

Reinforcement Learning With Hado van Hasselt

Yaro Kazakov

September 14, 2024

Abstract

These are study notes to prepare myself for PhD applications.

1 Lecture 3 - Finite Markov Decision Processes

We try to solve MDPs in the rest of this book. This problem involves evaluative feedback, as in bandits, but also an associative aspect—choosing different actions in different situations. MDPs are a classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards. Thus MDPs involve **delayed reward** and the need to **tradeoff immediate and delayed reward**. Whereas in bandit problems we estimated the value $q_*(a)$ of each action a , in MDPs we estimate the value $q_*(s, a)$ of each action a in each state s , or we estimate the value $v_*(s)$ of each state given optimal action selections.

As in all of artificial intelligence, there is a tension between **breadth of applicability** and **mathematical tractability**.

1.1 The Agent-Environment Interface

MDPs are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The learner and decision maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment.

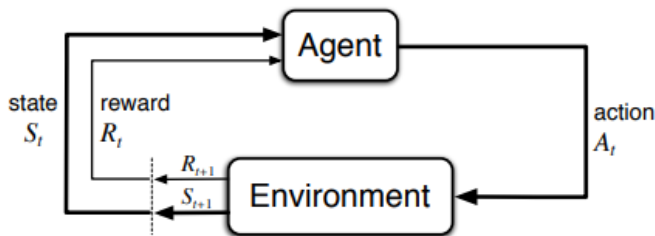


Figure 3.1: The agent–environment interaction in a Markov decision process.

Figure 1: Agent-environment Interaction

The MDP and agent together thereby give rise to a sequence or trajectory that begins like this: $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots$

In a finite MDP, the sets of states, actions, and rewards (S , A , and R) all have a **FINITE** number of elements.

That is, there is a probability of those values occurring at time t , given particular values of the preceding state and action:

$p(s', r|s, a) \doteq \Pr S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a$, the function p defines the **dynamics of the MDP**. The dot over the equals sign means it is a definition. Here this reminds us that:

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

In MDP, the probabilities given by p completely characterize the environment's dynamics. The state must include information about all aspects of the past agent-environment interaction that make a difference for the future. If it does, then the state is said to have the Markov property.

Markov property is assumed throughout this book.

For the **state transition probabilities** (not joint distribution), we do the following: $p(s'|s, a) \doteq \Pr S_t = s' | S_{t-1} = s, A_{t-1} = a$. $\sum_{r \in R} p(s', r|s, a)$, why is that?

This is a concept of **marginalization**. Marginalization in probability theory refers to **the process of summing (or integrating) over the possible values of one or more variables** to obtain the probability distribution of a subset of the variables. The goal is to "marginalize" out (i.e., eliminate) certain variables from a joint distribution to focus on the probability distribution of the remaining variables.

We can also compute state-action-next-state: $r(s, a) \doteq E[R_t | S_{t-1} = s, A_{t-1} = a] = \sum_{r \in R} r \sum_{s' \in S} p(s', r|s, a)$

The general rule we follow is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment.

1.1.1 Exercise 3.1

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.

1. Walk from a bedroom to a kitchen. States: in a bedroom, walking to the kitchen, in the kitchen. Actions: Walking, standing still. Rewards: 0, -1.
2. Room air-conditioning. State: Hot, OK, cold. Action: Cool down, warm up, maintain, idle. Reward: -1, -1, 1, 0.
3. Cruise control (assuming no cars and no traffic). States: Too fast, too slow, OK. Action: Accelerate, decelerate, maintain. Rewards: -1, -1, 0.

1.1.2 Exercise 3.2

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

1. No, a situation where states are not self-containing cannot be modelled with MDPs. For example a robot that is controlled through a virtual camera cannot send action signals to the robot based on one snapshot. The snapshot might relate to a situation where a robot is charging forward, hopping or turning. At some moment time t , this snapshot would be insufficient to judge whether a robot is doing any of the three actions. Hence, state history would be required.

1.1.3 Exercise 3.3

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?

1. High-level abstraction would be navigation. Actions are route choices, states are your position on the map, reward is the distance differential between your position and destination or time to get there. Every additional minute is a penalty of -1.
2. Autonomous driving algo is a low-level abstraction. Actions are steering, accelerating and braking.

1.1.4 Exercise 3.4

The joint probability would be the same because for every s,a,s' triple there is only one deterministic reward. Use the formula 3.6

1.2 Returns and Episodes

We seek to maximize expected return. Return is defined as before:

$$G_t = R_{t+1} + R_{t+2} + \dots + R_T$$

This notion works if there is a **final step**. That is when the agent-environment interaction breaks naturally into subsequences, called **episodes**. Each episode ends in a special terminal state.

On the other hand, an ongoing task of a robot with a life long span would be called **continuous tasks**.

Discounting is also an important concept. $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$ where γ is between 0 and 1

$G_t = R_{t+1} + \gamma G_{t+1}$, **this recursive formula is very important for RL.**

Note that even though the return is infinite, it is still finite if the reward is nonzero and constant. if $\gamma < 1$. If $R_t = +1$

$$G_t = 1/(1 - \gamma)$$

1.2.1 Exercise 3.5

S becomes S^+ because we need to include the terminal space explicitly.

1.3 Exercise 3.8

$$G_5 = 0, G_4 = 2 + 0.5 * G_5 = 2, G_3 = 3 + 0.25 * G_4 = 3.5 \text{ etc..}$$

1.3.1 Exercise 3.9

Same as before but using sum to infinity.

1.4 Policies and Value Functions

State value function under/for policy π $v_{\pi}(s) \doteq E_{\pi}[G_t | S_t = s] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s]$ Action-value function for policy π : $q_{\pi}(s, a) \doteq E_{\pi}[G_t | S_t = s, A_t = a] = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a]$

1.4.1 Exercise 3.12

Exercise 3.12 Give an equation for v in terms of q and π .

This is yet another marginalization Rule $P(X|Y) = \sum_{z \in Z} P(X|Y, Z) * P(Z|Y)$ the state value function is the weighted sum of the state action value function, with the weights equal to the probabilities of selecting each action.

$$v_{\pi}(s) = \sum_a \pi(a|s) * q(s, a)$$

1.4.2 Exercise 3.13

Just like Bellman but without the first term, i.e. averaging by the probability of taking action a given s .

1.5 Bellman equation for v_{π}

$$v_{\pi}(s) = \sum_a \pi(a, s) \sum_{(s', r)} p(s', r|s, a) [r + \gamma v_{\pi}(s')] \text{ for all } s \in S.$$

The Bellman equation (3.14) averages over all the possibilities, weighting each by its probability of occurring. It states that the value of the start state must equal the (discounted) value of the expected next state, plus the reward expected along the way.

1.6 Exercise 3.14

$P(a=s)$ is equal to $1/4$, four-term probability is 1, because the state is fully deterministic once the action is taken. r is equal to 0. Plug in the numbers from the near cells and multiply by 0.9.