Lecture Notes on Reinforcement Learning

Amir-massoud Farahmand

March 19, 2021

Contents

Preface

These are lectures notes for a graduate-level Introduction to Reinforcement Learning (RL) course, taught at the Department of Computer Science, University of Toronto, in Spring 2021. The course is introductory in the sense that it does not assume prior exposure to reinforcement learning. It is not, however, merely a collection of algorithms. But instead, it tries to build the mathematical intuition behind many important ideas and concepts often encountered in RL. During the course of these lectures notes, I prove many basic, or sometimes not so basic, results in RL. If the proof of some result is too complicated, I prove a simplified version of it. This course requires some level of mathematical maturity.

These lectures notes are a work in progress. New chapters will be added each week during the course. The content may change dramatically in future revisions of the work. Sometimes I have not had a chance to do a close proofread of each chapter before posting them. I add a footnote at the beginning of each chapter showing what stage of maturity the chapter is. The version 0.05 is for the first full draft, version 0.1 is after its first proofread and possible revisions, and the versions below 0.05 are for incomplete chapters (which means that I have the content ready, but I haven't typed it yet).

If you find any typos or you find some parts not very clear or you have any suggestion on how the content can be improved, please send an email to me at csc2547-2021-01@cs.toronto.edu (especially if you are a student of the course, so I won't miss your email) or farahmand@vectorinstitute.ai. I would appreciate your feedback.

Chapter 1

Introduction

Reinforcement Learning (RL) refers to both a type of problem and a set of computational methods for solving that type of problem.[1] The RL problem is the problem of how to act so that some notion of long-term performance is maximized. The RL problem, by its very definition, is about acting and interaction of an entity, which we call an agent, with its surrounding, which we call an environment. This is a very general objective. One may argue that solving the AI problem is equivalent to the RL problem.

Reinforcement learning also refers to a set of computational methods to solving the RL problem. What kind of computation does an agent need to do in order to ensure that its actions lead to good (or even optimal) long-term performance? The methods that achieve these are known as RL methods.

Historically, only a subset of all computational methods that attempt to solve the RL problem are known as the RL methods. For example, a method such as Q-Learning (which we shall study in this course) is a well-regarded RL method, but a method on evolutionary computation, such as the genetic algorithms, is not. One can argue that evolutionary computation methods do not have much of a "learning" component, or that they do not act at the timescale of an agent's lifetime, but act at the timescale of generations. While these are true distinctions, this way of demarcation is somewhat arbitrary. In this lecture notes, we focus on methods that are commonly studied within the "RL Community".

Next, we informally discuss the setup of the RL problem. Before proceeding, I would like to mention that there are very good textbooks on RL, which I encourage you to consult. A very well-known textbook is by Sutton and Barto [2019]. It

[1]Chapter's Version: 0.05 (2021 January 11).

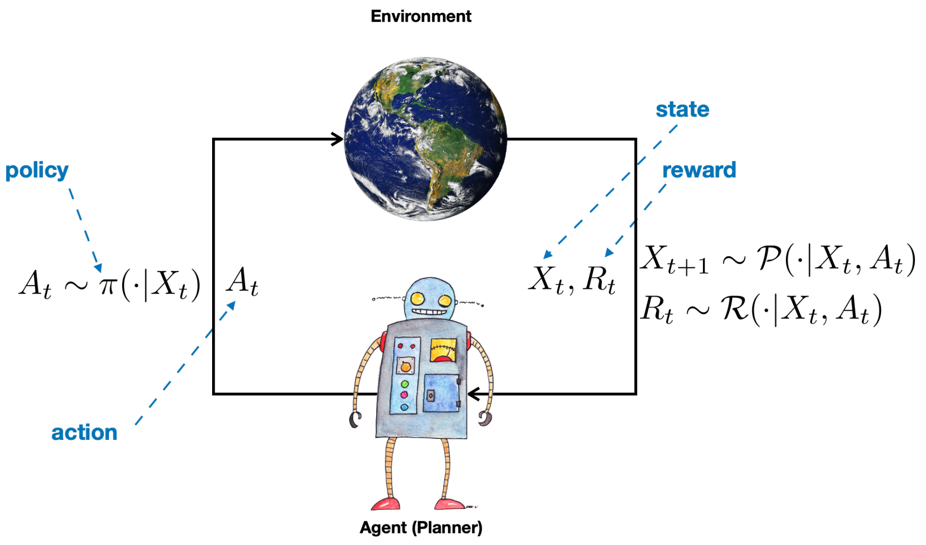
Environment
state
reward
policy
$X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$
$R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
X_t, R_t
$A_t \sim \pi(\cdot|X_t)$
A_t
action
Agent (Planner)

Figure 1.1: Reinforcement Learning Agent

provides a good intuition on many of the concepts and algorithms that we discuss in these lectures notes.

1.1 Setup

In RL, we often talk about an agent and its environment, and their interaction. Figure 1.1 depicts the schematic of how they are related. The agent is the decision maker and/or learner, and the environment is anything outside it with which the agent interacts. For example, an agent can be the decision-maker part of a robot. Or it can be the decision-maker of a medical diagnosis and treatment system. For the robot agent, the environment is whatever is outside the robot, i.e., the physical system. For the medical agent, the environment is the patient.

The interaction of the agent and its environment follows a specific protocol. The current discussion is somewhat informal, but may help you understand the concept before we formalize it. At time t, the agent observes its state X_t in the environment. For example, this is the position of the robot in the environment. Or it can be the vital information of a patient such as their temperature, blood pressure, EKG signal, etc.

The agent then picks an action A_t according to an action-selection mechanism. This mechanism is called a policy π. It usually depends on the agent's current state X_t. The policy can be *deterministic*, which means that π is a function from the state space to the action space and $A_t = \pi(X_t)$, or it can be *stochastic* (or *randomized*), which means that π defines a probability distribution over the action space that depends on the state variable, i.e., $A_t \sim \pi(\cdot|X_t)$. Here $\sim$ refers to the random variable A_t being drawn from the distribution $\pi(\cdot|X_t)$. For example, the action can be a "move forward with velocity of 1m/s" command for the robot problem, or "inject 10mg of Amoxicillin".

Based on the selected action, the state of the agent in the environment changes and becomes X_{t+1}. The state evolves according to the dynamics of the agent in the environment, which is shown by $\mathcal{P}$ in the figure. This means that $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$. The conditional distribution $\mathcal{P}$ is called *transition probability kernel* (or distribution). For the robot example, the dynamics can be described by a set of electromechanical equations that describe how the position of the robot (including its joints) change when a certain command is sent to its motor. For the medical agent, the dynamics is described by how the patient's physiology changes after the administration of the treatment. This is a very complex dynamics, which we may not have a set of equation to describe.

The agent also receives a reward signal R_t. The reward signal is a real number, and it specifies how "desirable" the choice of action A_t at the state X_t (possibly leading to state X_{t+1}) has been. Therefore, $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$ or $R_t \sim \mathcal{R}(\cdot|X_t, A_t, X_{t+1})$ (we use the former in the rest, as it does not matter). The reward is a measure of the performance of the agent at time t. For example, if our goal is for a robot to go to a specific location and pick up an object, the reward might be defined as a positive value whenever the robot achieves that goal. And it can be a zero value whenever the robot is not doing anything relevant to the goal. It maybe even be negative when it does something that ruins achieving the goal, for example breaks the object. In this case, the negative reward is actually a punishment. For the medical agent case, the reward might be defined based on the vital signs of the patient. For example, if the patient at time t had an infection, and the action was an appropriate choice of antibiotics, and at the next time $t+1$ (maybe a day later), the infection has subsided, the agent receives a positive reward, say, $+10$.

This process repeats and as a result, the agent receives a sequence of state, actions, and rewards:

$$X_1, A_1, R_1, X_2, A_2, R_2, \cdots .$$

This sequence might terminate after a fixed number of time steps (say, T), or until the agent gets to a certain region of the state space, or it might continue forever.

1.2 Markov Decision Process (MDP)

In this section, we formally define some of the important concepts that we require throughout the course. The first important concept is the Markov Decision Process (MDP). An MDP essentially defines the environment with which the agent interacts.

In the rest of this lecture notes, we denote $\mathcal{M}(\Omega)$ as the space of all probability distributions defined over the space Ω, and $\mathcal{B}(\Omega)$ as the space of all bounded functions defined over Ω. So, for example, $\mathcal{M}(\mathbb{R})$ is the space of all probability distribution over real numbers, and similar for $\mathcal{B}(\mathbb{R})$. Refer to Appendix A.1 for more formal definition. We are ready to formally define elements of MDP.

Definition 1.1. *A* discounted MDP *is a 5-tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{X}$ is a measurable state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X})$ is the transition probability kernel with domain $\mathcal{X} \times \mathcal{A}$, $\mathcal{R} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathbb{R})$ is the immediate reward distribution, and $0 \leq \gamma < 1$ is the discount factor.*

MDPs encode the temporal evolution of a discrete-time stochastic process controlled by an *agent*. The dynamical system starts at time $t = 1$ with random initial state $X_1 \sim \rho$ where " $\sim$ " denotes that X_1 is drawn from the initial state distribution $\rho \in \mathcal{M}(\mathcal{X})$.[2] At time t, action $A_t \in \mathcal{A}$ is selected by the agent controlling the process. As a result, the agent goes to the next state X_{t+1}, which is drawn from $\mathcal{P}(\cdot|X_t, A_t)$ The agent also receives an immediate reward drawn from $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$. Note that in general X_{t+1} and R_t are random, unless the dynamics is deterministic. This procedure continues and leads to a random *trajectory* $\xi = (X_1, A_1, R_1, X_2, A_2, R_2, \cdots)$. We denote the space of all possible trajectories as Ξ.

This definition of MDP is quite general. If $\mathcal{X}$ is a finite state space, the result is called a *finite MDP*. The state space $\mathcal{X}$ can be more general. If we consider a measurable subset of $\mathbb{R}^d$ ($\mathcal{X} \subseteq \mathbb{R}^d$), such as $(0, 1)^d$, we get the so-called continuous state-space MDPs. We can talk about other state spaces too, e.g., the binary lattices $\{0, 1\}^d$, the space of graphs, the space of strings, the space of distributions, etc. In this course, we switch back and forth between finite MDPs and continuous MDPs.

Example 1.1. *When $\mathcal{X}$ is finite (i.e., $\mathcal{X} = \{x_1, x_2, \ldots, x_m\}$), the transition probability kernel $\mathcal{P}(\cdot|\cdot, a)$ is a matrix for any $a \in \mathcal{A}$.*

[2]The initial distribution μ is not a part of the definition of MDPs. When we talk about MDPs as the descriptor of temporal evolution of dynamical systems, we usually implicitly or explicitly define the initial state distribution.

Example 1.2 (Deterministic Dynamics)**.** *We can represent deterministic dynamics such as* (1.18) *within the framework. If $x_{t+1} = f(x_t, a_t)$ for $f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$ (we use x instead of z to gradually make it similar to the notation that we use here), then the transition probability kernel conditioned on a pair of (x, a) puts a probability mass of* 1 *at $f(x, a)$. Using Dirac's delta function's notation,*

$$\mathcal{P}(x'|x, a) = \delta(x' - f(x, a)).$$

Remark 1.1. *We use 'x' to denote the state and 'a' to denote the action in these lectures. This is similar to how Szepesvári [2010] use it too. These are not the only notation used in the literature, and definitely not the most commonly used one. Sutton and Barto [2019] use 's' for the state and 'a' for the action. The authors from the control theory background tend to use 'u' for the action, and 'i' [Bertsekas and Tsitsiklis, 1996] or 'x' for the state [Bertsekas, 2018].*

The reason I use 'x' for state is partly historical and partly because of the following justification: I find it more aligned with how the rest of ML, and even applied math, use x as the input to a function. The fact that the input is an agent's state does not mean that we have to use a different notation. I find it slightly more appealing to see $f(x)$ instead of $f(s)$, though nothing is inherently wrong with the latter usage. The reason I stick to 'a' for the action, instead of 'u' commonly used in control theory, does not have much of a justification other than showing the CS/AI roots of RL.

Let us tend to the policy π. Recall from Section 1.1 that the policy is the action-selection mechanism of the agent. The goal of the RL agent is to find a "good" policy, to be defined what it exactly means. Let us formally define it.

Definition 1.2 (Definition 8.2 and 9.2 of Bertsekas and Shreve [1978])**.** *A* ***policy*** *is a sequence $\bar{\pi} = (\pi_1, \pi_2, \dots)$ such that for each t,*

$$\pi_t(a_t|X_1, A_1, X_2, A_2, \dots, X_{t-1}, A_{t-1}, X_t)$$

is a universally measurable stochastic kernel on $\mathcal{A}$ given $\underbrace{\mathcal{X} \times \mathcal{A} \times \cdots \times \mathcal{X} \times \mathcal{A} \times \mathcal{X}}_{2t-1 \text{ elements}}$ satisfying

$$\pi_t(\mathcal{A}|X_1, A_1, X_2, A_2, \dots, X_{t-1}, A_{t-1}, X_t) = 1$$

for every $(X_1, A_1, X_2, A_2, \dots, X_{t-1}, A_{t-1}, X_t)$.

If π_t is parametrized only by X_t, that is

$$\pi_t(\cdot|X_1, A_1, X_2, A_2, \dots, X_{t-1}, A_{t-1}, X_t) = \pi_t(\cdot|X_t),$$

$\bar{\pi}$ is a Markov *policy.*

If for each t and $(X_1, A_1, X_2, A_2, \ldots, X_{t-1}, A_{t-1}, X_t)$, the policy π_t assigns mass one to a single point in $\mathcal{A}$, $\bar{\pi}$ is called a deterministic (nonrandomized) *policy; if it assigns a distribution over $\mathcal{A}$, it is called* stochastic *or* randomized *policy.*

If $\bar{\pi}$ is a Markov policy in the form of $\bar{\pi} = (\pi, \pi, \ldots)$, it is called a stationary *policy.*

This definition essentially categorizes whether the policy is time-dependent or not, and whether it uses only the current state X_t or looks at the previous state and action pairs too. Recall our earlier discussion on the definition of state, and that the observation O might be different from the state of the agent. That is one of the cases when we need to use a non-Markov policy. Since we assume that we have access to the state X_t, we do not need to use non-Markov policies. In most of these lectures, we only focus on stationary and Markov policies, and simply use "policy" to refer to a stationary Markov policy $\pi(\cdot|x)$.

