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 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! Consider what the agent, state, and possible reward signals are in each example.

These are environments where the **state transitions**, the "rules" of the environment, only depend on the *most recent* state and action. We can formalize such environments using **Markov decision processes** (MDPs). Formally, we say that the state transitions satisfy the **Markov property**:

$$\mathbb{P}(s_{t+1} \mid s_0, a_0, \dots, s_t, a_t) = P(s_{t+1} \mid s_t, a_t).$$

Where $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the state transition function. We'll see that this simple assumption leads to a rich set of problems and algorithms.

MDPs are usually classified as **finite-horizon** (aka **episodic**), 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 then discuss the infinite-horizon case.

In each setting, we'll describe how to evaluate different **policies** (strategies for choosing actions) and how to compute or approximate the **optimal policy**. 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 (episodic) MDPs

The components of a finite-horizon (aka **episodic**) 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

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	$P(\text{orderly} \mid s, a)$	$P(\text{messy} \mid s, a)$	$r(s, a)$
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 $t = 6$ corresponds to Sunday.

1.2.1 Policies

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 two axes along which policies can vary:

- Policies can be **deterministic** or **stochastic**. A deterministic policy maps "situations" to actions while a stochastic policy maps "situations" to *probability distributions* over actions. (The use of "situation" is clarified below.)
- Policies can be **stationary** or **history-dependent**. A stationary policy only depends on the current state, while a history-dependent policy can depend on the entire history of states, actions, and rewards. In the episodic setting, we'll also consider **time-dependent** policies that depend on the time step t, i.e. $\pi = \{\pi_0, \dots, \pi_{H-1}\}$ where $\pi_t : \mathcal{S} \to \mathcal{A}$ or $\Delta(\mathcal{A})$ for each $t \in [H]$.

A fascinating result is that every finite-horizon MDP has an optimal policy that is time-dependent and deterministic! 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 policy

Here are some possible policies for the tidying example:

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

1.2.2 Trajectories

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

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

(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 choosing actions according to the policy and observing the state transitions and 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

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 above policies have generated this trajectory?

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

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

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

For a deterministic policy π, we have that $\pi_t(a \mid s) = \mathbb{I}[a = \pi_t(s)]$; 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

We'll use G_t to denote the cumulative reward from a given time step onwards (called the **return-to-go**):

$$G_t(\tau) := r_t + \cdots + r_{H-1}$$

(The dependence on τ will be omitted for brevity.) The main goal of RL is to find an **optimal policy** $\pi^\star$ that maximizes the expected return:

$$\pi^\star = \arg\max_{\pi} \mathop{\mathbb{E}}_{\tau \sim \rho^\pi} [G_0].$$

We'll need a concise way to refer to the expected return conditional on *starting in a given state at a given time*. We call this the **value function** of π at time t and denote it by

$$V_t^\pi(s) := \mathop{\mathbb{E}}_{\tau \sim \rho^\pi} [G_t \mid s_t = s]$$

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

$$Q_t^\pi(s, a) := \mathop{\mathbb{E}}_{\tau \sim \rho^\pi} [G_t \mid s_t = s, a_t = a]$$

Remark 1.2.1: Connection between value and action-value functions

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

$$V_t^\pi(s) = \mathop{\mathbb{E}}_{a \sim \pi_t(s)} [Q_t^\pi(s, a)]$$

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

$$Q_t^\pi(s, a) = r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} [V_{t+1}^\pi(s')]$$

The one-step (Bellman) consistency equation

Note that by simply considering the return as the sum of the *current* reward and the *future* return, we can describe the value function recursively (in terms of itself):

$$V_t^\pi(s) = \mathop{\mathbb{E}}_{\substack{a \sim \pi_t(s) \\ s' \sim P(s, a)}} [r(s, a) + V_{t+1}^\pi(s')] \tag{1.1}$$

This is named the **Bellman consistency equation** after **Richard Bellman** (1920–1984), who is credited with introducing dynamic programming in 1953.

Remark 1.2.2: The Bellman consistency equation for deterministic policies

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

$$V_t^\pi(s) = r(s, \pi_t(s)) + \mathop{\mathbb{E}}_{s' \sim P(s, \pi_t(s))} [V_{t+1}^\pi(s')]$$

Remark 1.2.3: Bellman consistency for the action-value function

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

$$Q_t^\pi(s, a) = r(s, a) + \mathop{\mathbb{E}}_{\substack{s' \sim P(s, a) \\ a' \sim \pi_{t+1}(s')}} [Q_{t+1}^\pi(s', a')]$$

which can also be simplified for deterministic policies:

$$Q_t^\pi(s, a) = r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} [Q_{t+1}^\pi(s', \pi_{t+1}(s'))]$$

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":

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

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

$$V_t^\pi = \mathcal{J}^\pi(V_{t+1}^\pi)$$

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".

