Solutions to Reinforcement Learning: An Introduction (Second edition)

Shourya Bose (boseshourya1@gmail.com)

August 4, 2020

Contents

1 Chapter 2

Exercise 2.1

Let g denote the greedy action. Furthermore, let $A_t \sim \mathcal{G}$ denote that on timestep t the action is chosen in a greedy fashion, while let $A_t \sim \mathcal{NG}$ denote that on timestep t, the action is chosen non-greedily. Then, by the rule of total probability, we have

$$\begin{aligned} P(A_t = g) &= P(A_t = g \,|\, A_t \sim \mathcal{G})P(A_t \sim \mathcal{G}) + P(A_t = g \,|\, A_t \sim \mathcal{NG})P(A_t \sim \mathcal{NG}) \\ &= P(A_t = g \,|\, A_t \sim \mathcal{G})(1-\epsilon) + P(A_t = g \,|\, A_t \sim \mathcal{NG})(\epsilon) \\ &= 1.(1-\epsilon) + 0.5.(\epsilon) \\ &= 0.75 \end{aligned}$$

Exercise 2.2

Let $Q_n(a)$ denote the sample average action value estimate after action a has been chosen $n-1$ times. In other words, $Q_n(a) = (n-1)^{-1}\sum_{i=0}^{n-1} R_i$. Then we can write the values for all the 4 actions as

$$\begin{aligned} &Q_1(1) = 0,\ Q_2(1) = -1 \\ &Q_1(2) = 0,\ Q_2(2) = 2,\ Q_3(2) = 0,\ Q_4(2) = 2/3 \\ &Q_1(3) = 0,\ Q_2(3) = 0 \\ &Q_1(4) = 0. \end{aligned}$$

From the evolution of A_t and R_t over 5 timesteps, it can be noted that on timestep 5, action 3 is chosen whose value estimate is 0, while action 2 is not chosen whose value estimate is $2/3$. Thus on timestep **5** the ϵ case *definitely* occurred while on timesteps **1,2,3,4**, it might have *possibly* occurred.

Exercise 2.3

Using similar calculations (and employing similar notation) to the answer of Exercise 2.1, we see that for a k-armed bandit, the probability of choosing greedy action on any timestep is given by

$$\begin{aligned} P(A_t = g) &= 1.(1-\epsilon) + \frac{1}{k}.(\epsilon) \\ &= 1 - \frac{\epsilon(k-1)}{k}. \end{aligned}$$

As long as $\epsilon \neq 0$, the Law of Large Numbers dictates that $Q_t(a) \to q_*(a)$ in the long run. However, due to the ϵ-greedy policy, we see that the optimal reward will be generated only on a fraction of the timesteps with the fraction given by $1 - \frac{\epsilon(k-1)}{k}$.

Exercise 2.4

If the time-varying step sizes are given as $\{\alpha_n\}$, then the general recursion for estimate Q_n is given by

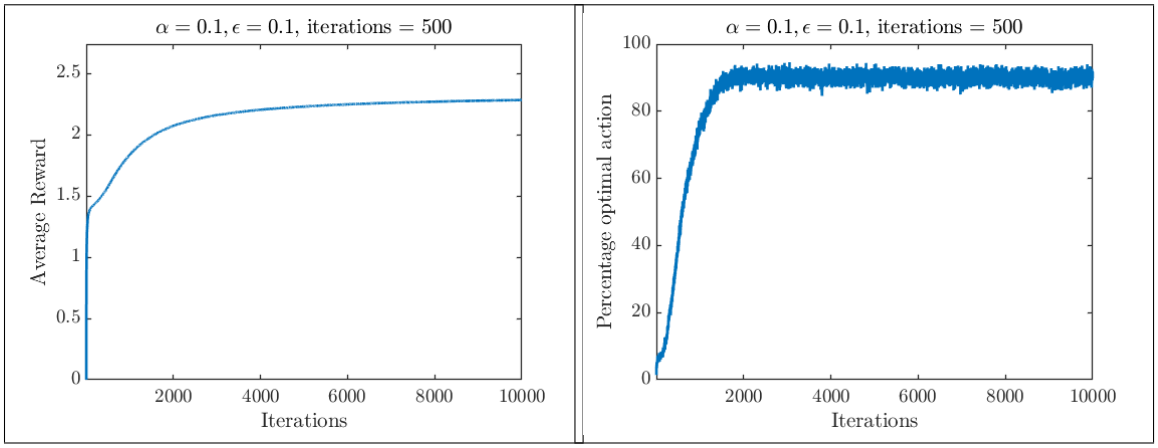
$$\begin{aligned} Q_{n+1} &= Q_n + \alpha_n \left[R_n - Q_n\right] \\ &= (1-\alpha_n)Q_n + R_n \\ &= (1-\alpha_n)(1-\alpha_{n-1})Q_{n-1} + \alpha_n(1-\alpha_{n-1})R_{n-1} + \alpha_n R_n \\ &= \vdots \end{aligned}$$

In general, we can write Q_{n+1} in terms of $\{R_1, R_2, \cdots, R_n\}$ and the initial estimate Q_1 as

$$Q_{n+1} = \left[\prod_{i=1}^{n}(1-\alpha_i)\right] Q_1 + \sum_{i=0}^{n-1}\left[\prod_{j=0}^{i-1}(1-\alpha_{n-j})\right]\alpha_{n-i}R_{n-i},$$

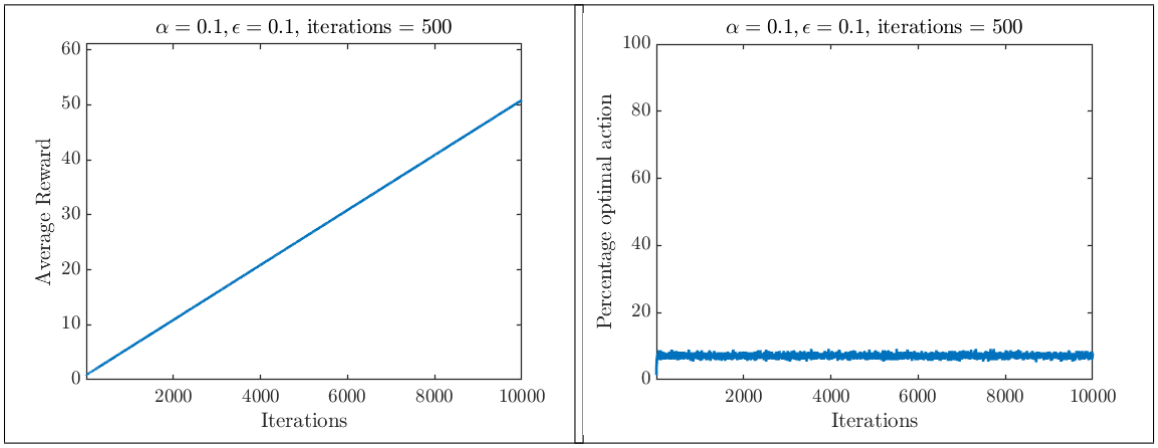
from where the coefficients of $R_1, R_2, \cdots, R_n$ as well as Q_1 can be inferred.

Exercise 2.5



(a) $q_*(a)$ is stationary

(b) $q_*(a)$ is stationary



(c) $\mathbb{E}[q_*(a)]$ increases by 0.01 on every timestep

(d) $\mathbb{E}[q_*(a)]$ increases by 0.01 on every timestep

From Figure 1b and Figure 1d, it can be seen that if $q_*(a)$ has a stationary distribution then the optimal action is chosen more and more as time progresses. However , in the non-stationary case, because the distribution of $q_*(a)$ is constantly changing, the percentage of times an optimal action is taken remains very low.

Exercise 2.6

All policies which 'explore' in the beginning, as opposed to immediately exploiting its knowledge from the start will show spikes on initial timesteps. This is because as the policy explores the action space, it will invariably explore the action with highest expected return, among all other actions. This will cause the observed spike. However, due to the optimistic initial values, the policy will not stop at that action but rather continue to search through other actions to determine optimality - thus leading to the 'dip' of the spike.

