Chapter 1

Markov Decision Processes

Contents

1.1 Introduction

The field of RL studies how an agent can learn to make sequential decisions in an interactive environment. This is a very general problem! How can we *formalize* this task in a way that is both *sufficiently general* yet also tractable enough for *fruitful analysis*?

Let's consider some examples of sequential decision problems to identify the key common properties we'd like to capture:

- **Board games** like chess or Go, where the player takes turns with the opponent to make moves on the board.
- **Video games** like Super Mario Bros or Breakout, where the player controls a character to reach the goal.
- **Robotic control**, where the robot can move and interact with the real-world environment to complete some task.

All of these fit into the RL framework. Furthermore, these are environments where the **state transitions**, the "rules" of the environment, only depend on the *most recent* state and action. This is called the **Markov property**.

Definition 1.1.1: Markov property

name: markov

An interactive environment satisfies the **Markov property** if the probability of transitioning to a new state only depends on the current state and action:

$$\mathbb{P}(s_{h+1} \mid s_0, a_0, \dots, s_h, a_h) = P(s_{h+1} \mid s_h, a_h)$$

where $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ describes the state transitions. (We'll elaborate on this notation later in the chapter.)

We'll see that this simple assumption leads to a rich set of problems and algorithms. Environments with the Markov property are called **Markov decision processes** (MDPs) and will be the focus of this chapter.

Exercise: What information might be encoded in the state for each of the above examples? What might the valid set of actions be? Describe the state transitions heuristically and verify that they satisfy the Markov property.

MDPs are usually classified as **finite-horizon**, where the interactions end after some finite number of time steps, or **infinite-horizon**, where the interactions can continue indefinitely. We'll begin with the finite-horizon case and discuss the infinite-horizon case in the second half of the chapter.

In each setting, we'll describe how to evaluate different **policies** (strategies for choosing actions) and how to compute (or approximate) the **optimal policy** for a given MDP. We'll introduce the **Bellman consistency condition**, which allows us to analyze the whole series of interactions in terms of individual timesteps.

1.2 Finite horizon MDPs

Definition 1.2.1: Finite-horizon Markov decision process

name: finite$_{m}dp$

The components of a finite-horizon Markov decision process are:

1. The **state** that the agent interacts with. We use $\mathcal{S}$ to denote the set of possible states, called the **state space**.
2. The **actions** that the agent can take. We use $\mathcal{A}$ to denote the set of possible actions, called the **action space**.
3. Some **initial state distribution** $\mu \in \Delta(\mathcal{S})$.
4. The **state transitions** (a.k.a. **dynamics**) $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ that describe what state the agent transitions to after taking an action.
5. The **reward** signal. In this course we'll take it to be a deterministic function on state-action pairs, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, but in general many results will extend to a *stochastic* reward signal.
6. A time horizon $H \in \mathbb{N}$ that specifies the number of interactions in an **episode**.

Combined together, these objects specify a finite-horizon Markov decision process:

$$M = (\mathcal{S}, \mathcal{A}, \mu, P, r, H).$$

Example 1.2.1: Tidying MDP

name: tidy$_{m}dp$

Let's consider an extremely simple decision problem throughout this chapter: the task of keeping your room tidy!

Your room has the possible states $\mathcal{S} = \{\text{orderly}, \text{messy}\}$. You can take either of the actions $\mathcal{A} = \{\text{tidy}, \text{ignore}\}$. The room starts off orderly.

The state transitions are as follows: if you tidy the room, it becomes (or remains) orderly; if you ignore the room, it might become messy.

The rewards are as follows: You get penalized for tidying an orderly room (a waste of time) or ignoring a messy room, but you get rewarded for ignoring an orderly room (since you can enjoy). Tidying a messy room is a chore that gives no reward.

These are summarized in the following table:

s	a	$P($orderly $\mid s, a)$	$P($messy $\mid s, a)$	$r(s, a)$
orderly	tidy	1	0	-1
orderly	ignore	0.7	0.3	1
messy	tidy	1	0	0
messy	ignore	0	1	-1

Consider a time horizon of $H = 7$ days (one interaction per day). Let $t = 0$ correspond to Monday and $t = 6$ correspond to Sunday.

1.2.1 Policies

Definition 1.2.2: Policies

name: policy

A **policy** π describes the agent's strategy: which actions it takes in a given situation. A key goal of RL is to find the **optimal policy** that maximizes the total reward on average.

There are three axes along which policies can vary: their outputs, inputs, and time-dependence. We'll discuss each of these in turn.

1. **Deterministic or stochastic.** A deterministic policy outputs actions while a stochastic policy outputs *distributions* over actions.
2. **State-dependent or history-dependent.** A state-dependent (a.k.a. "Markovian") policy only depends on the current state, while a history-dependent policy depends on the sequence of past states, actions, and rewards. We'll only consider state-dependent policies in this course.
3. **Stationary or time-dependent.** A stationary policy remains the same function at all time steps, while a time-dependent policy $\pi = \{\pi_0, \dots, \pi_{H-1}\}$ specifies a different function π_h at each time step h.

A fascinating result is that every finite-horizon MDP has an optimal deterministic time-dependent policy! Intuitively, the Markov property implies that the current state contains all the information we need to make the optimal decision. We'll prove this result constructively later in the chapter.

Example 1.2.2: Tidying policies

name: tidy$_policy$

Here are some possible policies for the tidying MDP (**??**):

- Always tidy: $\pi(s) =$ tidy.

- Only tidy on weekends: $\pi_h(s) =$ tidy if $h \in \{5, 6\}$ and $\pi_h(s) =$ ignore otherwise.
- Only tidy if the room is messy: $\pi_h(\text{messy}) =$ tidy and $\pi_h(\text{orderly}) =$ ignore for all h.

1.2.2 Trajectories

Definition 1.2.3: Trajectories

name: trajectory

A sequence of states, actions, and rewards is called a **trajectory**:

$$\tau = (s_0, a_0, r_0, \dots, s_{H-1}, a_{H-1}, r_{H-1})$$

where $r_h = r(s_h, a_h)$. (Note that sources differ as to whether to include the reward at the final time step. This is a minor detail.)