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 time t as a function of the value function at time $t+1$. 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 1.2.1: Dynamic programming for policy evaluation in finite-horizon MDPs

```
$V_t(s) \leftarrow 0$ 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
        $V_t(s) \leftarrow V_t(s) + \pi_t(a \mid s) P(s' \mid s, a)[r(s, a) + V_{t+1}(s')]$
    end for
end for
```

Example 1.2.4: Tidying policy evaluation

Let's evaluate the policy from 1.2.2 that tidies iff the room is messy. We'll use the Bellman consistency equation to compute the value function at each time step.

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

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

$$V_t^{\pi^\star}(s) \geq V_t^\pi(s) \quad \forall s \in \mathcal{S}, t \in [H]$$

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

We mentioned earlier that every MDP has an optimal policy that is stationary and deterministic. In particular, we can construct such a policy by acting *greedily* with respect to the optimal action-value function:

$$\pi_t^\star(s) = \arg\max_a Q_t^\star(s, a)$$

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

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

$$\hat{\pi}_t(s) := \arg\max_a Q_t^\star(s, a).$$

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 $t \in [H]$.

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

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

$$[\mathcal{J}^{\hat{\pi}}(V_{t+1}^\star)](s) \ge V_t^\star(s).$$

Proof:

$$\begin{aligned}
V_t^\star(s) &= \max_{\pi \in \Pi} V_t^\pi(s) \\
&= \max_{\pi \in \Pi} \mathop{\mathbb{E}}_{a \sim \pi(\dots)} \left[r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s,a)} V_{t+1}^\pi(s') \right] && \text{Bellman consistency} \\
&\le \max_{\pi \in \Pi} \mathop{\mathbb{E}}_{a \sim \pi(\dots)} \left[r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s,a)} V_{t+1}^\star(s') \right] && \text{definition of } V^\star \\
&= \max_a \left[r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s,a)} V_{t+1}^\star(s') \right] && \text{only depends on } \pi \text{ via } a \\
&= [\mathcal{J}^{\hat{\pi}}(V_{t+1}^\star)](s).
\end{aligned}$$

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

$$V_t^\star(s) \le V_t^{\hat{\pi}}(s)$$

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.)

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

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 and using **dynamic programming** (DP).

Definition 1.2.2: DP for optimal policy

We can solve for the optimal policy in an episodic 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.

$$V^\star_{H-1} = \max_a Q^\star_{H-1}(s, a)$$

- *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

$$\begin{aligned} Q^\star_t(s, a) &= r(s, a) + \mathop{\mathbb{E}}_{s' \sim P(s, a)} [V^\star_{t+1}(s')] \\ \pi^\star_t(s) &= \arg\max_a Q^\star_t(s, a) \\ V^\star_t(s) &= \max_a Q^\star_t(s, a) \end{aligned}$$

Analysis

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 $O(H|\mathcal{S}|^2|\mathcal{A}|)$.

Example 1.2.5: Optimal policy for the tidying MDP

Left as an exercise.

1.3 Infinite horizon MDPs

What happens if a trajectory is allowed to continue forever? This is the setting of **infinite horizon** MDPs. We'll need to make a few adjustments to make the problem tractable.

First of all, note that using the total cumulative reward 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:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$$

This also has a nice real-world interpretation: it's better to get a reward now than later, since you can invest it to get more reward in the future.

The other components of the MDP remain the same:

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

It also becomes impossible to store any quantities that scale with H, e.g. the time-dependent policies and value functions from the episodic case. We'll shift to time-independent policies $\pi : \mathcal{S} \to \mathcal{A}$ (deterministic) or $\Delta(\mathcal{A})$ (stochastic).

Exercise: Which of the policies in 1.2.2 are time-independent?