If a policy is stochastic, it would be a conditional distribution over the action space depending on the state, i.e., $\pi(\cdot|x) \in \mathcal{M}(\mathcal{A})$. If a policy is deterministic, it would be a function from the state space $\mathcal{X}$ to the action space $\mathcal{A}$, and we use $\pi(x)$ to refer to it.

We define the following terminology and notations in order to simplify our exposition.

Definition 1.3. *We say that an agent is "following" a Markov stationary policy π whenever A_t is selected according to the policy $\pi(\cdot|X_t)$, i.e., $A_t = \pi(X_t)$ (deterministic) or $A_t \sim \pi(\cdot|X_t)$ (stochastic). The policy π induces two transition probability kernels $\mathcal{P}^\pi : \mathcal{X} \to \mathcal{M}(\mathcal{X})$ and $\mathcal{P}^\pi : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X} \times \mathcal{A})$. For a measurable subset A of $\mathcal{X}$ and a measurable subset B of $\mathcal{X} \times \mathcal{A}$ and a deterministic policy π, denote*

$$\begin{aligned}(\mathcal{P}^\pi)(A|x) &\triangleq \int_{\mathcal{X}} \mathcal{P}(\mathrm{d}y|x, \pi(x)) \mathbb{I}_{\{y \in A\}},\\ (\mathcal{P}^\pi)(B|x, a) &\triangleq \int_{\mathcal{X}} \mathcal{P}(\mathrm{d}y|x, a) \mathbb{I}_{\{(y, \pi(y)) \in B\}}.\end{aligned}$$

If π is stochastic, we have

$$\begin{aligned}(\mathcal{P}^\pi)(A|x) &\triangleq \int_{\mathcal{X}} P(\mathrm{d}y|x, a) \pi(\mathrm{d}a|x) \mathbb{I}_{\{y \in A\}},\\ (\mathcal{P}^\pi)(B|x, a) &\triangleq \int_{\mathcal{X}} P(\mathrm{d}y|x, a) \pi(\mathrm{d}a'|y) \mathbb{I}_{\{(y, a') \in B\}}.\end{aligned}$$

The m-step transition probability kernels $(\mathcal{P}^\pi)^m : \mathcal{X} \to \mathcal{M}(\mathcal{X})$ and $(\mathcal{P}^\pi)^m : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X} \times \mathcal{A})$ for $m = 2, 3, \cdots$ for a deterministic policy π are inductively defined as[3]

$$
\begin{aligned}
(\mathcal{P}^\pi)^m(A|x) &\triangleq \int_{\mathcal{X}} \mathcal{P}(\mathrm{d}y|x, \pi(x))(\mathcal{P}^\pi)^{m-1}(A|y), \\
(\mathcal{P}^\pi)^m(B|x,a) &\triangleq \int_{\mathcal{X}} \mathcal{P}(\mathrm{d}y|x, a)(\mathcal{P}^\pi)^{m-1}(B|y, \pi(y)).
\end{aligned}
$$

The difference between the transition probability kernels $\mathcal{P}^\pi : \mathcal{X} \to \mathcal{M}(\mathcal{X})$ and $\mathcal{P}^\pi : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X} \times \mathcal{A})$ is in the way the policy affects the action selection: in the former, the action of the first step is chosen according to the policy, while in the latter the first action is pre-chosen and the policy chooses the action in the second step.

1.3 From Immediate to Long-Term Reward

Recall that the RL problem is the problem of how to act so that some notion of long-term performance is maximized. In this section, we would like to elaborate on the meaning of "long-term". Along the way, we learn about important concepts such as return and value functions. It turns out that we can define long-term in different ways. We discuss some of them here. Before that, however, let us start with a simpler problem of maximizing the immediate (or short-term) performance first.

Suppose that an agent starts at state $X_1 \sim \rho \in \mathcal{M}(\mathcal{X})$, chooses action $A_1 = \pi(X_1)$ (deterministic policy), and receives a reward of $R_1 \sim \mathcal{R}(\cdot|X_1, A_1)$. This ends one round of interaction of the agent and its environment. The agent then restarts, samples another (independent) $X_1 \sim \rho \in \mathcal{M}(\mathcal{X})$, and repeats as before again and again. We call each of these rounds an *episode*. Here the episode only lasts one time-step.

How should this agent chooses its policy in order to maximize its performance? To answer this question, we need to specify what performance actually refers too. There are several sensible ways to define it one of which is to talk about the *average (expected) reward* that the agent receives within one episode. This meaning of average here is that if the agent repeats this interaction with the environment for many episodes (infinitely), how much reward it receives in average. So the averaging is over the episodes.

If we define the performance in this way, answering the question of how the agent should act to maximize this notion of performance is easy. Let us define *expected*

[3]The definition for the stochastic policy would be similar.

reward as

$$r(x,a) \triangleq \mathbb{E}\left[R|X=x, A=a\right]. \tag{1.1}$$

Here the random variables (r.v.) R is distributed according to $\mathcal{R}(\cdot|x,a)$.

In order to maximize the expected reward, the best action depends on the state the agent initially starts with. At state x, it should choose

$$a^* \leftarrow \operatorname*{argmax}_{a\in\mathcal{A}} r(x,a).$$

This is the best, or *optimal*, action at state x.[4] By the definition of argmax, no choice of action can gather more rewards in expectation. With this choice, we can define the optimal policy $\pi^* : \mathcal{X} \to \mathcal{A}$ as the function that at each state x returns

$$\pi^*(x) \leftarrow \operatorname*{argmax}_{a\in\mathcal{A}} r(x,a). \tag{1.2}$$

Note that the optimal policy depends on the agent's initial state. It does not depend on initial distribution ρ.

Exercise 1.1. *Describe a similar setup where the optimal policy depends on ρ. The performance measure should still be the expected reward that the agent receives. But feel free to change some crucial aspect of the agent.*

Exercise 1.2. *Explain how a standard supervised learning problem can be formulated as finding the policy that maximizes the immediate expected reward. To be concrete, focus on the binary classification problem. What is the state x? What is the action a? And what is the reward $r(x,a)$?*

Exercise 1.3. *We equate the performance as maximizing the expected reward. What other sensible performance measures can you think of? It should still be related to the rewards that the agent receives in its episode.*

Let us consider some setups where the agent interacts with the environment for multiple steps.

[4] If there are more than one action that attains $\max_{a\in\mathcal{A}} r(x,a)$, the agent can choose any of them.

1.3.0.1 Finite Horizon Tasks

Here the idea is that agent interacts with the environment for a fixed $T \geq 1$ number of steps. When $T = 1$, this is the same as what we already discussed, but when $T > 1$, it is different.

To be more concrete, within each episode, the interaction of the agent following a policy π goes like this:

- The agent starts at $X_1 \sim \rho \in \mathcal{M}(\mathcal{X})$.
- It chooses action $A_1 = \pi(X_1)$ (or $A_1 \sim \pi(\cdot|X_1)$ for a stochastic policy)
- The agent goes to the next-state $X_2 \sim \mathcal{P}(\cdot|X_1, A_1)$ and receives reward $R_1 \sim \mathcal{R}(\cdot|X_1, A_1)$.
- The agent chooses $A_2 = \pi(X_2)$ (or $A_2 \sim \pi(\cdot|X_2)$ for a stochastic policy).
- The agent goes to the next-state $X_3 \sim \mathcal{P}(\cdot|X_2, A_2)$ and receives reward $R_2 \sim \mathcal{R}(\cdot|X_2, A_2)$.
- This process repeats for several steps until ...
- $X_T \sim \mathcal{P}(\cdot|X_{T-1}, A_{T-1})$.
- $R_T \sim \mathcal{R}(\cdot|X_{T-1}, A_{T-1})$.

Afterward, the agent starts a new episode.[5]

How should we evaluate the performance of the agent as a function of the reward sequence $(R_1, R_2, \ldots, R_T)$? A common choice is to compute the sum of rewards:

$$G^{\pi} \triangleq R_1 + \ldots + R_T. \tag{1.3}$$

The r.v. G^{π} is called the return of following policy π. As it is random, its value in each new episodes would be different (unless the dynamics and policy are deterministic, and ρ always selects the same initial state; or other similar cases).

Another choice is to consider *discounted* sum of rewards. Given a discount factor $0 \leq \gamma \leq 1$, we define the return as

$$G^{\pi} \triangleq R_1 + \gamma R_2 + \ldots + \gamma^{T-1} R_T. \tag{1.4}$$

[5] We could generalize the interaction by allowing $\mathcal{P}$ and $\mathcal{R}$ to be time-dependent. In that case, $X_{t+1} \sim \mathcal{P}_t(\cdot|X_t, A_t)$ and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$.

Whenever $\gamma < 1$, the reward that is received earlier contributes more to the return. Intuitively, this means that such a definition of return values earlier rewards more, e.g., you prefer to get a cookie today instead of tomorrow, and you prefer a cookie tomorrow to a cookie a week later (assuming that you like cookies). How much exactly your preference changes depends on the value of γ. The discount factor has a financial interpretation too and is related to the inflation rate.[6] The inflation is the rise over time in the average price (usually over a large part of the market, for example, the consumer goods and services). If the price of a certain set of goods has changed from \$1 to $\$(1 + \text{rate}_{\text{inflation}})$ next year, the inflation is $\text{rate}_{\text{inflation}}$ per year. This means that whenever rate > 0, the value of a dollar this year is more than a value of dollar next year. So if you have a choice in receiving a dollar this year or some amount of dollar next year, you need to consider the inflation rate, and discount the value of dollar next year by $\gamma = \frac{1}{1+\text{rate}_{\text{inflation}}}$. Of course, this is all based on the assumption that you do not have an immediate need for that dollar, so you can potentially postpone the time you receive it.

The return (1.4) (and (1.3) as a special case) is a random variable. To define a performance measure that is not random, we compute its expectation. We define

$$V^{\pi}(x) \triangleq \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} R_t | X_1 = x\right]. \tag{1.5}$$

This is the expected value of return if the agent starts at state x and follows policy π. The function $V^{\pi} : \mathcal{X} \to \mathbb{R}$ is called the *value* function of π.

More generally, we can define the return from time $\tau = 1, \dots, T$ until the end of episode (which is time T) as

$$G^{\pi}_{\tau} \triangleq \sum_{t=\tau}^{T} \gamma^{k-1} R_t. \tag{1.6}$$

And likewise, define the value function at time τ to be

$$V^{\pi}_{\tau}(x) \triangleq \mathbb{E}\left[G^{\pi}_{\tau} | X_{\tau} = x\right]. \tag{1.7}$$

Clearly, V^{π}_1 is the same as V^{π} in (1.5).

Comparing the expected return (1.1) and the value function is instructive. We first focus on $T = 1$. We get that

$$V^{\pi}(x) = \mathbb{E}\left[R_1 | X = x\right].$$

[6] I do not have any background in finance, so take my interpretation with a grain of salt.

This is similar to $r(x,a) = \mathbb{E}[R|X = x, A = a]$ with the difference that $r(x,a)$ is conditioned on both x and a, whereas V^π is conditioned on x. The choice of a in V^π is governed by the policy π, and is $a = \pi(x)$ (deterministic) or $A \sim \pi(\cdot|x)$ (stochastic). If we define

$$r^\pi(x) \triangleq \mathbb{E}[R|X = x] \tag{1.8}$$

with $A \sim \pi(\cdot|x)$, we get that $r^\pi = V^\pi$. Of course, this equality is only true for $T = 1$. For $T > 1$, V^π captures the long-term (discounted) average of the rewards, instead of the expected immediate reward captures by r^π.

For $T = 1$, finding the optimal policy given $r(x,a)$ is easy because we can simply find the action that maximizes it (1.2).[7] Finding the optimal policy given V^π may seem to be less straightforward. We need to search over the space of all deterministic or stochastic policies. For example, if we denote the space of all stochastic policies by

$$\Pi = \{\, \pi \,:\, \pi(\cdot|x) \in \mathcal{M}(\mathcal{A}), \forall x \in \mathcal{X} \,\} \tag{1.9}$$

we need to find

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} V^\pi.$$

It turns out that this problem is not too difficult when $T = 1$. As the values of V^π at two different states $x_1, x_2 \in \mathcal{X}$ do not have any interaction with each other, we find the optimal policy at each state separately. Note that for each $x \in \mathcal{X}$,

$$V^\pi(x) = \int \mathcal{R}(\mathrm{d}r|x,a)\pi(\mathrm{d}a|x) = \int \pi(\mathrm{d}a|x) r(x,a).$$

Find a $\pi(\cdot|x)$ that maximizes $V^\pi(x)$ means that

$$\sup_{\pi(\cdot|x) \in \mathcal{M}(\mathcal{A})} \int \pi(\mathrm{d}a|x) r(x,a). \tag{1.10}$$

The maximizing distribution can concentrate all its mass at the action a^* that maximizes $r(x,a)$ (assuming it exists). Therefore, $\pi^*(a|x) = \delta(a - \operatorname{argmax}_{a' \in \mathcal{A}} r(x,a'))$ (or equivalently, $\pi^*(x) = \operatorname{argmax}_{a \in \mathcal{A}} r(x,a)$) is an optimal policy at state x.

When $T > 1$, this argument does not hold anymore and finding the optimal policy is more difficult. The reason is that the choice of action at each time step affects the future states, so we have to be careful in choosing the policy. We spend a great deal of time on algorithms to solving this problem (though not for the finite horizon problems, but for another type that we shall introduce soon).

[7] Assuming that finding the maximizer is easy. For a finite (and small) action space, it is. But for a general action spaces, it is not.

1.3.0.2 Episodic Tasks

In some scenarios, there is a final time T that the episode ends (or terminates), but it is not fixed a priori. For example, think of playing of a board game such as chess (it ends whenever one side checkmates the other or they reach a draw), moving through a maze (it ends whenever the agent reaches a goal state), or a robot successfully picks up an object and places it in another location. For these problems, the episode ends (or terminates) whenever the agent reaches a certain state x_{terminal} within the state space, that is, it terminates whenever $X_T = x_{\text{terminal}}$.[8] For these problems, called episodic problems, the length of the episode T is a random variable. We can define the return as before: For $0 \leq \gamma \leq 1$, we have

$$G^\pi \triangleq \sum_{k=1}^{T} \gamma^{k-1} R_k, \tag{1.11}$$

and

$$V^\pi(x) \triangleq \mathbb{E}\left[G^\pi | X_1 = x\right]. \tag{1.12}$$

If $\gamma < 1$, these definitions are always well-defined. If $\gamma = 1$, we need to ensure that the termination time T is finite. Otherwise, the summation might be divergent (just imagine that all R_t are equal to 1). We do not get into analysis of episodic problem with $\gamma = 1$, so we do not get into more detail here anymore. Refer to Section 2.2 (Stochastic Shortest Path Problems) by Bertsekas and Tsitsiklis [1996].

