Chapter 3

Finite Markov Decision Processes

3.1 The Agent-Environment Interface

> *Exercise 3.1* 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.

- Example Task 1: Fill water bottle
  - Task Description: Consider a water dispenser that has a valve to let water flow into a bottle placed below it.
  - State space: The state space could be the total volume of the bottle being filled, the total volume of water currently in the bottle, and the total volume of water that enters the drain.
  - Action space: The action space in the simplest case could be whether the valve is open or closed with a constant flow rate of water occurring for an open valve and zero flow for a closed valve. The reward could be
  - Rewards: The reward could be $1 - \text{unfilled volume fraction} - (10 \times \text{overfilled volume fraction})$. This way the reward would be maximized when the bottle is completely filled and would be penalized for any overfilling that occurs. Since the overfilling negative reward could not be removed into the future it would accumulate forever discouraging this behavior. In contract, there will always be a benefit for the system to get arbitrarily close to being perfectly full without overfilling.
- Example Task 2: Automatic Vacuum Cleaner
  - Task Description: Consider a battery powered robotic vacuum with wheels that can drive around a room and clean the floor. The robot will have to decide which path to take through the room before it has to recharge in its charging dock. To simplify the problem consider a driving scheme where the robot can only move forward in a straight line and can stop and turn in place to one of 4 directions: N, S, E, W.
  - State space: Consider a simple vacuum that can only detect its own battery charge state as a percent of full, how much material it is collecting at each moment in time, and whether it has a clear path immediately in front of it in each of 4 possible directions.
  - Action space: The robot can select to drive forward, not drive forward, or turn to a new direction.
  - Rewards: It is critical that the robot not get stuck with 0 charge outside of its charging dock while still trying to collect as much material from the floor while it has available charge. So, the reward at each timestep could be $+1 \times \text{material collected} - 1 \times \text{missing charge percentage}$. Depending on the units of material collected, these coefficients would have to be normalized to ensure that the robot doesn't sit in the charging dock indefinitely to avoid negative reward.
- Example Task 3: Firing Range Gun Aimer
  - Task Description: Consider a mechanism for holding a gun at a particular angle with the ability to pull the trigger and fire a round. The aim is to hit the target as close to the bullseye as possible. Consider an outdoor environment with wind and a target that can be placed at some arbitrary fixed distance away from the gun.
  - State space: The state space could be the vertical and horizontal angle of the gun relative to the straight line connecting the pivot point of the gun to the center of the target. In addition a wind speed sensor could detect the angle and speed of any airflow and a rangefinder can detect the distance to the target.
  - Action space: The agent can decide each of the two angles to position the gun restricted by some finite number of precise steps. If an angle is not being selected then the agent can decide whether to pull the trigger or wait.
  - Rewards: The agent could receive a large positive reward depending on how close the bullet hits to the bullseye and an infinitely negative reward for hitting too far away from the bullseye. Something like $r = \frac{1}{\max(0, \text{max distance} - \text{distance to bullseye})} + \frac{1}{\text{distance to bullseye}}$

> *Exercise 3.2* Is the MDP framework adequate to usefully represent *all* goal-directed learning tasks? Can you think of any clear exceptions?

If the environment depends heavily on the past history in terms of future rewards, but none of that information can be encoded into the current state, then the agent would be unable to learn the correct actions to take unless it had some internal memory of its own. But that would be akin to the agent having an internal state that varied over the course of a trajectory which is not part of the current framework. This could be solved by augmenting the environment state to contain whatever past information is necessary to specify the current state but that information may not always be accessible. In particular consider an environment with a person that has a particular action in mind that if repeated will cause large negative reward. The environment itself provides no record of which actions the agent has taken, so unless the agent saves that information itself, it would have no way of knowing.

> *Exercise 3.3* 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?

If you have a system like a car, it already has a mechanism for translating the pedal position and steering wheel into forces on the tires. If instead, we tried to have the system directly control the torque on the wheel, it would still have to control that through the pedal and steering wheel and rely on some other learned or explicit mechanism for translating those desires. Using the action space that relates most closely to what can actually be controlled would be the least prone to errors in which the desired actions are not implemented accurately. If we already had a built navigation system for the car in question and the desired task involves choosing the optimal path to navigate between many locations, then it might be appropriate to have the action space in terms of *where* to drive. If the car only has simple controls like accelerator and steering wheel as described above, then even if the ultimate task is more complicated, the natural action space is still the controls we have access to. The agent may effectively learn an intermediate task of how to navigate to a particular city, but putting that in the action space would give the agent no obvious way of performing that action. If we consider a human driving, it would be natural to have the action space in terms of actions that a human would know how to perform such as pressing the accelerator a set amount. Since people already know how to control muscles with electrical impulses, it would be an unnecessary layer of complexity to have an agent learn how to directly control the muscles of a person.

