Reading Summary for Reinforcement Learning: An Introduction Chapter 3 and Chapter 4

Yufeng Yuan(Arthur)

Finite Markov Decision Processes are a classical formalization of sequential decision making, where actions influence subsequent states and rewards. This interaction can be described as following: at each time step t, the agent receives some representation of the environment's state, $S_t \in \mathcal{S}$, and on that basis selects an action $A_t \in \mathcal{A}(s)$. One time step later, in part as a consequence of its action, the agent receives a numerical reward $R_t \in \mathcal{R}$, and finds itself in a new state S_{t+1}. Another important component of MDPs is the dynamics, denoted by p:

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

By definition, the conditional distribution p is an ordinary deterministic function of four arguments, which completely characterizes the environment's dynamics.

At each time step, the reward is a simple number, $R_t \in \mathbb{R}$ and the agent's goal is to maximize the total amount of reward it received. In general, we seek to maximize the expected return, where the return, denoted G_t, is defined as some specific function of the reward sequence:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where γ is a parameter $0 \leq \gamma \leq 1$ called the discount rate. The above can also be written in recursive form: $G_t = R_{t+1} + \gamma G_{t+1}$

Almost all reinforcement learning algorithms involve estimating value functions, which tell the agent how good a state or a state-action pair is and is defined in terms of future rewards that can be expected. The expectation in value functions is defined with respect to policy, which is a mapping from states to probabilities of selecting each possible action. Policies are denoted as $\pi(a|s)$ which is the probability that $A_t = a$ if $S_t = s$. The value-function of a state under policy π, denoted $v_\pi(s)$, is the expected return when starting in s and following π thereafter. For MDPs, the value function is defined as:

$$v_\pi(s) = E_\pi[G_t|S_t = s] = E_\pi\Big[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\Big|S_t = s\Big], s \in \mathcal{S}$$

Similarly, the action-value function $q_\pi(s)$ is the expected return starting from s, taking the action a, and thereafter following policy π:

$$q_\pi(s) = E_\pi[G_t|S_t = s, A_t = a] = E_\pi\Big[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\Big|S_t = s, A_t = a\Big], s \in \mathcal{S}, a \in \mathcal{A}(s)$$

A fundamental property of value functions is the recursive relationship:

$$\begin{aligned} v_\pi(s) &= E_\pi[G_t|S_t = s] \\ &= E_\pi[R_t + \gamma G_{t+1}|S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\Big[r + \gamma E_\pi[G_{t+1}|S_{t+1} = s']\Big] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)\Big[r + \gamma v_\pi(s')]\Big], s \in \mathcal{S} \end{aligned}$$

For finite MDPs, $\pi' \geq \pi$ if and only if $v_{\pi'}(s) \geq v_\pi(s)$ for all $s \in \mathcal{S}$. There is always at least one policy that is better than or equal to all other policies, which is the optimal policy. The optimal state-value function v_* is defined as: $v_*(s) = \max_\pi v_\pi(s)$. Similarly, the optimal action-value function q_* is defined as: $q_*(s) = \max_\pi q_\pi(s, a)$

Further, we can derive the Bellman optimality equation for both optimal value function and optimal action-value function:

$$\begin{aligned} v_*(s) &= \max_a E_{\pi_*}[G_t | S_t = s, A_t = a] \\ &= \max_a E_{\pi_*}[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a] \\ &= \max_a E[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \end{aligned}$$

$$\begin{aligned} q_*(s, a) &= E\Big[R_{t+1} + \gamma q_*(S_{t+1}, a') \Big| S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s', r|s, a)\Big[r + \gamma \max_a{}' q_*(s', a')\Big] \end{aligned}$$

Dynamic programming is a collection of algorithms that can be used to compute optimal policies for a MDP. We first consider policy evaluation which is to compute the state-value function v_π for an arbitrary policy π:

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_k(s')]$$

With value functions, we can do policy improvement, by making a new policy greedy with respect to the value function of the original policy:

$$\begin{aligned} \pi'(s) &= \arg\max_a q_\pi(s, a) \\ &= \arg\max_a E[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big] \end{aligned}$$

By doing iterative policy evaluation and policy improvement, we obtain policy iteration. Each policy is guaranteed to be strict improvement over the previous one unless it's already the optimal policy.

One drawback of policy iteration is that policy evaluation in it requires mutiple sweeps through the state set. One way to improve it is that stop policy evaluation just after one sweep and we obtain value iteration:

$$\begin{aligned} v_{k+1}(s) &= \max_a E[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_k(s')\Big] \end{aligned}$$

A major drawback to the DP methods so far is that they involve operations over the entire state set. Asynchronous DP methods are in-place iterative DP algorithms that are not organized in terms of systematic sweeps of the state set. These algorithms update the values of states in any order whatsoever, using whatever values of other states happen to be available.

Generalized policy iteration refers to the general idea of letting policy-evaluation and policy-improvement processes interact, independent of the granularity and other details of the two processes. In this case, the value function stabilizes only when it is consistent with the current policy, and the policy stabilizes only when it is greedy with respect to the current value function. Thus both process stabilize only when a policy has been found that is greedy with respect to its own evaluation function.