My Solutions for Exercises of Sutton and Barto 2nd Edition

Andres Espinosa

May 18, 2024

Contents

1 Introduction

1.1 Exercise 1.1

The games would eventually result in a tie for every game. Since the model learns the optimal decision, and tic-tac-toe is a game where it is possible to always end in a tie in the case of optimal playing, the models would converge to always ending in a tie.

1.2 Exercise 1.2

We could model the value function in a way where symmetrical states have the same value function. This would benefit exploration because the agent would be able to learn the value function for a state that it has never truly reached. If the opponent did not take advantage of symmetries but the agent does, the opponent would have to experience a state before getting an approximation of the value function. The agent does not need to reach a state because it can use a symmetrical state as an accurate approximation of the value function for the other symmetrical states.

1.3 Exercise 1.3

Not necessarily, greedy actions can cause agents to get stuck in local minima. An agent could perform better if the greedy actions happen to be the best ones, but there could be a better set of actions that are in states the agent has never seen, and since it hasn't seen the state it may never due to its greedy actions.

1.4 Exercise 1.4

If an agent decides to learn from exploration, then it will learn the policy that it is using; a policy that combines optimal and suboptimal actions. If an agent explores but only learns from the greedy actions, then the value function it learns would be the value function that represents the optimal policy, not the policy it is actually executing (a mix of optimal and explorative).

1.5 Exercise 1.5

The tic-tac-toe is a rather simple problem but an agent could learn faster if it learned the value function of the opponent as well as itself. It could maybe use the opponent's value function to inform it's own since the environment is symmetrical to both of them.

2 Multi-armed Bandits

2.1 Exercise 2.1

Define a_1, a_2 as the two actions that can be taken, ε as the probability of selecting a random action, and r as a randomly generated number $\in [0, 1]$.

$$\text{action} = \begin{cases} \text{random choice of } a_1 \text{ or } a_2 & \text{if } r < \varepsilon \\ \text{greedy choice} & \text{if } r \geq \varepsilon \end{cases} \tag{1}$$

if ε is 0.5, there is a 0.5 probability of taking the greedy choice and 0.5 probability of picking the random choice. The random choice has two actions, therefore there is a choice of picking a_1 or a_2 with 0.25 prob each. $0.25 + 0.50 = 0.75$, therefore the probability of the greedy action getting selected is 0.75

2.2 Exercise 2.2

k-armed bandits with $k = 4$. ε-greedy algorithm setting $Q_1(a) = 0\ \forall a$. Sample a trajectory $(a_1, r_1, a_2, \dots)\ as\ (1, -1, 2, 1, 2, -2, 2, 2, 3, 0)$

- $A_1 = 1$, Greedy (tie broken) or random
- $A_2 = 2$, Greedy (tie broken) or random
- $A_3 = 2$, Greedy or random
- $A_4 = 2$, Must have been random because average reward of a_2 is now $-1/2$
- $A_5 = 3$, Greedy (tie broken) or random

2.3 Exercise 2.3

In the long run, the $\varepsilon = 0.01$ will perform the best. On average, it will pick the optimal action more frequently while still sampling each action an infinite number of times. In the long run, as $Q(a) \approx q_*(a)$, $\varepsilon = 0.01$ will sample the optimal action 99.1% of the time, whereas $\varepsilon = 0.1$ will sample the optimal action 91% of the time, resulting in a smaller frequency of the optimal action being picked and a smaller total reward over the long run.

2.4 Exercise 2.4

Deriving the weighting on prior reward for general case $\alpha_n \neq \alpha$

$$Q_{n+1} = Q_n + \alpha_n[R_n - Q_n] \tag{2}$$

$$Q_{n+1} = \alpha_n R_n + (1 - \alpha_n Q_n) \tag{3}$$

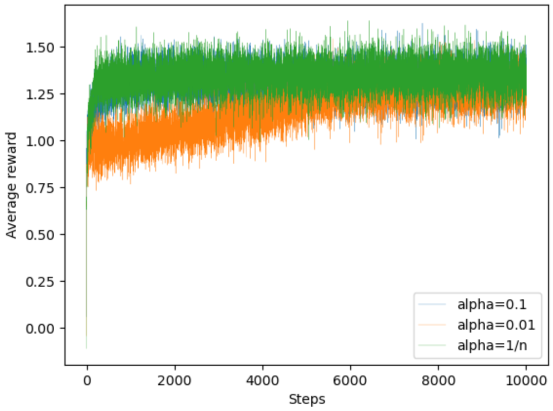
$$Q_{n+1} = \alpha_n R_n + (1 - \alpha_n[\alpha_{n-1} R_{n-1} + (1 - \alpha_{n-1})Q_{n-1}]) \tag{4}$$