> *Exercise 3.4* 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 $p(s', r|s, a) > 0$

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

3.3 Returns and Episodes

> *Exercise 3.5* 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).

From Section 3.1 we have equation (3.3):

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

In the episodic case there is an additional state outside of $\mathcal{S}$ called the terminal state, and the union of this with every other state is denoted $\mathcal{S}^+$. So equation (3.3) still applies, but only if we ensure that $\mathcal{S}$ is changed to $\mathcal{S}^+$ because there is some non-zero probability of entering the terminal state.

> *Exercise 3.6* 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?

The return at each time would be $G_t = -\gamma^{T-t-1}$ where T is the total number of steps in the episode. In the continuing formulation of the task there will be a series of failures each one occurring at a different time T_i. At any given time t, only the failures that occur in the future will contribute to the return: $G_t = \sum_{T_i > t} -\gamma^{T_i - t - 1}$

> *Exercise 3.7* Imagine you are designing a robot to run a maze. You decide to give it a reward of +1 for escaping 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 the 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?

According to equation 3.7, there is no discount factor for the reward signal. If we assume the maze is simple enough that an agent escapes in some finite time, then it will receive a reward signal of 1. Any agent that solves the maze faster will receive the same reward as do agents that take longer. Because of the lack of discounting within an episode, there is no incentive for agents to solve the maze faster so any agent that can solve the maze is equally good, which in this case would be almost any arbitrary agent except one that simply stands still or only goes in one direction that gets stuck.

> *Exercise 3.8* 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 $G_0,\ G_1, \ldots, G_5$

$G_5 = 0$ by definition

$G_4 = R_5 + \gamma G_5 = 2$

$G_3 = R_4 + \gamma G_4 = 3 + (0.5 \times 2) = 4$

$G_2 = R_3 + \gamma G_3 = 6 + (0.5 \times 4) = 8$

$G_1 = R_2 + \gamma G_2 = 2 + (0.5 \times 8) = 6$

$G_0 = R_1 + \gamma G_1 = -1 + (0.5 \times 6) = 2$

> *Exercise 3.9* 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?

$G_1 = 7 \times \sum_{k=0}^{\inf} \gamma^k = \frac{7}{1-\gamma} = 70$

$G_0 = R_1 + \gamma G_1 = 2 + (0.9 \times 70) = 65$

> *Exercise 3.10* Prove the second equality in (3.10).

Equation (3.10) is $G_t = \sum_{k=0}^{\inf} \gamma^k = \frac{1}{1-\gamma}$

$G_t = \gamma^0 + \gamma^1 + \cdots$

$\gamma \times G_t = \gamma^1 + \gamma^2 + \cdots = G_t - \gamma^0 = G_t - 1$

$\gamma \times G_t = G_t - 1 \implies G_t \times (\gamma - 1) = -1 \implies G_t = \frac{1}{1-\gamma}$

3.5 Policies and Value Functions

> *Exercise 3.11* If the current state is S_t, and actions are selected according to a stochastic policy π, then what is the expectation of R_{t+1} in terms of π and the four-argument function $p(s', r|s, a) \doteq Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$

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

> *Exercise 3.12* Give an equation for v_π in terms of q_π and π.

From (3.12) we have
$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s\right],\ \text{for all } s \in \mathcal{S}$

and from (3.13) we have

$q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a\right]$

So we need to average q over the actions, weighting them by the probability of taking those actions.

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

> *Exercise 3.13* Give an equation for q_π in terms of v_π and $p(s', r|s, a)$

If we are trying to estimate the value of the state action pair, we can use the transition probability function get the distribution over future states we will end up in. Then we can use the value of each subsequent state to work backwards to the value of that action. We can directly add to this the expected reward of the action we are guaranteed to take. Because the value function for a state starts discounting rewards from a new starting point, we need to discount our estimate of the new state by γ to have the correct form of q_π
$q_\pi(s,a) = \sum_{r\in\mathcal{R}} r \sum_{s'\in\mathcal{S}} p(s',r|s,a) + \gamma \sum_{s'\in\mathcal{S}} v_\pi(s') \sum_{r\in\mathcal{R}} p(s',r|s,a)$

> *Exercise 3.14* 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 $+2.3$, $+0.4$, and $+0.7$. (These numbers are accurate only to one decimal place.)

