Exercises - Sutton, Barto: Reinforcement Learning

Pablo Carrera Flórez de Quiñones

June 2019

Contents

1 Introduction

2 Multi-armed Bandits

2.1 Exercise 1

In ε-greedy action selection, for the case of two actions and $\varepsilon = 0.5$, what is the probability that the greedy action is selected?

In ε-greedy action selection we have two possibilities:

- Chose the greedy action set, with probability $1 - \varepsilon$.
- Chose a random action, with probability ε.

so, since there are only two possible actions, the probability that the greedy action is selected is given by

$$\begin{aligned} &P(greedy) = P(greedyset)P(greedy|greedyset) + P(random)P(greedy|random) \\ &\implies P(greedy) = (1-\varepsilon) \times 1 + \varepsilon \times 0.5 \\ &\implies P(greedy) = 1 - 0.5\varepsilon \\ &\implies P(greedy) = 0.75 \end{aligned}$$

2.2 Exercise 2

Bandit example Consider a k-armed bandit problem with $k = 4$ actions denoted as 1, 2, 3, and 4. Consider applying to this problem a bandit algorithm using ε-greedy action selection, sample-average action-value estimates, and initial estimates of $Q_1(a) = 0$, for all a. Suppose the initial sequence of actions and rewards is:

- $A_1 = 1 \implies R_1 = 1$
- $A_2 = 2 \implies R_2 = 1$
- $A_3 = 2 \implies R_3 = 2$
- $A_4 = 2 \implies R_4 = 2$
- $A_5 = 3 \implies R_5 = 0$

On some of these time steps the ε case may have occurred, causing an action to be selected at random. On which time steps did this definitely occur? On which time steps could this possibly have occurred?

In one hand, in steps 2 and 5 the random possibility has definitely occurred because actions 1 and 2 were the optimal ones and not 2 and 3 respectively. In the other hand, the random possibility is likely to occur in every step since the optimal action can be chosen anyway because of the randomness, so we have no evidence of which is the case for steps 3 and 4. Finally, note than in step 1 all actions are optimal, so the final decision has to be taken by randomness anyway.

2.3 Exercise 3

In the comparison shown in Figure 2.2, which method will perform best in the long run in terms of cumulative reward and probability of selecting the best action? How much better will it be? Express your answer quantitatively.

Given a ε-greedy algorithm, the expected value of an actions is given by

$$q_*(a) = (1 - \varepsilon)q_{opt}(a) + \varepsilon\bar{q}(a)$$

where $q_{opt}(a)$ is the value of the optimal action to take and $\bar{q}(a)$ is the mean value of all the possible actions. Looking at this expression we can see that in the long term, when the exploration is complete, the algorithm with lower ε will reach a higher performance, since the already known optimal action will be selected more times. This comparison can be seen in Figure 1