Exercise 2.7

Rearranging the terms for the recursive definition of $\bar{o}_n$, we get

$$\bar{o}_n - \alpha = (1 - \alpha)\bar{o}_{n-1}. \tag{1}$$

We now consider the recursion equation of Q_n with step size $\beta = \alpha/\bar{o}_n$.

$$\begin{aligned} Q_{n+1} &= Q_n + \beta_n(R_n - Q_n) \\ &= (1-\beta_n)Q_n + \beta_n R_n \\ &= \left(1 - \frac{\alpha}{\bar{o}_n}\right) Q_n + \left(\frac{\alpha}{\bar{o}_n}\right) R_n \\ \therefore \bar{o}_n Q_{n+1} &= (\bar{o}_n - \alpha)Q_n + \alpha R_n. \end{aligned}$$

We replace the value of $(\bar{o}_n - \alpha)$ from (1) into the above.

$$\bar{o}_n Q_{n+1} = (1-\alpha)\bar{o}_{n-1}Q_n + \alpha R_n.$$

Letting $\phi_k \stackrel{\Delta}{=} \bar{o}_{k-1}Q_k$ for all $k \in \mathbb{N}$, we get

$$\phi_{n+1} = (1-\alpha)\phi_n + \alpha R_n.$$

This is similar to the recursion equation for Q_n in the constant step-size case given as

$$\begin{aligned} Q_{n+1} &= (1-\alpha)Q_n + \alpha R_n \\ &= (1-\alpha)^n Q_1 + \sum_{i=0}^{n-1} \alpha(1-\alpha)^i R_{n-1}. \end{aligned}$$

Thus, we can write the closed form solution of ϕ_n as

$$\begin{aligned} \phi_{n+1} &= (1-\alpha)^n \phi_1 + \sum_{i=0}^{n-1} \alpha(1-\alpha)^i R_{n-i} \\ \therefore \bar{o}_n Q_{n+1} &= (1-\alpha)^n \phi_0 Q_1 + \sum_{i=0}^{n-1} \alpha(1-\alpha)^i R_{n-i}. \end{aligned}$$

However, noting that $\bar{o}_0 = 0$ by definition, we have $\phi_1 = \bar{o}_0 Q_1 = 0.Q_1 = 0$. Thus we get

$$\bar{o}_n Q_{n+1} = \sum_{i=0}^{n-1} \alpha(1-\alpha)^i R_{n-i},$$

thereby proving that Q_{n+1} is *unbiased* by initial estimate Q_1.

Exercise 2.8

A 'spike' is observed in the graph of average return around timestep 11 because this corresponds to the first timestep when the action with the highest average return (call this action a') is chosen by the policy for the first time.
However, after choosing a', even though the value of $Q_t(a')$ increases, the upper confidence bound (UCB) term $c\sqrt{\frac{\ln t}{N_t(a)}}$ for other actions a increases when these actions are not selected. Thus the increase in the UCB term for other actions overpowers the increase in $Q_t(a')$ for action a', and thus the policy keeps 'exploring' and collecting lower rewards, rather than continuously perform action a', leading to a 'dip' in the average return graph.
It should also be noted that for smaller values of c, the UCB term has less emphasis in choice of action, and therefore the spike will be less pronounced.

Exercise 2.9

The sigmoid function $S(x)$ is defined as

$$S(x) \triangleq \frac{1}{1+e^{-x}}.$$

Suppose we only have 2 actions namely a and b. Then the softmax distribution gives the probability of each of the actions as

$$\begin{aligned} P\{A_t = a\} &= \frac{e^{H_t(a)}}{e^{H_t(a)} + e^{H_t(b)}} \\ &= \frac{1}{1+e^{H_t(b)-H_t(a)}} \\ &= \frac{1}{1+e^{-(H_t(a)-H_t(b))}}. \\ P\{A_t = b\} &= \frac{e^{H_t(b)}}{e^{H_t(a)} + e^{H_t(b)}} \\ &= \frac{1}{1+e^{-(H_t(b)-H_t(a))}}. \end{aligned}$$

As we can see, the probabilities for both actions a and b take form of a sigmoid function in the two action case.

Exercise 2.9

In case we are not told about the timestep on which case A or case B holds, we should treat the problem as a non-stationary bandit problem, and use any of the methods introduced in this chapter with a constant step size α for updating $Q_t(a)$.
In case we know about the case on each timestep, we should maintain 4 Q values. Calling the actions a and b, the Q-values will be $Q_t^A(a)$, $Q_t^B(a)$, $Q_t^A(b)$, and $Q_t^B(b)$, and update them on each timestep according to whether the timestep falls under Case A or Case B.

Exercise 2.10

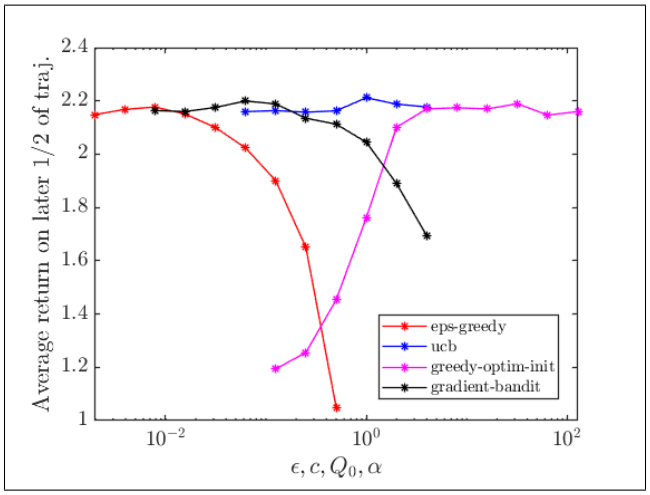


Figure 2: Parameter study of various bandit algorithms.

Improtant note: Gradient Bandits as Stochastic Gradient Ascent

In this section, we show how the gradient bandit algorithm can be derived from stochastic gradient ascent. The proof is taken verbatim from the textbook, except for trimming out some text and increasing clarity of

presentation. Consider the preference $H_t(a)$ for action a. Its evolution can be characterized in terms of its effect on the epected reward $\mathbb{E}[R_t]$, which is given as

$$H_{t+1}(a) = H_t(a) + \alpha \frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)},$$

where the expectation term $\mathbb{E}[R_t]$ can be expanded as

$$\mathbb{E}[R_t] = \sum_x \pi_t(x) q_*(x), \quad x \in \mathcal{A}.$$

The partial derivative can be evaluated as

$$\begin{aligned}
\frac{\partial \mathbb{E}[R_t]}{\partial H_t(a)} &= \frac{\partial}{\partial H_t(a)} \left[\sum_x \pi_t(x) q_*(x) \right] \\
&= \sum_x q_*(x) \frac{\partial \pi_t(x)}{\partial H_t(a)} \\
&\overset{[r1]}{=} \sum_x (q_*(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)}.
\end{aligned}$$

The equality [r1] contains the term B_t called *baseline*, which can be any time-varying function not dependent on x. [r1] is a valid equality since $-B_t \sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0$. This can be shown as

$$\sum_x \pi_t(x) = 1 \implies \frac{\partial}{\partial H_t(a)} \left[\sum_x \pi_t(x) \right] = 0 \implies \sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0 \overset{[r2]}{\implies} -B_t \sum_x \frac{\partial \pi_t(x)}{\partial H_t(a)} = 0,$$

where in [r2], we multiplied the LHS and RHS of the preceding equality with B_t.
Multiplying and dividing the RHS in [r1] with $\pi_t(x)$, we can represent the closed form of $\frac{\mathbb{E}[R_t]}{\partial H_t(a)}$ in the form of as expectation, as shown below.