1.3.0.3 Continuing Tasks

Sometimes the interaction between the agent and its environment does not break into episodes that terminates. It goes on continually forever. For example, this might be the case for a life-long robot or a chemical plant that is supposed to work for a long time. Of course, nothing in practice lasts forever, so the mathematical framework on continuing tasks is an abstract idealization of tasks that may take a long time.

Consider the sequence of rewards $(R_1, R_2, \dots)$ generated after the agent starts at state $X_1 = x$ and follows policy π. Given the discount factor $0 \leq \gamma < 1$, the return is

$$G_t^\pi \triangleq \sum_{k \geq t} \gamma^{k-t} R_k. \tag{1.13}$$

[8]It might be more intuitive to think about the terminal states $\mathcal{X}_{\text{terminal}}$, instead of a singular one. Mathematically, it does not matter.

We can also define the value function, as before. Since this is the value function that we will use for the rest of the lectures, we define it formally.

Definition 1.4 (Value Functions)**.** *The (state-)value function V^π and the action-value function Q^π for a policy π are defined as follows: Let $(R_t; t \geq 1)$ be the sequence of rewards when the process is started from a state X_1 (or (X_1, A_1) for the action-value function) drawn from a positive probability distribution over $\mathcal{X}$ (or $\mathcal{X} \times \mathcal{A}$) and follows the policy π for $t \geq 1$ (or $t \geq 2$ for the action-value function). Then,*

$$\begin{aligned} V^\pi(x) &\triangleq \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | X_1 = x\right], \\ Q^\pi(x,a) &\triangleq \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | X_1 = x, A_1 = a\right]. \end{aligned}$$

In words, the value function V^π evaluated at state x is the expected discounted return of following the policy π from state x. The action-value Q^π function evaluated at (x,a) is the expected discounted return when the agent starts at state x, takes action a, and then follows policy π.

The action-value function Q^π and value function V^π are closely related. The difference is that the first action A_1 in V^π is selected according to $\pi(\cdot|X_1)$, but the first action in $Q^\pi(x,a)$ is the pre-specified action a. So

$$V^\pi(x) = \mathbb{E}\left[Q^\pi(x,A)\right] = \int \pi(\mathrm{d}a|x) Q^\pi(x,a). \tag{1.14}$$

If $\gamma = 0$, $Q^\pi = \mathbb{E}\left[R_1 | X_1 = x, A_1 = a\right]$. This is the same as the expected immediate reward $r(x,a)$. The same way that we could easily compute the optimal action using $r(x,a)$ in the finite-horizon problem with $T = 1$, we shall see that we can use Q^π (in continual task) in order to easily compute the optimal policy.

Note that an episodic task can be seen as a continuing task with a special state x_{terminal} from which the agent cannot escape and it always gets a reward of zero, i.e.,

$$\begin{aligned} &\mathcal{P}(x_{\text{terminal}} | x_{\text{terminal}}, a) = 1, \qquad \forall a \in \mathcal{A} \\ &\mathcal{R}(r | x_{\text{terminal}}, a) = \delta(r), \qquad \forall a \in \mathcal{A}. \end{aligned}$$

1.4 Optimal Policy and Optimal Value Function

What does it mean for an agent to act optimally? To start thinking about this question, let us first think about how we can compare two policies π and π'. For

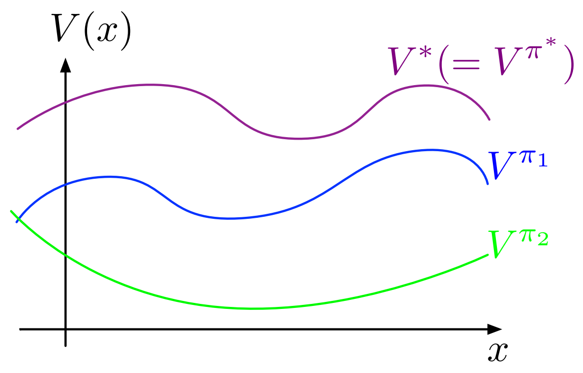

Figure 1.2: For any policy π, we have that $V^{\pi^*} \geq V^{\pi}$. Here the values V^{π_1} and V^{π_2} of two sub-optimal policies π_1 and π_2 are shown.

the moment, we can assume that they are Markov stationary policies, so the action selection is based on $A_t \sim \pi(\cdot|X_t)$ (and not $A_t \sim \pi(\cdot|X_t, X_{t-1}, X_{t-2}, \dots)$ or $A_t \sim \pi_t(\cdot|X_t)$). We say that π is better than or equal to π' (i.e., $\pi \geq \pi'$) iff $V^{\pi}(x) \geq V^{\pi'}(x)$ for all states $x \in \mathcal{X}$. This is shown in Figure 1.2.

If we can find a policy π^* that satisfies $\pi^* \geq \pi$ for any π, we call it an *optimal policy*. There may be more than one optimal policy. Despite that, their values should be the same, i.e., if we have two different π_1^* and π_2^*, we have $V^{\pi_1^*}(x) \geq V^{\pi_2^*}(x)$ and $V^{\pi_1^*}(x) \leq V^{\pi_2^*}(x)$ for all $x \in \mathcal{X}$, which entails that $V^{\pi_1^*} = V^{\pi_2^*}$. If we denote Π as the space of all stationary Markov polices, this means that

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} V^{\pi}, \tag{1.15}$$

where one of the maximizers is selected in an arbitrary way. The value function of this policy is the called the *optimal value function*, and is denoted by V^{π^*} or simply V^*. We can also define the optimal policy based on Q^{π}, i.e.,

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} Q^{\pi}. \tag{1.16}$$

The optimal action-value function is denoted by Q^{π^*} or Q^*.

For the immediate reward maximization problem (or equivalently, when $T = 1$ for a finite horizon problem), the solution was easy to find, see (1.2) and (1.10). It is not obvious, however, that such a policy exists for the continuing discounted tasks. It might be the case that no single policy can dominate all others for all states. For example, it is imaginable that at best we can only hope to find a π^* that is better than any other policy π only on a proper subset of $\mathcal{X}$, perhaps depending on π, but not everywhere.

It is also not obvious why we should focus on stationary policies. Isn't it possible to have a policy $\bar{\pi} = \{\pi_1, \pi_2, \dots\}$ that depends on the time step and acts better than any stationary policy $\bar{\pi} = \{\pi, \pi, \dots\}$?

If we find satisfactory answers to these questions, a more pragmatic question remains: How we can compute π^* if we know the MDP (which means that we know $\mathcal{P}$ and $\mathcal{R}$)?

And even more interesting is the question of how we can *learn* π^*, or a close approximation thereof, without actually knowing the MDP, but only have samples coming from interacting with the MDP.

We study the question about the existence and properties of the optimal policy in Chapter 2. The short answer is that for continual discounted problems, the optimal policy is indeed a stationary Markov policy. Moreover, we can always find a deterministic optimal policy too.

Chapter 3 introduces several methods for computing the optimal policy given a known model $\mathcal{P}$ and $\mathcal{R}$. We study some of their properties, and prove their convergence to the optimal policy. We call the setting when the model is known as the *planning* setting.

When we do not know $\mathcal{P}$ or $\mathcal{R}$, we are in the *reinforcement learning* setting. In that setting, we do not have a direct access to the model, but instead we can only interact with the MDP by selecting action A_t at state X_t, and getting a reward $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$ and going to the next state X_{t+1} according to the transition probability kernel. It turns out that many of the methods for planning can be modified to become a learning algorithm. Therefore, it is good to get a good grasp of planning methods first instead of delving into RL from the beginning. We introduce and analyze some methods for solving RL problems in Chapter 4. The focus of that chapter is on the RL problems with finite state and action spaces. We turn to problems with large state and action spaces (e.g., when $\mathcal{X}$ is a subset of $\mathbb{R}^d$) in later chapters.[9]

1.5 An Instance of an RL Algorithm: Q-Learning

It takes a while before we get into any RL algorithm, so it is good to see an example of such an algorithm before starting our excursion into the properties of MDPs (Chapter 2) and the planning methods (Chapter 3) until we finally get to RL algorithms in Chapter 4.

[9]The detail of chapter information will be determined later.

Algorithm 1.1 Q-Learning

```
Require: Step size $\alpha \in (0, 1]$
1: Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, except that for $x_{\text{terminal}}$, set $Q(x_{\text{terminal}}, \cdot) = 0$.
2: for each episode do
3:     Initialize $X_1 \sim \rho$
4:     for each step $t$ of episode do
5:         $A_t \sim \pi(\cdot|X_t)$,                                    ▷ Action selection
6:         Take action $A_t$, observe $X_{t+1}$ and $R_t$          ▷ The environment chooses
   $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$ and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$.
7:         $Q(X_t, A_t) \leftarrow Q(X_t, A_t) + \alpha\,[R_t + \gamma \max_{a' \in \mathcal{A}} Q(X_{t+1}, a') - Q(X_t, A_t)]$.    ▷
   Q-Learning Update Rule
8:     end for
9: end for
```

Q-Learning (Algorithm 1.1) is the quintessential RL algorithm, introduced by Christopher Watkins [Watkins, 1989, Chapter 7 – Primitive Learning]. Q-Learning itself is an example of the Temporal Difference (TD) learning [Sutton, 1988].

The choice of policy π in Line 1.1.5 is not specified. The Q-Learning algorithm can work with variety of choices for π. A common choice is to use the ε-greedy policy. The ε-greedy policy $\pi_\varepsilon(Q)$ for an $0 \le \varepsilon \le 1$ chooses the action as follows: Given the current estimate of the action-value function Q, it chooses the action that maximizes the action-value function at the current state X_t with probability $1 - \varepsilon$, and chooses a (possibly uniformly) random action with probability ε. Mathematically,

$$A_t = \begin{cases} \operatorname{argmax}_{a \in \mathcal{A}} Q(X_t, a) & \text{w.p. } 1 - \varepsilon \\ \text{uniform}(\mathcal{A}) & \text{w.p. } \varepsilon \end{cases} \tag{1.17}$$

Usually the value of ε is small and may go to zero as the agent learns more about its environment.

The update rule for Q-Learning is Line 1.1.7. We notice that it does not directly use the model $\mathcal{P}$ or $\mathcal{R}$, but uses the tuple (X_t, A_t, R_t, X_{t+1}) in order to update the action-value function Q.

Under certain conditions, including how the learning rate α should be selected, the Q-Learning algorithm can be guaranteed to converge to the optimal action-value function Q^*. As we shall see later, we can use Q^* to find the optimal policy π^*. We shall try to understand why this is the case in the next few chapters.

1.6 A Few Remarks on the MDP Assumption

Before finishing this chapter, a few remarks are in order. Perhaps the most crucial one is the definition of state. What is a state? Is any variable that the agent observes a state? The way we use the state here is that the state of the agent at time t is a variable that summarizes whatever has happened to the agent up to that time step. Knowing the state is enough to know (probabilistically) what will happen to the agent in the future. In other words, the state is a *sufficient statistic* of the history.

What is a state variable?

To make this more clear, let us introduce another concept called *observation*. An observation O_t is the variable that the agent actually observes using its various sensors. For example, it might be the camera input for the robot agent, or the temperature and blood pressure for the medical agent. The observation alone may not be sufficient to know "everything" that we could know about the agent given the information so far. For example, by only having an access to the current camera image, we do not know whether the robot is moving forward or backward or something is getting close to it or far from it (as the velocity information cannot be inferred from the position information alone). Or as another example, if we can only observe the blood pressure and heart rate, we cannot know everything that could be known about the agent. The information might be there, if we looked at the previous observations.

Whatever has happened to the agent up to time t is in its history H_t variable

$$H_t = (O_1, A_1, \ldots, O_{t-1}, A_{t-1}, O_t).$$

The history H_t summarizes whatever has happened to the agent up to time t. Given H_t, we can inquire about the probability distribution

$$\mathbb{P}\left\{O_{t+1} | H_t, A_t\right\}.$$

This is all we can hope to know about the future, given the information that we have. Now, if we do not look at H_t, but only look at the current observation O_t, we can still form $\mathbb{P}\{O_{t+1}|O_t, A_t\}$, but it has more "uncertainty" about the probability of O_{t+1}. We are losing information by not looking at H_t.

The variable H_t is a state of the agent at time t. But it is not a compact one, as its size gradually increases. If it happens that we can find another variable X_t, which is a function of H_t but perhaps of a compact form, that satisfies

$$\mathbb{P}\left\{O_{t+1} | H_t, A_t\right\} = \mathbb{P}\left\{O_{t+1} | X_t, A_t\right\},$$

we can replace H_t with X_t. This X_t is the state of the system in the sense described above. In the rest of these lectures, we assume that the agent has access to such a state variable.

Finding such a summary is not always complicated. Consider a dynamical system described by the following equation:

$$z_{t+1} = f(z_t, a_t), \tag{1.18}$$

where $z \in \mathbb{R}^m$, $a \in \mathbb{R}^n$, and $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$. For example, if

$$f(z, a) = Mz + Na,$$

with $M \in \mathbb{R}^{m\times m}$ and $N \in \mathbb{R}^{m\times n}$, we have a linear dynamical system. Suppose that the observation is $o_t = z_t$. In this case, we do not need to keep $h_t = (z_1, a_1, \dots, z_{t-1}, a_{t-1}, z_t)$ as a state of the system; the observation o_t alone is enough to know whatever has happened to the system up to time t. We can disregard $z_{t-1}, a_{t-1}, z_{t-2}$, etc. Now suppose that the observation is $o_t = g(z_t)$ with $g : \mathbb{R}^m \to \mathbb{R}^d$. In this case, depending on the function g, the observation o_t may or may not be a state. For example, if g is not a bijection (one-to-one correspondence), it is likely that we lose o being a state. However it may be possible that we can still process h_t and find a compact representation x_t that is a state of the agent.