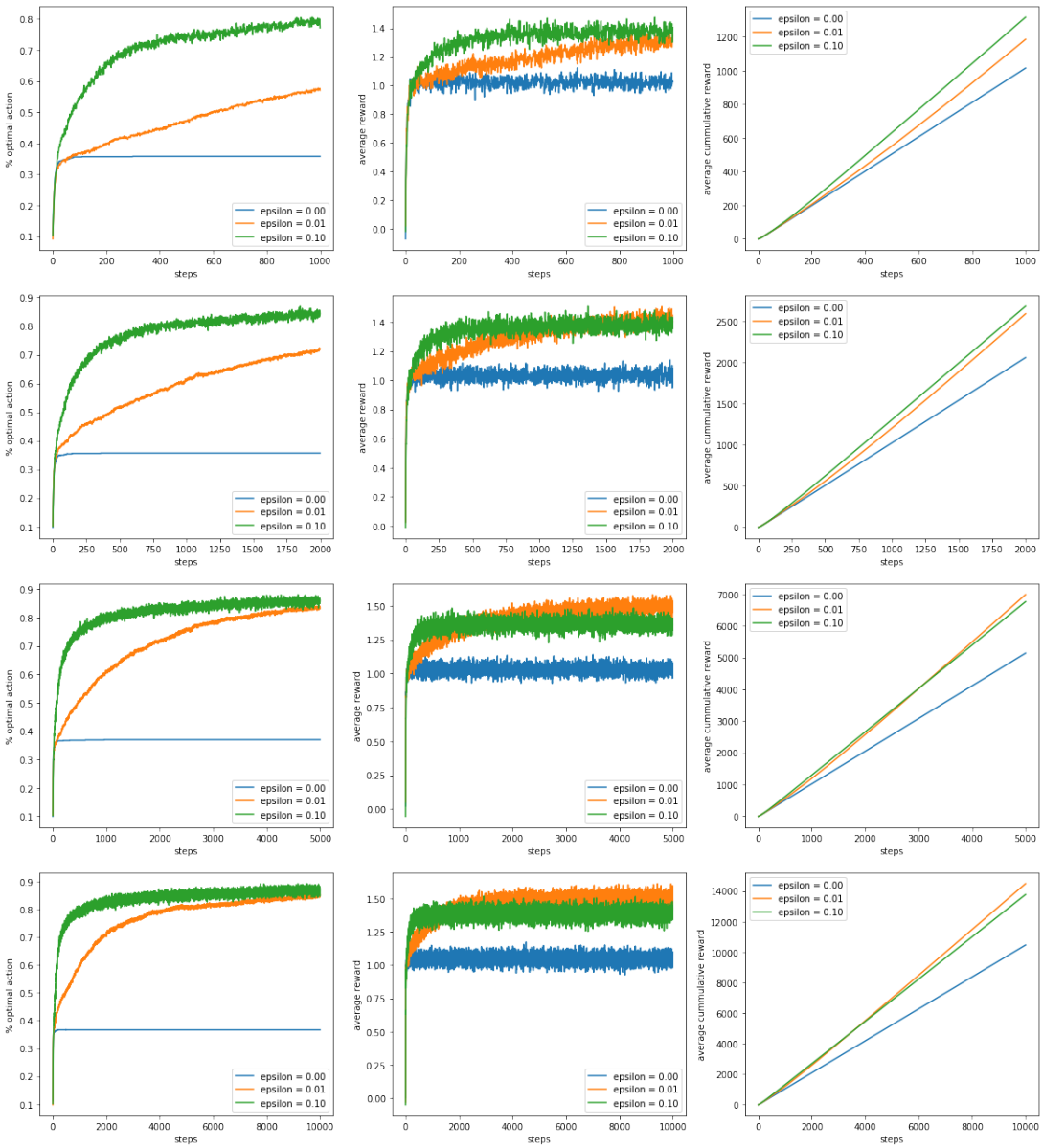


Figure 1: Reproduction of Figure 2.2 in Sutton and Barto. We show 1000, 2000, 5000 and 10000 steps.

2.4 Exercise 4

If the step-size parameters, α_n, are not constant, then the estimate Q_n is a weighted average of previously received rewards with a weighting different from that given by (2.6). What is the weighting on each prior reward for the general case, analogous to (2.6), in terms of the sequence of step-size parameters?

Starting with the relation of recurrence (2.5) and assuming that the step-size parameter is not constant

$$
\begin{aligned}
&Q_{n+1} = Q_n + \alpha_n(R_n - Q_n) \\
&\implies Q_{n+1} = \alpha_n R_n + (1-\alpha_n)Q_n \\
&\implies Q_{n+1} = \alpha_n R_n + (1-\alpha_n)\left[\alpha_{n-1}R_{n-1} + (1-\alpha_{n-1})Q_{n-1}\right] \\
&\implies Q_{n+1} = \alpha_n R_n + (1-\alpha_n)\alpha_{n-1}R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})Q_{n-1} \\
&\implies Q_{n+1} = \alpha_n R_n + (1-\alpha_n)\alpha_{n-1}R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\left[\alpha_{n-2}R_{n-2} + (1-\alpha_{n-2})Q_{n-2}\right] \\
&\implies Q_{n+1} = \alpha_n R_n + (1-\alpha_n)\alpha_{n-1}R_{n-1} + (1-\alpha_n)(1-\alpha_{n-1})\alpha_{n-2}R_{n-2} + (1-\alpha_n)(1-\alpha_{n-1})(1-\alpha_{n-2})Q_{n-2} \\
&\implies ... \\
&\implies Q_{n+1} = \left(\prod_{j=1}^{n}(1-\alpha_j)\right)Q_1 + \sum_{i=1}^{n}\alpha_i\left(\prod_{j=i+1}^{n}(1-\alpha_j)\right)R_i
\end{aligned}
$$

so we can see that the weight of each reward is given by $\alpha_i\left(\prod_{j=i+1}^{n}(1-\alpha_j)\right)$. If we set the step-size parameter to be a constant, $\alpha_i = \alpha$:

$$
\begin{aligned}
Q_{n+1} &= \left(\prod_{j=1}^{n}(1-\alpha)\right)Q_1 + \sum_{i=1}^{n}\alpha\left(\prod_{j=i+1}^{n}(1-\alpha)\right)R_i \\
&\implies Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n}\alpha(1-\alpha)^{n-(i+1)+1}R_i \\
&\implies Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n}\alpha(1-\alpha)^{n-i}R_i
\end{aligned}
$$

recovering the relation (2.6).

2.5 Exercise 5: programming

Design and conduct an experiment to demonstrate the difficulties that sample-average methods have for nonstationary problems. Use a modified version of the 10-armed testbed in which all the $q_*(a)$ start out equal and then take independent random walks (say by adding a normally distributed increment with mean zero and standard deviation 0.01 to all the $q_*(a)$ on each step). Prepare plots like Figure 2.2 for an action-value method using sample averages, incrementally computed, and another action-value method using a constant step-size parameter, $\alpha = 0.1$. Use $\varepsilon = 0.1$ and longer runs, say of 10,000 steps.

2.6 Exercise 6: Mysterious Spikes

The results shown in Figure 2.3 should be quite reliable because they are averages over 2000 individual, randomly chosen 10-armed bandit tasks. Why, then, are there oscillations and spikes in the early part of the curve for the optimistic method? In other words, what might make this method perform particularly better or worse, on average, on particular early steps?