$$\begin{aligned}
\frac{\mathbb{E}[R_t]}{\partial H_t(a)} &= \sum_x \pi_t(x)(q_*(x) - B_t) \frac{\partial \pi_t(x)}{\partial H_t(a)} \frac{1}{\pi_t(x)} \\
&\overset{[r3]}{=} \mathbb{E}_{A_t \in \mathcal{A}} \left[(q_*(A_t) - B_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \frac{1}{\pi_t(A_t)} \right] \\
&\overset{[r4]}{=} \mathbb{E}_{A_t \in \mathcal{A}} \left[(q_*(A_t) - \bar{R}_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \frac{1}{\pi_t(A_t)} \right] \\
&\overset{[r5]}{=} \mathbb{E}_{A_t \in \mathcal{A}} \left[(R_t - \bar{R}_t) \frac{\partial \pi_t(A_t)}{\partial H_t(a)} \frac{1}{\pi_t(A_t)} \right].
\end{aligned}$$

In the above, in [r2], we replace the term x with the random variable A_t over whose values the expectation is taken. In [r3], we chose $\bar{R}_t \triangleq t^{-1} \sum_{i=1}^t R_t$ as our baseline B_t, and finally in [r4] we replace $q_*(A_t)$ with R_t since $\mathbb{E}[R_t \,|A_t = a] = q_*(a)$.
We now find a closed form of $\frac{\partial \pi_t(A_t)}{\partial H_t(a)}$ which we will plug back into [r5]. Recall that in the gradient bandit algorithm, the action is chosen according to a softmax distribution given by

$$\pi_t(x) = \frac{\exp(H_t(x))}{\sum_{y=1}^k \exp(H_t(y))}.$$

Letting $\mu \triangleq \sum_{y=1}^k \exp(H_t(y))$, we carry out the differentiation in the partial derivative using the quotient rule of differentiation as follows.

$$\begin{aligned}
\frac{\partial \pi_t(x)}{\partial H_t(a)} &= \frac{\frac{\partial \exp(H_t(x))}{\partial H_t(a)} \mu - \exp(H_t(x)) \frac{\partial \mu}{\partial H_t(a)}}{\mu^2} \\
&= \mathbb{1}_{x=a} \frac{\exp(H_t(x))}{\mu} - \frac{\exp(H_t(x)) \exp(H_t(a))}{\mu^2} \\
&= \mathbb{1}_{x=a} \pi_t(x) - \pi_t(x) \pi_t(a) = \pi_t(x) \left[\mathbb{1}_{x=a} - \pi_t(a) \right].
\end{aligned}$$

Pluging the above result back in [r5], we get

$$\begin{aligned}\frac{\mathbb{E}[R_t]}{\partial H_t(a)} &= \mathbb{E}_{A_t \in \mathcal{A}}\left[(R_t - \bar{R}_t)\pi_t(A_t)(\mathbb{1}_{A_t=a} - \pi_t(a))\frac{1}{\pi_t(A_t)}\right] \\ &\overset{[r6]}{=} \mathbb{E}_{A_t \in \mathcal{A}}\left[(R_t - \bar{R}_t)(\mathbb{1}_{A_t=a} - \pi_t(a))\right].\end{aligned}$$

In the gradient bandit algorithm, we update the term using the formula inside the expectation in [r6] (as an approximation), thereby proving that the gradient bandit update rule is derived from stochastic gradient ascent.

2 Chapter 3

Exercise 3.1

Three applications are presented as follows.

1. **AI chess player:** We can design an autonomous agent that learns to play chess using reinforcement learning.
   - *State:* Position of various pieces on the chess board.
   - *Actions:* Agent can move each of its pieces to a legal position, and declare checkmate.
   - *Reward:* $+1$ when an opponent piece is knocked out, -1 when an own piece is knocked out, $+M$ for checkmate, where M is a large number.
2. **Autonomous driving agent** We can design an agent which can drive a car autonomously in the real world, like a Tesla car.
   - *State:* Input from the car's sensors like camera, LIDAR, etc.
   - *Actions:* Inputs to steering, brake, and accelerator.
   - *Reward:* The reward can be defined as a two-argument function $r(t', \mathbf{d})$, where t' is the remaining time to reach destination, and $\mathbf{d}$ is a vector that provides the vehicle's distance from other vehicles and obstacles. r should in general be decreasing in t' and increasing in each dimension of $\mathbf{d}$.
3. **Adaptive beam steering** Nowadays, there are multiple antennae which can electronically steer beams in order to provide better signal reception at the user. We consider an antenna which steers beams in 2D and uses reinforcement learning to reckon the optimal steering angle.
   - *State:* Current direction in which beam is focused.
   - *Action:* Angle θ by which to steer the beam.
   - *Reward:* SNR (signal to noise ratio) received at user terminal.

Exercise 3.2

One of the cases where MDP framework might be insufficient is where the reward might he a high-dimensional vector instead of a scalar. In this case, we cannot calculate which reward $\mathbf{r}$ or $\mathbf{r}'$ may be 'better', since for some indices $\mathbf{r}$ would be greater than $\mathbf{r}'$ and in other indices it could be smaller. Thus we would require additional context for determining optimality of received rewards.

Exercise 3.3

The correct interface between the environment and action in this case would be at the steering wheels, brake, and accelerator. This is because RL stipulates that the 'agent' should be the one which has complete control authority over its actions, and in this case, it is fair to assume that complete control authority over driving the car can be achieved by manipulating the steering wheel, accelerator, and brake pedal. Going to a 'lower' level than this, e.g. torques on the wheels, might make planning for activities like driving to a specific location difficult, while any 'higher' level, such as in the brain of a driver, would make the action space of the MDP so large that it would become intractable to use in a RL problem.

Exercise 3.4

Note that we denote the state space {High, Low} as $\{H, L\}$, the action space {Search, Wait, Recharge} as $\{S, W, R\}$ and the reward space $\{r_{search}, r_{wait}, r_{penalty}, 0\}$ as $\{s, w, p, 0\}$, where $r_{penalty} = -3$ is the reward (penalty) if the robot has to be manually recovered due to low charge.

s	a	s'	r	$p(s', r\|s, a)$
H	S	H	s	α
H	S	L	s	$1-\alpha$
L	S	H	p	$1-\beta$
L	S	L	s	β
H	W	H	w	1
L	W	L	w	1
L	R	H	0	1

Exercise 3.5

Let $\mathcal{S}$ be the space of all non-terminal states, $\mathcal{R}$ be the space of all rewards, and s^+ be the terminal state. Then (3.3) can be modified for episodic tasks as

$$\sum_{s\in\mathcal{S}}\sum_{r\in\mathcal{R}} p(s', r|s, a) + \sum_{r\in\mathcal{R}} p(s^+, r|s, a) = 1.$$

Exercise 3.6

In case of an episodic formulation, the return on any timestep t is given as $G_t = -\gamma^{K_t}$ where K_t is a random variable denoting the number of timesteps form t when the first failure occurs. Because we are in an episodic setting, K_t is also the final timestep of any episode.
In the continuing task formulation, the return is given as $G_t = -\gamma^{K_t^1} - \gamma^{K_t^2} - \gamma^{K_t^3} - \cdots$ where K_t^i is a random variable denoting the time to ith failure fom timestep t. Since in the episodic formulation the state is automatically reset after any failure, the return on any timestep t considers the possibilities of all failures that can happen in the future. Note that due to causality, $K_t^{i+1} > K_t^i$ for every timestep t for all $i \in \mathbb{N}$.

Exercise 3.7

If the robot's performance in escaping the maze is not improving after a while, the most probable reason would be that the reward scheme is not 'motivating' the learning agent sufficiently to escape the maze. One of the solutions is to increase the terminal reward to $+M$ where $M > 0$ is a very large value, in order to increase the incentive on the terminal state. Another solution would be to assign a penalty of -1 to all timesteps where the exit of the maze is not reached. This will place a negative reward on the time elapsed before the agent escapes the maze, thereby incentivizing it to do so.