Most (all?) physical systems can be written by an equation similar to (1.18).[10] If we have such a description of the dynamics, the state is often clear as long as we observe the right variable.

Exercise 1.4. *A ball is free falling under the Earth's gravity. The state is the vector described by its location $x(t)$ at its velocity $v(t) = \dot{x}(t)$. If we only observe $x(t)$, that is not enough to know the (physical) state of the ball. How can you estimate the state using only the location information?*

Exercise 1.5. *In Atari games, a single frame is not a state of the agent. Explain why.*

Where does reward come from?

The other remark is regarding the reward signal. How do we determine the reward signal R_t? The reward signal encodes the desire of the problem designer. It is a part of the problem specification. In biological systems, however, the reward signal has not been designed, but has been evolved. The reward mechanism of animals has been evolved so that the chance of survival increases. Throughout this course, we assume that the reward signal is given. We should mention that there has been work in evolving the reward signal itself.

[10] To be more accurate, almost all physical systems are written in the form of a differential equation, so we have $\frac{\mathrm{d}z}{\mathrm{d}t}(t) = f(z_t, a_t)$ instead. But this is not a crucial difference here.

Chapter 2

Structural Properties of Markov Decision Processes

In this chapter we study some important properties of the value function V^π and Q^π of a policy π, the optimal value functions V^* and Q^*, and the optimal policy π^* itself.[1] We also introduce the Bellman operators and study their properties such as monotonicity and contraction. What these mean will become clear soon. We focus on the discounted continuing tasks.

2.1 Bellman Equations

2.1.1 Bellman Equations for Value Functions of a Policy

Recall that the return G^π (1.13) is a r.v. defined as

$$G_t^\pi \triangleq \sum_{k \geq t} \gamma^{k-t} R_k.$$

Comparing G_t^π and G_{t+1}^π, we observe that

$$G_t^\pi = R_t + \gamma G_{t+1}^\pi. \tag{2.1}$$

This shows that the return at the current time is equal to the reward at time t plus the *discounted* return at the next time step. This recursive structure is very important in MDPs, and many Planning and RL algorithms benefit from it.

Let us take a closer look at the return and its recursive property.

[1]Chapter's Version: 0.05 (2021 January 26).

When the agent is at a particular state x at time t, it chooses $A_t \sim \pi(\cdot|X_t)$. The action is, in general, a random variable. The next-state $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$ and the reward $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$ are r.v. too. Continuing this process, we get a sequence of rewards, which define return G_t^π, which would be a r.v. in general.

Now suppose that the exact same agent restarts at state x and follows the same policy π. This time the draws of r.v. would be different, which means that the agent chooses a different A'_t, the next-state X'_{t+1}, reward R'_t, etc. The resulting return G'^π_t takes a different value, even though its distribution is the same.

Exercise 2.1. *Describe a situation when the return G_t^π would always be the same.*

Even though the actual value of the return G_t^π, starting from the same state x, is different in each run, they all have the same distribution. As a result, if we take its expectation, it would be the same (and would be equal to the value function V^π at state x). By computing the expected value of the return, we can reveal an important recursive property of the value function V^π. For any $x \in \mathcal{X}$, we have

$$\begin{aligned}
V^\pi(x) &= \mathbb{E}\left[G_t^\pi \mid X_t = x\right] \\
&= \mathbb{E}\left[R_t + \gamma G_{t+1}^\pi \mid X_t = x\right] \\
&= \mathbb{E}\left[R(X_t, A_t) \mid X_t = x\right] + \gamma \mathbb{E}\left[G_{t+1}^\pi \mid X_t = x\right] \\
&= r^\pi(x) + \gamma \mathbb{E}\left[V^\pi(X_{t+1}) \mid X_t = x\right],
\end{aligned}$$

where in the first equality, we simply substituted the definition of the value function (Definition 1.4); we used the recursive property of the return (2.1) in the second equality; and used the definition of the expected reward while choosing action according to π in the last one (1.8). This is similar to (2.1), except that as opposed to it, neither sides are random.

Let us pay attention to $\mathbb{E}\left[V^\pi(X_{t+1}) \mid X_t = x\right]$. This is the expected value of $V^\pi(X_{t+1})$ when the agent is at state x at time t, chooses action $A \sim \pi(\cdot|x)$. If we expand it, we get

$$\mathbb{E}\left[V^\pi(X_{t+1}) \mid X_t = x\right] = \int \mathcal{P}(\mathrm{d}x'|x, a)\pi(\mathrm{d}a|x)V^\pi(x').$$

Likewise, for countable state-action spaces, we have

$$\mathbb{E}\left[V^\pi(X_{t+1}) \mid X_t = x\right] = \sum_{x', a} \mathcal{P}(x'|x, a)\pi(a|x)V^\pi(x').$$

Therefore, we get that for any $x \in \mathcal{X}$, it holds that

$$V^\pi(x) = r^\pi(x) + \gamma \int \mathcal{P}(\mathrm{d}x'|x, a)\pi(\mathrm{d}a|x)V^\pi(x'). \tag{2.2}$$

This is known as the *Bellman equation* for a policy π. Using the notation of $\mathcal{P}^\pi$ (1.3), we can also write it as

$$V^\pi(x) = r^\pi(x) + \gamma \int \mathcal{P}^\pi(\mathrm{d}x'|x) V^\pi(x').$$

The Bellman equation gives us an interpretation of the value function V^π: The value of following a policy π starting from the state x is the reward that the agent receives at that state plus the discounted average (expected) value that the agent receives at the next-state that is generated by following policy π.

We can also write the Bellman equation more compactly:

$$V^\pi = r^\pi + \gamma \mathcal{P}^\pi V^\pi.$$

These are all known as the *Bellman equation* for a policy π. Note that it defines a linear system of equations in V^π. We will later show that the only V that satisfies this equation is V^π, i.e., if we find a V such that $V = r^\pi + \gamma \mathcal{P}^\pi V$, that V is necessarily the same as V^π, the value function of a policy π.

The action-value function Q^π also satisfies a Bellman equation:

$$\begin{aligned} Q^\pi(x,a) &= r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^\pi(x') && (2.3) \\ &= r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \pi(\mathrm{d}a'|x') Q^\pi(x',a'), && (2.4) \end{aligned}$$

or more compactly,

$$Q^\pi = r + \gamma \mathcal{P} V^\pi,$$

with the understanding that V^π and Q^π are related as $V^\pi(x) = \int \pi(\mathrm{d}a|x) Q^\pi(x,a)$, see (1.14). The difference with the Bellman equation for V^π is that the choice of action at the first time step is pre-specified, instead of being selected by policy π.

Exercise 2.2. *What is the interpretation of $Q^\pi(x,a)$ based on the Bellman equation?*

Exercise 2.3. *Write down the Bellman equation for a deterministic policy $\pi : \mathcal{X} \to \mathcal{A}$.*

Exercise 2.4. *Write down the Bellman equation for a deterministic dynamical system ($x_{t+1} = f(x_t, a_t)$ – see Example 1.2) and deterministic policy $\pi : \mathcal{X} \to \mathcal{A}$.*

2.1.2 Bellman Equations for Optimal Value Functions

Recall that the optimal policy π^* is a policy that satisfies $\pi^* \geq \pi$ for any (stationary Markov) policy π. Based on this definition, it also satisfies (cf. (1.15))

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} V^{\pi}.$$

Given an optimal policy, the optimal value function would be V^{π^*}. Here we restricted the search of the policy to the space of stationary policies. We discussed in Section 1.4 that it is imaginable that one can find a non-stationary policy that is better than this stationary optimal policy. In that case, calling such a policy an "optimal" one would be questionable. Nevertheless, we should not worry about it as one can show that for discounted continuing MDPs, we can find a stationary policy that is optimal within the space of all stationary and non-stationary policies.

Does the optimal value function V^{π^*} satisfy a recursive relation similar to the Bellman equation for a policy π? It turns out that it does. We should proceed carefully in our claims.

First, we claim that there exists a unique value function V^* that satisfies the following equation: For any $x \in \mathcal{X}$, we have

$$V^*(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^*(x') \right\}. \tag{2.5}$$

This equation is called the *Bellman optimality equation* for the value function. We prove the existence and uniqueness of V^* later in this chapter.

The second claim is that it turns out V^* is indeed the same as V^{π^*}, the optimal value function when π is restricted to be within the space of stationary policies.

The third claim is that for discounted continuing MDPs, we can always find a stationary policy that is optimal within the space of all stationary and non-stationary policies.

These three claims together show that the Bellman optimality equation (2.5) reveals the recursive structure of the optimal value function $V^* = V^{\pi^*}$. In the rest of these lectures, we often use V^* to refer to the optimal value function, unless we want to emphasize its dependence on the optimal policy, in which case we use V^{π^*}.

To define the optimal state-value function, we started from the the relation that $\pi \geq \pi'$ iff for all states $x \in \mathcal{X}$, we have $V^{\pi}(x) \geq V^{\pi'}(x)$. This relation, however, could be defined based on the action-value functions. That is, we could say that $\pi \geq \pi'$ iff for all state-actions $(x,a) \in \mathcal{X} \times \mathcal{A}$, we have $Q^{\pi}(x,a) \geq Q^{\pi'}(x,a)$. This would allow us to define

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} Q^{\pi}.$$

Let us use $V^\pi \geq V^{\pi'}$ as a short form for $V^\pi(x) \geq V^{\pi'}(x)$ for all $x \in \mathcal{X}$; similarly, we use $Q^\pi \geq Q^{\pi'}$ as a short form for $Q^\pi(x,a) \geq Q^{\pi'}(x,a)$ for all $(x,a) \in \mathcal{X}$. The following proposition shows that these relations are equivalent.

Proposition 2.1. *Given two policies π, π', $V^\pi \geq V^{\pi'} \iff Q^\pi \geq Q^{\pi'}$.*

Proof. Suppose that $V^\pi(x) \geq V^{\pi'}(x)$ for all $x \in \mathcal{X}$. We write the action-value functions Q^π and $Q^{\pi'}$ in terms of the value function V^π and $V^{\pi'}$ using (2.3). For any $(x,a) \in \mathcal{X}$, we have

$$\begin{aligned} Q^\pi(x,a) &= r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^\pi(\mathrm{d}x') \\ &\geq r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^{\pi'}(\mathrm{d}x') = Q^{\pi'}(x,a). \end{aligned}$$

So whenever $V^\pi(x) \geq V^{\pi'}(x)$, we also have $Q^\pi(x,a) \geq Q^{\pi'}(x,a)$.

To see the other direction, recall from (1.14) that $V^\pi(x) = \int \pi(\mathrm{d}a|x) Q^\pi(x,a)$. Thus, if $Q^\pi(x,a) \geq Q^{\pi'}(x,a)$ for all $(x,a) \in \mathcal{X} \times \mathcal{A}$, we also have that for any $x \in \mathcal{X}$,

$$V^\pi(x) = \int \pi(\mathrm{d}a|x) Q^\pi(x,a) \geq \int \pi(\mathrm{d}a|x) Q^{\pi'}(x,a) = V^{\pi'}(x).$$

□

Therefore, the relation $V^\pi \geq V^{\pi'}$ is the same as $Q^\pi \geq Q^{\pi'}$. The result is that we can define the optimal policy as the one that satisfies $\pi^* \leftarrow \mathrm{argmax}_{\pi \in \Pi} Q^\pi$.

We can define the *Bellman optimality equation* for the action-value functions, similar to (2.5). We have that for any $(x,a) \in \mathcal{X} \times \mathcal{A}$,

$$Q^*(x,a) = r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \max_{a' \in \mathcal{A}} Q^*(x',a'). \tag{2.6}$$

This equation is called the *Bellman optimality equation* for the value function. Similar to V^*, it can be shown that such a Q^* exists and is unique. Moreover, the relation of the solution Q^* of this equation and the action-value function Q^{π^*} of the optimal policy is the same as the relation of V^* and V^{π^*}: we have that $Q^* = Q^{\pi^*}$.

2.2 From Optimal Value Function to Optimal Policy through Greedy Policy

If we know the optimal value functions V^* or Q^*, we can compute the optimal policy π^* relatively easily. We can choose a deterministic optimal policy $\pi^* : \mathcal{X} \to \mathcal{A}$. For any $x \in \mathcal{X}$, the optimal policy is[2]

$$\begin{aligned}\pi^*(x) &= \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(x,a) \\ &= \operatorname*{argmax}_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^*(x') \right\}.\end{aligned}$$

Let us discuss this to gain a better understanding. Suppose that the agent is state x. If it wants to act optimally, it needs to act optimality both at the current time step (Now) and then afterwards in the Future time steps. If at either Now or Future, it doesn't act optimally, the decision would not be optimal and it can improve its decisions at Now or the Future to obtain a higher value. Suppose that we know that the agent is going to act optimally in the Future. This means that when it get to the next state $X' \sim \mathcal{P}(\cdot|x,a)$, given its choice of action a, it follows the optimal policy π^*. The value of following the optimal policy is going to be $V^*(X')$. Since we do not know where the agent will be at the next time step, and our performance criteria is the expected return, the performance of acting optimally in the Future, given its current choice of action a, is $\int \mathcal{P}(\mathrm{d}x'|x,a)V^*(x')$. As we are dealing with discounted tasks, the performance of the agent at the current state x is going to be $r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)V^*(x')$. To act optimally Now, the agent should choose an action that maximizes this value, which is exactly what we have above. Of course, this is the same action that maximizes the right-hand side (RHS) of (2.5).

The mapping that selects an action by choosing the maximizer of the (action-)value function is called the greedy policy. For an action-value function $Q \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$ (not necessarily the optimal one or for a particular policy), the greedy policy $\pi_g : \mathcal{X} \times \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{A}$ is defined as

$$\pi_g(x; Q) = \operatorname*{argmax}_{a \in \mathcal{A}} Q(x,a).$$

Likewise, for a value function $V \in \mathcal{B}(\mathcal{X})$, the greedy policy is[3]

$$\pi_g(x; V) = \operatorname*{argmax}_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V(x') \right\}. \tag{2.7}$$

When we do not explicitly state the dependence on x, $\pi_g(V)$ and $\pi_g(Q)$ denote functions from $\mathcal{X}$ to $\mathcal{A}$. Clearly, $\pi_g(V^*) = \pi_g(Q^*) = \pi^*$.

