2}(s,a) = r(s,a) + f_{1\rightarrow 2}(s) - \gamma E_{a'\sim \pi_1(s)}[f_{1\rightarrow 2}(s')],$$

and $\bar{r}_{1\rightarrow 2}$ has the same unique optimal policy as $r$.

Proof. The proof is provided in the appendix. \qed

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 $\bar{r} : S \rightarrow \mathbb{R}$ and a shaping $f : S \rightarrow \mathbb{R}$ such that:

$$\bar{r}_{1\rightarrow 2}(s,a) = \bar{r}(s) + f(s) - \gamma E_{a'\sim \pi_1(s)}[f(s')].$$

If Eq. (4) holds everywhere, then $\bar{r}$ equals $\bar{r}_{1\rightarrow 2}$ 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:

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

This loss is convex in the case of linear parameterizations 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 $P$, the discount factors $\gamma$ and the trade-off $\alpha$) 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 $\{\pi_1, \ldots, \pi_K\}$, 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 $\{D_1, \ldots, D_K\}$, following a set of unknown improving policies:

$$D_1 = \{(a_1^1, s_1^1), \ldots, (a_1^T, s_1^T)\} \sim \pi_1$$

$$D_K = \{(a_K^1, s_K^1), \ldots, (a_K^T, s_K^T)\} \sim \pi_K$$

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 $\{\hat{\pi}_1, \ldots, \hat{\pi}_K\}$, 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 $\{\bar{r}_{1\rightarrow 2}, \ldots, \bar{r}_{K-1\rightarrow K}\}$.

5.2. Learning the target rewards

In practice, we found that training the targets $\bar{r}_{k\rightarrow k+1}(s,a)$ with separated networks for the policy terms $\hat{\pi}_{k+1}(a|s)$ and the divergence terms $\text{KL}(\hat{\pi}_k(.|s')||\pi_{k+1}(.|s'))$ 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,

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

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:

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

5.3. Consistency loss

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

$$\mathcal{L}(\phi,\{\psi_k\}) = \sum_{k=1}^{K-1} \sum_{s,a,s' \in D_k} \left( \bar{r}_\phi(s,a,s') - r_\phi(s,a) + \text{sh}_{\psi_k}(s,a) \right)^2,$$
Learning from a Learner

Algorithm 1 Recovering trajectory-consistent reward

input trajectories \(\{D_1, \ldots, D_N\}\)

1: for \(i = 1\) to \(N_0\) do \{train target policies \(\hat{\pi}_{\theta_k}\)\}
2: \(\forall k, \theta_k \leftarrow \theta_k + \eta_0 \nabla_{\theta_k} J(\{\theta_k\})\)
3: end for
4: for \(i = 1\) to \(N_\omega\) do \{train target divergences \(\rho_{\omega_k}\)\)
5: \(\forall k, \omega_k \leftarrow \omega_k - \eta_\omega \nabla_{\omega_k} L(\{\omega_k\})\)
6: end for
7: for \(i = 1\) to \(N_{\phi_0}\) do \{initialize reward \(r_\phi = \ln \pi_\phi\)\}
8: \(\phi \leftarrow \phi + \eta_\phi \nabla_{\phi} \sum_{s,a,s' \sim D_K} \ln \pi_\phi(a|s)\)
9: end for
10: for \(i = 1\) to \(N_{\phi, \psi}\) do \{train reward and shaping\}
11: \(\phi \leftarrow \phi - \eta_\phi \nabla_{\phi} L(\phi, \{\psi_k\})\)
12: \(\forall k, \psi_k \leftarrow \psi_k - \eta_\psi \nabla_{\psi_k} L(\phi, \{\psi_k\})\)
13: end for

where \(\bar{r}_K(s,a,s') = \alpha \pi_{\theta_{k+1}}(a|s) + \alpha \gamma \rho_{\omega_K}(s')\) and \(sh_{\psi_k}(s,s') = f_{\psi_k}(s) - \gamma f_{\psi_k}(s')\). 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 \(\phi\) 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 \(L(\phi, \{\psi_k\})\) 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 \(\phi\) that maximizes the log-likelihood of the last trajectory. The resulting reward function is then improved by searching for the set of parameters \(\phi\) 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.