Exercise 3.8

The various values of G_t can be calculated as follows, starting backward from G_5, whose value is 0 since $T = 5$ is the final timestep of the episode.

$$\begin{aligned}
G_5 &= 0,\\
G_4 &= R_5 + \gamma G_5 = 2 + (0.5)(0) = 2,\\
G_3 &= R_4 + \gamma G_4 = 3 + (0.5)(2) = 4,\\
G_2 &= R_3 + \gamma G_3 = 6 + (0.5)(4) = 8,\\
G_1 &= R_2 + \gamma G_2 = 2 + (0.5)(8) = 6,\\
G_0 &= R_1 + \gamma G_1 = -1 + (0.5)(6) = 2.
\end{aligned}$$

Exercise 3.9

We will first calculate G_1 and then use its value to calculate G_0. Since $R_t = 7$ for $t \geq 2$, we have

$$\begin{aligned} G_1 &= R_2 + \gamma R_3 + \gamma^2 R_4 + \cdots \\ &= 7 + \gamma.7 + \gamma^2.7 + \cdots \\ &= 7(1 + \gamma + \gamma^2 + \cdots) \\ &\overset{[r7]}{=} \frac{7}{1-\gamma} = \frac{7}{1-0.9} = 70. \end{aligned}$$

In [r7] we have used equation (3.10) from the textbook. We can now calculate G_0 as

$$\begin{aligned} G_0 &= R_1 + \gamma G_1 \\ &= 2 + (0.9)(70) = 65. \end{aligned}$$

Exercise 3.10

We want to prove that

$$\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}, \quad \forall 0 \leq \gamma < 1. \tag{2}$$

The proof of (2) is as follows: we define the partial sum S_k as

$$S_k \triangleq \sum_{j=0}^{k} \gamma^j = 1 + \gamma + \gamma^2 + \cdots + \gamma^k.$$

Now we carry out some algebraic manipulations on the partial sum S_k as given below.

$$\begin{aligned} \gamma S_k &= \gamma \sum_{j=0}^{k} \gamma^j = \sum_{j=1}^{k+1} \gamma^j. \\ \therefore \quad S_k - \gamma S_k &= \sum_{j=0}^{k} \gamma^j - \sum_{j=1}^{k+1} \gamma^j = 1 - \gamma^{k+1}. \\ \therefore \quad S_k &= \frac{1-\gamma^{k+1}}{1-\gamma}. \end{aligned}$$

Using the fact that $\lim_{k\to\infty} S_k = \sum_{k=0}^{\infty} \gamma^k$ and that $\lim_{k\to\infty} \gamma^{k+1} = 0$ for $|\gamma| < 1$, we have

$$\sum_{k=0}^{\infty} \gamma^k = \lim_{k\to\infty} \frac{1-\gamma^{k+1}}{1-\gamma} = \frac{1}{1-\gamma},$$

thereby completing the proof.

Exercise 3.11

We have to find out a closed form expression for $\mathbb{E}\left[R_{t+1} | S_t = s\right]$ in terms of the MDP dynamics $p(s', r|s, a)$ and the policy $\pi(a|s)$. We first use the law of total expectation to expand the aforementioned expectation term as

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

Noting that the above expectation is to be taken over all actions $a \in \mathcal{A}(s)$, we derived the closed form of $\mathbb{E}[R_{t+1}|S_t = s]$ in terms of p and π as

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

Exercise 3.12

Using the law of total expectations, we see that

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

Exercise 3.13

Using the definition of $q_\pi(s, a)$, we see that

$$\begin{aligned}q_\pi(s, a) &= \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \\ &= \mathbb{E}_\pi[R_t + \gamma G_{t+1}|S_t = s, A_t = a] \\ &\overset{[r8]}{=} \mathbb{E}_\pi\left[R_t + \gamma\mathbb{E}_\pi[G_{t+1}|S_{t+1} = s', S_t = s, A_t = a]\,|S_t = s, A_t = a\right] \\ &\overset{[r9]}{=} \mathbb{E}_\pi\left[R_t + \gamma\mathbb{E}_\pi[G_{t+1}|S_{t+1} = s']\,|S_t = s, A_t = a\right] \\ &= \mathbb{E}_\pi[R_t + \gamma v_\pi(s')|S_t = s, A_t = a] \\ &= \sum_{r\in\mathcal{R}}\sum_{s'\in\mathcal{S}} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right].\end{aligned}$$

Note that in the above calculations, in [r8] we use the law of total expectations to expand the inner expectation, while in [r9] we have used the fact that G_t follows Markov property.

Important note: Bellman Recusion

From the results of Exercise 3.12 and Exercise 3.13, we can derive a formula to calculate $v_\pi(s)$ in terms of $v_\pi(s')$.

$$\begin{aligned}v_\pi(s) &= \sum_{a\in\mathcal{A}(s)} \pi(a|s)q_\pi(a, s) \\ &= \sum_{a\in\mathcal{A}(s)} \pi(a|s)\sum_{r\in\mathcal{R}}\sum_{s'\in\mathcal{S}} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right].\end{aligned}$$

The above formula is called the **Bellman Recursion Equation**.

Exercise 3.14

Let the center state be denoted by c. Let the action space be denoted by $\{N, S, E, W\}$, which respectively represent the action of going north, south, east, and west. Corresponding to taking these actions, let the resulting states be denoted as $\{c^N, c^S, c^E, c^W\}$. Furthermore, note that $\pi(N|c) = \pi(S|c) = \pi(E|c) = \pi(W|c) = 0.25$ as mentioned in the book. Lastly, the only reward that can be gained by any action while on c is 0, and therefore $p(c^N, 0|c, N) = p(c^S, 0|c, S) = p(c^E, 0|c, E) = p(c^W, 0|c, W) = 1$. The Bellman recursion in this case is given as

$$\begin{aligned}v_\pi(c) &= \pi(N|c)\left(\gamma v_\pi(c^N)\right) + \pi(S|c)\left(\gamma v_\pi(c^S)\right) + \pi(E|c)\left(\gamma v_\pi(c^E)\right) + \pi(W|c)\left(\gamma v_\pi(c^W)\right) \\ &= (0.25)(0.9)\left(2.3 + 0.4 - 0.4 + 0.7\right) = 0.675 \approx 0.7 = v_\pi(c).\end{aligned}$$

Exercise 3.15

Let v_π^c represent the value function of states when a constant c is added to all the reward values. Correspondingly, let G_t^c represent the infinite horizon discounted reward when c is added to all rewards. We have

$$\begin{aligned} G_t^c &= (R_{t+1}+c) + \gamma(R_{t+2}+c) + \gamma^2(R_{t+3}+c) + \cdots \\ &= \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots\right) + c\left(1 + \gamma + \gamma^2 + \cdots\right) \\ &= G_t + \frac{c}{1-\gamma}. \end{aligned}$$

By the definition of $v_\pi^c(s)$, we have

$$v_\pi^c(s) = \mathbb{E}_\pi\left[G_t + \frac{c}{1-\gamma}\middle| S_t = s\right] = \mathbb{E}_\pi\left[G_t | S_t = s\right] + \frac{c}{1-\gamma} = v_\pi(s) + \frac{c}{1-\gamma}.$$

Thus, adding a constant c to all the rewards adds a constant of $\frac{c}{1-\gamma}$ to all the state values.

Exercise 3.16

Using the notation from Exercise 3.15, we can expand G_t^c in the episodic case as

$$\begin{aligned} G_t^c &= (R_{t+1}+c) + \gamma(R_{t+2}+c) + \gamma^2(R_{t+3}+c) + \cdots + \gamma^{T-(t+1)}(R_T + c) \\ &= \left(R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{T-(t+1)} R_T\right) + c\left(1 + \gamma + \cdots + \gamma^{T-(t+1)}\right) \\ &= G_t + \frac{c(1-\gamma^{(T-t)})}{1-\gamma}. \end{aligned}$$

Letting $\phi(T) \triangleq \frac{c(1-\gamma^{(T-t)})}{1-\gamma}$, we see that $v_\pi^c(s) = v_\pi(s) + \phi(T)$ for all states $s \in \mathcal{S}$ in the episodic case. Furthermore, note that $\phi(T)$ is *increasing* in T since $\frac{\partial\phi(T)}{\partial T} = \frac{c\gamma^{(T-t)}\log(1/\gamma)}{1-\gamma} > 0$ for all $c > 0$ and $t \geq T$. Thus adding constant c to all the rewards in the episodic case results in addition of the factor $\phi(T)$ to the return G_t. Since $\phi(T)$ is increasing in T, it means that the agent will have an incentive to take a longer amount of time to complete the task in any given episode.