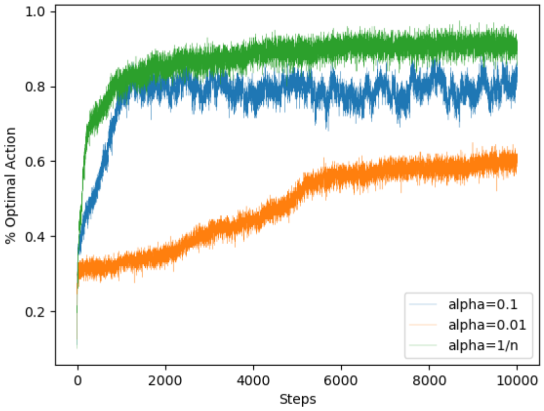
$$Q_{n+1} = \alpha_n R_n + (1 - \alpha_n)\alpha_{n-1} R_{n-1} + (1 - \alpha_n)(1 - \alpha_{n-1})Q_{n-1} \tag{5}$$

$$Q_{n+1} = Q_1 \prod_{i=1}^{n}(1 - \alpha_i) + \sum_{i=1}^{n} R_i \alpha_i \prod_{j=i+1}^{n}(1 - \alpha_j) \tag{6}$$

2.5 Exercise 2.5

Completed in code





2.6 Exercise 2.6

Optimism will cause every option to look appealing until the option has been sampled enough times. If the option that is appealing happens to be an optimal choice, the decay of the Q value will be slower than that of the other choices.

2.7 Exercise 2.7

$\bar{o}_0 = 0$, therefore $\bar{o}_1 = 0 + \alpha(1 - 0) = \alpha$. $\beta_1 = \alpha/\alpha = 1$. Using the equation derived at the end of exercise 2.4, $Q_1 \prod_{i=1}^{n}(1 - \alpha_i)$ is 0 since α_1 is 0. Q_n does not include Q_1 so it is without initial bias.

2.8 Exercise 2.8

In the UCB algorithm, each arm must be sampled once. Each arm that has not been pulled ($N_t(a) = 0$), is classified as a maximizing action and will likely be pulled in the first $|A|$ actions.

Since each arm has been sampled, on average, the optimal action will be the most appealing as it will provide the most reward in the long run. Therefore, each action will be made once and then the best seeming action is most likely to be selected.

2.9 Exercise 2.9

Soft-max as defined by equation 2.11 in the book is:

$$\pi_t(a) = \frac{e^{H_t(a)}}{\sum_{b=1}^{k} e^{H_t(b)}} \tag{7}$$

Sigmoid is defined as:

$$\sigma(x) = \frac{e^x}{e^x + 1} \tag{8}$$

When there are only two actions, a_1 and a_2, they end up being the same thing. This can be shown since the preference $H_t(a)$ only matters in relative importance to the other actions possible. To illustrate how this means the sigmoid is the same as soft-max, define $H_t(a_1) = n$ and $H_t(a_2) = m$. Subtracting m from both results in $H_t(a_1) = n - m$ and $H_t(a_2) = 0$. Since the relative aspect is the only thing that matters, the relative preference is still retained and this is a valid transformation. Substituting n and m into the softmax equation yields:

$$\pi_t(a_1) = \frac{e^{H_t(a_1)}}{e^{H_t(a_1)} + e^{H_t(a_2)}} \tag{9}$$

$$\pi_t(a_1) = \frac{e^{n-m}}{e^{n-m} + e^0} \tag{10}$$

$$\pi_t(a_1) = \frac{e^{n-m}}{e^{n-m} + 1} \tag{11}$$

$$\therefore \pi_t(a_1) = \sigma(n - m) \tag{12}$$

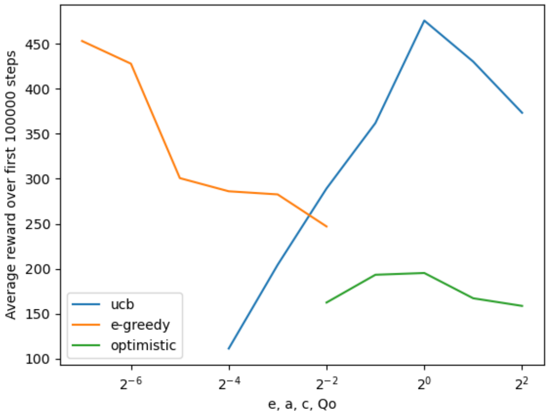
2.10 Exercise 2.10

If you do not know the case, then the problem can be framed as a 2-armed bandit where the reward structure is: a_1 gives 10 w/ probability 0.50 and 90 w/ probability 0.50, a_2 gives 20 w/ probability 0.50 and 80 w/ probability 0.50.

The expected reward for always picking a_1 is $E[a_1] = \frac{1}{2} * 10 + \frac{1}{2} * 90 = 50$ and the expected reward for always picking a_2 is $E[a_2] = \frac{1}{2} * 20 + \frac{1}{2} * 80 = 50$. Since the expected rewards are the same for both actions, the best expected reward is 50 and the policy does not affect the expected reward.

In event that the case is known, it is indeed important which action you take in order to maximize reward. You should pick a_2 when in case B and a_1 when in case A. The best expected reward that can be achieved with this strategy is $E[pi] = \frac{1}{2} * 20 + \frac{1}{2} * 90 = 55$

2.11 Exercise 2.11


Average reward over first 100000 steps
450
400
350
300
250
200
150
100
2^{-6}
2^{-4}
2^{-2}
2^{0}
2^{2}
e, a, c, Qo
ucb
e-greedy
optimistic

3 Finite Markov Decision Processes

3.1 Exercise 3.1

- An LLM is being fine tuned through reinforcement learning through human feedback. The states are a numerical representation of the inputted responses from a human and a numerical representation of the LLM's responses and it's weights. The action of the MDP is a shift in each of the weights and a novel generated prompt attempting to correct the response from earlier. The agent is given a reward that corresponds to a sentiment analysis of the human's response to the correction.

- A robot must learn to replicate the movements of a human. The states would be a representation of the human's movements. One way of this could be fixing dots on joints of the human and feeding the position of the dots to the robot. The robot then uses these dots and its own dots to move it's body in anticipation of the human's movements. The actions of the robot are movements of its possible joints, these may not correspond 100% to the human dots or it's own dots if there are differences in how it can move it and how its movement is represented. The robot could be given a reward that is inversely proportional to the distance of its dots and the humans.

- An AI must solve through a complex video game like final fantasy 7. The states would be incredibly complex as it would have to be designed with great detail and consideration. The state could include features such as: The screen (or a CNN output of the screen to reduce number of states), the items and materia in possession, the state of battle (0 if not in battle or 1 if in battle), the number of characters in the party. The actions could be the buttons that the player can press on a controller like start, o, x, and stick direction. The reward structure has a lot of freedom. The AI could be intrinsically motivated with curiosity through rewarding it based on an unseen state, or it could be given a bonus proportional to the EXP and gold earned.

3.2 Exercise 3.2

It is not, the markov property assumes that the current state is a unique sufficient representation for the agent's past and its future. This could be violated in many scenarios. One of them could be a robot that is designed to escape a maze and it is given the state representation of the $[s_1, s_2, s_3, s_4]$ where $s_i = 1$ if the robot senses a wall within 3 feet of it's current position in each direction north, east, south, west. There are likely many parts of the maze that have identical state representations and yet are completely different parts of the maze. However, many MDPs can be turned into MDP if given a state that reflects it's history. For example, in this robot you could include a vector containing each of it's prior actions as a state and would therefore be able to know if it is repeating movements.

3.3 Exercise 3.3

The answer really depends on the underlying goal of the person creating the agent-environment system. In this driving scenario, it is likely that the person already knows where they are driving to, the just don't know how to get there. So an incredibly high level would not be a great place to draw the line because it wouldn't align the agent's actions with the designer's goals.

