Chapter 3 solutions **Hanna Gábor**

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.*

   - Games are an easy example. For example, in Snake the state is the current board, actions are turning left or right or continuing forward. Rewards are whether you picked up a chip or not.
   - Public transportation development. Suppose we measured when and how many people use which line. (E.g. if we have an electronic ticket system, then this can be easily done.) Also suppose that we can simulate how people choose transportation vehicles. The actions would be like building a bus line here or a tram line there. The state is the graph created from the current lines and the possible new lines with their costs. Your current budget is also part of the state. The reward can be the number of people using public transport (the more the better) or the overall time needed for transportation (the less the better). We can add many things to this example. For example, you might need to bar some roads while you're building a new railway line.

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

   I'm not sure I'm working with the correct definition of "goal-directed" learning tasks in this exercise.

   An exception is if we can't make good intermediate rewards and reaching a good end-state is extremely unlikely. For example, if I don't know anything about how a car works and want a program that can learn to make a car. I can give a robot tools and give a reward if it builds a working car, but it will never get into a state like that.

   Another exception is if we can't really simulate the thing and can't try different methods in real life. E.g. I would like to make people happier. Should I start researching positive psychology or teach programming to poor kids or do something else? It's impossible to try taking different routes, to measure the resulting happiness or make a useful simulation for a virtual learning algorithm.

3. *Consider the problem of driving. You could define the actions in terms of the accelerator, steering wheel, and brake, that is, where your body meets the machine. Or you could define them farther out—say, where the rubber meets the road, considering your actions to be tire torques. Or you could define them farther in—say, where your brain meets your body, the actions being muscle twitches to control your limbs. Or you could go to a really high level and say that your actions are your choices of where to drive. What is the right level, the right place to draw the line between agent and environment? On what basis is one location of the line to be preferred over another? Is there any fundamental reason for preferring one location over another, or is it a free choice?*

   If it's about learning to drive a car, then the actions should be defined in terms of the accelerator, steering wheel and brake. If it's about learning to plan my days more efficiently then the actions should be defined as where I drive.

I'm pretty sure that these are the good boundaries, but can't think of good rules.

4. *Give a table analogous to that in Example 3.3, but for $p(s', r|s, a)$. It should have columns for s, a, s', r, and $p(s', r|s, a)$, and a row for every 4-tuple for which $p(s', r|s, a) > 0$.*

s	a	s'	r	$p(s', r\|s, a)$
high	search	high	r_{search}	α
high	search	low	r_{search}	$1-\alpha$
low	search	high	-3	$1-\beta$
low	search	low	r_{search}	β
high	wait	high	r_{wait}	1
low	wait	low	r_{wait}	1
low	recharge	low	0	1

5. *The equations in Section 3.1 are for the continuing case and need to be modified (very slightly) to apply to episodic tasks. Show that you know the modifications needed by giving the modified version of (3.3).*

$$\sum_{s' \in \mathcal{S}^+} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$$

6. *Suppose you treated pole-balancing as an episodic task but also used discounting, with all rewards zero except for -1 upon failure. What then would the return be at each time? How does this return differ from that in the discounted, continuing formulation of this task?*

   The return would be $-\gamma^K$ where K is the number of time steps before the next failure. Opposed to the continuing formulation, only the reward for the next failure is included in the return. In the continuing formulation, all the later failures have an effect on the return. (Although the later the failure occurs, the smaller the effect becomes.)

7. *Imagine that you are designing a robot to run a maze. You decide to give it a reward of +1 for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes—the successive runs through the maze—so you decide to treat it as an episodic task, where the goal is to maximize expected total reward (3.7). After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?*

   The expected total reward is always 1, the agent has no motivation to escape the maze early.

8. *Suppose $\gamma = 0.5$ and the following sequence of rewards is received $R_1 = -1$, $R_2 = 2, R_3 = 6, R_4 = 3$, and $R_5 = 2$, with $T = 5$. What are $G_0, G_1, ..., G_5$? Hint: Work backwards.*

$$
\begin{aligned}
G_5 &= 0 \\
G_4 &= 2 \\
G_3 &= 3 + 0.5 \cdot 2 = 4 \\
G_2 &= 6 + 0.5 \cdot 4 = 8 \\
G_1 &= 2 + 0.5 \cdot 8 = 6 \\
G_0 &= -1 + 0.5 \cdot 6 = 2
\end{aligned}
$$

9. *Suppose* $\gamma = 0.9$ *and the reward sequence is* $R_1 = 2$ *followed by an infinite sequence of 7s. What are* G_1 *and* G_0*?*

   $G_1 = 7 + \gamma 7 + \gamma^2 7 + \cdots = 7 \sum_{i=0}^{\infty} \gamma^i = 7 \frac{1}{1-\gamma} = \frac{7}{0.1} = 70$

   $G_0 = 2 + 0.9 \cdot 70 = 65$