Exercise 3.17

Using the definition of $q_\pi(s,a)$, we see that

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}\left[G_t | S_t = s, A_t = a\right] \\ &= \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a\right] \\ &\overset{[r10]}{=} \mathbb{E}_\pi\left[R_{t+1} + \gamma \mathbb{E}_\pi\left[G_{t+1} | S_{t+1} = s'\right] | S_t = s, A_t = a\right] \\ &\overset{[r11]}{=} \mathbb{E}_\pi\left[R_{t+1} + \gamma \mathbb{E}_\pi\left[\mathbb{E}_\pi\left[G_{t+1} | S_{t+1} = s', A_{t+1} = a'\right] | S_{t+1} = s'\right] | S_t = s, A_t = a\right] \\ &\overset{[r12]}{=} \mathbb{E}_\pi\left[R_{t+1} + \gamma \mathbb{E}_\pi\left[q_\pi(s',a') | S_{t+1} = s'\right] | S_t = s, A_t = a\right] \\ &= \mathbb{E}_\pi\left[R_{t+1} + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') q_\pi(s',a') | S_t = s, A_t = a\right] \\ &= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a)\left[r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(s'|a') q_\pi(s', a')\right]. \end{aligned}$$

In the above, in [r10] and [r11], we have used the law of total expectations which says that $\mathbb{E}[\mathbb{E}[X|Y,Z]|Y] = \mathbb{E}[X|Y]$ for random variables X, Y, and Z, while in [r12] we have used the definition of $q_\pi(s,a)$.

Exercise 3.18

$V_\pi(s)$ as an expectation over $q_\pi(s,a)$ is given by

$$v_\pi(s) = \mathbb{E}_\pi \left[q_\pi(s,a) | S_t = s \right].$$

In terms of $\pi(a|s)$,

$$v_\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) q_\pi(s,a).$$

Exercise 3.19

We can write $q_\pi(s,a)$ in terms of value function of the future state $v_\pi(S_{t+1})$ as follows.

$$\begin{aligned}
q_\pi(s,a) &= \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} || S_t = s, A_t = a \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \mathbb{E}_\pi \left[G_{t+1} | S_{t+1} \right] | S_t = s, A_t = a \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + v_\pi(S_{t+1}) | S_t = s, A_t = a \right] \\
&\overset{[r13]}{=} \mathbb{E} \left[R_{t+1} + v_\pi(S_{t+1}) | S_t = s, A_t = a \right].
\end{aligned}$$

In [r13], the outer expectation is not conditioned upon the policy π because once the values of S_t and A_t are fixed as s and π respectively, then the probability of values of S_{t+1} and R_{t+1} is described by the system model $p(s', t | s, a)$ and not the policy probabilities $\pi(a|s)$. To show the same, we expand the expectation above in terms of $p(s', t | s, a)$

$$q_\pi(s,a) = \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r | s, a) \left[r + \gamma v_\pi(s') \right].$$

Important note: Bellman Optimality Condition

In this section we will prove the Bellman optimality condition. It is given as a relation between $v_*(s)$ and $q_{\pi_*}(s,a)$ as follows.

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a).$$

We will now prove the above. Note that

$$\begin{aligned}
v_*(s) &= \max_\pi v_\pi(s), \ \forall s \in \mathcal{S} \\
&= \max_\pi \sum_{a \in \mathcal{A}(s)} \pi(a|s) q_\pi(s,a).
\end{aligned}$$

Let π_* be an optimal policy. Then by bellman recursion equation we have

$$v_*(s) = \sum_{a \in \mathcal{A}(s)} \pi_*(a|s) q_{\pi_*}(s,a).$$

Let $a_*(s) \triangleq \arg\max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a)$. Then, by definition, $q_{\pi_*}(s, a_*(s)) \geq q_{\pi_*}(s,a), \ \forall a \in \mathcal{A}(s)$. Since $v_*(s)$ is a convex combination of $q_{\pi_*}(s,a)$ for different values of a, we can maximize $v_*(s)$ by setting $\pi_*(a_*(s)|s) = 1$ and $\pi_*(a|s) = 0$ for all other values of $a \in \mathcal{A}(s)$. Thus, we have

$$\begin{aligned}
v_*(s) &= \sum_{\substack{a \in \mathcal{A}(s) \\ a \neq a_*(s)}} \pi_*(a|s) q_{\pi_*}(s,a) + \pi_*(a_*(s)|s) q_{\pi_*}(s, a_*(s)) \\
&= q_{\pi_*}(s, a_*(s)) = \arg\max_{s \in \mathcal{A}(s)} q_{\pi_*}(s,a).
\end{aligned}$$

The corresponding optimal policy is deterministic and is given by

$$\pi_*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a' \in \mathcal{A}(s)} q_{\pi_*}(s, a') \\ 0, & \text{otherwise.} \end{cases}$$

Note that throughout the proof, we have assumed that $a_*(s) = \arg\max_{a \in \mathcal{A}(s)} q_{\pi_*}(s, a)$ is unique. However, in case $a_*(s)$ is not unique, we can use any of the maximizing values, with ties broken arbitrarily.

Exercise 3.20

Since the optimal sequence of actions from the starting tee is two drives and one putt (thereby sinking the ball in 3 strokes), the optimal state value function and the optimal actions for each state is given in the following figure.

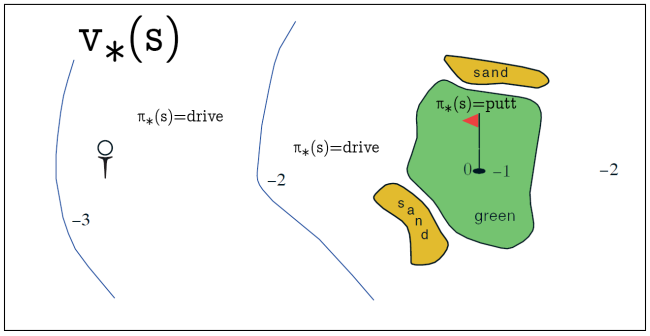


Figure 3: Optimal state-value function for the golf example.

Exercise 3.21

The optimal action-value function for the action `putt` is described in the following figure.

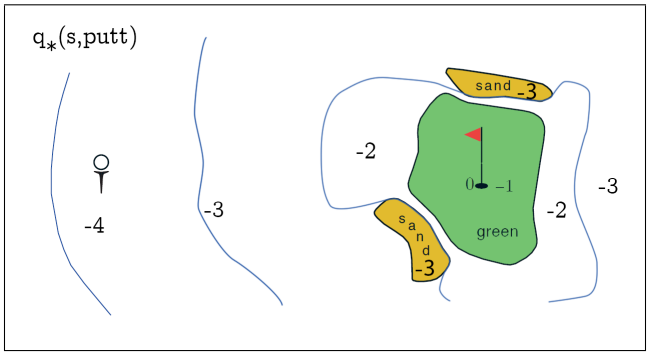


Figure 4: Optimal action-value function for the golf example.

For states s where $q_*(s, \texttt{putt}) = -4$, the optimal sequence of actions is `putt` $\rightarrow$ `drive` $\rightarrow$ `putt` $\rightarrow$ `putt`. For states s where $q_*(s, \texttt{putt}) = -3$, the optimal sequence of actions is either `putt` $\rightarrow$ `drive` $\rightarrow$ `putt` or `putt` $\rightarrow$ `putt` $\rightarrow$ `putt` depending upon the location of s. For states s where $q_*(s, \texttt{putt}) = -2$, the optimal sequence of actions is `putt` $\rightarrow$ `putt`. For states s where $q_*(s, \texttt{putt}) = -1$, the optimal action is `putt`. For states in the sand zone, $q_*(s, \texttt{putt}) = -3$ because a putt action cannot take the golf ball out of the sand (while a drive can), and therefore the optimal sequence of actions `putt` $\rightarrow$ `drive` $\rightarrow$ `putt`.