2.7 Exercise 7: Unbiased Constant-Step-Size Trick

In most of this chapter we have used sample averages to estimate action values because sample averages do not produce the initial bias that constant step sizes do (see the analysis leading to (2.6)). However,

sample averages are not a completely satisfactory solution because they may perform poorly on nonstationary problems. Is it possible to avoid the bias of constant step sizes while retaining their advantages on nonstationary problems? One way is to use a step size of

$$\beta_n = \alpha / \bar{\sigma}_n$$

to process the nth reward for a particular action, where $\alpha > 0$ is a conventional constant step size, and $\bar{\sigma}_n$ is a trace of one that starts at 0:

$$\bar{\sigma}_n = \bar{\sigma}_{n-1} + \alpha(1 - \bar{\sigma}_{n-1}) \text{ for } n \geq 0 \text{ with } \bar{\sigma}_0 = 0$$

Carry out an analysis like that in (2.6) to show that Q_n is an exponential recency-weighted average without initial bias.

2.8 Exercise 8: UCB Spikes

In Figure 2.4 the UCB algorithm shows a distinct spike in performance on the 11-th step. Why is this? Note that for your answer to be fully satisfactory it must explain both why the reward increases on the 11-th step and why it decreases on the subsequent steps. Hint: if $c = 1$, then the spike is less prominent.

2.9 Exercise 9

Show that in the case of two actions, the soft-max distribution is the same as that given by the logistic, or sigmoid, function often used in statistics and artificial neural networks.

2.10 Exercise 10

Suppose you face a 2-armed bandit task whose true action values change randomly from time step to time step. Specifically, suppose that, for any time step, the true values of actions 1 and 2 are respectively 0.1 and 0.2 with probability 0.5 (case A), and 0.9 and 0.8 with probability 0.5 (case B). If you are not able to tell which case you face at any step, what is the best expectation of success you can achieve and how should you behave to achieve it? Now suppose that on each step you are told whether you are facing case A or case B (although you still don't know the true action values). This is an associative search task. What is the best expectation of success you can achieve in this task, and how should you behave to achieve it?

2.11 Exercise 11: programming

Make a figure analogous to Figure 2.6 for the nonstationary case outlined in Exercise 2.5. Include the constant-step-size ε-greedy algorithm with $\varepsilon = 0.1$. Use runs of 200,000 steps and, as a performance measure for each algorithm and parameter setting, use the average reward over the last 100,000 steps.

3 Finite Markov Decision Processes

3.1 Exercise 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.

3.2 Exercise 2

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

3.3 Exercise 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?

3.4 Exercise 4

Give a table analogous to that in Example 3.3, but for $p(s', r|s, a)$. It should have columns for s, a, s' , r, and $p(s', r|s, a)$, and a row for every 4-tuple for which $p(s', r|s, a) > 0$.

3.5 Exercise 5

The equations in Section 3.1 are for the continuing case and need to be modified (very slightly) to apply to episodic tasks. Show that you know the modifications needed by giving the modified version of (3.3).

The function defining the discrete probability distributions governing the dynamics of a finite MDP is:

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

so since it involves a conditional probability, it satisfies

$$\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p(s', r|s, a) = 1$$

and if we consider a system with episodic tasks then $\mathcal{S}$ is denoting the set of all possible states but the terminal state, since this will not allow any next action.

3.6 Exercise 6

Suppose you treated pole-balancing as an episodic task but also used discounting, with all rewards zero except for -1 upon failure. What then would the return be at each time? How does this return differ from that in the discounted, continuing formulation of this task?

3.7 Exercise 7

Imagine that you are designing a robot to run a maze. You decide to give it a reward of $+1$ for escaping from the maze and a reward of zero at all other times. The task seems to break down naturally into episodes -the successive runs through the maze- so you decide to treat it as an episodic task, where the goal is to maximize expected total reward (3.7). After running the learning agent for a while, you find that it is showing no improvement in escaping from the maze. What is going wrong? Have you effectively communicated to the agent what you want it to achieve?

The actual goal of the problem is not only to escape from the maze, but doing it as fast as possible, so giving a non-zero reward only at the final step is not the best strategy for accomplishing this goal. If we give a reward of -1 at each step, but the terminal one, we make the agent, in its goal of maximizing the return, to learn to escape the maze with the less number of steps possible.

3.8 Exercise 8

Suppose $\gamma = 0.5$ and the following sequence of rewards is received $R_1 = 1$, $R_2 = 2$, $R_3 = 6$, $R_4 = 3$, and $R_5 = 2$, with $T = 5$. What are $G_0, G_1, ..., G_5$? Hint: Work backwards.

From the definition of G_t we can obtain the recursive relation in (3.9)

$$\begin{aligned} &G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... \\ &\implies G_t = R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + ... \right) \\ &\implies G_t = R_{t+1} + \gamma G_{t+1} \end{aligned}$$

and since $G_T = 0$, then we have

$$\begin{aligned} G_5 &= 0 \\ G_4 &= R_5 + \gamma G_5 = 2 \\ G_3 &= R_4 + \gamma G_4 = 4 \\ G_2 &= R_3 + \gamma G_3 = 8 \\ G_1 &= R_2 + \gamma G_2 = 6 \\ G_0 &= R_1 + \gamma G_1 = 3 \end{aligned}$$

3.9 Exercise 9

Suppose $\gamma = 0.9$ and the reward sequence is $R_1 = 2$ followed by an infinite sequence of 7s. What are G_1 and G_0?