Once we've chosen a policy, we can sample trajectories by repeatedly choosing actions according to the policy, transitioning according to the state transitions, and observing the rewards. That is, a policy induces a distribution ρ^π over trajectories. (We assume that μ and P are clear from context.)

Example 1.2.3: Trajectories in the tidying environment

name: tidy$_t raj$

Here is a possible trajectory for the tidying example:

t	0	1	2	3	4	5	6
s	orderly	orderly	orderly	messy	messy	orderly	orderly
a	tidy	ignore	ignore	ignore	tidy	ignore	ignore
r	-1	1	1	-1	0	1	1

Could any of the policies in **??** have generated this trajectory?

Note that for a state-dependent policy, using the Markov property (**??**), we can specify this probability distribution in an **autoregressive** way (i.e. one timestep at a time):

Definition 1.2.4: Autoregressive trajectory distribution

name: autoregressive$_t rajectories$

$$\rho^\pi(\tau) := \mu(s_0)\pi_0(a_0 \mid s_0)P(s_1 \mid s_0, a_0) \cdots P(s_{H-1} \mid s_{H-2}, a_{H-2})\pi_{H-1}(a_{H-1} \mid s_{H-1})$$

Exercise: How would you modify this to include stochastic rewards?

For a deterministic policy π, we have that $\pi_h(a \mid s) = \mathbb{I}[a = \pi_h(s)]$; that is, the probability of taking an action is 1 if it's the unique action prescribed by the policy for that state and 0 otherwise. In this case, the only randomness in sampling trajectories comes from the initial state distribution μ and the state transitions P.

1.2.3 Value functions

The main goal of RL is to find a policy that maximizes the average total reward $r_0 + \cdots + r_{H-1}$. (Note that this is a random variable that depends on the policy.) Let's introduce some notation for analyzing this quantity.

A policy's **value function** is its expected total reward *starting in a given state at a given time*:

Definition 1.2.5: Value function

name: value

$$V_h^\pi(s) := \mathop{\mathbb{E}}_{\tau \sim \rho^\pi} [r_h + \cdots + r_{H-1} \mid s_h = s]$$

Similarly, we can define the **action-value function** (aka the **Q-function**) as the expected total reward when starting in a given state and taking a given action:

Definition 1.2.6: Action-value function

name: action$_value$

$$Q_h^\pi(s, a) := \mathop{\mathbb{E}}_{\tau \sim \rho^\pi} [r_h + \cdots + r_{H-1} \mid s_h = s, a_h = a]$$

Note that the value function is just the average action-value over actions drawn from the policy:

$$V_h^\pi(s) = \mathop{\mathbb{E}}_{a \sim \pi_h(s)} [Q_h^\pi(s, a)]$$

and the action-value can be expressed in terms of the value of the following state:

$$Q_h^\pi(s, a) = r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} [V_{h+1}^\pi(s')]$$

The one-step (Bellman) consistency equation

Note that by simply considering the cumulative reward as the sum of the *current* reward and the *future* cumulative reward, we can describe the value function recursively (in terms of itself). This is named the **Bellman consistency equation** after **Richard Bellman** (1920–1984), who is credited with introducing dynamic programming in 1953.

Definition 1.2.7: Bellman consistency equation for the value function

name: bellman$_consistency$

$$V_h^\pi(s) = \mathop{\mathbb{E}}_{\substack{a \sim \pi_h(s) \\ s' \sim P(s, a)}} [r(s, a) + V_{h+1}^\pi(s')] \tag{1.1}$$

Exercise: Verify that this equation holds by expanding $V_h^\pi(s)$ and $V_{h+1}^\pi(s')$.

One can analogously derive the Bellman consistency equation for the action-value function:

Definition 1.2.8: Bellman consistency equation for action-values

name: bellman$_consistency_action$

$$Q_h^\pi(s, a) = r(s, a) + \mathop{\mathbb{E}}_{\substack{s' \sim P(s, a) \\ a' \sim \pi_{h+1}(s')}} [Q_{h+1}^\pi(s', a')]$$

The Bellman consistency equation for deterministic policies :name: bellman$_det$

Note that for deterministic policies, the Bellman consistency equation simplifies to

$$\begin{aligned} V_h^\pi(s) &= r(s, \pi_h(s)) + \mathop{\mathbb{E}}_{s' \sim P(s, \pi_h(s))} [V_{h+1}^\pi(s')] \\ Q_h^\pi(s, a) &= r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} [Q_{h+1}^\pi(s', \pi_{h+1}(s'))] \end{aligned}$$

The Bellman operator

Fix a policy π. Consider the higher-order operator that takes in a "value function" $v : \mathcal{S} \to \mathbb{R}$ and returns the r.h.s. of the Bellman equation for that "value function":

Definition 1.2.9: Bellman operator

name: bellman$_operator$

$$[\mathcal{J}^\pi(v)](s) := \mathop{\mathbb{E}}_{\substack{a \sim \pi(s) \\ s' \sim P(s, a)}} [r(s, a) + v(s')].$$

We'll call $\mathcal{J}^\pi : (\mathcal{S} \to \mathbb{R}) \to (\mathcal{S} \to \mathbb{R})$ the **Bellman operator** of π. Note that it's defined on any "value function" mapping states to real numbers; v doesn't have to be a well-defined value function for some policy (hence the lowercase notation). The Bellman operator also gives us a concise way to express the Bellman consistency equation (**??**):

$$V_h^\pi = \mathcal{J}^\pi(V_{h+1}^\pi)$$

Intuitively, the output of the Bellman operator, a new "value function", evaluates states as follows: from a given state, take one action according to π, observe the reward, and then evaluate the next state using the input "value function".

When we discuss infinite-horizon MDPs, the Bellman operator will turn out to be more than just a notational convenience: We'll use it to construct algorithms for computing the optimal policy.

1.2.4 Policy evaluation

How can we actually compute the value function of a given policy? This is the task of **policy evaluation**.

Dynamic programming

The Bellman consistency equation (**??**) gives us a convenient algorithm for evaluating stationary policies: it expresses the value function at timestep h as a function of the value function at timestep $h+1$. This means we can start at the end of the time horizon, where the value is known, and work backwards in time, using the Bellman consistency equation to compute the value function at each time step.