The intuition behind the greedy policy is that it chooses the action only based on the *local* information. It does not compute the value of all possible future actions

[2]This is the consequence of Proposition 2.5, which we shall prove.

[3]If there are more than one maximizer, any of them can be chosen.

and picks the action sequence that maximizes the expected return. Instead, it only looks one step ahead (for V) or even no-step ahead (for Q) in order to pick the action. This is a myopic action selection mechanism. Nevertheless, if we pass V^* or Q^* to the greedy policy, the selected action is going to be the optimal one. This is because the optimal value functions encodes the information about the future, so we do not need to explicitly consider all possible futures.

2.3 Bellman Operators

The Bellman equations can be seen as the fixed point equation of certain operators known as the *Bellman operators*.[4] The Bellman operators are mapping from the space of value functions (or action-value function) to the space of value functions (or action-value functions). They are formally defined as follows.

Definition 2.1 (Bellman Operators for policy π)**.** *Given a policy $\pi : \mathcal{X} \to \mathcal{M}(\mathcal{A})$, the Bellman operators $T^\pi : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$ and $T^\pi : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$ are defined as the mapping*

$$(T^\pi V)(x) \triangleq r^\pi(x) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a|x)V(x'), \qquad \forall x \in \mathcal{X}$$
$$(T^\pi Q)(x,a) \triangleq r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x')Q(x',a'), \qquad \forall (x,a) \in \mathcal{X} \times \mathcal{A}.$$

Here we are overloading the same notation to refer to two different operators. Its interpretation should be clear from the context and whether the function they are applied to is of the dimension of the value function or the action-value function.

If the policy is deterministic, the Bellman operators become

$$(T^\pi V)(x) \triangleq r^\pi(x) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,\pi(x))V(x'), \qquad \forall x \in \mathcal{X}$$
$$(T^\pi Q)(x,a) \triangleq r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)Q(x',\pi(x')), \qquad \forall (x,a) \in \mathcal{X} \times \mathcal{A}.$$

Comparing the Bellman equations (2.2) and (2.4) with the definition of the Bellman operator, we observe that the Bellman equations are in fact the fixed point equation defined based on the Bellman operators, i.e.,

$$V^\pi = T^\pi V^\pi,$$
$$Q^\pi = T^\pi Q^\pi.$$

[4]Recall that if $L : \mathcal{Z} \to \mathcal{Z}$ is an operator (or mapping) from a space $\mathcal{Z}$ to $\mathcal{Z}$, the point z satisfying $Lz = z$ is called the fixed point of L. Refer to Definition A.7 in Appendix A.3.

This is a compact form of Bellman equations.

We can define the *Bellman optimality operators* similarly.

Definition 2.2 (Bellman Optimality Operators)**.** *The Bellman operators $T^* : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$ and $T^* : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$ are defined as the mapping*

$$(T^*V)(x) \triangleq \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)V(x') \right\}, \qquad \forall x \in \mathcal{X}$$

$$(T^*Q)(x,a) \triangleq r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \max_{a' \in \mathcal{A}} Q(x',a'), \qquad \forall (x,a) \in \mathcal{X} \times \mathcal{A}.$$

Remark 2.1. *We often use $\max_{a \in \mathcal{A}}$ in the definition of the Bellman optimality operator and the Bellman optimality equation. But we could also have $\sup_{a \in \mathcal{A}}$ instead. The supremum of a function within a set is the least upper bound of the function, and may or may not belong to $\mathcal{A}$. As a simple example, if $f(a) = a$ and the domain is $(0,1)$, then $\sup_{a \in (0,1)} f(a) = 1$, and is attained at $a^* = 1$, but $a^* = 1$ does not belong to $(0,1)$, so the maximum does not exist.*

Comparing with the Bellman optimality equations (2.5) and (2.6), we see that V^* and Q^* can be written as the solution of the fixed-point equations defined based on these operators, i.e.,

$$V^* = T^*V^*,$$
$$Q^* = T^*Q^*.$$

The maximization in the definition of the Bellman optimality operator for V is over the action space $\mathcal{A}$. We can also write as the maximization over the space of stochastic or deterministic policies, and the result would be the same. Recall from (1.9) that the space of stochastic policies is $\Pi = \{ \pi : \pi(\cdot|x) \in \mathcal{M}(\mathcal{A}), \forall x \in \mathcal{X} \}$. The space of determinstic policies is defined as

$$\Pi_{\text{det}} = \{ \pi : \pi(x) \in \mathcal{A}, \forall x \in \mathcal{X} \} = \mathcal{A}^{\mathcal{X}}. \tag{2.8}$$

We have that for all $x \in \mathcal{X}$,

$$\begin{aligned}
(T^*V)(x) &= \sup_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)V(x') \right\} \\
&= \sup_{\pi \in \Pi_{\text{det}}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)V(x') \right\} \\
&= \sup_{\pi \in \Pi} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)V(x') \right\}.
\end{aligned}$$

The reason the second equality holds is that Π_{det} is the Cartesian product of $\mathcal{A}$, so the maximizing action $a^*(x)$ over each dimension $x \in \mathcal{X}$ can be combined to define a function $\pi^* = \prod_{x \in \mathcal{X}} a^*(x) \in \Pi_{\text{det}}$.

To see why the last equality holds, let us focus only on a single state. The claim is that for any $f \in \mathcal{B}(\mathcal{A})$, which in our case is the quantity within the brackets ($f(a) = r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)V(x')$), we have

$$\sup_{\mu \in \mathcal{M}(\mathcal{A})} \int \mu(\mathrm{d}a) f(a) = \max_{a \in \mathcal{A}} f(a). \tag{2.9}$$

This is true because we can define μ as a Dirac's delta function at one of the maximizers $a^* \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} f(a)$, and the value would be the same. This shows that

$$\sup_{\mu \in \mathcal{M}(\mathcal{A})} \int \mu(\mathrm{d}a) f(a) \geq \int \delta(a - a^*) f(a) \mathrm{d}a = f(a^*) = \max_{a \in \mathcal{A}} f(a).$$

On the other hand, as $f(a) \leq f(a^*)$ for any $a \in \mathcal{A}$, the integral (or expectation) of f w.r.t. any distribution μ is not going to be greater than $f(a^*)$, so it is not possible to find a distribution μ such as that the left-hand side (LHS) is strictly greater than the RHS.

All this, in summary, show that

$$TV = \sup_{\pi \in \Pi_{\text{det}}} T^\pi V = \sup_{\pi \in \Pi} T^\pi V. \tag{2.10}$$

Therefore, the Bellman optimality operator is the supremum of the Bellman operator T^π over all stochastic or deterministic policies.

We shall define some extensions of the Bellman operators later. Next, we focus on studying some important properties of the Bellman operators.

2.4 Properties of the Bellman Operators

The Bellman operators have some interesting and important properties. These properties are crucially used in some proofs as basic as the existence and uniqueness of the solution for the Bellman equations. They are used, directly or indirectly, in many RL/Planning algorithms too. The properties that matters for us the most are

- Monotonicity
- Contraction

We shall define what these mean next.

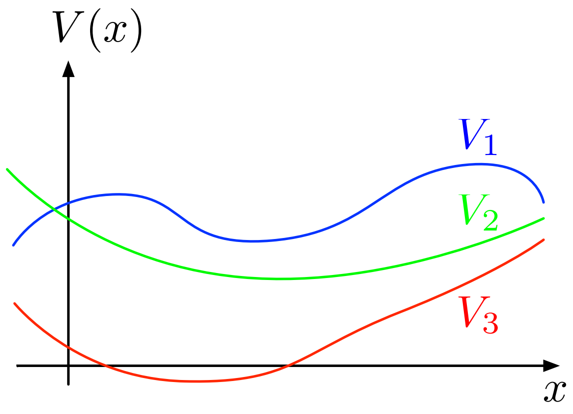

Figure 2.1: A few examples of the order relation between values: $V_3 \leq V_1$ and $V_3 \leq V_2$, but neither $V_2 \leq V_1$, nor $V_1 \leq V_2$.

2.4.1 Monotonicity

For two functions $V_1, V_2 \in \mathcal{B}(\mathcal{X})$, we use $V_1 \leq V_2$ if and only if $V_1(x) \leq V_2(x)$ for all $x \in \mathcal{X}$. This is the same notation as we used in Section 2.1.2 for comparing the value of two policies ($V^\pi \geq V^{\pi'}$), except that here we extend it to any arbitrary function defined over $\mathcal{X}$, as opposed to the value function of a particular policy. Figures 2.1 depicts an example.

The *monotonicity* of the Bellman operator means that if $V_1 \leq V_2$ and we apply the Bellman operator to both sides, we get $T^\pi V_1 \leq T^\pi V_2$, i.e., the Bellman operator does not change the order relationship. The next result shows that this is indeed true for both T^π and T^*.

Lemma 2.2 (Monotonicity). *Fix a policy π. If $V_1, V_2 \in \mathcal{B}(\mathcal{X})$, and $V_1 \leq V_2$, then we have*

$$T^\pi V_1 \leq T^\pi V_2,$$
$$T^* V_1 \leq T^* V_2.$$

Proof. Let us expand $T^\pi V_1$. As $V_1(x') \leq V_2(x')$ for any $x' \in \mathcal{X}$, we get that for any

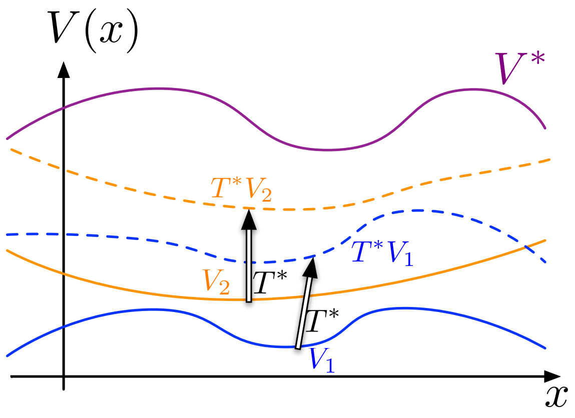

Figure 2.2: When $V_1 \leq V_2$, we also have $T^*V_1 \leq T^*V_2$.

$x \in \mathcal{X}$,

$$
\begin{aligned}
(T^\pi V_1)(x) &= r^\pi(x) + \gamma \int \mathcal{P}^\pi(\mathrm{d}x'|x) \underbrace{V_1(x')}_{\leq V_2(x')} \\
&\leq r^\pi(x) + \gamma \int \mathcal{P}^\pi(\mathrm{d}x'|x) V_2(x') = (T^\pi V_2)(x).
\end{aligned}
$$

Therefore, $T^\pi V_1 \leq T^\pi V_2$. This is the first claim.

For the Bellman optimality operator, we follow almost the same argument. For any $x \in \mathcal{X}$, we have

$$
\begin{aligned}
(T^*V_1)(x) &= \max_{a\in\mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \underbrace{V_1(x')}_{\leq V_2(x')} \right\} \\
&\leq \max_{a\in\mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V_2(x') \right\} = (T^*V_2)(x).
\end{aligned}
$$

Therefore, $T^*V_1 \leq T^*V_2$. $\square$

Figure 2.2 graphically shows the monotonicity property for T^*. A similar graph would hold for T^π. Note that V_1 or V_2 do not have to be smaller than V^* as they are arbitrary value functions, and not the value of a policy. As such, T^*V_1 is not necessarily smaller than V^* either, but it just happens to be in this depiction.[5]

[5]If it happens that for a value function V, we have that $V \leq T^*V$, we can prove that V and T^*V are both smaller or equal to V^*. We shall show this later. This is the case in this figure.

Exercise 2.5 ($\star$)**.** *Prove that the Bellman operators $T^\pi : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$ and $T^* : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$ applied on Q satisfy the monotonicity property.*

Exercise 2.6 ($\star\star\star$)**.** *Suppose that the Bellman operator was defined as*

$$(T^\pi_{mult} V)(x) \triangleq r^\pi(x) \int \mathcal{P}^\pi(\mathrm{d}x'|x) V(x'),$$

for any $x \in \mathcal{X}$. Answer the following questions:

(a) Is operator T^π_{mult} monotonic? If yes, prove it. If not, what assumptions do we need to make in order to guarantee its monotonicity?

(b) What notion of long-term reward this operator corresponds to?

2.4.2 Contraction

The Bellman operators for discounted problems are contraction mappings. Refer to Appendix A.3 for a brief introduction on contraction mapping, why they are important, and the statement of the contraction mapping (or Banach fixed point) theorem (Theorem A.1).

We often use the supremum norm of value functions (see Example A.4). Let us write them down here: For $V \in \mathcal{B}(\mathcal{X})$ and $Q \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$, their supremum norms are

$$\begin{aligned} \|V\|_\infty &= \sup_{x \in \mathcal{X}} |V(x)|, \\ \|Q\|_\infty &= \sup_{(x,a) \in \mathcal{X} \times \mathcal{A}} |Q(x,a)|. \end{aligned}$$

We can show that the the Bellman operators T^π and T^* are contraction mappings.

Lemma 2.3 (Contraction)**.** *For any π, the Bellman operator T^π is a γ-contraction mapping. Moreover, the Bellman operator T^* is a γ-contraction mapping.*

Proof. We show it for the Bellman operators applied on action-value function, i.e. $T^\pi : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$ and $T^* : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$.

Consider two action-value functions $Q_1, Q_2 \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$. Consider the metric $d(Q_1, Q_2) = \|Q_1 - Q_2\|_\infty$. We show the contraction w.r.t. this metric.

We start with the proof for T^π. For any $(x,a) \in \mathcal{X} \times \mathcal{A}$, we have

$$\begin{aligned}
|(T^\pi Q_1)(x,a) - (T^\pi Q_2)(x,a)| = \Bigg| & \left[r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x')Q_1(x',a') \right] - \\
& \left[r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x')Q_2(x',a') \right] \Bigg| \\
= \gamma & \left| \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x') \left(Q_1(x',a') - Q_2(x',a') \right) \right| .
\end{aligned}$$

We are going to upper bound the RHS. Since we will see similar arguments frequently, let us do each step of it in detail. We have an integral of the form $\left| \int P(\mathrm{d}x) f(x) \right|$ (or a summation $|\sum_x P(x) f(x)|$ for a countable state space). This can be upper bounded as

$$\begin{aligned}
\left| \int P(\mathrm{d}x) f(x) \right| \leq \int |P(\mathrm{d}x) f(x)| = \int |P(\mathrm{d}x)|.|f(x)| \leq \int P(\mathrm{d}x). \sup_{x \in \mathcal{X}} |f(x)| \\
= \sup_{x \in \mathcal{X}} |f(x)| \int P(\mathrm{d}x) = \|f\|_\infty ,
\end{aligned}$$

where in the last equality we used the fact that for a probability distribution P, we have $\int P(\mathrm{d}x) = 1$.