Since $R_t = R_2 = 7$ for $t \geq 2$, using the relation in (3.10)

$$\begin{aligned}
&G_1 = R_2 + \gamma R_3 + \gamma^2 R_4 + ... \\
&\implies G_1 = R_2 + \gamma R_2 + \gamma^2 R_2 + ... \\
&\implies G_1 = R_2 \left(1 + \gamma + \gamma^2 + ...\right) \\
&\implies G_1 = R_2 \sum_{k=0}^{\infty} \gamma^k \\
&\implies G_1 = R_2 \frac{1}{1-\gamma} \\
&\implies G_1 = \frac{R_2}{1-\gamma}
\end{aligned}$$

and the using the relation (3.9) we have

$$\begin{aligned}
G_1 &= \frac{R_2}{1-\gamma} = 70 \\
G_0 &= R_1 + \gamma G_1 = 16
\end{aligned}$$

3.10 Exercise 10

Prove the second equality in (3.10).

This is the classical computation of the convergence of the sum of a geometrical series

$$\begin{aligned}
&G_t = \sum_{k=0}^{\infty} \gamma^k \\
&\implies G_t = 1 + \gamma + \gamma^2 + ... \\
&\implies G_t = 1 + \gamma \left(1 + \gamma + ...\right) \\
&\implies G_t = 1 + \gamma G_t \\
&\implies G_t - \gamma G_t = 1 \\
&\implies G_t = \frac{1}{1-\gamma}
\end{aligned}$$

3.11 Exercise 11

If the current state is S_t, and actions are selected according to stochastic policy π, then what is the expectation of R_{t+1} in terms of π and the four-argument function p defined in (3.2)?

The expected value of R_{t+1} will be the sum of all the possible values of $r \in \mathcal{R}$ weighted for the probability of, starting in $S_t = s$ chose the action $a \in \mathcal{A}$ that gives rise to them

$$\begin{aligned}
&\mathbb{E}_\pi \left[R_{t+1} | S_t = s\right] = \sum_{a \in \mathcal{A}} \pi(a|s) \mathbb{E}_\pi \left[R_{t+1} | S_t = s, A_t = a\right] \\
&\implies \mathbb{E}_\pi \left[R_{t+1} | S_t = s\right] = \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) r
\end{aligned}$$

3.12 Exercise 12

Give an equation for v_π in terms of q_π and π.

From the definition of v_π in (3.12) and the definition of q_π in (3.13) we have

$$\begin{aligned}
&v_\pi(s) = \mathbb{E}_\pi\left[G_t | S_t = s\right] \\
&\implies v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s)\mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] \\
&\implies v_\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) q_\pi(s,a)
\end{aligned}$$

3.13 Exercise 13

Give an equation for q_π in terms of v_π and the four-argument p.

From the definition of q_π in (3.13) we have

$$\begin{aligned}
&q_\pi(s,a) = \mathbb{E}_\pi\left[G_t | S_t = s, A_t = a\right] \\
&\implies q_\pi(s,a) = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \\
&\implies q_\pi(s,a) = \mathbb{E}_\pi\left[R_{t+1} | S_t = s, A_t = a\right] + \mathbb{E}_\pi\left[\sum_{k=1}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right] \\
&\implies q_\pi(s,a) = \sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}} p(s',r|s,a) r + \gamma\mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+2} | S_t = s, A_t = a\right] \\
&\implies q_\pi(s,a) = \sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}} p(s',r|s,a) r + \gamma \sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}} p(s',r|s,a)\mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_{t+1} = s\right] \\
&\implies q_\pi(s,a) = \sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}} p(s',r|s,a) r + \gamma \sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}} p(s',r|s,a) v_\pi(S_{t+1}) \\
&\implies q_\pi(s,a) = \sum_{s'\in\mathcal{S}}\sum_{r\in\mathcal{R}} p(s',r|s,a)[r + \gamma v_\pi(S_{t+1})]
\end{aligned}$$

3.14 Exercise 14

The Bellman equation (3.14) must hold for each state for the value function v_π shown in Figure 3.2 (right) of Example 3.5. Show numerically that this equation holds for the center state, valued at $+0.7$, with respect to its four neighboring states, valued at $+2.3$, $+0.4$, -0.4, and $+0.7$. (These numbers are accurate only to one decimal place.)

Using the Bellman equation, with the probability distributions of the example 3.5 we have