Definition 1.2.10: Dynamic programming for policy evaluation

name: dp$_e val_f inite$

```
$V_h(s) \leftarrow 0$ for all $t \in \{0, \dots, H\}, s \in S$
for $t = H - 1, \dots, 0$ do
    for $s \in \mathcal{S}, a \in \mathcal{A}, s' \in \mathcal{S}$ do
        $V_h(s) \leftarrow V_h(s) + \pi_h(a \mid s)P(s' \mid s, a)[r(s, a) + V_{h+1}(s')]$
    end for
end for
```

This clearly runs in time $O(H \cdot |\mathcal{S}|^2 \cdot |\mathcal{A}|)$ by counting the loops.

Exercise: Do you see where we compute Q^π_h along the way? Make this step explicit.

Example 1.2.4: Tidying policy evaluation

name: tidy$_e val_f inite$

Let's evaluate the policy from **??** that tidies if and only if the room is messy. We'll use the Bellman consistency equation to compute the value function at each time step.

$$\begin{aligned}
V^\pi_{H-1}(\text{orderly}) &= r(\text{orderly}, \text{ignore}) \\
&= 1 \\
V^\pi_{H-1}(\text{messy}) &= r(\text{messy}, \text{tidy}) \\
&= 0
\end{aligned}$$

$$\begin{aligned}
V^\pi_{H-2}(\text{orderly}) &= r(\text{orderly}, \text{ignore}) + \mathop{\mathbb{E}}_{s' \sim P(\text{orderly}, \text{ignore})} [V^\pi_{H-1}(s')] \\
&= 1 + 0.7 \cdot V^\pi_{H-1}(\text{orderly}) + 0.3 \cdot V^\pi_{H-1}(\text{messy}) \\
&= 1 + 0.7 \cdot 1 + 0.3 \cdot 0 \\
&= 1.7 \\
V^\pi_{H-2}(\text{messy}) &= r(\text{messy}, \text{tidy}) + \mathop{\mathbb{E}}_{s' \sim P(\text{messy}, \text{tidy})} [V^\pi_{H-1}(s')] \\
&= 0 + 1 \cdot V^\pi_{H-1}(\text{orderly}) + 0 \cdot V^\pi_{H-1}(\text{messy}) \\
&= 1 \\
V^\pi_{H-3}(\text{orderly}) &= r(\text{orderly}, \text{ignore}) + \mathop{\mathbb{E}}_{s' \sim P(\text{orderly}, \text{ignore})} [V^\pi_{H-2}(s')] \\
&= 1 + 0.7 \cdot V^\pi_{H-2}(\text{orderly}) + 0.3 \cdot V^\pi_{H-2}(\text{messy}) \\
&= 1 + 0.7 \cdot 1.7 + 0.3 \cdot 1 \\
&= 2.49 \\
V^\pi_{H-3}(\text{messy}) &= r(\text{messy}, \text{tidy}) + \mathop{\mathbb{E}}_{s' \sim P(\text{messy}, \text{tidy})} [V^\pi_{H-2}(s')] \\
&= 0 + 1 \cdot V^\pi_{H-2}(\text{orderly}) + 0 \cdot V^\pi_{H-2}(\text{messy}) \\
&= 1.7
\end{aligned}$$

etc. You may wish to repeat this computation for the other policies to get a better sense of this algorithm.

1.2.5 Optimal policies

We've just seen how to *evaluate* a given policy. But how can we find the **optimal policy** for a given environment?

Definition 1.2.11: Optimal policies

name: optimal$_{policy}$

We call a policy optimal, and denote it by $\pi^\star$, if it does at least as well as *any* other policy π (including stochastic and history-dependent ones) in all situations:

$$\begin{aligned}
V_h^{\pi^\star}(s) &= \mathop{\mathbb{E}}_{\tau \sim \rho^{\pi^\star}} [r_h + \cdots + r_{H-1} \mid s_h = s] \\
&\geq \mathop{\mathbb{E}}_{\tau \sim \rho^{\pi}} [r_h + \cdots + r_{H-1} \mid \tau_h] \quad \forall \pi, \tau_h, h \in [H]
\end{aligned} \tag{1.2}$$

where we condition on the trajectory up to time h, denoted $\tau_h = (s_0, a_0, r_0, \ldots, s_h)$, where $s_h = s$.

Convince yourself that all optimal policies must have the same value function. We call this the **optimal value function** and denote it by $V_h^\star(s)$. The same goes for the action-value function

$Q^\star_h(s, a)$.

It is a stunning fact that **every finite-horizon MDP has an optimal policy that is time-dependent and deterministic.** In particular, we can construct such a policy by acting *greedily* with respect to the optimal action-value function:

$$\pi^\star_h(s) = \arg\max_a Q^\star_h(s, a).$$

Theorem 1.2.1: It is optimal to be greedy w.r.t. the optimal value function

name: optimal$_{greedy}$

Let $V^\star$ and $Q^\star$ denote the optimal value and action-value functions. Consider the greedy policy

$$\hat\pi_h(s) := \arg\max_a Q^\star_h(s, a).$$

We aim to show that $\hat\pi$ is optimal; that is, $V^{\hat\pi} = V^\star$.

Fix an arbitrary state $s \in \mathcal{S}$ and time $h \in [H]$.

Firstly, by the definition of $V^\star$, we already know $V^\star_h(s) \ge V^{\hat\pi}_h(s)$. So for equality to hold we just need to show that $V^\star_h(s) \le V^{\hat\pi}_h(s)$. We'll first show that the Bellman operator $\mathcal{J}^{\hat\pi}$ never decreases $V^\star_h$. Then we'll apply this result recursively to show that $V^\star = V^{\hat\pi}$.

