Róbert Leó Jónsson (robertt20@ru.is))

Chapter 3

Exercise 3.1

> Devise three example tasks of your own that fit into the MDP framework, identifying for each its states, actions, and rewards. Make the three examples as different from each other as possible. The framework is abstract and flexible and can be applied in many different ways. Stretch its limits in some way in at least one of your examples.

Controlling a rocket. A reinforcement learning agent is to control a rocket that should fly into orbit.

The agent will want to know all about the forces acting on the rocket, and the current position, velocity, acceleration, heading, and rotation forces of the rocket. It might need to know the fuel level, and the temperature of the engine in order to complete its task.

Finally, most rockets have discrete stages; the first transition might perhaps be to discard the level 1 boosters.

Thus the state might be composed of

- Position
- Velocity
- Acceleration
- Fuel status
- Engine temperature
- Heading
- Rotation
- The phase of the launch

The agent should control the engine and controls of the rocket. Maybe the agent also controls when the rocket releases its boosters. The agent might also want to warn the humans that something is wrong, so it might want to send a distress signal.

Thus the actions could be

- Accelerate engine
- Turn (or something to control the heading of the engine)
- Transition into next phase of launch
- Send distress signal

The agent should fly the rocket so that it heads forward, with minimum turbulance, with fuel efficiency, and a nice ascent into orbit. Its reward might be some weighted combination of these factors.

Programming. We could face a reinforcement learner create programs for tasks, given a description in some form. In fact, this has already been done.

The agent might receive as a state the programming language it should use, if it should not decide that itself. Of course it should want a description of the program.

An important thing to know when programming is the current state of the code, so the agent should receive that as well.

Now, the agent might want to run the program to see if it produces the desired output. So that might be a possible part of the state.

For the actions, the agent might choose to input a character to the file, delete text, or any other aciton that modifies the text. The agent might also decide to run to the program to see the results. Finally, when the agent is happy, it should be able to submit its code.

Cook food. We might produce a chef agent. It might control some kind of robot that has access to all necessary appliances in a kitchen, and the ingredient, and the agent's goal is to produce the requested dish.

The states might consist of visual input, temperature of stoves and ovents, and the sensorymotor data from the robot's actuators.

The agent needs to control the robot, so the actions should include all the possible controls of the robot. Also, for better automation, stove and oven controls might be actions seperate from the robot's movement.

The rewards should be negative for breaking stuff, or otherwise reaking havoc in the kitchen, and there should be positive rewards for creating the requested dish, where the size of the reward is a funciton of the dish's quality.

Exercise 3.2

> Is the MDP framework adequate to usefully represent all goal-directed learning tasks? Can you think of any clear exceptions?

I cannot find any counterexamples.

Exercise 3.3

The actions that the agent can perform instantaneously at any time step should be the defined actions for the agent. The choice of where to drive should not be an output for a driving agent, as that is not how you drive a car, in the immediate sense.

The choice of where to drive should be something that the agent decides makes plans on how to execute, and then uses its actuators/available actions to perform.

A driving agent that would output a coordinate of where to drive would not be a driving agent, rather a guidance agent.

Of course, a driving agent could have some other agent that controls where it should drive, and the driving agent implements that plan, but that would be a seperate agent, distinct form the driving agent.

Now, as stated, the actions should be something that can be outputted, and executed immediately. If the agent outputs a rotational acceleration of the steering wheel, or simply its goal angle should be something that could be immediately implemented by some microcontroller or other machinery.

Therefore that is an acceptable action.

Now, if there is some automatic system that can better automate a task than the agent can, say, a good brake system in a car that prevents skidding, the agent should not directly control the force of the brake pads, but should simply gain access to the brake pedal, and let the brake system do the rest.

Exercise 3.4

Table 1. The Table

s	a	s'	r	$p(s', r \mid s, a)$
high	search	high	1	0.4
high	search	high	0	0.6
high	search	low	1	0.4
high	search	low	0	0.6
high	wait	high	1	0.1
high	wait	high	0	0.9
high	wait	low	1	0.1
high	wait	low	0	0.9
low	search	low	1	0.2
low	search	low	0	0.8
low	wait	low	1	0
low	wait	low	0	1
low	recharge	low	1	0
low	recharge	low	0	1

Exercise 3.5

The formula simply becomes

$$p(s', r \mid s, a) = 1, \text{ for all } s \in \mathcal{S} \setminus \mathcal{S}^+$$

Exercise 3.6

The return would not just be related to $-\gamma^K$, it would equal $-\gamma^K$, where K is the number of steps until failure.

In the continuing task formulation, the agent might also take into account the next failure, in the next reset, in a given state.

Exercise 3.7

The expected return is always one, as the agent always escapes the maze eventually, and when it does, there is no discrimination between states. All states get an expected return of one.

To communicate to the agent that it should navigate the maze efficiently, we should penalize it for every move it makes, so that it wants to escape the maze quickly. We only instructed it to *eventually* escape the maze, no matter how long it takes.