$$\begin{aligned}
&v_\pi(s) = \sum_{a\in\mathcal{A}}\sum_{r\in\mathcal{R}}\sum_{s'\in\mathcal{S}} \pi(a|s) p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \\
&\implies v_\pi(s_0) = \pi(\uparrow|s_0) p(s_\uparrow, 0|s_0,\uparrow)\left[0 + \gamma v_\pi(s_\uparrow)\right] + \pi(\rightarrow|s_0) p(s_\rightarrow, 0|s_0,\rightarrow)\left[0 + \gamma v_\pi(s_\rightarrow)\right] \\
&\qquad\qquad + \pi(\downarrow|s_0) p(s_\downarrow, 0|s_0,\downarrow)\left[0 + \gamma v_\pi(s_\downarrow)\right] + \pi(\leftarrow|s_0) p(s_\leftarrow, 0|s_0,\leftarrow)\left[0 + \gamma v_\pi(s_\leftarrow)\right] \\
&\implies v_\pi(s_0) = \gamma\pi(\uparrow|s_0) v_\pi(s_\uparrow) + \gamma\pi(\rightarrow|s_0) v_\pi(s_\rightarrow) + \gamma\pi(\downarrow|s_0) v_\pi(s_\downarrow) + \gamma\pi(\leftarrow|s_0) v_\pi(s_\leftarrow) \\
&\implies v_\pi(s_0) = \gamma\pi\left[v_\pi(s_\uparrow) + v_\pi(s_\rightarrow) + v_\pi(s_\downarrow) + v_\pi(s_\leftarrow)\right] \\
&\implies v_\pi(s_0) = +0.675 \approx +0.7
\end{aligned}$$

3.15 Exercise 15

In the gridworld example, rewards are positive for goals, negative for running into the edge of the world, and zero the rest of the time. Are the signs of these rewards important, or only the intervals between them? Prove, using (3.8), that adding a constant c to all the rewards adds a constant, v_c, to the values of

all states, and thus does not affect the relative values of any states under any policies. What is v_c in terms of c and γ?

If we add a c to all the return values, then the returns, using (3.8) will be affected as

$$\begin{aligned}
&G'_t = \sum_{k=0}^{\infty} \gamma^k R'_{k+(t+1)} \\
&\implies G'_t = \sum_{k=0}^{\infty} \gamma^k (R_{k+(t+1)} + c) \\
&\implies G'_t = \sum_{k=0}^{\infty} \gamma^k R_{k+(t+1)} + \sum_{k=0}^{\infty} \gamma^k c \\
&\implies G'_t = G_t + c \sum_{k=0}^{\infty} \gamma^k \\
&\implies G'_t = G_t + \frac{c}{1-\gamma}
\end{aligned}$$

then the value of a state will be given by

$$\begin{aligned}
&v_\pi(s)' = \mathbb{E}_\pi \left[G'_t | S_t = s \right] \\
&\implies v_\pi(s)' = \mathbb{E}_\pi \left[G_t + \frac{c}{1-\gamma} | S_t = s \right] \\
&\implies v_\pi(s)' = \mathbb{E}_\pi \left[G_t | S_t = s \right] + \mathbb{E}_\pi \left[\frac{c}{1-\gamma} | S_t = s \right] \\
&\implies v_\pi(s)' = v_\pi(s) + \frac{c}{1-\gamma}
\end{aligned}$$

3.16 Exercise 16

Now consider adding a constant c to all the rewards in an episodic task, such as maze running. Would this have any effect, or would it leave the task unchanged as in the continuing task above? Why or why not? Give an example.

3.17 Exercise 17

What is the Bellman equation for action values, that is, for q_π? It must give the action value $q_\pi(s, a)$ in terms of the action values, $q_\pi(s', a')$, of possible successors to the state–action pair (s, a). Hint: the backup diagram to the right corresponds to this equation. Show the sequence of equations analogous to (3.14), but for action values.

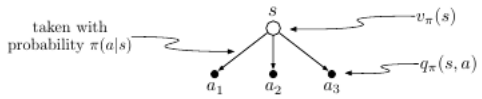
Following the same development that (3.14) we have

$$\begin{aligned}
&q_\pi(s,a) \doteq \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] \\
&\implies q_\pi(s,a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a \right] \\
&\implies q_\pi(s,a) = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} p(s', r | s, a) \left[r + \gamma v_\pi(s') \right] \\
&\implies q_\pi(s,a) = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} p(s', r | s, a) \left[r + \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a') \right]
\end{aligned}$$

3.18 Exercise 18

The value of a state depends on the values of the actions possible in that state and on how likely each action is to be taken under the current policy. We can think of this in terms of a small backup diagram

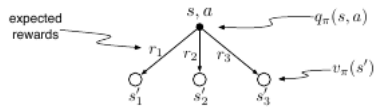
rooted at the state and considering each possible action:



Give the equation corresponding to this intuition and diagram for the value at the root node, $v_\pi(s)$, in terms of the value at the expected leaf node, $q_\pi(s,a)$, given $S_t = s$. This equation should include an expectation conditioned on following the policy, π. Then give a second equation in which the expected value is written out explicitly in terms of $\pi(a|s)$ such that no expected value notation appears in the equation.

3.19 Exercise 19

The value of an action, $q_\pi(s,a)$, depends on the expected next reward and the expected sum of the remaining rewards. Again we can think of this in terms of a small backup diagram, this one rooted at an action (state–action pair) and branching to the possible next states:



Give the equation corresponding to this intuition and diagram for the action value, $q_\pi(s,a)$, in terms of the expected next reward, R_{t+1} , and the expected next state value, $v_\pi(S_{t+1})$, given that $S_t = s$ and $A_t = a$. This equation should include an expectation but not one conditioned on following the policy. Then give a second equation, writing out the expected value explicitly in terms of $p(s', r|s, a)$ defined by (3.2), such that no expected value notation appears in the equation.