Lemma: $\mathcal{J}^{\hat\pi}$ never decreases $V^\star_h$ (elementwise):

$$[\mathcal{J}^{\hat\pi}(V^\star_{h+1})](s) \ge V^\star_h(s).$$

Proof:

$$\begin{aligned}
V^\star_h(s) &= \max_{\pi \in \Pi} V^\pi_h(s) \\
&= \max_{\pi \in \Pi} \mathop{\mathbb{E}}_{a \sim \pi(\dots)} \left[r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} V^\pi_{h+1}(s') \right] && \text{Bellman consistency} \\
&\le \max_{\pi \in \Pi} \mathop{\mathbb{E}}_{a \sim \pi(\dots)} \left[r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} V^\star_{h+1}(s') \right] && \text{definition of } V^\star \\
&= \max_{a} \left[r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} V^\star_{h+1}(s') \right] && \text{only depends on } \pi \text{ via } a \\
&= [\mathcal{J}^{\hat\pi}(V^\star_{h+1})](s).
\end{aligned}$$

Note that the chosen action $a \sim \pi(\dots)$ above might depend on the past history; this isn't shown in the notation and doesn't affect our result (make sure you see why).

We can now apply this result recursively to get

$$V^\star_t(s) \le V^{\hat\pi}_t(s)$$

as follows. (Note that even though $\hat{\pi}$ is deterministic, we'll use the $a \sim \hat{\pi}(s)$ notation to make it explicit that we're sampling a trajectory from it.)

$$
\begin{aligned}
V_t^\star(s) &\le [\mathcal{J}^{\hat{\pi}}(V_{h+1}^\star)](s) \\
&= \mathbb{E}_{a \sim \hat{\pi}(s)} \left[r(s,a) + \mathbb{E}_{s' \sim P(s,a)} \left[V_{h+1}^\star(s') \right] \right] && \text{definition of } \mathcal{J}^{\hat{\pi}} \\
&\le \mathbb{E}_{a \sim \hat{\pi}(s)} \left[r(s,a) + \mathbb{E}_{s' \sim P(s,a)} \left[[\mathcal{J}^{\hat{\pi}}(V_{t+2}^\star)](s') \right] \right] && \text{above lemma} \\
&= \mathbb{E}_{a \sim \hat{\pi}(s)} \left[r(s,a) + \mathbb{E}_{s' \sim P(s,a)} \left[\mathbb{E}_{a' \sim \hat{\pi}} r(s',a') + \mathbb{E}_{s''} V_{t+2}^\star(s'') \right] \right] && \text{definition of } \mathcal{J}^{\hat{\pi}} \\
&\le \cdots && \text{apply at all timesteps} \\
&= \mathbb{E}_{\tau \sim \rho^{\hat{\pi}}} [G_t \mid s_h = s] && \text{rewrite expectation} \\
&= V_t^{\hat{\pi}}(s) && \text{definition}
\end{aligned}
$$

And so we have $V^\star = V^{\hat{\pi}}$, making $\hat{\pi}$ optimal.

Dynamic programming

Now that we've shown this particular greedy policy is optimal, all we need to do is compute the optimal value function and optimal policy. We can do this by working backwards in time using **dynamic programming** (DP).

Definition 1.2.12: DP for optimal policy

name: pi$_s tar_d p$

We can solve for the optimal policy in an finite-horizon MDP using **dynamic programming**.

- *Base case.* At the end of the episode (time step $H-1$), we can't take any more actions, so the Q-function is simply the reward that we obtain:

  $$Q_{H-1}^\star(s,a) = r(s,a)$$

  so the best thing to do is just act greedily and get as much reward as we can!

  $$\pi_{H-1}^\star(s) = \arg\max_a Q_{H-1}^\star(s,a)$$

  Then $V_{H-1}^\star(s)$, the optimal value of state s at the end of the trajectory, is simply whatever action gives the most reward.

  $$V_{H-1}^\star = \max_a Q_{H-1}^\star(s,a)$$

- *Recursion.* Then, we can work backwards in time, starting from the end, using our consistency equations! i.e. for each $t = H - 2, \dots, 0$, we set

$$\begin{aligned} Q_t^\star(s, a) &= r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} [V_{h+1}^\star(s')] \\ \pi_t^\star(s) &= \arg \max_a Q_t^\star(s, a) \\ V_t^\star(s) &= \max_a Q_t^\star(s, a) \end{aligned}$$

At each of the H timesteps, we must compute $Q^\star$ for each of the $|\mathcal{S}||\mathcal{A}|$ state-action pairs. Each computation takes $|\mathcal{S}|$ operations to evaluate the average value over s'. This gives a total computation time of $O(H|\mathcal{S}|^2|\mathcal{A}|)$.

Note that this algorithm is identical to the policy evaluation algorithm **??**, but instead of *averaging* over the actions chosen by a policy, we instead simply take a *maximum* over the action-values. We'll see this relationship between **policy evaluation** and **optimal policy computation** show up again in the infinite-horizon setting.

Example 1.2.5: Optimal policy for the tidying MDP

name: tidy$_dp$

Left as an exercise.

1.3 Infinite horizon MDPs

What happens if a trajectory is allowed to continue forever (i.e. $H = \infty$)? This is the setting of **infinite horizon** MDPs.

In this chapter, we'll describe the necessary adjustments from the finite-horizon case to make the problem tractable. We'll show that the Bellman operator (**??**) in the discounted reward setting is a **contraction mapping** for any policy. We'll discuss how to evaluate policies (i.e. compute their corresponding value functions). Finally, we'll present and analyze two iterative algorithms, based on the Bellman operator, for computing the optimal policy: **value iteration** and **policy iteration**.

1.3.1 Differences from finite-horizon

Discounted rewards

First of all, note that maximizing the cumulative reward $r_h + r_{h+1} + r_{h+2} + \cdots$ is no longer a good idea since it might blow up to infinity. Instead of a time horizon H, we now need a **discount factor** $\gamma \in [0, 1)$ such that rewards become less valuable the further into the future they are:

$$r_h + \gamma r_{h+1} + \gamma^2 r_{h+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{h+k}.$$

We can think of γ as measuring how much we care about the future: if it's close to 0, we only care about the near-term rewards; it's close to 1, we put more weight into future rewards.

You can also analyze γ as the probability of *continuing* the trajectory at each time step. (This is equivalent to H being distributed by a First Success distribution with success probability γ.) This accords with the above interpretation: if γ is close to 0, the trajectory will likely be very short, while if γ is close to 1, the trajectory will likely continue for a long time.

Exercise: Assuming that $r_h \in [0, 1]$ for all $h \in \mathbb{N}$, what is the maximum **discounted** cumulative reward? You may find it useful to review geometric series.