The Bellman equation states

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

The value function in Figure 3.2 shows a policy in which each of the four possible actions is selected with equal probability so we can consider each action component of the sum separately.

north case: for a move north from the center, the agent with 100% probability receives a reward of 0 and ends up in the square directly north of the current square which has a value estimate. So the sums can be replaced with the certain outcomes and the term becomes.
$\sum_{s',r} p(s',r|s,north)\left[r + \gamma v_\pi(s')\right], \text{ for all } s \in \mathcal{S}$
$0 + 0.9 \times 2.3 = 2.07$

Since every other possible action has a completely deterministic state transition and reward outcome, we can directly write down the sum contribution for each one.

south case: $0 + 0.9 \times -0.4 = -0.36$

east case: $0 + 0.9 \times 0.4 = 0.36$

west case: $0 + 0.9 \times 0.7 = 0.63$

Now applying the Bellman equation: $v_\pi(s) = 0.25 \times [2.07 - 0.36 + 0.36 + 0.63] = 0.25 \times 2.7 = 0.675$, which rounded to the nearest decimal value matches the value in the figure of 0.7.

> *Exercise 3.15* 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 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?

Equation 3.8 states:

$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

So if we add a constant c to every reward value then the expected discounted return becomes:

$G'_t = \sum_{k=0}^{\infty} \gamma^k (c + R_{t+k+1}) = \sum_{k=0}^{\infty} c\gamma^k + \gamma^k R_{t+k+1} = \left(c \sum_{k=0}^{\infty} \gamma^k\right) + G_t = \frac{c}{1-\gamma} + G_t$

So the modified estimate is simply the previous estimate plus a constant value which does not change under any state or policy. The constant added is just $\frac{c}{1-\gamma}$. So if a constant value is added to all rewards to remove negative values, it will not affect the relative differences between value estimates of any state under any policy.

> *Exercise 3.16* Now consider adding a constant c to all 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.

In an episodic case, equation 3.8 becomes:

$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$

where T is the length of a particular episode. We can try the same substitution here of adding a constant value to each reward.

$G'_t = \sum_{k=0}^{T-t-1} \gamma^k (c + R_{t+k+1}) = \sum_{k=0}^{T-t-1} c\gamma^k + \gamma^k R_{t+k+1} = \left(c \sum_{k=0}^{T-t-1} \gamma^k\right) + G_t$

Unlike in the previous case, the sum term does not become a constant of c and γ, but is a sum of the form $\sum_{k=0}^{N} \gamma^k$ which can be simplified as follows:

$S = \sum_{k=0}^{N} \gamma^k = \gamma^0 + \gamma^1 + \gamma^2 + \cdots + \gamma^N$
$S\gamma = \gamma^1 + \gamma^2 + \cdots + \gamma^{N+1} = S - 1 + \gamma^{N+1}$
$S(\gamma - 1) = -1 + \gamma^{N+1} \implies S = \frac{\gamma^{N+1}-1}{\gamma-1}$

Substituting this into the modified equation for G we get:

$G'_t = c\frac{\gamma^{T-t}-1}{\gamma-1} + G_t = \left(G_t + \frac{c}{1-\gamma}\right) + \frac{c\gamma^{T-t}}{\gamma-1}$

The part of this equation in parentheses is identical to what we had in the continuing case, but there is an additional term that depends on T (the total episode length) and t (the step we are on of the current episode). To see what this term does to G, we can consider the case of $\gamma = 0.9$ and $T = 100$.

On the first timestep $t = 1$, this term becomes $\frac{c \times 0.9^{99}}{-0.1} \approx c \times -0.0003$. Let's compare that to the final step of the episode $t = T$. Then the term becomes $\frac{c}{-0.1} = c \times -10$. So early on in an episode the constant acts nearly the same as in the continuing case, but as the terminal state approaches there is an increasingly large reduction in the discounted reward estimate which actually cancels out the constant term addition. So we have $G'_t \rightarrow G_t \text{ as } t \rightarrow T$ for the episodic case.