3.20 Exercise 20

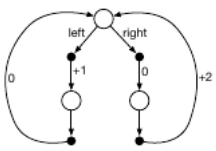
Draw or describe the optimal state-value function for the golf example.

3.21 Exercise 21

Draw or describe the contours of the optimal action-value function for putting, $q_*(s, putter)$, for the golf example.

3.22 Exercise 22

Consider the continuing MDP shown on to the right. The only decision to be made is that in the top state, where two actions are available, left and right. The numbers show the rewards that are received deterministically after each action. There are exactly two deterministic policies, π_{left} and π_{right} . What policy is optimal if $\gamma = 0$? If $\gamma = 0.9$? If $\gamma = 0.5$?



3.23 Exercise 23

Give the Bellman equation for q_* for the recycling robot.

3.24 Exercise 24

Figure 3.5 gives the optimal value of the best state of the gridworld as 24.4, to one decimal place. Use your knowledge of the optimal policy and (3.8) to express this value symbolically, and then to compute it to three decimal places.

3.25 Exercise 25

Give an equation for v_* in terms of q_*.

From the definitions of v_* and q_* we can see that

$$v_*(s) = \max_a q_*(s, a)$$

3.26 Exercise 26

Give an equation for q_* in terms of v_* and the four-argument p.

3.27 Exercise 27

Give an equation for π_* in terms of q_*.

3.28 Exercise 28

Give an equation for π_* in terms of v_* and the four-argument p.

3.29 Exercise 29

Rewrite the four Bellman equations for the four value functions (v_π , v_* , q_π, and q_*) in terms of the three argument function p (3.4) and the two-argument function r (3.5).

4 Dynamic Programming

4.1 Exercise 1

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

4.2 Exercise 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?

4.3 Exercise 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 $q_0, q_1, q_2, ...$?

From the Bellman equation for action values (Exercise 3.17),

$$
\begin{aligned}
& q_\pi(s,a) \doteq \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] \\
& \implies q_\pi(s,a) = \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a \right] \\
& \implies q_\pi(s,a) = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} p(s', r | s, a) \left[r + \gamma v_\pi(s') \right] \\
& \implies q_\pi(s,a) = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} p(s', r | s, a) \left[r + \sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a') \right]
\end{aligned}
$$

so the update rule will be

$$
q_{k+1}(s,a) = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} p(s', r | s, a) \left[r + \sum_{a' \in \mathcal{A}} \pi(a'|s') q_k(s', a') \right]
$$

4.4 Exercise 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.

4.5 Exercise 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.

We just have to add an extra loop over the actions at policy evaluation and treat $\pi(s)$ as a vector in which each position denotes the probability of taking and action, being this in our case a 0 or a 1.

Algorithm 1 Policy iteration, for estimating $\pi \approx \pi_*$

Initialization

$Q(s,a) \longleftarrow 0$
$\pi(s) \longleftarrow$ 0 and one 1 at random action

Policy evaluation

loop until $\Delta < \theta$
  $\Delta \longleftarrow 0$
  loop for $s \in \mathcal{S}$
    loop for $a \in \mathcal{A}$
      $q \longleftarrow Q(s,a)$
      $Q(s,a) \longleftarrow \sum_{r\in\mathcal{R}} \sum_{s'\in\mathcal{S}} p(s',r|s,\pi(s)) \left[r + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')Q(s',a')\right]$
      $\Delta \longleftarrow \max(\Delta, |q - Q(s,a)|)$
    end loop
  end loop
end loop

Policy improvement

policy stable $\longleftarrow$ *True*
loop for $s \in \mathcal{S}$
  old action $\longleftarrow \pi(s)$
  $\pi(s) \longleftarrow \text{argmax}_a \left[\sum_{r\in\mathcal{R}} \sum_{s'\in\mathcal{S}} p(s',r|s,a) \left[r + \gamma \sum_{a'\in\mathcal{A}} \pi(a'|s')Q(s',a')\right]\right]$
  if *old action* $\neq \pi(s)$ **then**
    policy stable $\longleftarrow$ *False*
  end if
end loop
if *policy stable* $=$ *True* **then**
  return Q and π
else
  go back to policy evaluation
end if

4.6 Exercise 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_π on page 80.

4.7 Exercise 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.

4.8 Exercise 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?

4.9 Exercise 9: programming

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* $\theta \to 0$*?*

4.10 Exercise 10

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

5 Monte Carlo Methods

5.1 Exercise 1

Consider the diagrams on the right in Figure 5.1. Why does the estimated value function jump up for the last two rows in the rear? Why does it drop off for the whole last row on the left? Why are the frontmost values higher in the upper diagrams than in the lower?

The diagrams in Figure 5.1 show a representation of the value function (axis z) in terms of the card showed by the dealer (axis x) and the player sum (axis y). In those diagrams we can see that the last two rows in the rear have much higher value than the rest, this is due to the fact that when the sum of the player is either 20 or 21 our policy makes us stick, but with any other sum we must hit, this makes us lose too many times, when the sum is small due to the lack of points and when the sum is high due to the posterior bust. On the other hand, 20 and 21 will result in a secured draw or victory the majority of the times independently of our policy.