The other components of the MDP remain the same:

$$M = (\mathcal{S}, \mathcal{A}, \mu, P, r, \gamma).$$

Stationary policies

The time-dependent policies from the finite-horizon case become difficult to handle in the infinite-horizon case. In particular, many of the DP approaches we saw required us to start at the end of the trajectory, which is no longer possible. We'll shift to **stationary** policies $\pi : \mathcal{S} \to \mathcal{A}$ (deterministic) or $\Delta(\mathcal{A})$ (stochastic).

Exercise: Which of the policies in **??** are stationary?

Value functions and Bellman consistency

We also consider stationary value functions $V^\pi : \mathcal{S} \to \mathbb{R}$ and $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. We need to insert a factor of γ into the Bellman consistency equation (**??**) to account for the discounting:

$$\begin{aligned}
V^\pi(s) &= \mathop{\mathbb{E}}_{\tau \sim \rho^\pi}[r_h + \gamma r_{h+1} + \gamma^2 r_{h+2} \cdots \mid s_h = s] && \text{for any } h \in \mathbb{N} \\
&= \mathop{\mathbb{E}}_{\substack{a \sim \pi(s) \\ s' \sim P(s,a)}} [r(s,a) + \gamma V^\pi(s')] \\
Q^\pi(s,a) &= \mathop{\mathbb{E}}_{\tau \sim \rho^\pi}[r_h + \gamma r_{h+1} + \gamma^2 r_{h+2} + \cdots \mid s_h = s, a_h = a] && \text{for any } h \in \mathbb{N} \\
&= r(s,a) + \gamma \mathop{\mathbb{E}}_{\substack{s' \sim P(s,a) \\ a' \sim \pi(s')}} [Q^\pi(s', a')]
\end{aligned} \tag{1.3}$$

Exercise: Heuristically speaking, why does it no longer matter which time step we condition on when defining the value function?

1.3.2 The Bellman operator is a contraction mapping

Recall from **??** that the Bellman operator $\mathcal{J}^\pi$ for a policy π takes in a "value function" $v : \mathcal{S} \to \mathbb{R}$ and returns the r.h.s. of the Bellman equation for that "value function". In the infinite-horizon setting, this is

$$[\mathcal{J}^\pi(v)](s) := \mathop{\mathbb{E}}_{\substack{a \sim \pi(s) \\ s' \sim P(s,a)}} [r(s,a) + \gamma v(s')].$$

The crucial property of the Bellman operator is that it is a **contraction mapping** for any policy. Intuitively, if we start with two "value functions" $v, u : \mathcal{S} \to \mathbb{R}$, if we repeatedly apply the Bellman operator to each of them, they will get closer and closer together at an exponential rate.

Definition 1.3.1: Contraction mapping

name: contraction

Let X be some space with a norm $\|\cdot\|$. We call an operator $f : X \to X$ a **contraction mapping** if for any $x, y \in X$,

$$\|f(x) - f(y)\| \le \gamma \|x - y\|$$

for some fixed $\gamma \in (0, 1)$.

Exercise: Show that for a contraction mapping f with coefficient γ, for all $t \in \mathbb{N}$,

$$\|f^{(t)}(x) - f^{(t)}(y)\| \le \gamma^t \|x - y\|,$$

i.e. that any two points will be pushed closer by at least a factor of γ at each iteration.

It is a powerful fact (known as the **Banach fixed-point theorem**) that every contraction mapping has a unique **fixed point** $x^\star$ such that $f(x^\star) = x^\star$. This means that if we repeatedly apply f to any starting point, we will eventually converge to $x^\star$:

$$\|f^{(t)}(x) - x^\star\| \le \gamma^t \|x - x^\star\|. \tag{1.4}$$

Let's return to the RL setting and apply this result to the Bellman operator. How can we measure the distance between two "value functions" $v, u : \mathcal{S} \to \mathbb{R}$? We'll take the **supremum norm** as our distance metric:

$$\|v - u\|_\infty := \sup_{s \in \mathcal{S}} |v(s) - u(s)|,$$

i.e. we compare the "value functions" on the state that causes the biggest gap between them. Then (**??**) implies that if we repeatedly apply $\mathcal{J}^\pi$ to any starting "value function", we will eventually converge to V^π:

$$\|(\mathcal{J}^\pi)^{(t)}(v) - V^\pi\|_\infty \le \gamma^t \|v - V^\pi\|_\infty. \tag{1.5}$$

We'll use this useful fact to prove the convergence of several algorithms later on.

Theorem 1.3.1: The Bellman operator is a contraction mapping

name: bellman$_contraction$

We aim to show that

$$\|\mathcal{J}^\pi(v) - \mathcal{J}^\pi(u)\|_\infty \le \gamma \|v - u\|_\infty.$$

Proof: for all states $s \in \mathcal{S}$,

$$\begin{aligned}
|[\mathcal{J}^\pi(v)](s) - [\mathcal{J}^\pi(u)](s)| &= \Bigg| \mathop{\mathbb{E}}_{a \sim \pi(s)} \left[r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(s,a)} v(s') \right] \\
&\qquad - \mathop{\mathbb{E}}_{a \sim \pi(s)} \left[r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(s,a)} u(s') \right] \Bigg| \\
&= \gamma \left| \mathop{\mathbb{E}}_{s' \sim P(s,a)} [v(s') - u(s')] \right| \\
&\le \gamma \mathop{\mathbb{E}}_{s' \sim P(s,a)} |v(s') - u(s')| \qquad \text{(Jensen's inequality)} \\
&\le \gamma \max_{s'} |v(s') - u(s')| \\
&= \gamma \|v - u\|_\infty.
\end{aligned}$$

1.3.3 Tabular case (linear algebraic notation)

When there are **finitely** many states and actions, i.e. $|\mathcal{S}|, |\mathcal{A}| < \infty$, we call the MDP **tabular** since we can express the relevant quantities as vectors and matrices (i.e. *tables* of values):

$$r \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{A}|} \qquad P \in [0,1]^{(|\mathcal{S} \times \mathcal{A}|) \times |\mathcal{S}|} \qquad \mu \in [0,1]^{|\mathcal{S}|}$$

$$\pi \in [0,1]^{|\mathcal{A}|\times|\mathcal{S}|} \qquad V^\pi \in \mathbb{R}^{|\mathcal{S}|} \qquad Q^\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}.$$

(Verify that these types make sense!)

