Imitation Learning over Heterogeneous Agents with Restraining Bolts

Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, Fabio Patrizi

Sapienza University of Rome
{lastname@diag.uniroma1.it}

Motivation

- A common problem in Reinforcement Learning (RL) is that often the reward function is hard to express. This can be overcome by resorting to Inverse Reinforcement Learning (IRL), which consists in first obtaining a reward function from a set of execution traces generated by an expert agent, and then making the learning agent learn the expert's behavior – this is known as Imitation Learning (IL).
- Typical IRL solutions rely on a numerical representation of the reward function, which raises problems related to the adopted optimization procedures.
- We describe an IL method where the execution traces generated by the expert agent, possibly via planning, are used to produce a logical (as opposed to numerical) specification of the reward function, to be incorporated into a device known as Restraining Bolt (RB) [1]. The RB can be attached to the learning agent to drive the learning process and ultimately make it imitate the expert.
- We show that IL can be applied to heterogeneous agents, with the expert, the learner and the RB using different representations of the environment's actions and states, without specifying mappings among their representations.

Restraining Bolts

A Restraining Bolt (RB) [1] is a tuple $RB = \langle \mathcal{L}, \{(\varphi_i, r_i)\}_{i=1}^{m} \rangle$, where each φ_i is an LDL_f formula [2] over a set of fluents $\mathcal{L}$ and each r_i is a reward value. In Figure 1 you can see a standard RL scenario, with the environment, the RL agent, its features and the reward function, extended with the RB, i.e., a device that observes the environment and, based on its own fluents, offers rewards to the agent. Fluents constitute the RB's representation of the environment state and need not match the RL agent features.

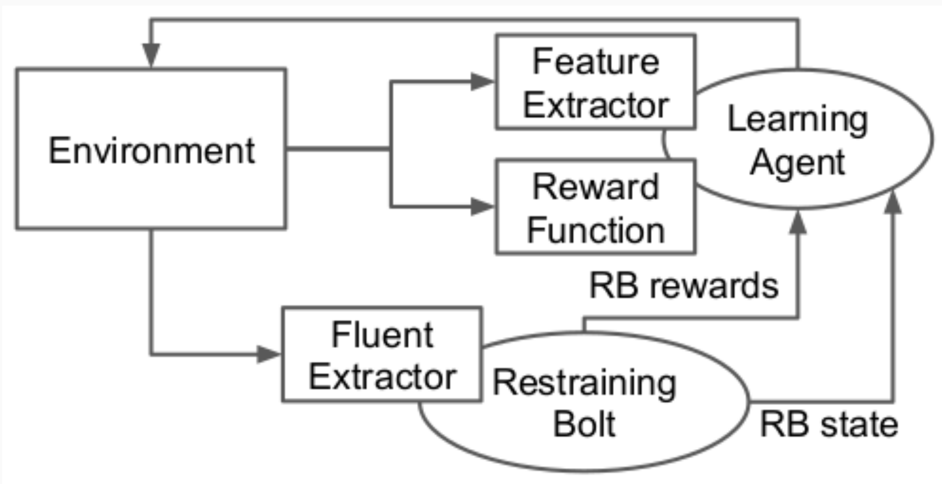

Figure 1: The RB setting.

Formulas φ_i specify the behaviors that should be rewarded, each with its respective r_i, to constrain an agent's behavior to fulfill high-level (i.e., fluent-based) goals. In our context, the RB state is the equivalent DFA of the formula. RBs were introduced in [2], to constrain an agent's behavior to fulfill high-level (i.e., fluent-based) goals.

Problem definition

We use RBs to address the problem of transferring a task from an expert to a learner agent. The task is represented by a formula φ in a RB or, more precisely, by the corresponding DFA $\mathcal{A}$. As a result, we consider RBs of the form $\langle \mathcal{L}, \mathcal{A}, r \rangle$, where $\mathcal{A}$ is a DFA representing an $\text{LTL}_f/\text{LDL}_f$ formula and r is a reward value associated with the accepting states of $\mathcal{A}$.

The agent can execute optimal policies of a given target task represented by a DFA $\mathcal{A}$, but cannot make the corresponding reward function explicit; in other words, the agent knows how to accomplish the task but cannot describe it. As the agent executes the policy, some traces are produced, some of which are desirable (positive) and some other are not. The expert can correctly classify the traces as positive or negative, based on its own state representation.

On the other hand, the traces can also be seen from the RB perspective, through the RB sensors. Thus, from each state, the fluents can be extracted to produce the corresponding representation in the RB space. Notice the expert does not know anything about fluents, in particular, it cannot interpret them, as belonging to a different representation space. In fact, the expert is not even aware of the RB. This scenario is illustrated in Figure 2.

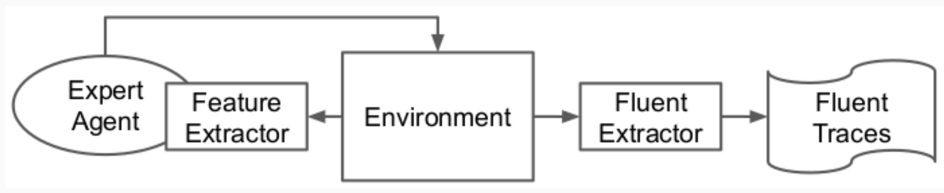

Figure 2: Trace generation for RB-IRL.