In our case, we get that

$$\begin{aligned}
|(T^\pi Q_1)(x,a) - (T^\pi Q_2)(x,a)| &= \gamma \left| \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x') \left(Q_1(x',a') - Q_2(x',a') \right) \right| \\
&\leq \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x') \left| Q_1(x',a') - Q_2(x',a') \right| \\
&\leq \gamma \|Q_1 - Q_2\|_\infty \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x') \\
&= \gamma \|Q_1 - Q_2\|_\infty .
\end{aligned}$$

This inequality holds for any $(x,a) \in \mathcal{X} \times \mathcal{A}$, so it holds for its supremum over $\mathcal{X} \times \mathcal{A}$ too, that is,

$$\|(T^\pi Q_1) - (T^\pi Q_2)\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty .$$

This shows that T^π is a γ-contraction.

Showing the contraction property of T^* is similar. We consider two action-value functions $Q_1, Q_2 \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$. We get

$$\begin{aligned}|(T^*Q_1)(x,a) - (T^*Q_2)(x,a)| &= \gamma \left| \int \mathcal{P}(\mathrm{d}x'|x,a) \left(\max_{a' \in \mathcal{A}} Q_1(x',a') - \max_{a' \in \mathcal{A}} Q_2(x',a') \right) \right| \\ &\leq \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \sup_{x' \in \mathcal{X}} \left| \max_{a' \in \mathcal{A}} Q_1(x',a') - \max_{a' \in \mathcal{A}} Q_2(x',a') \right|. \end{aligned} \tag{2.11}$$

We have that for two functions $f_1, f_2 : \mathcal{A} \to \mathbb{R}$,

$$\left| \max_{a \in \mathcal{A}} f_1(a) - \max_{a \in \mathcal{A}} f_2(a) \right| \leq \max_{a \in \mathcal{A}} |f_1(a) - f_2(a)|.$$

Therefore, we get that the RHS of (2.11) is upper bounded by

$$\begin{aligned}(2.11) &\leq \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \sup_{x' \in \mathcal{X}} \max_{a' \in \mathcal{A}} |Q_1(x',a') - Q_2(x',a')| \\ &= \gamma \sup_{(x,a) \in \mathcal{X} \times \mathcal{A}} |Q_1(x,a) - Q_2(x,a)| \int \mathcal{P}(\mathrm{d}x'|x,a) \\ &= \gamma \|Q_1 - Q_2\|_\infty .\end{aligned}$$

This proves the second claim. □

Exercise 2.7 ($\star\star$). *Prove that for two functions $f_1, f_2 : \mathcal{A} \to \mathbb{R}$, we have*

$$\left| \max_{a \in \mathcal{A}} f_1(a) - \max_{a \in \mathcal{A}} f_2(a) \right| \leq \max_{a \in \mathcal{A}} |f_1(a) - f_2(a)|.$$

2.5 Some Consequences of Monotonicity and Contraction

We have shown that the Bellman operators for a discounted MDP are both monotonic (Lemma 2.2) and γ-contraction w.r.t. the supremum norm (Lemma 2.3). These properties have important consequences, which we study some of them here. One of the most important ones is that these operators have a unique fixed point. This, in turn, shows that the Bellman equations have a unique solution.

2.5.1 Uniqueness of Fixed Points

Proposition 2.4 (Uniqueness of Fixed Points)**.** *The operators T^π and T^* have unique fixed points, denoted by V^π (Q^π) and V^* (Q^*), i.e.,*

$$\begin{aligned} V^\pi &= T^\pi V^\pi; \quad Q^\pi = T^\pi Q^\pi, \\ V^* &= T^* V^*; \quad Q^* = T^* Q^*. \end{aligned}$$

Moreover, they can be computed from any $V_0 \in \mathcal{B}(\mathcal{X})$ or $Q_0 \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$, by iteratively computing

$$\begin{aligned} V_{k+1} &\leftarrow T^\pi V_k; \quad Q_{k+1} \leftarrow T^\pi Q_k, \\ V_{k+1} &\leftarrow T^* V_k; \quad Q_{k+1} \leftarrow T^* Q_k, \end{aligned}$$

for $k = 0, 1, \dots$. We have that $V_k \to V^\pi$ and $Q_k \to Q^\pi$ (for T^π) and $V_k \to V^*$ and $Q_k \to Q^*$ (for T^*).

Proof. Consider the space of bounded functions $\mathcal{B}(\mathcal{X})$ with the metric d based on the uniform norm, i.e., $d(V_1, V_2) = \|V_1 - V_2\|_\infty = \sup_{x \in \mathcal{X}} |V_1(x) - V_2(x)|$. The space $(\mathcal{B}(\mathcal{X}), d)$ is a complete metric space.

Lemma 2.3 shows that for any π, the operator T^π is a γ-contraction. The same lemma shows that T^* has the same property too.

By the Banach fixed point theorem (Theorem A.1), they have a unique fixed point. Moreover, any sequence (V_k) with $V_0 \in \mathcal{B}(\mathcal{X})$ and $V_{k+1} \leftarrow T^\pi V_k$ $(k = 0, 1, \dots)$ is convergent, which means that $\lim_{k \to \infty} \|V_k - V^\pi\|_\infty = 0$. The same is true for the sequence generated by the repeated application of T^*, with appropriate modifications. □

This result suggests a way to find V^π and V^* by repeated application of the Bellman operators. This procedure will be one of the main approaches to find the value function (either for a policy π or the optimal one). It is called *Value Iteration*. We shall define and study it later in Chapter 3.

In the proof of the result, we used the completeness property of $\mathcal{B}(\mathcal{X})$, so that we can use the Banach fixed point theorem. We do not prove its completeness property (or even define what it actually means). For the proof, see Theorem 43.6 of Munkres [2018].

Also note that we only used the contraction property of the Bellman operators, and not their monotonicity. We will use the monotonicity later.

The next result shows that the greedy policy of the optimal value function has the value of V^*. This partially justifies the use of the greedy policy.

Proposition 2.5 (Proposition 2.1.1(c) of Bertsekas 2018)**.** *We have $T^{\pi}V^{*} = T^{*}V^{*}$ if and only if $V^{\pi} = V^{*}$.*

Proof. Let us first assume that $T^{\pi}V^{*} = T^{*}V^{*}$. We try to prove that $V^{\pi} = V^{*}$. As V^{*} is the solution of the Bellman optimality equation, we have $T^{*}V^{*} = V^{*}$. Therefore,

$$T^{\pi}V^{*} = T^{*}V^{*} = V^{*}.$$

This shows that V^{*} is a fixed point of T^{π}. The fixed point of T^{π}, however, is unique (Proposition 2.4) and is equal to V^{π}. So V^{π} and V^{*} should be the same, i.e., $V^{\pi} = V^{*}$.

To prove the other direction, we apply T^{π} to both sides of $V^{*} = V^{\pi}$ to get

$$T^{\pi}V^{*} = T^{\pi}V^{\pi}.$$

As V^{π} is the solution of the Bellman equation for policy π, we have $T^{\pi}V^{\pi} = V^{\pi}$. Therefore,

$$T^{\pi}V^{*} = T^{\pi}V^{\pi} = V^{\pi}.$$

By assumption, $V^{\pi} = V^{*}$. So we have $T^{\pi}V^{*} = V^{\pi} = V^{*}$. On the other hand, we have $V^{*} = T^{*}V^{*}$, so

$$T^{\pi}V^{*} = V^{*} = T^{*}V^{*},$$

which is the desired result. □

The same holds for Q too, i.e., $T^{\pi}Q^{*} = T^{*}Q^{*} \iff Q^{\pi} = Q^{*}$.

Exercise 2.8 ($\star$)**.** *Prove that $T^{\pi}Q^{*} = T^{*}Q^{*}$ if and only if $Q^{\pi} = Q^{*}$.*

This proposition shows if $T^{\pi}V^{\pi} = T^{*}V^{*}$ for some policy π, the value function V^{π} of that policy is the same as the fixed point of T^{*}, which is V^{*}. Note that we have not yet shown that the fixed point of T^{*} is an optimal value function, in the sense that it is $\pi^{*} \leftarrow \operatorname{argmax}_{\pi \in \Pi} V^{\pi}(x)$ (for all $x \in \mathcal{X}$) over the space of all stationary policies Π, or even more generally, over the set of all non-stationary policies. But it is indeed true, as we shall see.

To see the connection to the greedy policy (2.7) more clearly, note that given any V, the greedy policy selects

$$\operatorname*{argmax}_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V(x') \right\}.$$

So the Bellman operator of the greedy policy of V being applied to V (i.e., $T^{\pi_g(V)}V$) is

$$T^{\pi_g(V)}V = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V(x') \right\}. \tag{2.12}$$

Let us compare it with T^* being applied to V. We have

$$(T^*V)(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)V(x') \right\}. \tag{2.13}$$

Both are the same, so $T^{\pi_g(V)}V = T^*V$. In particular, if $V = V^*$, we have $T^{\pi_g(V^*)}V^* = T^*V^*$. This proposition then states that the value of $\pi_g(V^*)$, that is $V^{\pi_g(V^*)}$ is the same as V^*.

This means that if we find V^*, we can compute its greedy policy $\pi_g(V^*)$, and the greedy policy would have the value of V^* (which so far we do not know if it is the same as the optimal value function, but it is).

Its consequence is that we can find the optimal value function V^*, and get the optimal policy by simply computing its greedy policy.

2.5.2 Error Bounds

The consequence of the uniqueness property of the Bellman operators is that if we find a function $V = T^\pi V$ (or $V = T^*V$), we know that $V = V^\pi$ (or $V = V^*$). What happens if we find a function V that only approximately satisfies the Bellman equations? That is, suppose we only have $V \approx T^\pi V$ (or $V \approx T^*V$). Can we say anything about these approximate solutions to the Bellman equations, and how close they are to V^π (or V^*)?

Answering such a question is important because in practice, we may not be able solve the Bellman equations exactly, but only approximately. There are several reasons for approximation. Some of them are computational, which limit how much we can get close to the solution of the fixed-point equations, some are because we can only represent value functions approximately due to the use of function approximations, and some are due to the statistical errors. We get to these in later chapters.

It turns out that we can relate the closeness of V to V^π based on the size of $\mathrm{BR}(V) \triangleq T^\pi V - V$ or $\mathrm{BR}^*(V) \triangleq T^*V - V$. The functions $\mathrm{BR}(V)$ and $\mathrm{BR}^*(V)$ are called the *Bellman Residual*. There are several such results in the literature. The next one is a simple one, which shows the flavour of these results.

Proposition 2.6. *For any $V \in \mathcal{B}(\mathcal{X})$ or $Q \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$, we have*

$$\|V - V^*\|_\infty \le \frac{\|V - T^*V\|_\infty}{1-\gamma}, \qquad \|Q - Q^*\|_\infty \le \frac{\|Q - T^*Q\|_\infty}{1-\gamma}.$$

Proof. We add and subtract T^*V to $V - V^*$, take the supremum norm, and use the triangle inequality to get

$$\begin{aligned} V - V^* &= V - T^*V + T^*V - V^* \\ \Rightarrow \|V - V^*\|_\infty &= \|V - T^*V + T^*V - V^*\|_\infty \\ &\leq \|V - T^*V\|_\infty + \|T^*V - V^*\|_\infty . \end{aligned}$$

Let us focus on the term $\|T^*V - V^*\|_\infty$. We make two observations: (1) $V^* = T^*V^*$, and (2) the Bellman optimality operator is a γ-contraction w.r.t. the supremum norm. Thus,

$$\|T^*V - V^*\|_\infty = \|T^*V - T^*V^*\|_\infty \leq \gamma \|V - V^*\|_\infty .$$

Therefore,

$$\|V - V^*\|_\infty \leq \|V - T^*V\|_\infty + \gamma \|V - V^*\|_\infty .$$

Re-arranging this, we get that

$$(1-\gamma) \|V - V^*\|_\infty \leq \|V - T^*V\|_\infty ,$$

which is the desired result. The proof for the second part is the same. □

The norm of the Bellman Residual is called the *Bellman Error*. Here we use the supremum norm to quantify its size, but we could also define it w.r.t. other norms. In that case, however, we do not get the same result, as a crucial step in the proof was the γ-contraction of the Bellman operator w.r.t. the supremum norm. If we change the norm, the Bellman operator would not necessarily be a contraction anymore.

Exercise 2.9 ($\star$)**.** *Why don't we get the same result if we change the norm from the supremum norm to the ℓ_2-norm or other ℓ_p-norms (with $p < \infty$), e.g., $\|V\|_2 = \sqrt{\sum_{x \in \mathcal{X}} |V(x)|^2}$ (for countable state space), see Example A.3.*

Proposition 2.7. *For any $V \in \mathcal{B}(\mathcal{X})$ or $Q \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$, and any $\pi \in \Pi$, we have*

$$\|V - V^\pi\|_\infty \leq \frac{\|V - T^\pi V\|_\infty}{1-\gamma}, \qquad \|Q - Q^\pi\|_\infty \leq \frac{\|Q - T^\pi Q\|_\infty}{1-\gamma}.$$

Exercise 2.10 ($\star$)**.** *Prove Proposition 2.7.*

2.5.3 Fixed Point of T^* is the Optimal Value Function

Next we show that the fixed point of T^* is indeed the optimal value function within the space of stationary policies Π. This result uses the monotonicity of T^*, in addition to contraction, which was used before.

Proposition 2.8 (Proposition 2.1.2 of Bertsekas 2018)**.** *Let V^* be the fixed point of T^*, i.e., $V^* = T^*V^*$. We have*

$$V^*(x) = \sup_{\pi \in \Pi} V^\pi(x), \qquad \forall x \in \mathcal{X}.$$

Proof. We first show that $V^*(x) \leq \sup_{\pi\in\Pi} V^\pi(x)$ (for all $x \in \mathcal{X}$). We then show the opposite direction, that is, $\sup_{\pi\in\Pi} V^\pi(x) \leq V^*(x)$. These two combined prove the statement.

