Chapter 4 solutions **Hanna Gábor**

1. *In Example 4.1, if π is the equiprobable random policy, what is $q_\pi(11, down)$? What is $q_\pi(7, down)$?*

$$q_\pi(11, \text{down}) = 0$$
$$q_\pi(7, \text{down}) = -1 + v_\pi(11) = -15$$

2. *2 In Example 4.1, suppose a new state 15 is added to the gridworld just below state 13, and its actions, left, up, right, and down, take the agent to states 12, 13, 14, and 15, respectively. Assume that the transitions from the original states are unchanged. What, then, is $v_\pi(15)$ for the equiprobable random policy? Now suppose the dynamics of state 13 are also changed, such that action down from state 13 takes the agent to the new state 15. What is $v_\pi(15)$ for the equiprobable random policy in this case?*

   In the first case

$$\begin{aligned} v_\pi(15) &= -1 + (v_\pi(12) + v_\pi(13) + v_\pi(14) + v_\pi(15))/4 \\ &= -1 + (-22 - 20 - 14 + v_\pi(15))/4 = -15 + v_\pi(15)/4 \end{aligned}$$

   From this $v_\pi(15) = 4 \cdot (-15)/3 = -20.0$

   $v_\pi(15) = -20.0$ in the second case as well, because we changed a state transition to a another state transition, where the two states have the same value.

3. *What are the equations analogous to (4.3), (4.4), and (4.5), but for action-value functions instead of state-value functions?*

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

4. *The policy iteration algorithm on page 80 has a subtle bug in that it may never terminate if the policy continually switches between two or more policies that are equally good. This is okay for pedagogy, but not for actual use. Modify the pseudocode so that convergence is guaranteed.*

   The easiest would be to break the ties in a deterministic ways when applying argmax. We can also change if old-action $\neq \pi(s)$ to

$$\sum_{s',r} p(s', r|s, \text{old-action})(r + \gamma V(s')) \neq \sum_{s',r} p(s', r|s, \pi(s))(r + \gamma V(s'))$$

5. *How would policy iteration be defined for action values? Give a complete algorithm for computing q_π, analogous to that on page 80 for computing v_π. Please pay special attention to this exercise, because the ideas involved will be used throughout the rest of the book.*

1. Initialization
$Q(s,a) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ for all $s \in \mathcal{S}$

2. Policy Evaluation
repeat
  $\Delta \leftarrow 0$
  for all $s \in \mathcal{S}$ **do**
    for all $a \in \mathcal{A}(s)$ **do**
      $q \leftarrow Q(s,a)$
      $Q(s,a) \leftarrow \sum_{r,s'} p(s',r|s,a)\Big(r + Q(s',\pi(s'))\Big)$
      $\Delta \leftarrow \max(\Delta, |q - Q(s,a)|)$
    end for
  end for
until $\Delta < \Theta$

3. Policy Improvement
policy-stable $\leftarrow$ *true*
for all $s \in \mathcal{S}$ **do**
  old-action $\leftarrow \pi(s)$
  $\pi(s) \leftarrow \text{argmax}_a\, Q(s,a)$
  If $Q(s, \textit{old-action}) \neq Q(s,\pi(s))$, then *policy-stable* $\leftarrow false$
end for
If *policy-stable*, then stop and return $Q \approx q_*$ and $\pi \approx \pi_*$; else go to 2.

6. *Suppose you are restricted to considering only policies that are ϵ-soft, meaning that the probability of selecting each action in each state, s, is at least $\epsilon/|A(s)|$. Describe qualitatively the changes that would be required in each of the steps 3, 2, and 1, in that order, of the policy iteration algorithm for v_π on page 80.*

   In step 3, we would need to make a new policy that is ϵ-soft. E.g. by assigning the probability $1 - \epsilon(|A(s)| - 1)/|A(s)|$ to the action from argmax and $\epsilon/|A(s)|$ for the other actions. The comparison should check if the sum for *old-policy*(s) matches the sum for $\pi(s)$.

   In step 2, $V(s)$ should be calculated by taking an expected value of the value on the right, based on the distribution $\pi(a|s)$.

   In step 1, we should start with an ϵ-soft policy π.

7. *Programming. Write a program for policy iteration and re-solve Jack's car rental problem with the following changes. One of Jack's employees at the first location rides a bus home each night and lives near the second location. She is happy to shuttle one car to the second location for free. Each additional car still costs \$2, as do all cars moved in the other direction. In addition, Jack has limited parking space at each location. If more than 10 cars are kept overnight at a location (after any moving of cars), then an additional cost of \$4 must be incurred to use a second parking lot (independent of how many cars are kept there). These sorts of nonlinearities and arbitrary dynamics often occur in real problems and cannot easily be handled by optimization methods other than dynamic programming. To check your program, first replicate the results given for the original problem.*

The code can be found here. The x axis shows the number of cars at the second location, while the y axis corresponds to the number of cars at the first location.