Note that when the policy π is deterministic, the actions can be determined from the states, and so we can chop off the action dimension for the rewards and state transitions:

$$\begin{aligned} &r^\pi \in \mathbb{R}^{|\mathcal{S}|} && P^\pi \in [0,1]^{|\mathcal{S}|\times|\mathcal{S}|} && \mu \in [0,1]^{|\mathcal{S}|} \\ &\pi \in \mathcal{A}^{|\mathcal{S}|} && V^\pi \in \mathbb{R}^{|\mathcal{S}|} && Q^\pi \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{A}|}. \end{aligned}$$

For P^π, we'll treat the rows as the states and the columns as the next states. Then $P^\pi_{s,s'}$ is the probability of transitioning from state s to state s' under policy π.

Example 1.3.1: Tidying MDP

name: tidy$_tabular$

The tabular MDP from before has $|\mathcal{S}| = 2$ and $|\mathcal{A}| = 2$. Let's write down the quantities for the policy π that tidies if and only if the room is messy:

$$r^\pi = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad P^\pi = \begin{bmatrix} 0.7 & 0.3 \\ 1 & 0 \end{bmatrix}, \quad \mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

We'll see how to evaluate this policy in the next section.

1.3.4 Policy evaluation

The backwards DP technique we used in the finite-horizon case (**??**) no longer works since there is no "final timestep" to start from. We'll need another approach to policy evaluation.

The Bellman consistency conditions yield a system of equations we can solve to evaluate a policy *exactly*. For a faster approximate solution, we can iterate the policy's Bellman operator, since we know that it has a unique fixed point at the true value function.

Tabular case for deterministic policies

The Bellman consistency equation for a deterministic policy can be written in tabular notation as

$$V^\pi = r^\pi + \gamma P^\pi V^\pi.$$

(Unfortunately, this notation doesn't simplify the expression for Q^π.) This system of equations can be solved with a matrix inversion:

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi. \tag{1.6}$$

Note we've assumed that $I - \gamma P^\pi$ is invertible. Can you see why this is the case?

(Recall that a linear operator, i.e. a square matrix, is invertible if and only if its null space is trivial; that is, it doesn't map any nonzero vector to zero. In this case, we can see that $I - \gamma P^\pi$ is invertible because it maps any nonzero vector to a vector with at least one nonzero element.)

Example 1.3.2: Tidying policy evaluation

name: tidy$_{e}val_{i}nfinite$

Let's use the same policy π that tidies if and only if the room is messy. Setting $\gamma = 0.95$, we must invert

$$I - \gamma P^\pi = \begin{bmatrix} 1 - 0.95 \times 0.7 & -0.95 \times 0.3 \\ -0.95 \times 1 & 1 - 0.95 \times 0 \end{bmatrix} = \begin{bmatrix} 0.335 & -0.285 \\ -0.95 & 1 \end{bmatrix}.$$

The inverse to two decimal points is

$$(I - \gamma P^\pi)^{-1} = \begin{bmatrix} 15.56 & 4.44 \\ 14.79 & 5.21 \end{bmatrix}.$$

Thus the value function is

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi = \begin{bmatrix} 15.56 & 4.44 \\ 14.79 & 5.21 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 15.56 \\ 14.79 \end{bmatrix}.$$

Let's sanity-check this result. Since rewards are at most 1, the maximum cumulative return of a trajectory is at most $1/(1-\gamma) = 20$. We see that the value function is indeed slightly lower than this.

Iterative policy evaluation

The matrix inversion above takes roughly $O(|\mathcal{S}|^3)$ time. Can we trade off the requirement of finding the *exact* value function for a faster *approximate* algorithm?

Let's use the Bellman operator to define an iterative algorithm for computing the value function. We'll start with an initial guess $v^{(0)}$ with elements in $[0, 1/(1-\gamma)]$ and then iterate the Bellman operator:

$$v^{(t+1)} = \mathcal{J}^\pi(v^{(t)}) = r^\pi + \gamma P^\pi v^{(t)},$$

i.e. $v^{(t)} = (\mathcal{J}^\pi)^{(t)}(v^{(0)})$. Note that each iteration takes $O(|\mathcal{S}|^2)$ time for the matrix-vector multiplication.

Then, as we showed in (**??**), by the Banach fixed-point theorem:

$$\|v^{(t)} - V^\pi\|_\infty \le \gamma^t \|v^{(0)} - V^\pi\|_\infty.$$

How many iterations do we need for an ϵ-accurate estimate? We can work backwards to solve for t:

$$\begin{aligned} \gamma^t \|v^{(0)} - V^\pi\|_\infty &\le \epsilon \\ t &\ge \frac{\log(\epsilon / \|v^{(0)} - V^\pi\|_\infty)}{\log \gamma} \\ &= \frac{\log(\|v^{(0)} - V^\pi\|_\infty / \epsilon)}{\log(1/\gamma)}, \end{aligned}$$

and so the number of iterations required for an ϵ-accurate estimate is

$$T = O\left(\frac{1}{1-\gamma}\log\left(\frac{1}{\epsilon(1-\gamma)}\right)\right). \tag{1.7}$$

Note that we've applied the inequalities $\|v^{(0)} - V^\pi\|_\infty \le 1/(1-\gamma)$ and $\log(1/x) \ge 1 - x$.

1.3.5 Optimal policies

Now let's move on to solving for an optimal policy in the infinite-horizon case. As in the finite-horizon case (**??**), an **optimal policy** $\pi^\star$ is one that does at least as well as any other policy in all situations. That is, for all policies π, states $s \in \mathcal{S}$, times $h \in \mathbb{N}$, and initial trajectories $\tau_h = (s_0, a_0, r_0, \dots, s_h)$ where $s_h = s$,

$$\begin{aligned} V^{\pi^\star}(s) &= \mathop{\mathbb{E}}_{\tau \sim \rho^{\pi^\star}}[r_h + \gamma r_{h+1} + \gamma^2 r_{h+2} + \cdots \mid s_h = s] \\ &\ge \mathop{\mathbb{E}}_{\tau \sim \rho^{\pi}}[r_h + \gamma r_{h+1} + \gamma^2 r_{h+2} + \cdots \mid \tau_h] \end{aligned} \tag{1.8}$$

Once again, all optimal policies share the same **optimal value function** $V^\star$, and the greedy policy w.r.t. this value function is optimal.