3.4 Exercise 3.4

s	a	s'	r	$p(s', r \mid s, a)$
high	search	high	r_{search}	α
high	search	low	r_{search}	$1-\alpha$
high	wait	high	r_{wait}	1
low	search	low	r_{search}	β
low	search	high	-3	$1-\beta$
low	recharge	high	0	1
low	wait	low	r_{wait}	1

3.5 Exercise 3.5

Equation 3.3 is:

$$\sum_{s' \in S} \sum_{r \in R} p(s', r|s, a) = 1, \forall s \in S, a \in A(s) \tag{13}$$

Modifying this for the episodic case:

$$\sum_{s' \in S^+} \sum_{r \in R} p(s', r|s, a) = 1, \forall s \in S, a \in A(s) \tag{14}$$

3.6 Exercise 3.6

The return for this discounted episodic case would be $-\gamma^{K-1}$ where K is the number of time steps in the episode, normally T in episodic cases but for the sake of comparison we use K. This is identical to the discounted continuous case.

3.7 Exercise 3.7

Since this is an episodic case and is not discounted, the robot will receive a reward of 1 every time it escapes the maze no matter the amount of time spent in the maze. The fact that the return at each time step is 1 no matter the state the robot is in or the action it chooses. The robot will eventually escape the maze and so there is no hurry in finding an optimal path.A better approach would be to give the robot a negative reward for every time step in order to incentivize the robot to escape as soon as possible.

3.8 Exercise 3.8

$\gamma = 0.5$ and $R_1 = -1, R_2 = 2, R_3 = 6, R_4 = 3, R_5 = 2$ and $T = 5$

$$G_0 = -1 + 0.5 * 2 + 0.5^2 * 6 + 0.5^3 * 3 + 0.5^4 * 2 = 2 \tag{15}$$

$$G_1 = 2 + 0.5 * 6 + 0.5^2 * 3 + 0.5^3 * 2 = 6 \tag{16}$$

$$G_2 = 6 + 0.5 * 3 + 0.5^2 * 2 = 8 \tag{17}$$

$$G_3 = 3 + 0.5 * 2 = 4 \tag{18}$$

$$G_4 = 2 = 2 \tag{19}$$

$$G_5 = 0 = 0 \tag{20}$$

3.9 Exercise 3.9

$\gamma = 0.9, R_1 = 2, R_i = 7 \ \forall i \in [2, \infty]$

$$
\begin{aligned}
G_0 &= 2 + \sum_{i=2}^{\infty} 0.9^{i-1}7 = 2 - 7 + 7\sum_{i=0}^{\infty} 0.9^i && (21)\\
&= -5 + 7/0.1 = 65 && (22)\\
G_1 &= \sum_{i=1}^{\infty} 0.9^{i-1}7 = 7\sum_{i=0}^{\infty} 0.9^i && (23)\\
&= 7/0.1 = 70 && (24)
\end{aligned}
$$

3.10 Exercise 3.10

Proving $\sum_{k=0}^{\infty} \gamma^k = \frac{1}{1-\gamma}$:

$$
\begin{aligned}
\sum_{k=0}^{\infty} \gamma^k &= \lim_{k\to\infty} \gamma^0 + \gamma^1 + \gamma^2 + \cdots + \gamma^k && (25)\\
\lim_{k\to\infty} \gamma^0 + \gamma^1 + \gamma^2 + \cdots + \gamma^k &= \lim_{k\to\infty} \gamma^0 + \gamma^1 + \gamma^2 + \cdots + \gamma^k + \gamma^{k+1} && (26)\\
(1-\gamma)\lim_{k\to\infty} \gamma^0 + \gamma^1 + \gamma^2 + \cdots + \gamma^k &= (1-\gamma)\lim_{k\to\infty} \gamma^0 + \gamma^1 + \gamma^2 + \cdots + \gamma^k + \gamma^{k+1} && (27)\\
&= 1 + \lim_{k\to\infty} (\gamma-\gamma)^0 + (\gamma-\gamma)^1 + (\gamma-\gamma)^2 + \cdots + (\gamma-\gamma)^k + \gamma^{k+1} && (28)\\
&= 1 + 0 && (29)\\
(1-\gamma)\sum_{k=0}^{\infty} \gamma^k &= 1 && (30)\\
\sum_{k=0}^{\infty} \gamma^k &= \frac{1}{1-\gamma} && (31)
\end{aligned}
$$