The problem we address in this paper is that of reconstructing a DFA $\mathcal{A}$ that is consistent with the traces, i.e., that accepts all of its positive traces and none of its negative. The approach proposed in this paper allows for generating a new RB $\langle \mathcal{L}, \mathcal{A}', r \rangle$, where $\mathcal{A}'$ is the DFA built from the traces, and r is a reward value associated with the accepting states of $\mathcal{A}'$. After the training phase, the generated RB can be placed on a learner agent to drive the learning process of a behaviour imitating the expert's one, using the technique explained in [1]. Crucially, the observation space and the action space of the expert and the learning agent can be different in general, as long as they allow to solve the task, and no explicit mapping between them is needed.

Solution Method

- Make the expert interacting with an environment, and producing a set of high-level traces $\mathcal{T}$ (offline).
- Learn a DFA that models the traces using the L^* [3] algorithm, by using the set traces $\mathcal{T}$ as oracle (see Figure 3;
- The learning agent learns the optimal policy against the learned Restraining Bolt.

Figure 3: RB's DFA learning.

References

[1] Giuseppe De Giacomo et al. "Foundations for Restraining Bolts: Reinforcement Learning with LTLf/LDLf Restraining Specifications". In: *Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS)*. 2019.

[2] Giuseppe De Giacomo and Moshe Y. Vardi. "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces". In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. 2013.

[3] Dana Angluin. "Learning regular sets from queries and counterexamples". In: *Information and Computation* (1987).

Case Studies

- Breakout: The goal is to break the columns of bricks from left to right. The expert is able to shoot (Figure 4), whereas the learner can only hit the ball (Figure 5).
- Sapientino: the goal is to visit color in a certain order in a gridworld-like environment. The expert agent can use directional moves (Figure 6) whereas the learning agent can only use differential moves (turn left/right, move forward/backward) (Figure 7).
- Minecraft: the goal is to accomplish a sequential composite task, getting resources and using tools placed on a grid. The expert agent was able to "teleport" himself to any resource/tool on the grid (Figure 8), and the learning agent could only walk through the grid (Figure 9).

In all the cases, the traces generated by the expert agent were complete enough to learn a good restraining bolt to let the learning agent to learn the optimal policy.

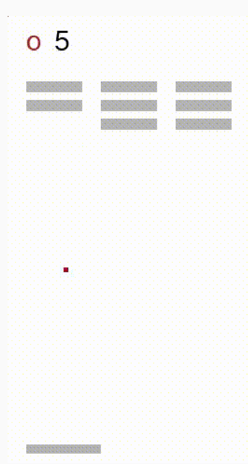

Fig. 4: Breakout (expert)

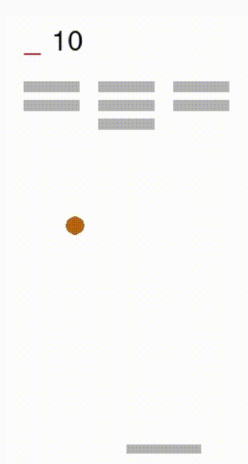

Fig. 5: Breakout (apprentice)

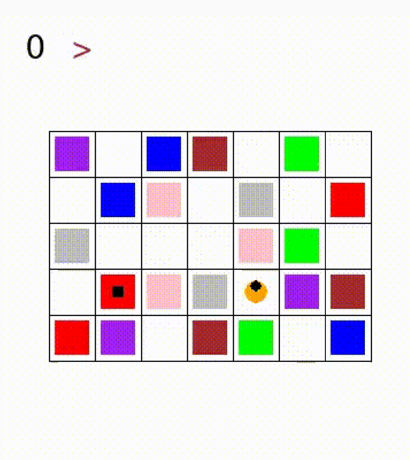

Fig. 6: Sapientino (expert)

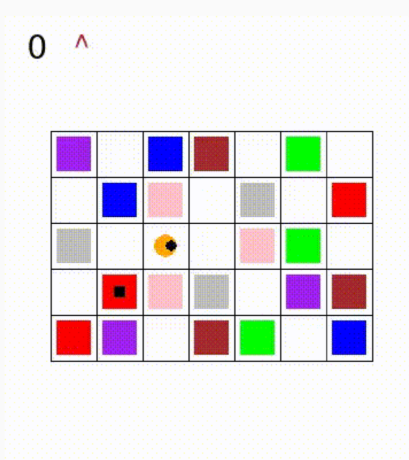

Fig. 7: Sapientino (apprentice)

Fig. 8: Minecraft (expert)

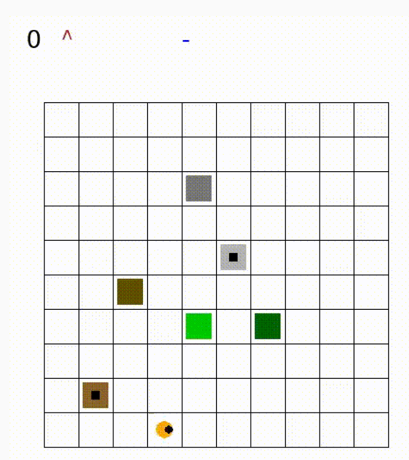

Fig. 9: Minecraft (apprentice)

Code and description of the experiments can be found at: https://whitemech.github.io/Imitation-Learning-over-Heterogeneous-Agents-with-Restraining-Bolts/