Exercise 3.22

We refer to the top state as A, the left state as B, and the right state as C. We also refer to the left action as L and the right action as R. By the Bellman optimality condition, the value of $v_*(A)$ is given as

$$\begin{aligned} v_*(A) &= \max\left\{p(B,+1|A,L)\left[+1+\gamma v_*(B)\right], p(C,0|A,R)\left[0+\gamma v_*(C)\right]\right\} \\ &= \max\{1+\gamma v_*(B), \gamma v_*(C)\}. \end{aligned}$$

The optimal state values $v_*(B)$ and $v_*(C)$ using the Bellman optimality condition is given as

$$v_*(B) = \gamma v_*(A), \quad v_*(C) = 2 + \gamma v_*(A).$$

Putting the previous equation together, we get

$$\begin{aligned} v_*(A) &= \max\{1+\gamma^2 v_*(A), 2\gamma + \gamma^2 v_*(A)\} \\ &= \gamma^2 v_*(A) + \max\{1, 2\gamma\} \\ \therefore v_*(A) &= \frac{\max\{1,2\gamma\}}{1-\gamma^2}. \end{aligned}$$

From the above, when $\gamma = 0$, the optimal action is left and $v_*(A) = 1$, when $\gamma = 0.9$ the optimal action is right and $v_*(A) = 9.47$, and finally, when $\gamma = 0.5$, *both* actions left and right are optimal and $v_*(A) = 1.33$.

Exercise 3.23

The two states in the recycling robot problem are `High` and `Low`. Correspondingly, we denote the actions search, wait, and recharge with S, W, and R respectively. r_s and r_w denote the rewards for searching and waiting, respectively, while recharging from `Low` to `High` incurs 0 cost, and being 'rescued', i.e. going from `Low` to `High` while searching incurs a cost of -3. We are going to find out the expressions for $q_*(\texttt{High}, S)$, $q_*(\texttt{High}, W)$, $q_*(\texttt{Low}, S)$, $q_*(\texttt{Low}, W)$ and $q_*(\texttt{Low}, R)$.

$$\begin{aligned} q_*(\texttt{High}, S) &= p(\texttt{High}, r_s|\texttt{High}, S)\left[r_s + \gamma \max_{a'} q_*(\texttt{High}, a')\right] + p(\texttt{Low}, r_s|H, S)\left[r_s + \gamma \max_{a'} q_*(\texttt{Low}, a')\right] \\ &= \alpha\left[r_s + \gamma \max_{a'} q_*(\texttt{High}, a')\right] + (1-\alpha)\left[r_s + \gamma \max_{a'} q_*(\texttt{Low}, a')\right] \\ &= r_s + \gamma\left[\alpha \max_{a'} q_*(\texttt{High}, a') + (1-\alpha)\max_{a'} q_*(\texttt{Low}, a')\right]. \\ q_*(\texttt{Low}, S) &= p(\texttt{Low}, r_s|\texttt{High}, S)\left[r_s + \gamma \max_{a'} q_*(\texttt{Low}, a')\right] + p(\texttt{High}, -3|\texttt{High}, S)\left[-3 + \gamma \max_{a'} q_*(\texttt{High}, a')\right] \\ &= \beta\left[r_s + \gamma \max_{a'} q_*(\texttt{Low}, a')\right] + (1-\beta)\left[-3 + \gamma \max_{a'} q_*(\texttt{High}, a')\right] \\ &= -3(1-\beta) + \beta r_s + \gamma\left[\beta \max_{a'} q_*(\texttt{Low}, a') + (1-\beta)\max_{a'} q_*(\texttt{High}, a')\right]. \\ q_*(\texttt{High}, W) &= p(\texttt{High}, r_w|\texttt{High}, W)\left[r_w + \gamma \max_{a'} q_*(\texttt{High}, a')\right] \\ &= r_w + \gamma \max_{a'} q_*(\texttt{High}, a'). \\ q_*(\texttt{Low}, W) &= p(\texttt{Low}, r_w|\texttt{Low}, W)\left[r_w + \gamma \max_{a'} q_*(\texttt{Low}, a')\right] \\ &= r_w + \gamma \max_{a'} q_*(\texttt{Low}, a'). \\ q_*(\texttt{Low}, R) &= p(\texttt{High}, 0|\texttt{Low}, R)\left[0 + \gamma \max_{a'} q_*(\texttt{High}, a')\right] \\ &= \gamma \max_{a'} q_*(\texttt{High}, a'). \end{aligned}$$

Exercise 3.24

Since all the actions $\{\texttt{north}, \texttt{south}, \texttt{east}, \texttt{west}\}$ are optimal for optimal state A, we choose the action $a = \texttt{north}$ for our calculations. We have

$$v_*(A) = p(A', +10|A, \texttt{north})\left[+10 + \gamma v_*(A')\right] = \left[10 + (0.9)(16)\right] = 24.4.$$

Exercise 3.25

The value of $v_*(s)$ in terms of $q_*(s, a)$ is given as

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_{\pi^*}(s, a) = \max_{a \in \mathcal{A}(s)} q_*(s, a).$$

Exercise 3.26

We can write down $q_*(s, a)$ in terms of $v_*(s)$ as follows.

$$\begin{aligned}
q_*(s, a) &= \max_{\pi} \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a\right] \\
&= \max_{\pi} \mathbb{E}_\pi \left[R_{t+1} + \gamma \mathbb{E}_\pi \left[G_{t+1} | S_{t+1} = s'\right] | S_t = s, A_t = a\right] \\
&\overset{[r14]}{=} \mathbb{E}\left[R_{t+1} + \gamma \max_{\pi} \mathbb{E}_\pi \left[G_{t+1} | S_{t+1}\right] | S_t = s, A_t = a\right] \\
&\overset{[r15]}{=} \mathbb{E}\left[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a\right] \\
&= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a)\left[r + \gamma v_*(s')\right].
\end{aligned}$$

In the above, in [r14] we have used the linearity of expectation operator and the non-dependence of R_{t+1} on π once S_t and A_t are known, while in [r15] we have used the definition of $v_*(s)$.

Exercise 3.27

The optimal value function $v_*(s)$ in terms of optimal action value function $q_*(s, a)$ is given as

$$v_*(s) = \max_{a \in \mathcal{A}(s)} q_*(s, a).$$

Corresponding to the optimal value, there is a *deterministic* policy π^* which is optimal. It is given as

$$\pi_*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a' \in \mathcal{A}(s)} q_*(s, a') \\ 0, & \text{otherwise.} \end{cases}$$

In case the a'-maximiser of $q_*(s, a')$ is not unique, select *one* of the maximisers, with ties broken arbitrarily.

Exercise 3.28

We know that the Bellman optimality condition for $v_*(s)$ is given as

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a)\left[r + \gamma v_*(s')\right].$$

From the above recursion equation of the optimal value function, we can derive an optimal deterministic policy π^* as

$$\pi_*(a|s) = \begin{cases} 1, & \text{if } a = \arg\max_{a' \in \mathcal{A}(s)} \sum_{r \in \mathcal{R}} p(s', r|s, a)\left[r + \gamma v_*(s')\right] \\ 0, & \text{otherwise.} \end{cases}$$

In case the a'-maximiser in the above equation is not unique, select *one* of the maximisers, with ties broken arbitrarily.

Exercise 3.29

We use the following equations to resolve $p(s', r|s, a)$ with $r(s, a)$ and $p(s', s, a)$.

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} r p(s', r|s, a) = \mathbb{E}\left[R_{t+1} | S_t = s, A_t = a\right] = r(s, a)$$
$$\sum_{r \in \mathcal{R}} p(s', r|s, a) = p(s'|s, a).$$

We first consider Bellman recursion equation for v_π.

$$\begin{aligned} v_\pi(s) &= \sum_{a \in \mathcal{A}(s)} \pi(a|s) \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma v_*(s')\right] \\ &= \sum_{a \in \mathcal{A}(s)} \pi(a|s) \left[r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_\pi(s')\right]. \end{aligned}$$

Next, we consider the Bellman recursion equation for q_π.

$$\begin{aligned} q_\pi(s, a) &= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma \sum_{a' \in \mathcal{A}(s')} \pi(a'|s') q_\pi(s', a')\right] \\ &= r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \sum_{a' \in \mathcal{A}(s')} p(s'|s, a) \pi(s'|s) q_\pi(s', a') \end{aligned}$$

Next, we consider the Bellman optimality condition for v_*.