Since early values are shifted up but reduce to the case without the reward constant addition towards the end of episodes, the agent behavior could vary as t increases. If we consider the maze running task, the only reward occurs upon exiting the maze, and it is likely that states closer to the exit would have higher values. At any given step, the relative differences in value between states would be unchanged, so an agent that simply tries to move towards higher value states would not be affected despite the decrease in bonus after each time step. There could be other agents though with more complicated strategies that would exhibit different behavior if value estimates change by a fixed amount across all states with each time step.

> *Exercise 3.17* 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 example in (3.14) but for $q_\pi(s, a)$ intsead of $v_\pi(s)$ we have:

$q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$
$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a]$
$= \sum_{s',r} p(s', r|s, a) \left[r + \gamma \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s']\right]$
$= \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a', s') \mathbb{E}_\pi[G_{t+1} | S_{t+1} = s', A_{t+1} = a']\right]$
$= \sum_{s',r} p(s', r|s, a) \left[r + \gamma \sum_{a'} \pi(a', s') q_\pi(s', a')\right], \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}(s)$

> *Exercise 3.18* The value of a state depends on the values of the actions possible in that state and 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.

In the diagram we see the value function at the root connecting to all of the the possible actions from that state with a corresponding q value. Each action is taken with a probability given by the policy. Since $v_\pi(s)$ is an average over the value of all actions that could be taken by the policy from this point, we can write it in terms of the expected action.

$v_\pi(s) = \mathbb{E}_\pi[q_\pi(S_t, A_t) | S_t = s]$

We can rewrite this using the probabilities given by the policy at each action explicitly:

$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a) \text{ for all } a \in \mathcal{A}(s)$

> *Exercise 3.19* 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 the following 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.

The diagram shows a root for the action value estimate and all of the (future state, reward) pairs that are possible from that action. Since there is a distribution over these pairs, we can write the equation in terms of expected value:

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

From equation 3.2 we have:

$p(s', r|s,a) \doteq \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$
Since this provides the probability for each (future-state, reward) pair that is possible after the current (state, action) pair, we can directly compute the expected action value:

$q_\pi(s,a) = \sum_{r,s'} p(s', r|s,a)(r + \gamma v_\pi(s'))$ for all $r \in \mathcal{R}, s' \in \mathcal{S}$

> *Exercise 3.20* Draw or describe the optimal state-value function for the golf example.

The optimal state-value function assumes that the optimal action is taken which in this case is simply the choice between putter and driver. The optimal action-value function is already shown, so as long as the optimal choice in each region is driver, this function will be identical to the state-value for that state. For all states on the green, the optimal action is putter, not driver, so unlike the bottom of Figure 3.3, any point on the green should have a value of -1. Any point outside the green but within the -2 contour will still have a value of -2 since these states can all reach the green by using the driver. Any point that could reach the green already shares the same value as the optimal state-value function because the optimal action is selected for the subsequent shot.

> *Exercise 3.21* Draw of describe the contours of the optimal action-value function for putting, $q_*(s, \text{putter})$, for the golf example.

$q_*(s, \text{putter})$ restricts the selected initial action to be putter, but any subsequent action selected will be the optimal one. Starting on the green, we still have a value of -1 because any ball on the green can reach the hole in one shot. Next we have the contour shown in the top of Figure 3.3 for -2 which will also be the same since any ball in this state can reach the green and then the hole in the next two strokes. The sand pit however will also share a value of -3 with the rest of the -3 contour. Balls in the sandpit cannot move with the putter so a shot will be wasted, but the driver will be used on the next shot to reach the green and then the hole using a total of 3 shots. The -3 contour will remain -3 for the optimal policy because one putt will be used to reach the -2 contour and then whether driver or putter is selected next, it will take exactly 2 strokes to reach the hole. The contour labeled -4 can reach the -3 region with a putt; however at that point the optimal action would be to use the driver and get a hole in another 2 strokes. Therefore, the -4 contour will be merged into the -3 contour, taking on its value. The -5 contour can reach the -4 contour with a putt. If we approximate that this lies within the driver range of the green then much of this region will also share a value of -3. Only the remaining contour of -6 will take on a value of -4 since puts from this region can only reach a region that is still 3 shots away from a hole.

> *Exercise 3.22* Consider the continuing MDP shows 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 $\gamma = 0.5$?

For $\gamma = 0$ the only reward considered is the immediate one from the chosen action. If we select left, the immediate reward is +1 vs 0 so π_{left} is optimal.