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:

\[\pi_1(\cdot|s) = U(A)\] and \(\pi_2 = SPI_\{\pi_1\}\).

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

- the score of an agent trained with \(\bar{r}_{1 \rightarrow 2}\) 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-
Learning from a Learner

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.

Table 2. Comparison of score \( J(\pi) \) between the learner’s best policy and an observer using an optimal policy based on the recovered reward function \( \bar{r}_o \) 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.

<table>
<thead>
<tr>
<th>learner</th>
<th>( K )</th>
<th>learner score</th>
<th>observer score</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPI</td>
<td>3</td>
<td>4.18</td>
<td>4.68 ± 0.24</td>
</tr>
<tr>
<td>SVI</td>
<td>20</td>
<td>3.59</td>
<td>4.73 ± 0.61</td>
</tr>
<tr>
<td>Q-learning</td>
<td>50</td>
<td>3.99 ± 0.88</td>
<td>3.65 ± 0.78</td>
</tr>
<tr>
<td>rand. impro.</td>
<td>10</td>
<td>1.76 ± 2.64</td>
<td>3.95 ± 0.49</td>
</tr>
</tbody>
</table>

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

We normalize scores by setting to 1 the score of the last observed policy of the learner, 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 LFIL, 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, LFIL learns a reward that leads to better performance for the observer than for the last observed policy from the learner. LFIL 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

\[
\phi(s,a) = \begin{cases} 
1 & \text{if } \langle s,a \rangle \text{ is absorbing} \\
0 & \text{otherwise}
\end{cases}
\]

\[
r_1 \rightarrow 2(s,a) = r(s,a) + \alpha r_\phi(s,a)
\]

\[
\bar{r}_o(s) = \frac{1}{K} \sum_{k=1}^{K} r_1 \rightarrow 2(s_k,a_k)
\]

\[
\bar{r}_o(s) = \frac{1}{K} \sum_{k=1}^{K} r(s_k,a_k) + \alpha \phi(s)
\]

\[
\text{Algorithm 1:}
\]

1. Initialize a reward function \( r_0 \), set \( K \) to its final value and \( \alpha \) to zero.
2. For each of the last \( K \) observed trajectory:
   - Update the reward function according to the update rule.
   - If the trajectory is absorbing, set \( r_0 \) to 1.
3. Output the final reward function \( \bar{r}_o \).

\[
\bar{r}_o(s) = \frac{1}{K} \sum_{k=1}^{K} r(s_k,a_k) + \alpha \phi(s)
\]
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 \( \phi \) and \( \psi_k \). 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 \( \theta_k \) as described in Algorithm 1. We use 10 gradient steps containing the full observed set of transitions for each trajectory \( D_k \). 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 \( \psi_k \) and \( \rho_k \), 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 \( \phi \) as well as for policies parameters \( \theta_k \). Our PPO implementation conserves the set of hyperparameters described in the original paper, at the exception that we parallelise 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.

<table>
<thead>
<tr>
<th>Environment</th>
<th>AIRL</th>
<th>DAC</th>
<th>LfL (inverted SPI)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reacher</td>
<td>0.99</td>
<td>-0.99 ± 0.78</td>
<td></td>
</tr>
<tr>
<td>Hopper</td>
<td>0.99</td>
<td>1.54 ± 0.11</td>
<td></td>
</tr>
<tr>
<td>HalfCheetah</td>
<td>1.01</td>
<td>1.40 ± 0.25</td>
<td></td>
</tr>
<tr>
<td>Ant</td>
<td>0.80</td>
<td>1.53 ± 0.60</td>
<td></td>
</tr>
</tbody>
</table>