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma v_*(s')\right] \\ &= \max_{a \in \mathcal{A}(s)} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) v_*(s') \right\}. \end{aligned}$$

Finally, we consider the Bellman optimality condition for q_*.

$$\begin{aligned} q_*(s, a) &= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma \max_{a' \in \mathcal{A}(s')} q_*(s', a')\right] \\ &= r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) \max_{a' \in \mathcal{A}(s')} \left\{q_*(s', a')\right\}. \end{aligned}$$

3 Chapter 4

Exercise 4.1

In this exercise, we note that $\gamma = 1$, the value function $v_\pi(s)$ is provided in Figure 4.1, and we recall the following Bellman iteration.

$$q_\pi(a,s) = \sum_{s\in\mathcal{S}^+}\sum_{r\in\mathcal{R}} p(s',r|s,a)\left[r+\gamma v_\pi(s')\right].$$

We can calculate $q_\pi(11,\texttt{down})$ and $q_\pi(7,\texttt{down})$ as

$$\begin{aligned} q_\pi(11,\texttt{down}) &= p(\texttt{terminal},-1|11,\texttt{down})\left[-1+\gamma v_\pi(\texttt{terminal})\right] \\ &= 1\left[-1+\gamma.0\right] = -1. \\ q_\pi(7,\texttt{down}) &= p(11,-1|7,\texttt{down})\left[-1+\gamma v_\pi(11)\right] \\ &= 1\left[-1+\gamma(-11)\right] = -15. \end{aligned}$$

Exercise 4.2

In this exercise, we use the Bellman iteration formula for state values given as

$$v_\pi(s) = \sum_{a\in\mathcal{A}(s)}\pi(a|s)\sum_{s\in\mathcal{S}^+}\sum_{r\in\mathcal{R}} p(s',r|s,a)\left[r+\gamma v_\pi(s')\right].$$

First, we evaluate $v_\pi(13)$ in terms of $v_\pi(15)$ as follows.

$$\begin{aligned} v_\pi(13) &= 0.25\left[-4+\gamma\left(v_\pi(9)+v_\pi(12)+v_\pi(14)+v_\pi(15)\right)\right] \\ &= -1+0.25\left(-20-22-14+v_\pi(15)\right) \\ &= -15+0.25v_\pi(15). \end{aligned} \tag{3}$$

We now evaluate $v_\pi(15)$ in terms of $v_\pi(13)$ as follows.

$$\begin{aligned} v_\pi(15) &= 0.25\left[-4+\gamma\left(v_\pi(13)+v_\pi(12)+v_\pi(14)+v_\pi(15)\right)\right] \\ &= -1+0.25\left[-22-14+v_\pi(13)+v_\pi(15)\right] \\ v_\pi(15)-0.25v_\pi(15) &= -10+0.25v_\pi(13) \\ 0.75v_\pi(15) &= -10+0.25v_\pi(13). \end{aligned} \tag{4}$$

Note that (3) and (4) form a system of 2 variables in 2 equations. Solving the same, we get

$$v_\pi(15) = -20, \quad v_\pi(13) = -20.$$

Thus, $v_\pi(15) = -20$, and the value function of state 13 does not change, i.e. $v_\pi(13) = -20$.

Exercise 4.3

We can obtain a recursive equation for $q_\pi(s,a)$ as follows.

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

Now, starting from q_0, we can successively generate $q_1, q_2, q_3, \cdots$ for all state-action pairs (s, a) by using the above Bellman recursion equation as a fixed point equation.

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

Exercise 4.4

In order to circumvent the scenario presented in this exercise, in policy improvement, the line "*old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*" should be replaced with "$v_{\textit{old-action}}(s) \neq v_\pi(s)$, then *policy-stable* $\leftarrow$ *false*".

Exercise 4.5

Algorithm 1 Policy Iteration for $q_*(s, a)$

Input: Initialize $Q(s, a)$ and $\pi(s) \in \mathcal{A}(s)$ for all $a \in \mathcal{A}(s)$ and $s \in \mathcal{S}$ arbitrarily.
Output: $Q \approx q_*$ and $\pi \approx \pi_*$.

procedure Policy Evaluation ▷ Compute v_π from π
  do
    $\Delta \leftarrow 0$
    for $s \in \mathcal{S}$ **do**
      for $a \in \mathcal{A}(s)$ **do**
        $q \leftarrow Q(s, a)$
        $Q(s, a) \leftarrow \sum_{s', r} p(s', r|s, a) \left[r + \gamma Q(s, \pi(s)) \right]$
        $\Delta \leftarrow \max\{\Delta, |q - Q(s, a)|\}$
      end for
    end for
  while $\Delta \geq \theta$
end procedure

procedure Policy Improvement ▷ Get improved policy π using state-action values q_π
  policy-stable $\leftarrow$ *true*
  for $s \in \mathcal{S}$ **do**
    old-action $\leftarrow \pi(s)$
    $\pi(s) = \arg\max_{a \in \mathcal{A}(s)} \sum_{s \in \mathcal{S}^+} \sum_{r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma Q(s, \pi(s)) \right]$
    If $Q(s, \textit{old-action}) \neq Q(s, \pi(s))$ then *policy-stable* = *false*
  end for
  if *policy-stable* **then**
    return Q and π
  else
    goto Policy Evaluation
  end if
end procedure

Exercise 4.6

Let the ϵ-soft policy for $\epsilon \leq 1$ be denoted by π_ϵ. Then in terms of the policy π and probabilities $\pi_\epsilon(a|s)$, π_ϵ can be described as

$$\pi_\epsilon(s|a) = \begin{cases} (1 - \epsilon) - \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{if } a = \pi(s), \\ \frac{\epsilon}{|\mathcal{A}(s)|}, & \text{otherwise.} \end{cases} \tag{5}$$

We define $n_s \triangleq |\mathcal{A}(s)|$. In order to account for the ϵ-soft policy, the update rule in Policy Evaluation for $V(s)$ should be modified as

$$V(s) \leftarrow \sum_{a \in \mathcal{A}(s)} \left[\left((1-\epsilon) - \frac{\epsilon}{n_s} \right) \mathbb{1}_{a=\pi(s)} + \left(\frac{\epsilon}{n_s} \right) \mathbb{1}_{a \neq \pi(s)} \right] \sum_{s \in \mathcal{S}^+} \sum_{r \in \mathcal{R}} p(s', r|s, a) \left[r + \gamma V(s') \right]$$

However, the Policy Improvement procedure will remain the same and will calculate the greedy policy $\pi(s)$ corresponding to values of $q_\pi(s, a)$. Thus nothing changes in (3), the update rule for $V(s)$ is changed in (2), and nothing changes for (1).

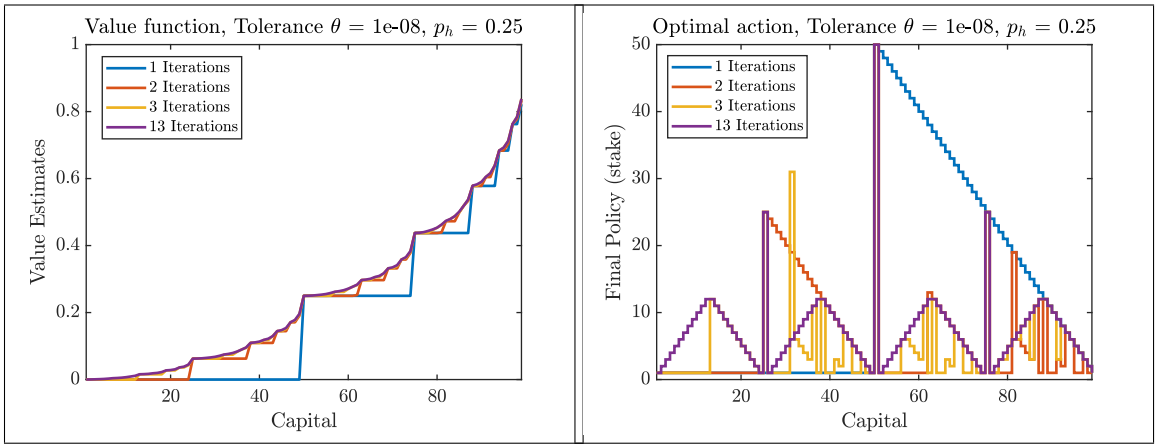
Exercise 4.7

Todo!