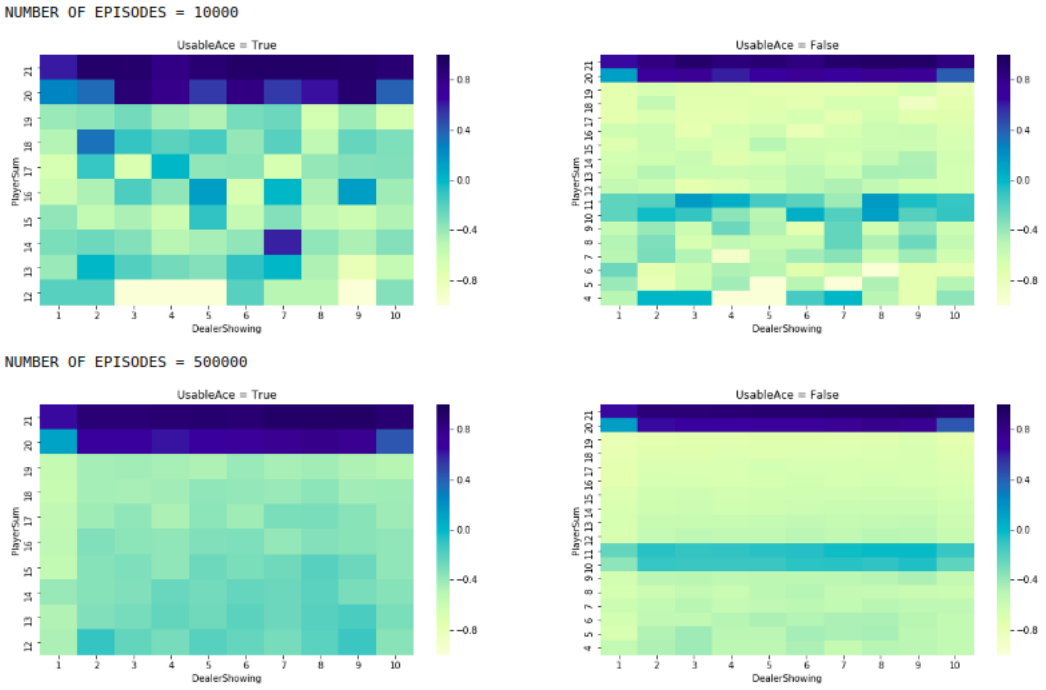


Figure 2: Reproduction of Figure 5.1 in Sutton and Barto with the first-visit algorithm.

5.2 Exercise 2

Suppose every-visit MC was used instead of first-visit MC on the blackjack task. Would you expect the results to be very different? Why or why not?

Since the first-visit method and the every-visit method are shown to converge to the expected value v_π we can expect to obtain the same results. Maybe in the case of 10000 episodes could be some differences, but in the case of 500000 episodes we expect the results to be indentical.

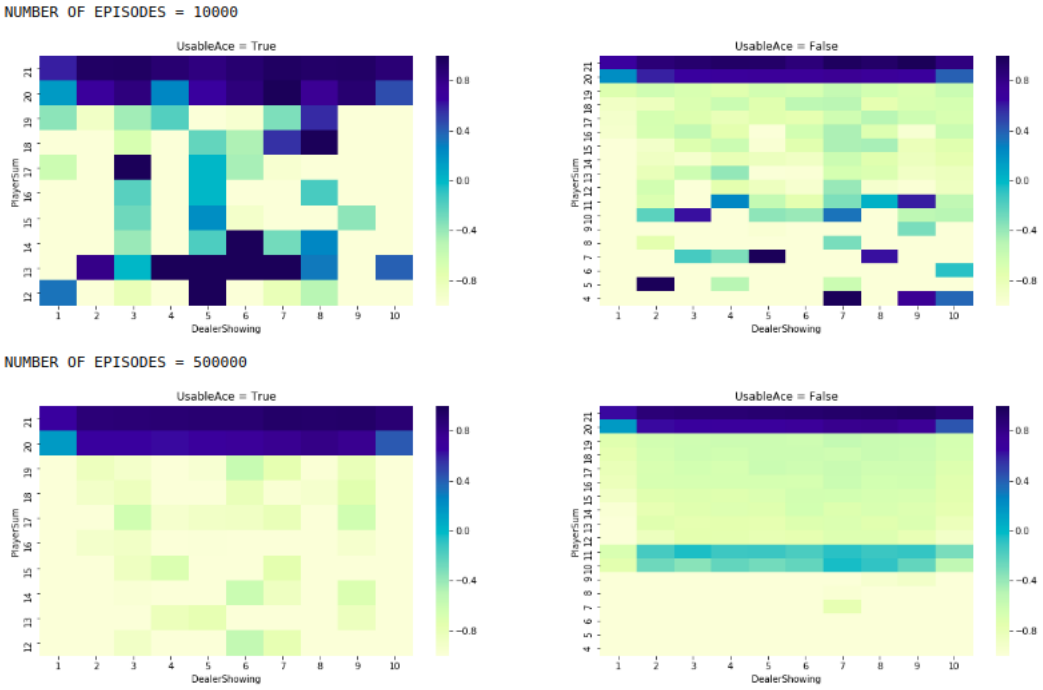


Figure 3: Reproduction of Figure 5.1 in Sutton and Barto but with the every-visit algorithm.