3.11 Exercise 3.11

Finding $E[r|\pi, p]$ where π is a stochastic mapping of states to actions and $p(s', r|s, a) = Pr(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a)$

$$E[r|\pi, p, S_t = s] = \sum_a \pi(a|S_t) \sum_{s',r} rp(s', r|s, a) \qquad (32)$$

3.12 Exercise 3.12

Giving an equation for v_π in terms of q_π and π:

$$v_\pi(s) = \sum_a q_\pi(s, a)\pi(a|s) \qquad (33)$$

3.13 Exercise 3.13

Giving an equation for q_π in terms of v_π and p:

$$q_\pi(s,a) = \sum_{s'}\sum_{r} p(s',r|s,a)[r + \gamma v_\pi(s')] \tag{34}$$

3.14 Exercise 3.14

$$v_\pi(s) = \sum_{a}\pi(a|s)\sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')] \tag{35}$$

$$\text{Verifying this for the gridworld example} \tag{36}$$

$$0.7 = 0.25 * 1[0 + 0.9 * 0.7] + 0.25[0.9 * 2.3] + 0.25[0.9 * 0.4] + 0.25[0.9 * (-0.4)] \tag{37}$$

$$0.7 = \frac{63}{400} + \frac{207}{400} \tag{38}$$

$$0.7 = \frac{270}{400} \approx 0.7 \tag{39}$$

3.15 Exercise 3.15

Adding a constant c to the rewards:

$$v_\pi(s) = E[G_t|S_t = s] \tag{40}$$

$$v_\pi = E[\sum_{k=0}^{\infty}\gamma^k(R_{t+k+1} + c)|S_t = s] \tag{41}$$

$$= E[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} + \sum_{k=0}^{\infty}\gamma^k c|S_t = s] \tag{42}$$

$$= E[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1}|S_t = s] + \frac{c}{1-\gamma} \tag{43}$$

$$v_\pi(s) = v_\pi(s) + \frac{c}{1-\gamma} \tag{44}$$

3.16 Exercise 3.16

Adding a reward can change the task in an episodic case as imagine if in the gridworld case, the user was given a -0.1 reward for going north, if the periodic case worked out where on the second to last time step, the user landed in A', normally it's best action would be to travel north in order to get access to the reward associated with $A \rightarrow A'$. However, adding a total constant would change the appeal of going north.

3.17 Exercise 3.17

Finding $q_\pi(s,a)$bellman equation

$$q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a] \tag{45}$$
$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a] \tag{46}$$
$$q_\pi(s,a) = E_\pi[R_{t+1}|S_t = s, A_t = a] + E_\pi[\gamma G_{t+1}|S_t = s, A_t = a] \tag{47}$$
$$= \sum_{s',r} p(s', r|s,a)[r + \gamma \sum_{a'} \pi(a'|s')q_\pi(s', a')] \tag{48}$$

3.18 Exercise 3.18

Equation of expected state value based off state action pairs.

$$v_\pi(s) = E_\pi[q_\pi(s,a)] \tag{49}$$

Equation without the expectation

$$v_\pi(s) = \sum_a \pi(a|s)q_\pi(s,a) \tag{50}$$

3.19 Exercise 3.19

Equation of expected state-action value based off the future states and rewards.

$$q_\pi(s,a) = E[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s, A_t = a] \tag{51}$$

Equation without the expectation

$$q_\pi(s,a) = \sum_{s'} \sum_{r} p(s', r|s,a)[r + \gamma v_\pi(s')] \tag{52}$$

3.20 Exercise 3.20

Describing the optimal state-value function for the golf example.

$$v_*(s) = \max_a \sum_{s',r} p(s', r|s,a)[r + \gamma v_*(s')] \tag{53}$$

The optimal state value function would require summing over each state from a given state and weighing it by the probability of it occuring based off each action. So an example is for the -6 state, the possible actions are putt or driver. The state-value function would include the impossibility of moving from -6 to -4 using a putt and moving from -6 to -5. The way this would look is similar to the lower part of figure 3.3 with the exception that optimally, if the ball landed in the green the optimal action is to switch to putt. The diagram would then look exactly like the $q_*(s, driver)$ drawing but the -1 circle would be expanded to be the same as the v_{putt} drawing.

