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). 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.

Formulas $\varphi$ 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. 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.

Refraining Bolts

A Refraining Bolt (RB) $\mathcal{B}$ is a tuple $\mathcal{B} = (L \{\varphi_i, r_i\})^m_{i=1}$, where each $\varphi_i$ is an LTLf formula $\mathcal{B}$ over a set of fluents $L$ and each $r_i$ is a reward value. In Figure 1 you can see a standard IL 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.

Solution Method

- Make the expert interacting with an environment, and producing a set of high-level traces $T$ (offline).
- Learn a DFA that models the traces using the $L^*$ algorithm, by using the set traces $T$ as oracle (see Figure 2).
- The learning agent learns the optimal policy against the learned Refraining Bolt.

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 $\varphi$ in a RB or, more precisely, by the corresponding DFA $A$. As a result, we consider RBs of the form $(L \{\varphi, r\})$, where $A$ is a DFA representing an LTLf/LDLf formula and $r$ is a reward value associated with the accepting states of $A$.

The agent can execute optimal policies of a given target task represented by a DFA $A$, 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 3.

References


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 Refraining Bolt (RB).
- 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.

Restrainting Bolts

A Restrainting Bolt (RB) is a tuple $RB = (\mathcal{L}, \{\varphi_i, r_i\})^m_{i=1}$, where each $\varphi_i$ is an LTLf formula over a set of fluents $\mathcal{L}$ and each $r_i$ is a reward value. In Figure 1 you can see a standard IL 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.

Solution Method

- Make the expert interacting with an environment, and producing a set of high-level traces $T$ (offline).
- Learn a DFA that models the traces using the $L^*$ algorithm, by using the set traces $T$ as oracle (see Figure 2).
- The learning agent learns the optimal policy against the learned Restrainting Bolt.

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 $\varphi$ in a RB or, more precisely, by the corresponding DFA $A$. As a result, we consider RBs of the form $(\mathcal{L}, \varphi, r)$, where $A$ is a DFA representing an LTLf/LDLf formula and $r$ is a reward value associated with the accepting states of $A$.

The agent can execute optimal policies of a given target task represented by a DFA $A$, 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 3.

References