Exercise: Verify this by modifying the proof **??** from the finite-horizon case.

So how can we compute such an optimal policy? We can't use the backwards DP approach from the finite-horizon case (**??**) since there's no "final timestep" to start from. Instead, we'll exploit the fact that the Bellman consistency equation (**??**) for the optimal value function doesn't depend on any policy:

$$V^\star(s) = \max_a \left[r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(s,a)} V^\star(s'). \right]$$

Exercise: Verify this by substituting the greedy policy into the Bellman consistency equation.

As before, thinking of the r.h.s. as an operator on value functions gives the **Bellman optimality operator**

$$[\mathcal{J}^\star(v)](s) = \max_a \left[r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(s,a)} v(s') \right].$$

Value iteration

Since the optimal policy is still a policy, our result that the Bellman operator is a contracting map still holds, and so we can repeatedly apply this operator to converge to the optimal value function! This algorithm is known as **value iteration**.

Definition 1.3.2: Value iteration pseudocode

name: value$_{i}teration_{p}seudocode$

```
$v^{(0)} \leftarrow 0$
for $t = 0, 1, 2, \dots, T-1$ do
    $v^{(t+1)} \leftarrow \mathcal{J}^\star(v^{(t)})$
end for
return $v^{(T)}$
```

Note that the runtime analysis for an ϵ-optimal value function is exactly the same as iterative policy evaluation (**??**)! This is because value iteration is simply the special case of applying iterative policy evaluation to the *optimal* value function.

As the final step of the algorithm, to return an actual policy $\hat{\pi}$, we can simply act greedily w.r.t. the final iteration $v^{(T)}$ of our above algorithm:

$$\hat{\pi}(s) = \arg\max_a \left[r(s,a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(s,a)} v^{(T)}(s') \right].$$

We must be careful, though: the value function of this greedy policy, $V^{\hat{\pi}}$, is *not* the same as $v^{(T)}$, which need not even be a well-defined value function for some policy!

The bound on the policy's quality is actually quite loose: if $\|v^{(T)} - V^\star\|_\infty \le \epsilon$, then the greedy policy $\hat{\pi}$ satisfies $\|V^{\hat{\pi}} - V^\star\|_\infty \le \frac{2\gamma}{1-\gamma}\epsilon$, which might potentially be very large.

Theorem 1.3.2: Greedy policy value worsening

name: greedy$_{w}orsen$

We aim to show that

$$\|V^{\hat{\pi}} - V^\star\|_\infty \le \frac{2\gamma}{1-\gamma}\|v - V^\star\|_\infty$$

where $\hat{\pi}(s) = \arg\max_a q(s,a)$ is the greedy policy w.r.t.

$$q(s,a) = r(s,a) + \mathop{\mathbb{E}}_{s' \sim P(s,a)} v(s').$$

Proof: We first have

$$\begin{aligned} V^\star(s) - V^{\hat{\pi}}(s) &= Q^\star(s, \pi^\star(s)) - Q^{\hat{\pi}}(s, \hat{\pi}(s)) \\ &= [Q^\star(s, \pi^\star(s)) - Q^\star(s, \hat{\pi}(s))] + [Q^\star(s, \hat{\pi}(s)) - Q^{\hat{\pi}}(s, \hat{\pi}(s))]. \end{aligned}$$

Let's bound these two quantities separately.

For the first quantity, note that by the definition of $\hat{\pi}$, we have

$$q(s, \hat{\pi}(s)) \ge q(s, \pi^\star(s)).$$

Let's add $q(s, \hat{\pi}(s)) - q(s, \pi^\star(s)) \ge 0$ to the first term to get

$$\begin{aligned}
Q^\star(s, \pi^\star(s)) - Q^\star(s, \hat{\pi}(s)) &\le [Q^\star(s, \pi^\star(s)) - q(s, \pi^\star(s))] + [q(s, \hat{\pi}(s)) - Q^\star(s, \hat{\pi}(s))] \\
&= \gamma \mathop{\mathbb{E}}_{s' \sim P(s, \pi^\star(s))} [V^\star(s') - v(s')] + \gamma \mathop{\mathbb{E}}_{s' \sim P(s, \hat{\pi}(s))} [v(s') - V^\star(s')] \\
&\le 2\gamma \|v - V^\star\|_\infty.
\end{aligned}$$

The second quantity is bounded by

$$\begin{aligned}
Q^\star(s, \hat{\pi}(s)) - Q^{\hat{\pi}}(s, \hat{\pi}(s)) &= \gamma \mathop{\mathbb{E}}_{s' \sim P(s, \hat{\pi}(s))} \left[V^\star(s') - V^{\hat{\pi}}(s') \right] \\
&\le \gamma \|V^\star - V^{\hat{\pi}}\|_\infty
\end{aligned}$$

and thus

$$\begin{aligned}
\|V^\star - V^{\hat{\pi}}\|_\infty &\le 2\gamma \|v - V^\star\|_\infty + \gamma \|V^\star - V^{\hat{\pi}}\|_\infty \\
\|V^\star - V^{\hat{\pi}}\|_\infty &\le \frac{2\gamma \|v - V^\star\|_\infty}{1 - \gamma}.
\end{aligned}$$

So in order to compensate and achieve $\|V^{\hat{\pi}} - V^\star\| \le \epsilon$, we must have

$$\|v^{(T)} - V^\star\|_\infty \le \frac{1 - \gamma}{2\gamma} \epsilon.$$

This means, using (**??**), we need to run value iteration for

$$T = O\left(\frac{1}{1 - \gamma} \log \left(\frac{\gamma}{\epsilon (1 - \gamma)^2} \right) \right)$$

iterations to achieve an ϵ-accurate estimate of the optimal value function.

Policy iteration

Can we mitigate this "greedy worsening"? What if instead of approximating the optimal value function and then acting greedily by it at the very end, we iteratively improve the policy and value function *together*? This is the idea behind **policy iteration**. In each step, we simply set the policy to act greedily with respect to its own value function.