3.21 Exercise 3.21

Describing the optimal action-value function for putting in the golf example.

$$q_*(s,a) = \sum_{s',r} p(s', r|s,a)[r + \gamma \max_{a'} q_*(s', a')] \tag{54}$$

The drawing would likely look similar to the $q_*(s, driver)$ with the exception that the -6 sliver at the tee would be -4 and the contour lines would shift inward from the s driver drawing.

3.22 Exercise 3.22

If $\gamma = 0$, the optimal policy would be π_{left} since the 0 discount rate would cause the value function for the state to be 1 for left and 0 for the right.
If $\gamma = 0.9$, the optimal policy would be π_{right} since the value for the state would be:

$$\begin{aligned} v_{right}(s) &= \sum_{k=0}^{\infty} 0.9^{2k} * 0 + \sum_{k=0}^{\infty} 0.9^{2k+1} * 2 && (55) \\ &= 1.8 \sum_{k=0}^{\infty} 0.9^{2k} = 1.8 * \frac{1}{1-0.81} && (56) \\ &= \frac{1.8}{0.19} \approx 9.48 && (57) \end{aligned}$$

$$\begin{aligned} v_{left}(s) &= \sum_{k=0}^{\infty} 0.9^{2k} * 1 + \sum_{k=0}^{\infty} 0.9^{2k+1} * 0 && (58) \\ &= \frac{1}{1-0.81} \approx 5.26 && (59) \end{aligned}$$

If $\gamma = 0.5$, both policies are the optimal policy.

$$\begin{aligned} v_{right}(s) &= \sum_{k=0}^{\infty} 0.5^{2k} * 0 + \sum_{k=0}^{\infty} 0.5^{2k+1} * 2 && (60) \\ &= \frac{1}{1-0.25} = 4 && (61) \end{aligned}$$

$$\begin{aligned} v_{right}(s) &= \sum_{k=0}^{\infty} 0.5^{2k} * 1 + \sum_{k=0}^{\infty} 0.5^{2k+1} * 0 && (62) \\ &= \frac{1}{1-0.25} = 4 && (63) \end{aligned}$$

3.23 Exercise 3.23

Bellman equation for q:

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')] \quad (64)$$

Finding one bellman equation for the recycling robot, there are 6:

$$\begin{aligned} q_*(high, search) &= \alpha[r_{search} + \gamma \max_{a'}(q_*(high, search), q_*(high, wait))] && (65) \\ &+(1-\alpha)[r_{search} + \gamma \max_{a'}(q_*(low, search), q_*(low, wait), q_*(low, recharge))] && (66) \end{aligned}$$

3.24 Exercise 3.24

Optimal policy is to travel to A, which would bring the agent to A' and then go back north to A and repeat infinitely.

$$v_\pi(s_A) = E_\pi[\sum_{k=0}^{\infty} \gamma^k R_{k+t+1} | S_t = s] \tag{67}$$

$$= 10 \sum_{k=0}^{\infty} 0.9^{5k} \tag{68}$$

$$= \frac{10}{1 - 0.9^5} \tag{69}$$

$$= 24.419 \tag{70}$$

3.25 Exercise 3.25

$$v_*(s) = \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q(s', a')] \tag{71}$$

$$v_*(s) = \max_{a'} q(s, a') \tag{72}$$

3.26 Exercise 3.26

$$q_*(s, a) = \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{73}$$

3.27 Exercise 3.27

$$\pi_*(a|s) = \arg\max_a q(s, a) \tag{74}$$

3.28 Exercise 3.28

$$\pi_*(a|s) = \arg\max_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \tag{75}$$

3.29 Exercise 3.29

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} p(s'|s, a)[r(s, a) + \gamma v_\pi(s')] \tag{76}$$

$$v_*(s) = \max_a \sum_{s'} p(s'|s, a)[r(s, a) + \gamma v_\pi(s')] \tag{77}$$

$$q_\pi(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \sum_{a'} q_\pi(s', a')\pi(a'|s') \tag{78}$$

$$q_\pi(s, a) = r(s, a) + \gamma \sum_{s'} p(s'|s, a) \sum_{a'} q_*(s', a')\pi(a'|s') \tag{79}$$