Hierarchical Reinforcement Learning

Carlos Diuk
Rutgers University, USA

Michael Littman
Rutgers University, USA

INTRODUCTION

Reinforcement learning (RL) deals with the problem of an agent that has to learn how to behave to maximize its utility by its interactions with an environment (Sutton & Barto, 1998; Kaelbling, Littman & Moore, 1996). Reinforcement learning problems are usually formalized as Markov Decision Processes (MDP), which consist of a finite set of states and a finite number of possible actions that the agent can perform. At any given point in time, the agent is in a certain state and picks an action. It can then observe the new state this action leads to, and receives a reward signal. The goal of the agent is to maximize its long-term reward.

In this standard formalization, no particular structure or relationship between states is assumed. However, learning in environments with extremely large state spaces is infeasible without some form of generalization. Exploiting the underlying structure of a problem can effect generalization and has long been recognized as an important aspect in representing sequential decision tasks (Boutilier et al., 1999).

Hierarchical Reinforcement Learning is the subfield of RL that deals with the discovery and/or exploitation of this underlying structure. Two main ideas come into play in hierarchical RL. The first one is to break a task into a hierarchy of smaller subtasks, each of which can be learned faster and easier than the whole problem. Subtasks can also be performed multiple times in the course of achieving the larger task, reusing accumulated knowledge and skills. The second idea is to use state abstraction within subtasks: not every task needs to be concerned with every aspect of the state space, so some states can actually be *abstracted away* and treated as the same for the purpose of the given subtask.

BACKGROUND

In this section, we will introduce the MDP formalism, where most of the research in standard RL has been done. We will then mention the two main approaches used for learning MDPs: model-based and model-free RL. Finally, we will introduce two formalisms that extend MDPs and are widely used in the Hierarchical RL field: semi-Markov Decision Processes (SMDPs) and Factored MDPs.

Markov Decision Processes (MDPs)

A Markov Decision Process consists of:

- a set of states S
- a set of actions A
- a transition probability function: $Pr(s' \mid s, a)$, representing the probability of the environment transitioning to state s' when the agent performs action a from state s. It is sometimes notated $T(s, a, s')$.
- a reward function: $E[r \mid s, a]$, representing the expected immediate reward obtained by taking action a from state s.
- a discount factor $\gamma \in (0, 1]$, that downweights future rewards and whose precise role will be clearer in the following equations.

A *deterministic policy* $\pi: S \rightarrow A$ is a function that determines, for each state, what action to take. For any given policy π, we can define a *value function* V^{π}, representing the *expected infinite-horizon discounted return* to be obtained from following such a policy starting at state s:

$$V^{\pi}(s) = E[r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots].$$

Bellman (1957) provides a recursive way of determining the value function when the reward and transition probabilities of an MDP are known, called the *Bellman equation*:

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \, S} T(s, \pi(s), s') V\pi(s'),$$

commonly rewritten as an *action-value function* or *Q-function*:

$$Q\pi(s,a) = R(s, a) + \gamma \sum_{s' \, S} T(s, a, s') V\pi(s').$$

An *optimal policy* $\pi^*(s)$ is a policy that returns the action *a* that maximizes the value function:

$$\pi^*(s) = argmax_a \, Q^*(s,a)$$

States can be represented as a set of *state variables or factors*, representing different features of the environment: $s = <f_1, f_2, f_3, \ldots, f_n>$.

Learning in Markov Decision Processes (MDPs)

The reinforcement-learning problem consists of determining or approximating an optimal policy through repeated interactions with the environment (*i.e.*, based on a sample of *experiences* of the form <*state – action – next state – reward*>).

There are three main approaches to learning such an optimal or near-optimal policy:

- **Policy-search methods:** learn a policy directly via evaluation in the environment.
- **Model-free (or direct) methods:** learn the policy by directly approximating the *Q* function with updates from direct experience.
- **Model-based (or indirect) methods:** first learn the transition probability and reward functions, and use those to compute the *Q* function by means of , for example, the *Bellman equations*.

Model-free algorithms are sometimes referred to as the *Q-learning family* of algorithms. See Sutton (1988) or Watkins (1989) for the first best-known examples. It is known that model-free methods make inefficient use of experience, but they do not require expensive computation to obtain the *Q* function and the corresponding optimal policy.

Model-based methods make more efficient use of experience, and thus require less data, but they involve an extra *planning* step to compute the value function, which can be computationally expensive. Some well-known algorithms can be found in the literature (Sutton, 1990; Moore & Atkeson, 1993; Kearns & Singh, 1998; and Brafman & Tennenholtz, 2002).

Algorithms for reinforcement learning in MDP environments suffers from what is known as the *curse of dimensionality*: an exponential explosion in the total number of states as a function of the number of *state variables*. To cope with this problem, *hierarchical methods* try to break down the intractable state space into smaller pieces, which can be learned independently and reused as needed. To achieve this goal, changes need to be introduced to the standard MDP formalism. In the introduction we mentioned the two main ideas behind hierarchical RL: task decomposition and state abstraction. Task decomposition implies that the agent will not only be performing single-step actions, but also full subtasks which can be extended in time. Semi-Markov Decision Processes (SMDPs) will let us represent these extended actions. State abstraction means that, in certain contexts, certain aspects of the state space will be ignored, and states will be grouped together. Factored-state representations is one way of dealing with this. The following section introduces these two common formalisms used in the HRL literature.

Beyond MDPs: SMDPs and Factored-State Representations

We'll consider the limitations of the standard MDP formalism by means of an illustrating example. Imagine an agent whose task is to exit a multi-storyed office building. The starting position of the agent is a certain office in a certain floor, and the goal is to reach the front door at ground level. To complete the task, the agent has to first exit the room, find its way through the hallways to the elevator, take the elevator to the ground floor, and finally find its way from the elevator to the exit. We would like to be able to reason in terms of subtasks (e.g., *"exit room", "go to elevator", "go to floor X"*, etc.), each of them of different durations and levels of abstraction, each encompassing a series of