For the first direction, we use Proposition 2.7 with the choice of $V = V^*$. We get that for any $\pi \in \Pi$,

$$\|V^* - V^\pi\|_\infty \leq \frac{\|V^* - T^\pi V^*\|_\infty}{1-\gamma}. \tag{2.14}$$

Let $\varepsilon > 0$. Choose a policy a π_ε such that

$$\|V^* - T^{\pi_\varepsilon} V^*\|_\infty \leq (1-\gamma)\varepsilon.$$

This is possible because we have

$$(T^*V^*)(x) = \sup_{a\in\mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^*(x') \right\},$$

so it is sufficient to pick a π_ε that solves the optimization problem up to $(1-\gamma)\varepsilon$-accuracy of the supremum at each state x (if we find the maximizer, then $\varepsilon = 0$).

For policy π_ε, (2.14) shows that

$$\|V^* - V^{\pi_\varepsilon}\|_\infty \leq \varepsilon.$$

This means that

$$V^*(x) \leq V^{\pi_\varepsilon}(x) + \varepsilon, \qquad \forall x \in \mathcal{X}.$$

Notice that $V^{\pi_\varepsilon}(x) \leq \sup_{\pi\in\Pi} V^\pi(x)$ (as $\pi_\varepsilon \in \Pi$). We take $\varepsilon \to 0$ to get that

$$V^*(x) \leq \lim_{\varepsilon\to 0} \{V^{\pi_\varepsilon}(x) + \varepsilon\} \leq \lim_{\varepsilon\to 0} \left\{ \sup_{\pi\in\Pi} V^\pi(x) + \varepsilon \right\} = \sup_{\pi\in\Pi} V^\pi(x) \qquad \forall x \in \mathcal{X}. \tag{2.15}$$

This shows that V^*, the fixed point of T^*, is smaller or equal to the optimal value function within the space of stationary policies.

To show the other direction, consider any $\pi \in \Pi$. By the definition of T^π and T^*, for any $V \in \mathcal{B}(\mathcal{X})$, we have that for any $x \in \mathcal{X}$,[6]

$$\begin{aligned}
(T^\pi V)(x) &= \int \pi(\mathrm{d}a|x) \left[r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V(x') \right] \\
&\leq \sup_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V(x') \right\} \\
&= (T^* V)(x).
\end{aligned}$$

In particular, with the choice of $V = V^*$, we have $T^\pi V^* \leq T^* V^*$. As $T^* V^* = V^*$, we have

$$T^\pi V^* \leq V^*. \tag{2.16}$$

We use the monotonicity of T^π (Lemma 2.2) to conclude that

$$T^\pi (T^\pi V^*) \leq T^\pi V^*,$$

which by (2.16) shows that

$$(T^\pi)^2 V^* \leq V^*.$$

We repeat this argument for k times to get that

$$(T^\pi)^k V^* \leq V^*.$$

As $k \to \infty$, Proposition 2.4 shows that $(T^\pi)^k V^*$ converges to V^π (the choice of V^* is irrelevant). Therefore,

$$V^\pi = \lim_{k \to \infty} (T^\pi)^k V^* \leq V^*.$$

As this holds for any $\pi \in \Pi$, we take the supremum over $\pi \in \Pi$ to get

$$\sup_{\pi \in \Pi} V^\pi \leq V^*. \tag{2.17}$$

The inequalities (2.16) and (2.17) together show that

$$V^* = \sup_{\pi \in \Pi} V^\pi,$$

which is the desired result. $\square$

[6] If this is not clear, see the discussion around (2.9).

2.5.3.1 Optimality Over the Space of Non-stationary Policies

We have not ruled out the possibility that there exists a non-stationary policy (Definition 1.2) in the form of $\bar{\pi} = (\pi_1, \pi_2, \dots)$ with each π_t being a Markov policy such that following it may lead to higher expected return compared to following any single stationary Markov policy π, including the optimal one $\pi^* \leftarrow \operatorname{argmax}_{\pi \in \Pi} V^\pi$ (1.15). By following $\bar{\pi}$, we mean that at the first time step, the agent chooses action from π_1, and the second time step, the agent chooses action from π_2, and so on.

Let us introduce some notations. Let $\bar{\Pi}$ denote the space of all non-stationary policies, so $\bar{\pi} \in \bar{\Pi}$. More formally,

$$\bar{\Pi} = \{\, \bar{\pi} = (\pi_1, \pi_2, \dots) \,:\, \pi_t : \mathcal{X} \to \mathcal{M}(\mathcal{A}), t = 1, 2, \dots \,\}\,.$$

Any stationary policy $\bar{\pi} = (\pi, \pi, \dots)$ with $\pi : \mathcal{X} \to \mathcal{M}(\mathcal{A})$, is a member of $\bar{\Pi}$ too.

We can convince ourselves that the expected return of following $(\pi_1, \pi_2, \dots, \pi_k)$ and terminating at time k with the terminal reward of V_0 is

$$V^{\pi_1:\pi_k} = T^{\pi_1} T^{\pi_2} \dots T^{\pi_k} V_0.$$

If we let $k \to \infty$, this would be the value of following the infinite sequence of $\bar{\pi} = (\pi_1, \pi_2, \dots)$, i.e.,

$$V^{\bar{\pi}} = \liminf_{k \to \infty} V^{\pi_1:\pi_k}.$$

When $k \to \infty$, the choice of V_0 does not matter, as it is discounted by $\lim_{k \to \infty} \gamma^k = 0$.

Define the optimal value function within the space of non-stationary policy as V^+:

$$V^+ = \sup_{\bar{\pi} \in \bar{\Pi}} V^{\bar{\pi}}.$$

Proposition 2.9. *We have*

$$V^* = V^+.$$

Proof. Since the space of stationary policies is a subset of the space of non-stationary policies, we have

$$V^* = \sup_{\pi \in \Pi} V^\pi \le \sup_{\bar{\pi} \in \bar{\Pi}} V^{\bar{\pi}} = V^+, \tag{2.18}$$

where the equality of V^* and $\sup_{\pi \in \Pi} V^\pi$ is because of Proposition 2.8.

Now consider an arbitrary sequence $(\pi_1, \dots, \pi_k)$, and any value function V_0. By the optimality property of T^*, we have that

$$T^{\pi_k} V_0 \le T^* V_0.$$

By the monotonicity of $T^{\pi_{k-1}}$, we get that

$$T^{\pi_{k-1}} T^{\pi_k} V_0 \leq T^{\pi_{k-1}} T^* V_0.$$

Again, by the optimality property of T^*, we have

$$T^{\pi_{k-1}} T^* V_0 \leq T^* T^* V_0 = (T^*)^2 V_0.$$

Repeating this argument, we get that

$$T^{\pi_1} T^{\pi_2} \cdots T^{\pi_k} V_0 \leq (T^*)^k V_0.$$

By letting $k \to \infty$, the LHS converges to $V^{\bar{\pi}}$, and the RHS converges to V^* by Proposition 2.4, i.e.,

$$V^{\bar{\pi}} = \liminf_{k \to \infty} T^{\pi_1} T^{\pi_2} \cdots T^{\pi_k} V_0 \leq \lim_{k \to \infty} T^k V_0 = V^*.$$

This is true for any $\bar{\pi} \in \bar{\Pi}$, so we have

$$\sup_{\bar{\pi} \in \bar{\Pi}} V^{\bar{\pi}} \leq V^*. \tag{2.19}$$

By (2.18) and (2.19) we get the desired result. □

Chapter 3

Planning with a Known Model

Our goal is to design an RL agent that should be able to act optimally (or close to optimally) in an environment.[1] So we we have defined the value function, policy, and studied of their important properties. We have not presented any mechanism to find the optimal policy though. This chapter provides general approaches for finding the optimal policy. The underlying assumption here is that the MDP is known, i.e., we know $\mathcal{R}$ and $\mathcal{P}$.

The assumption of knowing the MDP does not hold in the RL setting. In that setting, the agent can only observe data coming from interaction with the environment. Designing methods for finding the optimal policy given that knowledge, however, would provide the foundation for developing methods for the RL setting. We shall see that many methods that can handle the RL setting are sample-based variants of the methods in this chapter.

The methods for finding the optimal policy π^* can be roughly divided to three categories:

- Value-based
- Direct policy search
- Hybrid methods

The first category tries to find V^* or Q^* first, and then use it to compute the optimal policy. This is often done by simply computing the greedy policy π_g w.r.t. the approximation of the optimal policy. As we discussed in Section 2.2 and formally proved in Proposition 2.5, this is a sensible approach.

[1]Chapter's Version: 0.04 (2021 January 26). Some examples are incomplete and need to be typed.

The methods in the second category directly search in the space of policies without explicitly constructing the optimal vale function, though they may still implicitly use a quantity that has the same interpretation as the value function.

There are also hybrid methods that use the explicitly constructed value function to guide the search in the policy space.

We should note that the boundaries between these categories are not clear cut.

In this chapter, our focus will be on the value-based methods. We talk about policy search ones in a later chapter (Chapter XXX).

3.1 Policy Evaluation vs. Control Problems

We need a distinction between two types of problems that we often solve.

- Policy Evaluation (PE)
- Control

The problem of *policy evaluation* refers to the problem of computing the value function of a given policy π. This is not the ultimate goal of an RL agent, finding the optimal policy is, but is often needed as an intermediate step in finding the optimal policy. The problem of *control* refers to the problem of finding the optimal value function V^* or Q^* or optimal policy π^*. This is admittedly a confusing terminology.

We would also want to mention that some of the described methods are considered as *dynamic programming* methods. These methods benefit from the structure of the MDP, such as the recursive structure encoded in the Bellman equation, in order to compute the value function.

Let us focus on the policy evaluation problem first.

3.2 Policy Evaluation: Two Initial Attempts

Problem Statement: Given an MDP $(\mathcal{X}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ and a policy π, we would like to compute V^π or Q^π.

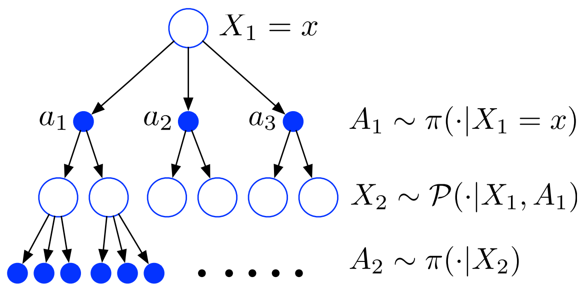
$X_1 = x$
a_1
a_2
a_3
$A_1 \sim \pi(\cdot|X_1 = x)$
$X_2 \sim \mathcal{P}(\cdot|X_1, A_1)$
$A_2 \sim \pi(\cdot|X_2)$

Figure 3.1: All future possibilities starting from state $X_1 = x$ up to depth 2.

3.2.1 A Naive Solution

Let us start from a naive solution. For simplicity, we assume finite $\mathcal{X} \times \mathcal{A}$. To compute V^π at a state $x \in \mathcal{X}$, we refer to its definition:

$$V^\pi(x) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | X_1 = x\right].$$

This expectation is w.r.t. the sequence of r.v. $(X_1 = x, A_1, X_1, X_2, A_2, R_2, \dots)$ with $A_t \sim \pi(\cdot|X_t)$, $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$, and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$. Depending on the draw of each r.v., we get different sequences. We can represent all possibilities in a tree structure, partially depicted in Figure 3.1.

In principle, we can compute the probability of being in each state. For example, at time $t = 2$, the probability of the agent being in state x' and choosing action a' is

$$\sum_{a \in \mathcal{A}} \pi(a|x)\mathcal{P}(x'|x, a)\pi(a'|x')$$

The expected reward that the agent receives at time $t = 2$ is then

$$\sum_{a, x', a'} \pi(a|x)\mathcal{P}(x'|x, a)\pi(a'|x')r(x', a').$$

We can continue the expansion to compute the expected reward at other time steps too, and eventually add them in a discounted way to compute the value $V^\pi(x)$.

Nevertheless, this is not a satisfactory solution as the size of the tree grows exponentially fast. Also for continuing problem, the depth of the tree is going to be infinity.

Exercise 3.1 ($\star$)**.** *What is the size of the tree by the time that we get to X_t ($t \geq 1$)?*

Exercise 3.2 ($\star\star$)**.** *Suppose that we expand the tree only up to horizon $H < \infty$, and use it to compute an H-truncated approximation V_H^π of V^π, defined as*

$$V_H^\pi(x) = \mathbb{E}\left[\sum_{t=1}^{H} \gamma^{t-1} R_t | X_1 = x\right].$$

Assume that $|r(x,a)| \leq R_{max}$ for all $(x,a) \in \mathcal{X} \times \mathcal{A}$. Provide an upper bound on $|V^\pi(x) - V_H^\pi(x)|$.

3.2.2 Linear System of Equations

A much better way to compute V^π is to benefit from the recursive structure of the value function, represented by the Bellman equation $V^\pi = T^\pi V^\pi$. In the discrete state-action case, the Bellman equation defines a set of $n = |\mathcal{X}|$ equations with $|\mathcal{X}|$ unknowns $(V(x_1), \dots, V(x_n))$ and has the the form of

$$V(x) = r^\pi(x) + \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^\pi(x'|x) V(x'),$$

for each $x \in \mathcal{X}$. As the solution of the Bellman equation is unique (Proposition 2.4), the solution V of these equations would be the same as V^π.

How can we solve this set of equations? As the Bellman equation for a policy π is a linear system of equations, we can simply use any standard solver to compute $V = V^\pi$. To see the linearity more clearly, we re-arrange the equation to get

$$V(x) - \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^\pi(x'|x) V(x') = r^\pi(x),$$

or more compactly in the matrix form as

$$(\mathbf{I} - \gamma \mathcal{P}^\pi) V = r^\pi,$$

where $\mathbf{I}$ is an $n \times n$-identity matrix, and $\mathcal{P}^\pi$ is overloaded to denote an $n \times n$ stochastic matrix with $[\mathcal{P}^\pi]_{i,j} = \mathcal{P}^\pi(x_j|x_i)$. This has the same form as

$$A_{n\times n} x_{n\times 1} = b_{n\times 1},$$

commonly used to represent a linear system of equations.