Exercise 3.8

$$G_5 = 0 \tag{1}$$
$$G_4 = R_5 + \gamma G_5 = 2 + 0.5 \cdot 0 = 2 \tag{2}$$
$$G_3 = R_4 + \gamma G_4 = 3 + 0.5 \cdot 2 = 4 \tag{3}$$
$$G_2 = R_3 + \gamma G_3 = 6 + 0.5 \cdot 4 = 8 \tag{4}$$
$$G_1 = R_2 + \gamma G_2 = 2 + 0.5 \cdot 8 = 6 \tag{5}$$
$$G_0 = R_1 + \gamma G_1 = -1 + 0.5 \cdot 6 = 2 \tag{6}$$

Exercise 3.9

We know that

$$G_1 = \sum_{i=0}^{\infty} 7\gamma^i = 7\sum_{i=0}^{\infty} \gamma^i \tag{7}$$
$$= 7\frac{1}{1-\gamma} = 7\frac{1}{0.1} \tag{8}$$
$$= 7 \cdot 10 = 70 \tag{9}$$

So

$$G_0 = 2 + \gamma G_1 \tag{10}$$
$$= 2 + 0.9 \cdot 70 \tag{11}$$
$$= 2 + 7 = 65 \tag{12}$$

Exercise 3.10

Let

$$\Gamma = \sum_{k=0}^{\infty} \gamma^k$$

Then

$$\Gamma - \gamma\Gamma = \sum_{k=0}^{\infty} \gamma^k - \sum_{k=1}^{\infty} \gamma^k \tag{13}$$
$$= \gamma^0 = 1 \tag{14}$$

And hence

$$(1-\gamma)\Gamma = 1 \tag{15}$$
$$\implies \Gamma = \frac{1}{1-\gamma} \tag{16}$$

Exercise 3.11

If A is the set of actions, and R the set of possible rewards, it is

$$\mathbb{E}(R_{t+1}) = \sum_{a \in A} \left[\pi(a \mid S_t) \sum_{r \in R} \left(r \sum_{s' \in S} p(s', r \mid S_t, a) \right) \right]$$

Exercise 3.12

We want to give an equation for v_π in terms of q_π and π. This can be done by taking the action-value function, and averaging out the action, that is

$$v_\pi(s) = \mathbb{E}\left[q_\pi(s,a)\right] = \sum_{a \in A} \pi(a \mid s) q_\pi(s,a)$$

Exercise 3.13

We want to give an equation for q_π in terms of v_π and $p(s', r \mid s, a)$. We define $\mathbb{E}_x\left[f(x)\right]$ as the expectation of f with respect to x. With that definition, we see that

$$\begin{aligned} q_\pi &= \mathbb{E}_{s',r}\left[r + \gamma v(s')\right] && (17)\\ &= \sum_{s' \in S} \pi(a \mid s) \sum_{r \in R} p(s', r \mid s, a)(r + \gamma v(s')) && (18) \end{aligned}$$

Exercise 3.14

We represent the state s as its coordinate $s = (x, y)$.

Thus,

$$\begin{aligned} v_\pi(2,2) &= 0.7 && (19)\\ v_\pi(2,1) &= 2.5 && (20)\\ v_\pi(1,2) &= 0.7 && (21)\\ v_\pi(3,2) &= 0.4 && (22)\\ v_\pi(2,3) &= -0.4 && (23) \end{aligned}$$

Now, the Bellman equation for the value function v_π is

$$v_\pi(s) = \sum_{a} \pi(a \mid a) \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_\pi(s')\right]$$

We how that the Bellman equation for the value function holds for $v_\pi(2,2)$ with respect to its neighbors. The set of actions is $A = (UP, RIGHT, DOWN, LEFT)$. However, all teh actions from the state $s = (2,2)$ result in a reward of 0.

Also, the environment is deterministic. We can now perform the calculation:

$$\begin{aligned} v_\pi(2,2) &= \sum_{a} \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_\pi(s')\right] && (24)\\ &= \frac{1}{4}(\gamma v_\pi(2,1) + \gamma v_\pi(1,2) + \gamma v_\pi(1,2) + \gamma v_\pi(3,2) + \gamma v_\pi(2,3)) && (25)\\ &= \frac{0.9}{4}(2.5 + 0.7 + 0.4 + (-0.4)) && (26)\\ &= 0.72 \approx 0.7 && (27) \end{aligned}$$

Exercise 3.15

If we add a constant c to all the rewards, each reward becomes $R_t + c$, so the return becomes

$$G'_t = (R_{t+1} + c) + (R_{t+2} + c) + \cdots \tag{28}$$

$$= \sum_{k=0}^{\infty} \gamma^k (R_{t+k+1} + c) \tag{29}$$

$$= \sum_{k=0}^{\infty} \gamma^k c + \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{30}$$

$$= \frac{c}{1-\gamma} + \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{31}$$

$$= G_t + \frac{c}{1-\gamma} \tag{32}$$

So the value v'_π becomes

$$v'_\pi(s) = \mathbb{E}\left[G'_t \mid S_t = s\right] \tag{33}$$

$$= \mathbb{E}\left[G_t + \frac{c}{1-\gamma} \mid S_t = s\right] \tag{34}$$

$$= \mathbb{E}\left[G_t \mid S_t = s\right] + \mathbb{E}\left[\frac{c}{1-\gamma} \mid S_t = s\right] \tag{35}$$

$$= \mathbb{E}\left[G_t \mid S_t = s\right] + \frac{c}{1-\gamma} \tag{36}$$

$$= v_\pi(s) + \frac{c}{1-\gamma} \tag{37}$$

$$= v_\pi(s) + v_c \tag{38}$$

where the constant added is $v_c = \frac{c}{1-\gamma}$

Exercise 3.16

Lets say the task is episodic, and some episode lasts k steps. Then, at time t, the return is

$$G'_t = \sum_{i=0}^{k-t-1} \gamma^i (R_{t+i+1} + c) \tag{39}$$

$$= \sum_{i=0}^{k-t-1} \gamma^i c + \sum_{i=0}^{k-t-1} R_{t+i+1} \tag{40}$$

$$= \frac{c(\gamma^{k-t-1} - 1)}{\gamma - 1} + \sum_{i=0}^{k-t-1} R_{t+i+1} \tag{41}$$

$$\tag{42}$$

So here we see that the return depends on the number of steps remaining until the end of the episode.

The feature of the continuing tasks that caused there to be a constant added to the value (or return), is that the sum was infinite.