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. 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 hyp-
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 $\tau$, 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 Awareness (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.
References


A. Proof of Lemma 1

Using the Bellman evaluation equation, we have

$$Q_{\text{soft}}^{\pi_2}(s, a) = r_2(s, a) + \gamma E_{s', a'} [Q_{\text{soft}}^{\pi_1}(s', a') - \alpha \ln \pi(a'|s')] ,$$

$$\Leftrightarrow Q_{\text{soft}}^{\pi_2}(s, a) + g(s) = r_2(s, a) + g(s) - \gamma E_{s', a'}[g(s')] + E_{s', a'} \left[ Q_{\text{soft}}^{\pi_1}(s', a') + g(s') - \alpha \ln \pi(a'|s') \right]$$

$$\Leftrightarrow Q_{\text{soft}}^{\pi_1}(s, a) = r_1(s, a) + \gamma E_{s', a'} [Q_{\text{soft}}^{\pi_1}(s', a') - \alpha \ln \pi(a'|s')] .$$

This proves the stated result.

B. Proof of Theorem 1

Let $\pi' = \text{SPI}_{r_2}\{\pi\}$. We have, for any state-action couple,

$$\pi'(a|s) = \frac{\exp\{Q_{\text{soft}}^{\pi_1}(s, a)\}}{Z(s)} \cdot \frac{\exp\{Q_{\text{soft}}^{\pi_1}(s, a) + g(s)\}}{Z(s) \exp g(s)} \cdot \frac{\exp\{Q_{\text{soft}}^{\pi_1}(s, a)\}}{Z'(s)} .$$

The last equations means that $\pi' = \text{SPI}_{r_2}\{\pi\}$, and so $\text{SPI}_{r_1}\{\pi\} = \text{SPI}_{r_2}\{\pi\}$, in particular if the optimal policy is unique, it is sufficient to notice that an optimal policy is the unique policy being greedy respectively to itself, that is, $\pi_* = \text{SPI}_{r_1}\{\pi_*\}$. So, $\text{SPI}_{r_1}\{\pi\}$ and $\text{SPI}_{r_2}\{\pi\}$ have necessarily the same fixed point.

C. Proof of Theorem 2

Let $\pi_1$ and $\pi_2$ be two successive policies such that $\pi_2 = \text{SPI}_{r_2}\{\pi_1\}$. This means that, for any state $s$ and action $a$, we have:

$$\pi_2(a|s) = \frac{\exp\{Q_{\text{soft}}^{\pi_1}(s, a)\}}{Z_1(s)} ,$$

where $Z_1(s)$ is a normalization factor. Taking the logarithm of this expression, we get:

$$\alpha \ln \pi_2(a|s) = Q_{\text{soft}}^{\pi_1}(s, a) - \ln Z_1(s) = Q_{\text{soft}}^{\pi_2}(s, a) + f(s) .$$

According to Lemma 1, this means that $\alpha \ln \pi_2(a|s)$ is the $Q$-function associated to the shaped reward function $\bar{r}(s, a) = r(s, a) + f(s) - \gamma E_{s'} [f(s')]$ for the policy $\pi_1$. Using the fact that this $Q$-function satisfies the Bellman equation, we have:

$$\alpha \ln \pi_2(a|s) = \bar{r}(s, a) + \gamma E_{s', a'} [\alpha \ln \pi_2(a'|s') - \alpha \ln \pi_1(a'|s')]$$

$$= \bar{r}(s, a) - \alpha \gamma E_{s'}[\text{KL}(\pi_1(.|s')\|\pi_2(.|s'))]$$

$$\Leftrightarrow \bar{r}(s, a) = \alpha \ln \pi_2(a|s) + \alpha \gamma E_{s'\sim P(.|a,s)} [\text{KL}(\pi_1(.|s')\|\pi_2(.|s'))] .$$

The fact that both $r$ and $\bar{r}$ have the same optimal policy is due to theorem 1. This proves the stated result.