Exercise 4.8

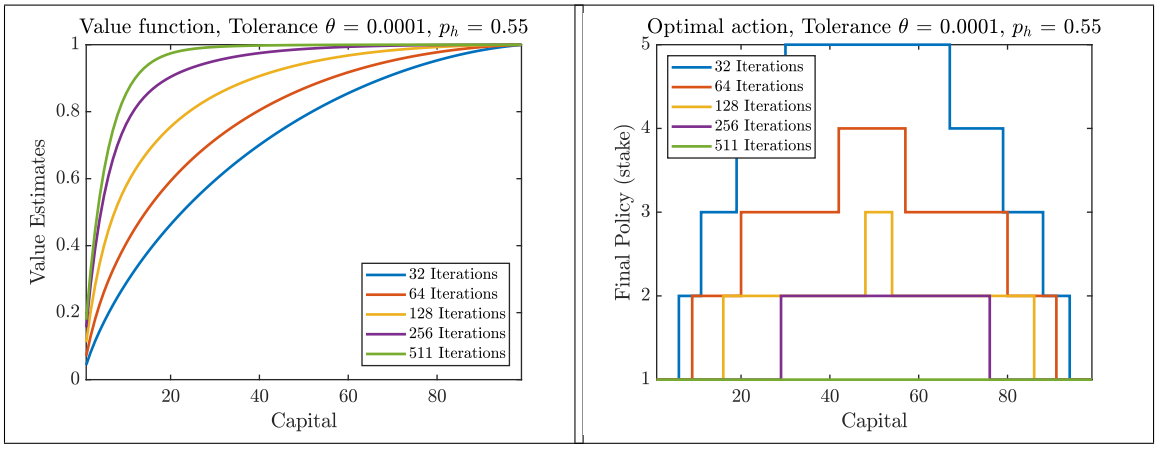
One thing to be noted about the current system is that $p_h < 0.5$, i.e. the probability of getting winning stakes on any timestep is lower than the probability of losing the stakes. Thus, the optimal policy would be one which ensures that a loss on any timestep can do no worse than reverse the gains from a previous timestep. Thus we get the curious final policy when $p_h < 5$. As can be noted from Exercise 4.9, the same is not the case when $p_h > 0.5$, where the optimal policy is to bet \$1 on every timestep.

Exercise 4.9



(a) Optimal state value for $p_h = 0.25$

(b) Optimal actions for $p_h = 0.25$



(c) Optimal state value for $p_h = 0.55$

(d) Optimal actions for $p_h = 0.55$

For $p_h = 0.25$, convergence is achieved for $\theta = 10^{-8}$ within 13 sweeps of the state space, while for $p_h = 0.55$, convergence is achieved for $\theta = 10^{-4}$ within 511 sweeps of the state space. For both choices of p_h, the optimal value function and optimal actions are stable as $\theta \to 0$.

Exercise 4.10

The analog of value iteration update (4.10) for action values $q_{k+1}(s, a)$ is given as

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

Important note: Complexity of solving DP's

In this section, we discuss a few points about the complexity of solving Dynamic Programming problems, and Markov Decision Process problems in general.

- Suppose the total number of states are n and correspondingly there are k available actions for each state. Even though the size of the state-action space is n^k, DP can find the optimum policy in polynomial time in k and n.
- Linear programming can be used to find optimal policies and its worst case runtime is lower than DP. However, in practice, the runtime of linear programming can become $100\times$ that of DP as the state space size increases.
- **Curse of dimensionality** refers to the fact that as the number of states grows exponentially in the number of state variables. For example, consider the simplest case where we have p state variables, each of which can assume q values. Then the total number of states is given by p^q.
- For large problems, asynchronous DP and variations of GPI (Generalized Policy Iteration) can help in reducing solution time.

4 Chapter 5

Extra: MATLAB Implementation of blackjack problem

The following results were derived by implementing Monte Carlo (MC) method for calculating value function in case of the blackjack problem (as given in the textbook).

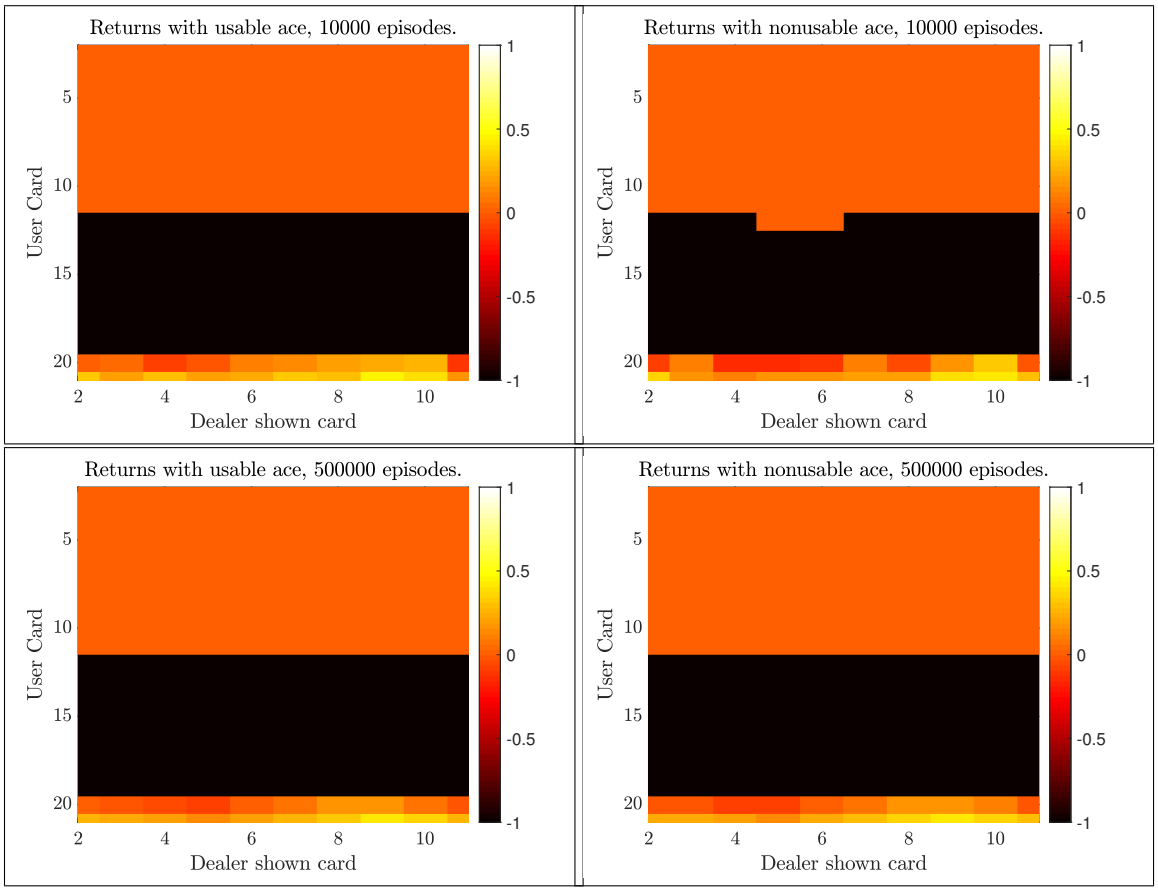


Figure 6: Monte Carlo value estimates for both usable and non-usable cases, for different number of simulation episodes.