Chapter 4 Exercises

Tags: #Exercises

Exercise 4.1

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

Answer

$r = -1$ for all states except the terminal state.
$\gamma = 1$, undiscounted and episodic.

$$
\begin{aligned}
q_\pi(s, a) &= r + \gamma v_\pi(s') \\
&= -1 + v_\pi(s')
\end{aligned}
$$

$$
\begin{aligned}
q_\pi(11, down) &= -1 + v_\pi(15) \\
&= -1 + 0 \\
&= -1
\end{aligned}
$$

$$
\begin{aligned}
q_\pi(7, down) &= -1 + v_\pi(11) \\
&= -1 + -14 \\
&= -15
\end{aligned}
$$

Exercise 4.2:

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?

Answer

Case unchanged:

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

$$\begin{aligned} v_\pi(15) &= 0.25(-1 - 22 - 1 - 20 - 1 - 14 - 1 + v_\pi(15)) \\ &= 0.25(-60 + v_\pi(15)) \\ &= -\frac{15}{0.75} \\ &= -20 \end{aligned}$$

Case state 13 changed:

$$\begin{aligned} V_0(15) &= -20 \\ V_1(13) &= 0.25(-1 - 22 - 1 - 20 - 1 - 14 - 1 + V_0(15)) \\ &= 0.25(-60 - 20) \\ &= -20 \end{aligned}$$

Now since $V_0(13) = -20$, we see that is it unchanged, so $V_1(15)$ will also be unchanged from $V_0(15) = -20$.

Exercise 4.3

What are the equations analogous to (4.3), (4.4), and (4.5) for the action-value function q_π and its successive approximation by a sequence of functions $q_0, q_1, q_2, \ldots$?

Answer

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a') \mid S_t = s, A_t = a \right] \\ &= \sum_{s',r} p(s',r|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a') \right] \\ q_{k+1}(s,a) &= \sum_{s',r} p(s',r|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_k(s',a') \right] \end{aligned}$$

Exercise 4.4

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 ok for pedagogy, but not for actual use. Modify the pseudocode so that convergence is guaranteed.

Answer

1. Initialization
   $V(s) \in \mathbb{R} \cap \pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$
2. Policy Evaluation
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)
3. Policy Improvement
   $\text{policy-stable} \leftarrow true$
   For each $s \in \mathcal{S}$:
   $\text{old-action} \leftarrow \pi(s)$
   If $\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')] > \sum_{s',r} p(s', r|s, \pi(s))[r + \gamma V(s')]$: ***CHANGED HERE***
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
   $\text{If old-action} \neq \pi(s), \text{ then policy-stable} \leftarrow \text{false}$
   If policy-stable, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Exercise 4.5

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

Answer

1. Initialization
   $Q(s, a) \in \mathbb{R}$ arbitrarily for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$
2. Q-value Policy Iteration
   Loop:
   $\Delta \leftarrow 0$
   Loop for each $s \in \mathcal{S}$:
   Loop for each $a \in \mathcal{A}$:
   $q \leftarrow Q(s, a)$
   $Q(s, a) \leftarrow \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} Q(s', a')]$
   $\Delta \leftarrow max(\Delta, |q - Q(s, a)|)$
   until $\Delta < \theta$

$\pi(s) \leftarrow \arg\max_a Q(s,a)$
Return $Q \approx q_*$ and $\pi \approx \pi_*$

Exercise 4.6:

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 $\varepsilon/|A(s)|$. 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_* on page 80.

Answer

Step 3:

Policy cannot become deterministic and we must give actions other than the greedy action a probability of at least $\varepsilon/|A(s)|$, which would converge to this value as the minimum for each exploratory action. Then our greedy action would have probability $1 - \varepsilon + \varepsilon/|A(s)|$.

Step 2:

We would have to add the policy into the value function calculation. The values would be lower than v_* because we cannot commit 100% to optimal actions and we have to maintain ε probability of taking non-optimal actions.

Step 1:

We have to initialize in a way that our policy follows ε-soft, so initial probabilities must satistfy:

- $\pi(s|a) \geq \varepsilon/|A(s)|$
- $\sum_a \pi(a|s) = 1$ for all $s \in \mathcal{S}$

Exercise 4.7:

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.

Programming

Exercise 4.8:

Why does the optimal policy for the gambler's problem have such a curious form? In particular, for capital of 50 it bets it all on one flip, but for capital of 51 it does not. Why is this a good policy?

Answer

We can think of 50 as a checkpoint of halfway, where we have an advantage to staying above this so we are closer to winning. Because 50 is right where we have a single step to victory, it has a high value. Otherwise, we want to preserve our position and get to the next checkpoint, which would be 75. So for 51, we want to reduce our risk now and not fall below 50, which would place us farther from victory.

Exercise 4.9

Implement value iteration for the gambler's problem and solve it for $p_h = 0.25$ and $p_h = 0.55$. In programming, you may find it convenient to introduce two dummy states corresponding to termination with capital of 0 and 100, giving them values of 0 and 1 respectively. Show your results graphically, as in Figure 4.3. Are your results stable as $\theta \to 0$?

Programming

Exercise 4.10

What is the analog of the value iteration update (4.10) for action values, $q_{k+1}(s, a)$?

Answer

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $Q(terminal, a) = 0$

Loop:
$\Delta \leftarrow 0$
Loop for each $s \in \mathcal{S}$:
Loop for each $a \in \mathcal{A}$:
$q \leftarrow Q(s, a)$
$Q(s, a) \leftarrow \sum_{s', r} p(s', r|s, a)[r + \gamma \max_{a'} Q(s', a')]$
$\Delta \leftarrow \max(\Delta, |q - Q(s, a)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
$\pi(s) = \arg\max_a Q(s, a)$