For $\gamma \neq 0$, we can calculate the future discounted reward of each policy:

$G_{\pi_{\text{left}}} = 1 + \gamma^2 + \gamma^4 + \cdots = \frac{1}{1-\gamma^2}$
$G_{\pi_{\text{right}}} = 2 \times (\gamma + \gamma^3 + \cdots) = \frac{2\gamma}{1-\gamma^2} = 2\gamma G_{\pi_{\text{left}}}$

So it is clear that if $\gamma > 0.5$ then π_{right} is more optimal than π_{left} and they are equal if $\gamma = 0.5$.

> *Exercise 3.23* Give the Bellman equation for q_* for the recycling robot.

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

As in example 3.9 we will abbreviate the two states high and low with h, l and the three possible actions of search, wait, and recharge by s, w, re.

Starting with the h state, there are two possible actions of w and s.

$q_*(h,s) = p(h|h,s)[r(h,s,h) + \gamma \max_{a'} q_*(h,a')] + p(l|h,s)[r(h,s,l) + \gamma \max_{a'} q_*(l,a')]$
$q_*(h,s) = \alpha[r_s + \gamma \max_{a'} q_*(h,a')] + (1-\alpha)[r_s + \gamma \max_{a'} q_*(l,a')] = r_s + \gamma[\alpha \max_{a'} q_*(h,a') + (1-\alpha)\max_{a'} q_*(l,a')]$

$q_*(h,w) = r_w + \gamma \max_{a'} q_*(h,a')$

Starting with the l state, there are three possible actions of w, s, and re.

$q_*(l,s) = \beta[r_s + \gamma \max_{a'} q_*(l,a')] + (1-\beta)[-3 + \gamma \max_{a'} q_*(h,a')]$
$q_*(l,w) = r_w + \gamma \max_{a'} q_*(l,a')$
$q_*(l,re) = \gamma \max_{a'} q_*(h,a')$

Together these 5 equations can be solved simultaneously to give 5 values for q_* for each state-action pair.

> *Exercise 3.24* 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.

Equation 3.8 provides the expected discounted return as:

$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

If we assume the same discount factor as when the problem was introduced of $\gamma = 0.9$, then we can iterate through the grid using the optimal policy and collect a sequence of rewards. So for that square, the expected discounted return is:

$G_t = 10[1 + \gamma^5 + \gamma^{10} + \gamma^{15} + \cdots]$

$G_t \times \gamma^5 = 10[\gamma^5 + \gamma^{15} + \cdots] = G_t - 10 \implies G_t = \frac{10}{1-\gamma^5}$

So for $\gamma = 0.9$, $G_t = \frac{10}{1-0.59049} \approx 24.419$

> *Exercise 3.25* Give an equation for v_* in terms of q_*.

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

> *Exercise 3.26* Give an equation for q_* in terms of v_* and the four-argument p.

$q_*(s,a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a]$
$q_*(s,a) = \sum_{r,s'} p(s',r|s,a)[r + \gamma v_*(s')]$

> *Exercise 3.27* Give an equation for π_* in terms of q_*.

$\pi_*(a|s) = 1 \iff a = \text{argmax}_{a \in \mathcal{A}(s)} q_*(s,a) \text{ else } 0$

> *Exercise 3.28* Give an equation for π_* in terms of v_* and the four-argument p.

$\pi_*(a|s) = 1 \iff a = \text{argmax}_{a \in \mathcal{A}(s)} q_*(s,a) = \text{argmax}_{a \in \mathcal{A}(s)} \sum_{r,s'} p(s',r|s,a)[r + \gamma v_*(s')] \text{ else } 0$

> *Exercise 3.29* Rewrite the four Bellman equations for the four value functions $(v_\pi,\ v_*,\ q_\pi,\ \text{and } q_*)$ in terms of the three argument function p (3.4) and the two-argument function r (3.5).

From (3.4) we have:

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

and from (3.5) we have:

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

Starting with v_π:

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

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

Next for q_π:

$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a',s') q_\pi(s',a')\right]$
$q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) \sum_{a'} \pi(a',s') q_\pi(s',a')$
Next for q_*:
$q_*(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} q_*(s',a')\right]$
$q_*(s,a) = r(s,a) + \sum_{s'} p(s'|s,a) \gamma \max_{a'} q_*(s',a')$