Definition 1.3.3: Policy Iteration

```
name: pi$_iter$
  $\pi^{(0)} : \mathcal{S} \to \mathcal{A}$ arbitrary
  for $t = 0, \dots, T - 1$ do
    $V^{\pi^{(t)}} \leftarrow (I - \gamma P^{\pi^{(t)}})^{-1} r^{\pi^{(t)}}$        ▷ (Exact) Policy Evaluation (??)
    $Q^{\pi^{(t)}}(s, a) \leftarrow r(s, a) + \gamma \mathbb{E}_{s' \sim P(s, a)} [V^{\pi^{(t)}}(s')]$
```

```
        $\pi^{(t+1)}(s) \leftarrow \arg\max_a Q^{\pi^{(t)}}(s, a)$        ▷ Policy Improvement
    end for
```

Although PI appears more complex than VI, we'll use the same contraction property (**??**) to show convergence. This will give us the same runtime bound as value iteration and iterative policy evaluation for an ϵ-optimal value function (**??**), although in practice, PI often converges much faster.

Theorem 1.3.3: Policy Iteration runtime and convergence

name: pi$_iter_analysis$

We aim to show that the number of iterations required for an ϵ-accurate estimate of the optimal value function is

$$T = O\left(\frac{1}{1-\gamma} \log\left(\frac{1}{\epsilon(1-\gamma)}\right)\right).$$

This bound follows from the contraction property (**??**):

$$\|V^{\pi^{t+1}} - V^\star\|_\infty \le \gamma \|V^{\pi^t} - V^\star\|_\infty.$$

We'll prove that the iterates of PI respect the contraction property by showing that the policies improve monotonically:

$$V^{\pi^{t+1}}(s) \ge V^{\pi^t}(s).$$

Then we'll use this to show $V^{\pi^{t+1}}(s) \ge [\mathcal{J}^\star(V^{\pi^t})](s)$. Note that

$$\begin{aligned} [\mathcal{J}^\star(V^{\pi^t})](s) &= \max_a \left[r(s, a) + \gamma \mathop{\mathbb{E}}_{s' \sim P(s,a)} V^{\pi^t}(s') \right] \\ &= r(s, \pi^{t+1}(s)) + \gamma \mathop{\mathbb{E}}_{s' \sim P(s, \pi^{t+1}(s))} V^{\pi^t}(s') \end{aligned}$$

Since $[\mathcal{J}^\star(V^{\pi^t})](s) \ge V^{\pi^t}(s)$, we then have

$$\begin{aligned} V^{\pi^{t+1}}(s) - V^{\pi^t}(s) &\ge V^{\pi^{t+1}}(s) - \mathcal{J}^\star(V^{\pi^t})(s) \\ &= \gamma \mathop{\mathbb{E}}_{s' \sim P(s, \pi^{t+1}(s))} \left[V^{\pi^{t+1}}(s') - V^{\pi^t}(s') \right]. \end{aligned} \tag{1.9}$$

But note that the expression being averaged is the same as the expression on the l.h.s. with s replaced by s'. So we can apply the same inequality recursively to get

$$\begin{aligned} V^{\pi^{t+1}}(s) - V^{\pi^t}(s) &\ge \gamma \mathop{\mathbb{E}}_{s' \sim P(s, \pi^{t+1}(s))} \left[V^{\pi^{t+1}}(s') - V^{\pi^t}(s') \right] \\ &\ge \gamma^2 \mathop{\mathbb{E}}_{\substack{s' \sim P(s, \pi^{t+1}(s)) \\ s'' \sim P(s', \pi^{t+1}(s'))}} \left[V^{\pi^{t+1}}(s'') - V^{\pi^t}(s'') \right] \end{aligned}$$

$$\geq \cdots$$

which implies that $V^{\pi^{t+1}}(s) \geq V^{\pi^t}(s)$ for all s (since the r.h.s. converges to zero). We can then plug this back into (**??**) to get the desired result:

$$\begin{aligned} V^{\pi^{t+1}}(s) - \mathcal{J}^\star(V^{\pi^t})(s) &= \gamma \mathop{\mathbb{E}}_{s' \sim P(s, \pi^{t+1}(s))} \left[V^{\pi^{t+1}}(s') - V^{\pi^t}(s') \right] \\ &\geq 0 \\ V^{\pi^{t+1}}(s) &\geq [\mathcal{J}^\star(V^{\pi^t})](s) \end{aligned}$$

This means we can now apply the Bellman convergence result (**??**) to get

$$\|V^{\pi^{t+1}} - V^\star\|_\infty \leq \|\mathcal{J}^\star(V^{\pi^t}) - V^\star\|_\infty \leq \gamma \|V^{\pi^t} - V^\star\|_\infty.$$

1.4 Summary

- Markov decision processes (MDPs) are a framework for sequential decision making under uncertainty. They consist of a state space $\mathcal{S}$, an action space $\mathcal{A}$, an initial state distribution $\mu \in \Delta(\mathcal{S})$, a transition function $P(s' \mid s, a)$, and a reward function $r(s, a)$. They can be finite-horizon (ends after H timesteps) or infinite-horizon (where rewards scale by $\gamma \in (0, 1)$ at each timestep).
- Our goal is to find a policy π that maximizes expected total reward. Policies can be **deterministic** or **stochastic**, **state-dependent** or **history-dependent**, **stationary** or **time-dependent**.
- A policy induces a distribution over **trajectories**.
- We can evaluate a policy by computing its **value function** $V^\pi(s)$, which is the expected total reward starting from state s and following policy π. We can also compute the **state-action value function** $Q^\pi(s, a)$, which is the expected total reward starting from state s, taking action a, and then following policy π. In the finite-horizon setting, these also depend on the timestep h.
- The **Bellman consistency equation** is an equation that the value function must satisfy. It can be used to solve for the value functions exactly. Thinking of the r.h.s. of this equation as an operator on value functions gives the **Bellman operator**.
- In the finite-horizon setting, we can compute the optimal policy using **dynamic programming**.
- In the infinite-horizon setting, we can compute the optimal policy using **value iteration** or **policy iteration**.