We also consider time-independent value functions $V^\pi : \mathcal{S} \to \mathbb{R}$ and $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. We need to adjust the Bellman consistency functions accordingly to account for the discounting:

$$\begin{aligned}
V^\pi(s) &= \mathop{\mathbb{E}}_{\tau \sim \rho^\pi} [G_t \mid s_t = s] && \text{for any } t \in \mathbb{N} \\
&= \mathop{\mathbb{E}}_{\substack{a \sim \pi(s) \\ s' \sim P(s, a)}} [r(s, a) + \gamma V^\pi(s')] \\
Q^\pi(s, a) &= \mathop{\mathbb{E}}_{\tau \sim \rho^\pi} [G_t \mid s_t = s, a_t = a] && \text{for any } t \in \mathbb{N} \\
&= r(s, a) + \gamma \mathop{\mathbb{E}}_{\substack{s' \sim P(s, a) \\ a' \sim \pi(s')}} [Q^\pi(s', a')]
\end{aligned}$$

Exercise: Heuristically speaking, why doesn't it matter which time step we condition on when defining the value functions?

In this chapter, we'll show that the Bellman operator 1.2.3 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 The Bellman operator is a contraction mapping

Recall from 1.2.3 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. A useful fact known as the **Banach fixed-point theorem** tells us that this procedure converges to the true value function! (We won't prove this here, though it only takes some algebraic manipulation.)

Let's make this more rigorous. How can we measure the distance between two value functions? We'll take the **supremum norm** as our distance metric:

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

We aim to show that

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

Proof:

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

Then the Banach fixed-point theorem tells us that repeatedly applying the Bellman operator will converge to the true value function exponentially:

$$\|(\mathcal{J}^\pi)^{(t)}(v) - V^\pi\|_\infty \le \gamma^t \|v - V^\pi\|_\infty \tag{1.2}$$

where $(J^\pi)^{(t)}(v)$ denotes applying J^π to v t times. We'll use this useful fact to prove the convergence of several algorithms later on.

1.3.2 Tabular case (linear algebraic notation)

When there's not many states and actions, we call the MDP **tabular** since we can express the relevant quantities as low-dimensional vectors and matrices:

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

(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

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 iff 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.

1.3.3 Policy evaluation

The backwards DP technique we used in the episodic 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:

$$V^\pi = (I - \gamma P^\pi)^{-1} r^\pi. \tag{1.3}$$

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

Left as an exercise.

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 1.2, 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

$$T = O\left(\frac{\log(1/(\epsilon(1-\gamma)))}{1-\gamma}\right). \tag{1.4}$$

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

1.3.4 Optimal policies

Now let's move on to solving for the optimal policy. Once again, we can't use the backwards DP approach from the episodic 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:

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

To see this, recall we showed that the greedy policy w.r.t. the optimal value function is optimal; substituting this policy into the standard Bellman consistency equation gives the above expression.

Exercise: Verify this.

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 1.3.1: Value iteration pseudocode

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

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 1.3.1: Greedy policy value degradation

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 one 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 1.4, we need to run the algorithm for

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

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 w.r.t. it, we iteratively improve the policy and value function together? This is the idea behind **policy iteration**.

Theorem 1.3.2: Policy Iteration

Remember, for now we're only considering policies that are *stationary and deterministic*. There's $|\mathcal{S}|^{|\mathcal{A}|}$ of these, so let's start off by choosing one at random. Let's call this initial policy π^0, using the superscript to indicate the time step.

Now for $t = 0, 1, \ldots$, we perform the following:

1. *Policy Evaluation*: First use the exact policy evaluation algorithm 1.3 to calculate $V^{\pi^t}(s)$ for all states s. Then use this to calculate the state-action values:

$$Q^{\pi^t}(s, a) = r(s, a) + \gamma \sum_{s'} P(s' \mid s, a) V^{\pi^t}(s')$$

2. *Policy Improvement*: Update the policy so that, at each state, it chooses the action with the highest action-value (according to the current iterate):

$$\pi^{t+1}(s) = \arg\max_a Q^{\pi^t}(s, a)$$

In other words, we're setting it to act greedily with respect to the current Q-function.

Analysis

As in 1.2, we'd like to show

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

We'll prove convergence by showing that the policies improve monotonically:

$$V^{\pi^{t+1}}(s) \ge [\mathcal{J}^\star(V^{\pi^t})](s) \ge V^{\pi^t}(s).$$

The right-hand inequality follows from the fact that $\mathcal{J}^\star$ is a contraction mapping with a fixed point at $V^\star$.

To show the left-hand inequality, we'll first show that iterates improve monotonically, that is, $V^{\pi^{t+1}}(s) \ge V^{\pi^t}(s)$ for all 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

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

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

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

which implies that $V^{\pi^{t+1}}(s) \ge V^{\pi^t}(s)$ for all s. We can then plug this back into 1.5 to get the desired result.

This means we can now apply the Bellman convergence result 1.2 to get

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