Finite Markov Decision Processes: Reinforcement Learning

Sanyam Singhal

March 18, 2022

1 Summary

1. MDPs are formalization of sequential decision making process. The learner/ decision maker is called the agent and thing it interacts with is called the environment. Any aspect of the problem that the agent does not have an absolute control over are part of the environment.

2. At each time step t (assuming discete time steps), the agent interprets some representation of the environment as the state $S_t \in \mathcal{S}$ and takes an action $A_t \in \mathcal{A}(S_t)$, consequently it lands in a state $S_{t+1} \in \mathcal{S}$ and fetches a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$. In finite MDPs, the sets of states, actions and rewards ($\mathcal{S}, \mathcal{A}$ and $\mathcal{R}$) are finite.

3. The Markov property defines that the next immediate state and reward only depend on the present state and action i.e. immediate future is independent of immediate past, conditioned on present. So, we can completely define the dynamics of the MDP by the probability function p:

$$p(s', r|s, a) = Pr(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a) \tag{1}$$

4. From $p(s', r|s, a)$ we can obtain other quantities of interest. Such as the state-transition probability:

$$p(s'|s, a) = \sum_{r \in \mathcal{R}} p(s', r|s, a) \tag{2}$$

   Expected reward upon taking an action a from a state s:

$$r(s, a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a) \tag{3}$$

   Expected reward given we land in a state s', starting from a state s and upon taking an action a:

$$r(s, a, s') = \sum_{r \in \mathcal{R}} r \frac{p(s', r|s, a)}{p(s'|s, a)} \tag{4}$$

5. The goal of the agent is formalized in terms of the reward it receives from its choice of actions. The agent's objective is to maximize the total reward it receives. More formally, the goal is to maximize the expected value of the cumulative sum of a received scalar signal (reward hypothesis).

6. We define the return G_t as (with T as the final time step):

$$G_t = R_{t+1} + R_{t+2} + \cdots + R_T \tag{5}$$

   So, our goal is to maximize the expected return. At time step T, the agent lands in a terminal state that marks the end of a sequence of agent-environment interaction. Such sequences are called episodes. An example would be the ending of a chess game, in either a win/lose/draw for the agent. Learning tasks that run in such episodes are called episodic tasks (if not, then they are called continuing tasks). In episodic tasks, the non-terminal states are represented as $\mathcal{S}$ and all states including terminal states are represented as $\mathcal{S}^+$.

7. To accommodate continuing tasks, we introduce discounted return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \tag{6}$$

$\gamma \in [0, 1]$ is called the discounting rate. $\gamma = 0$ implies that the agent only maximizes the immediate reward while choosing the action A_t. As $\gamma \to 1$, the agent becomes more far sighted.

We can recursively relate the returns as:

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{7}$$

This works for all time steps $t < T$ even if termination occurs at $t + 1$, if we define $G_T = 0$.

8. To include both episodic and continuing tasks under a single notation, we assume that the episodic tasks continue to run indefinitely within the terminal state i.e. they self-loop over the terminal state while obtaining 0 reward on every move. This way, the return can be written as:

$$G_t = \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{8}$$

with the possibility that $T = \infty$ or $\gamma = 1$ but not both.

9. A policy is a mapping from states to probabilities of selecting each possible action. If the agent is following a policy π at time t then $\pi(a|s)$ is the probability that $A_t = a$ given that $S_t = s$.

10. The value function $v_\pi(s)$ of a state s under a policy π is defined as:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}[\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k | S_t = s] \ \forall \ s \in S \tag{9}$$

11. Similarly, action-value $q_\pi(s, a)$ of choosing an action a from a state s and thereafter following policy π is defined as:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}[\sum_{k=t+1}^{T} \gamma^{k-t-1} R_k | S_t = s, A_t = a] \ \forall \ s \in S \ , a \in A \tag{10}$$

12. From some algebraic manipulation, we can obtain a relation between the value of a state and the values of its successor states. Such an expression is called the Bellman equation for v_π. We can also have the Bellman equation for $q_\pi(s, a)$.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{11}$$

Since, in q_π, we fix the action, it becomes:

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \tag{12}$$

Now, note that $v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$, so:

$$q_\pi(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma \sum_{a'} \pi(a|s') q_\pi(s', a')] \tag{13}$$

These equations can be graphically visualized using backup diagrams.

13. A policy π is said to be better than another polciy π' if $v_\pi(s) \geq v_{\pi'}(s)$ for all $s \in S$. There is always one policy that is better than all other policies. It is called the optimal policy. There can be multiple optimal policies but the state values associated with them, called the optimal state values is unique.

$$v_*(s) = \max_\pi v_\pi(s) \tag{14}$$

for all $s \in S$.

Similarly, optimal action value function:

$$q_*(s,a) = \max_\pi q_\pi(s,a) \tag{15}$$

for all $s \in S$ and $a \in A$.

14. Since v_* and q_* must still satisfy Bellman equations. The corresponding equations are called Bellman optimality equations. Instead of summing over all possible actions, we choose the maximizing action:

$$v_*(s) = \max_a \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] = \max_a q_*(s,a) \tag{16}$$

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')] = \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \tag{17}$$

15. If we assign a zero value to terminal states then the Bellman optimality equations have a unique solution for finite MDPs. Many reinforcement learning methods can be understood as approximately solving the Bellman optimality equations.