5.3 Exercise 3

What is the backup diagram for Monte Carlo estimation of q_π?

5.4 Exercise 4

The pseudocode for Monte Carlo ES is inefficient because, for each state-action pair, it maintains a list of all returns and repeatedly calculates their mean. It would be more efficient to use techniques similar to those explained in Section 2.4 to maintain just the mean and a count (for each state-action pair) and update them incrementally. Describe how the pseudocode would be altered to achieve this.

In order to make this more memory efficient we can just keep two lists, one for the sum of the values of the returns in for each state, and another for the counts of its appearances. This allow us to compute the values without having to keep all the returns.

Algorithm 2 Monte Carlo ES, for estimating $\pi \approx \pi_*$

$\pi(s) \longleftarrow$ arbitrary
$Q(s,a) \longleftarrow$ arbitrary
$returnsSum(s,a) \longleftarrow$ zeros list
$returnsCount(s,a) \longleftarrow$ zeros list
loop forever
    Chose S_0, A_0 randomly such that all pairs have nonzero probability
    Generate a episode following π: $S_0, A_0, R_1, S_1, ..., S_{T-1}, A_{T-1}, R_T$
    $G \longleftarrow 0$
    loop for each step in reverse ordering: $t = T-1, T-2, ..., 0$
        $G \longleftarrow \gamma G + R_{t+1}$
        if (S_t, A_t) not in $(S_0, A_0), (S_1, A_1), ..., (S_{t-1}, A_{t-1})$ **then**
            $returnsCount(S_t, A_t) \longleftarrow returnsCount(S_t, A_t) + 1$
            $returnsSum(S_t, A_t) \longleftarrow returnsSum(S_t, A_t) + G$
            $Q(S_t, A_t) \longleftarrow returnsSum(S_t, A_t)/returnsCount(S_t, A_t)$
            $\pi(S_t) \longleftarrow \text{argmax}_a Q(S_t, a)$
        end if
    end loop
end loop

5.5 Exercise 5

Consider an MDP with a single nonterminal state and a single action that transitions back to the nonterminal state with probability p and transitions to the terminal state with probability $1-p$. Let the reward be $+1$ on all transitions, and let $\gamma = 1$. Suppose you observe one episode that lasts 10 steps, with a return of 10. What are the first-visit and every-visit estimators of the value of the nonterminal state?

5.6 Exercise 6

What is the equation analogous to (5.6) for action values $Q(s,a)$ instead of state values $V(s)$, again given returns generated using b?

5.7 Exercise 7

In learning curves such as those shown in Figure 5.3 error generally decreases with training, as indeed happened for the ordinary importance-sampling method. But for the weighted importance-sampling method error first increased and then decreased. Why do you think this happened?

5.8 Exercise 8

The results with Example 5.5 and shown in Figure 5.4 used a first-visit MC method. Suppose that instead an every-visit MC method was used on the same problem. Would the variance of the estimator still be infinite? Why or why not?

5.9 Exercise 9

Modify the algorithm for first-visit MC policy evaluation (Section 5.1) to use the incremental implementation for sample averages described in Section 2.4.

5.10 Exercise 10

Derive the weighted-average update rule (5.8) from (5.7). Follow the pattern of the derivation of the unweighted rule (2.3).

5.11 Exercise 11

In the boxed algorithm for off-policy MC control, you may have been expecting the W update to have involved the importance-sampling ratio $\pi(A_t|A_s)/b(A_t|A_s)$*, but instead it involves* $1/b(A_t|A_s)$*. Why is this nevertheless correct?*

5.12 Exercise 12: Racetrack (programming)

Consider driving a race car around a turn like those shown in Figure 5.5. You want to go as fast as possible, but not so fast as to run off the track. In our simplified racetrack, the car is at one of a discrete set of grid positions, the cells in the diagram. The velocity is also discrete, a number of grid cells moved horizontally and vertically per time step. The actions are increments to the velocity components. Each may be changed by $+1$*,* -1*, or* 0 *in each step, for a total of nine* (3×3) *actions. Both velocity components are restricted to be nonnegative and less than 5, and they cannot both be zero except at the starting line. Each episode begins in one of the randomly selected start states with both velocity components zero and ends when the car crosses the finish line. The rewards are* -1 *for each step until the car crosses the finish line. If the car hits the track boundary, it is moved back to a random position on the starting line, both velocity components are reduced to zero, and the episode continues. Before updating the car's location at each time step, check to see if the projected path of the car intersects the track boundary. If it intersects the finish line, the episode ends; if it intersects anywhere else, the car is considered to have hit the track boundary and is sent back to the starting line. To make the task more challenging, with probability* 0.1 *at each time step the velocity increments are both zero, independently of the intended increments. Apply a Monte Carlo control method to this task to compute the optimal policy from each starting state. Exhibit several trajectories following the optimal policy (but turn the noise off for these trajectories).*

5.13 Exercise 13

Show the steps to derive (5.14) from (5.12).

5.14 Exercise 14

Modify the algorithm for off-policy Monte Carlo control (page 111) to use the idea of the truncated weighted-average estimator (5.10). Note that you will first need to convert this equation to action values.