If the size of the state space n is not too large (say, a few thousands or so), we can easily solve this linear system of equation either by computing the inverse of A and writing it as $x = A^{-1}b$, or even better, using a standard linear equation solver. When the state space becomes very large, the standard solvers may not scale very well, and we have to use solvers that are somewhat specialized to the properties of the value function.[2]

We remark that to solve the control problem of finding V^*, we need to solve $V = T^*V$, whose unique solution is V^*. This would be n equations in the form of

$$V(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}(x'|x,a)V(x') \right\}.$$

These equations, however, are not linear in V anymore, and the use of a linear solver is not feasible. We shall see later that we can formulate it as a linear program (LP) instead.

Exercise 3.3 ($\star$). *Write down A, x, and b in terms $\mathcal{P}^\pi$, r^π, γ, and V. What is the form of A if the MDP is deterministic?*

Exercise 3.4 ($\star$). *Write down the linear system of equations needed for finding Q^π. How many equations do we have?*

Exercise 3.5 ($\star\star$). *What solvers for linear systems of equations do you know? Describe one or two in some detail. What is the computational complexity of getting to ε-approximate solution for the solver?*

Exercise 3.6 ($\star\star\star$). *(Project Idea? Open ended) Identify and implement a novel solver that we often do not use in DP. Can we find a sample-based version of it?*

3.3 Value Iteration

One of the main approaches to find the value function is called *Value Iteration* (VI), which we briefly encountered in Section 2.5.1. It is a direct consequence of Proposition 2.4, in which we showed that the fixed point of the Bellman operators are unique, and they can be computed by the iterative application of the Bellman

[2]One such solver is called Value Iteration, which we shall describe soon. Note that VI used for evaluating π can be interpreted as implementing the Jacobi iteration method for solving a linear system of equations, so it is a linear solver itself.

operator. The VI can be used for both PE and Control problems. We start with its description for the PE problem.

Starting from $V_0 \in \mathcal{B}(\mathcal{X})$, we compute a sequence of $(V_k)_{k \geq 0}$ by

$$V_{k+1} \leftarrow T^\pi V_k. \tag{3.1}$$

Proposition 2.4 shows that

$$\lim_{k \to \infty} \|V_k - V^\pi\|_\infty = 0.$$

This entails the pointwise convergence of V_k to V^π too, i.e., for any $x \in \mathcal{X}$, we have that $V_k(x) \to V^\pi(x)$. As the initial value function V_0, we may simply choose $V_0 = 0$, but other choices are possible too. The procedure to compute Q^π is very similar, with the difference that we start from $Q_0 \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$ and iteratively compute $Q_{k+1} \leftarrow T^\pi Q_k$. This procedure is called the Value Iteration algorithm, and is one of the fundamental algorithms for planning. We shall see that many RL algorithms are essentially the sample-based variants of VI too.

Remark 3.1. *Sutton and Barto [2019] call this method for PE "Iterative Policy Evaluation", and reserve Value Iteration for the procedure used to compute V^* or Q^*, as we encounter soon. I use VI for both procedures because the essence of both is the iterative application of the Bellman operator on the current approximation of the value function. It should not matter whether the Bellman operator is T^π or T^*, and whether the value function is V^π or V^*. Which one we are referring to should be clear from the context, and if not, we can always use VI for PE or VI for Control to distinguish them.*

Exercise 3.7 ($\star$)**.** *Assume that we have a finite state-action MDP.*

(a) Write down VI for the computation of V^π and Q^π.

(a) What is the computational cost of each iteration of VI, in terms of basic arithmetic operations? You can assume that we have $n = |\mathcal{X}|$ and $m = |\mathcal{A}|$.

3.3.1 Value Iteration for Control

We may use VI to compute V^* or Q^*. The procedure is very similar to (3.2), with the difference that we apply the Bellman optimality operator instead, i.e.,

$$V_{k+1} \leftarrow T^* V_k, \tag{3.2}$$

$$Q_{k+1} \leftarrow T^* Q_k. \tag{3.3}$$

Proposition 2.4 guarantees that $V_k \to V^*$ (or $Q_k \to Q^*$).

3.4 Policy Iteration

A different approach is based on the iterative application of the following two steps:

- (Policy Evaluation) Given a policy π_k, compute V^{π_k} (or Q^{π_k}).
- (Policy Improvement) Find a new policy π_{k+1} that is better than π_k, i.e., $V^{\pi_{k+1}} \geq V^{\pi_k}$ (with a strict inequality in at least one state, unless at convergence).

The policy evaluation step is clear. We can use any method, such as solving a linear system of equation or even VI (PE) to compute the value of a policy π_k.

To perform the policy improvement, we have to find a policy that is better than the current one. Commonly used approach is to use the greedy policy of the value function of π_k. That is, given a value function Q^{π_k}, we set the new policy as

$$\pi_{k+1}(x) \leftarrow \pi_g(x; Q^{\pi_k}) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi_k}(x,a), \qquad \forall x \in \mathcal{X}.$$

If we are given V^{π_k}, we set $\pi_{k+1} \leftarrow \pi_g(V^{\pi_k})$.

To gain an intuition on why the greedy policy is a better policy, assume that at state x, we act according to $\pi_g(x; Q^{\pi_k})$, and afterwards, we follow π_k. The value of this new policy is

$$Q^{\pi_k}(x, \pi_g(x; Q^{\pi_k})) = Q^{\pi_k}(x, \operatorname*{argmax}_{a \in \mathcal{A}} Q^{\pi_k}(x,a)) = \max_{a \in \mathcal{A}} Q^{\pi_k}(x,a).$$

Comparing $\max_{a \in \mathcal{A}} Q^{\pi_k}(x,a)$ with the value of following the current policy at state x, which is $V^{\pi_k}(x) = Q^{\pi_k}(x, \pi_k(x))$, we get that

$$Q^{\pi_k}(x, \pi_g(x; Q^{\pi_k})) \geq V^{\pi_k}(x). \tag{3.4}$$

So this new policy is equal to better than π_k at state x. We shall prove that if we choose greedy policy at all states, the value function would satisfy a similar inequality at all states.

The policy improvement step in the procedure described at the beginning of this section only required $V^{\pi_{k+1}} \geq V^{\pi_k}$. The *Policy Iteration* (PI) algorithm refers to the specific case that we pick the new policy π_{k+1} as $\pi_g(Q^{\pi_k})$.

As the value function of π_k is the unique fixed point of T^{π_k} (Proposition 2.4), and the greedy policy $\pi_g(Q^{\pi_k})$ satisfies $T^{\pi_{k+1}} Q^{\pi_k} = T^* Q^{\pi_k}$ (see (2.13) and (2.12)), we can summarize each iteration of the Policy Iteration algorithm as

- (Policy Evaluation) Given π_k, compute Q^{π_k}, i.e., find a Q that satisfies $Q = T^{\pi_k} Q$.
- (Policy Improvement) Obtain π_{k+1} as a policy that satisfies $T^{\pi_{k+1}} Q^{\pi_k} = T^* Q^{\pi_k}$.

Remark 3.2. *We also have* approximate *policy iteration algorithms too, in which we allow some level of approximations. The approximation can be either at the policy evaluation step (we only find $Q \approx T^{\pi_k} Q$), or the policy improvement step (we only find $T^{\pi_{k+1}} Q^{\pi_k} \approx T^* Q^{\pi_k}$), or both.*

Exercise 3.8 ($\star$)**.** *Write down the Policy Iteration algorithm based on V^{π_k}.*

3.4.1 Convergence of Policy Iteration

We establish the convergence of the PI algorithm. The following result is sometimes called the *policy improvement* theorem, and is a crucial part of the convergence proof of the PI algorithm. We can think of it as extending and formalizing what we have shown in (3.4).

Theorem 3.1 (Policy Improvement)**.** *If for policies π and π', it holds that $T^{\pi'} Q^{\pi} = T^* Q^{\pi}$, we have that $Q^{\pi'} \geq Q^{\pi}$.*

Proof. We have $T^* Q^{\pi} \geq T^{\pi} Q^{\pi} = Q^{\pi}$ because for any $(x, a) \in \mathcal{X} \times \mathcal{A}$, it holds that

$$r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \max_{a' \in \mathcal{A}} Q^{\pi}(x',a') \geq r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) Q^{\pi}(x', \pi(x')).$$

Therefore, $T^{\pi'} Q^{\pi} = T^* Q^{\pi} \geq T^{\pi} Q^{\pi} = Q^{\pi}$.

We apply $T^{\pi'}$ to both sides of $T^{\pi'} Q^{\pi} \geq Q^{\pi}$, and use the monotonicity property of the Bellman operator (Lemma 2.2) to conclude

$$T^{\pi'}(T^{\pi'} Q^{\pi}) \geq T^{\pi'} Q^{\pi} = T^* Q^{\pi} \geq Q^{\pi}.$$

So we also have $(T^{\pi'})^2 Q^{\pi} \geq T^* Q^{\pi} \geq Q^{\pi}$. By repeating this argument, we get that for any $m \geq 1$,

$$(T^{\pi'})^m Q^{\pi} \geq T^* Q^{\pi} \geq Q^{\pi}. \tag{3.5}$$

Take the limit of $m \to \infty$. Because of the contraction property of the Bellman operator $T^{\pi'}$ (Proposition 2.4), we get that

$$\lim_{m \to \infty} (T^{\pi'})^m Q^{\pi} = Q^{\pi'}. \tag{3.6}$$

By combining (3.5) and (3.6), we get that

$$Q^{\pi'} = \lim_{m\to\infty} (T^{\pi'})^m Q^\pi \geq T^* Q^\pi \geq Q^\pi, \tag{3.7}$$

which is the desired result. □

The same result holds for V^π and $V^{\pi'}$.

This result shows that by choosing the greedy policy w.r.t. the most recent value function, we get a new policy that is at least as good as the previous one. Based on this result, we can actually show that the PI algorithm converges to an optimal policy. And if $|\mathcal{X} \times \mathcal{A}| < \infty$, this happens in a finite number of iterations. We shall state this result soon. The basic idea behind the proof is that we either can strictly improve the policy, or if we cannot, we are already at the optimal policy.

Theorem 3.2 (Convergence of the Policy Iteration Algorithm – Proposition 2.4.1 of Bertsekas 2018)**.** *Let $(\pi_k)_{k\geq 0}$ be the sequence generated by the PI algorithm. For all k, we have that $V^{\pi_{k+1}} \geq V^{\pi_k}$, with equality if and only if $V^{\pi_k} = V^*$. Moreover,*

$$\lim_{k\to\infty} \|V^{\pi_k} - V^*\|_\infty = 0.$$

Furthermore, if the set of policies is finite, the PI algorithm converges in a finite number of iterations.

Proof. By Theorem 3.1, we have that $V^{\pi_{k+1}} \geq V^{\pi_k}$. Suppose that instead of a strict inequality, we have an equality of $V^{\pi_{k+1}} = V^{\pi_k}$. We get that

$$T^{\pi_{k+1}} V^{\pi_k} = T^{\pi_{k+1}} V^{\pi_{k+1}}.$$

As $T^{\pi_{k+1}} V^{\pi_k} = T^* V^{\pi_k}$ by the definition of the PI algorithm, we get that

$$T^{\pi_{k+1}} V^{\pi_{k+1}} = T^* V^{\pi_k} = T^* V^{\pi_{k+1}},$$

where in the last step we used $V^{\pi_{k+1}} = V^{\pi_k}$ again. By these equalities, we have

$$T^{\pi_{k+1}} V^{\pi_{k+1}} = T^* V^{\pi_{k+1}}.$$

As $V^{\pi_{k+1}}$ is the value function of π_{k+1}, it satisfies $T^{\pi_{k+1}} V^{\pi_{k+1}} = V^{\pi_{k+1}}$. Therefore, we also have

$$V^{\pi_{k+1}} = T^* V^{\pi_{k+1}}.$$

This means that $V^{\pi_{k+1}}$ is a fixed point of T^*. But the fixed point of T^* is unique and is equal to V^* by Proposition 2.4, so we must have that $V^{\pi_{k+1}} = V^*$.

The proof of the other direction is simple: If $V^{\pi_k} = V^*$, then π_k is an optimal policy. The greedy policy of $V^{\pi_k} = V^*$ is still an optimal policy, hence $V^{\pi_{k+1}} = V^* = V^{\pi_k}$.

To prove the convergence, recall that from (3.7), we have

$$Q^{\pi_{k+1}} \geq T^* Q^{\pi_k} \geq Q^{\pi_k}. \tag{3.8}$$

By induction,

$$Q^{\pi_{k+1}} \geq T^* Q^{\pi_k} \geq T^*(T^* Q^{\pi_{k-1}}) \geq \cdots \geq (T^*)^k Q^{\pi_0}.$$

By the definition of the optimal policy, we have $Q^\pi \leq Q^*$ for any π, including all π_k generated during the iterations of the PI algorithm. So $Q^{\pi_{k+1}}$ is sandwiched between Q^* and $(T^*)^k Q^{\pi_0}$, i.e.,

$$Q^* \geq Q^{\pi_{k+1}} \geq (T^*)^k Q^{\pi_0}.$$

By the contraction property of the Bellman optimality operator, we have that

$$\lim_{k\to\infty} \left\| (T^*)^k Q^{\pi_0} - Q^* \right\|_\infty = 0.$$

As $\|Q^{\pi_{k+1}} - Q^*\|_\infty \leq \left\| (T^*)^k Q^{\pi_0} - Q^* \right\|_\infty$, we have that

$$\lim_{k\to\infty} \|Q^{\pi_k} - Q^*\|_\infty = 0.$$

This implies the convergence of V^{π_k} too.

Finally, if the number of policies is finite, the number of times (3.8) can be a strict inequality is going to be finite too. □

This result shows that the PI algorithm converges to the optimal policy in a finite number of iterations whenever the number of policies is finite. This is the case if the state space $\mathcal{X}$ and the action space $\mathcal{A}$ are finite. To see this, note that each deterministic policy $\pi : \mathcal{X} \to \mathcal{A}$ is of the form $\pi = (a_1, a_2, \ldots, a_{|\mathcal{X}|}$ with $a_i \in \mathcal{A}$. Thus, we have a total of $|\mathcal{A}|^{|\mathcal{X}|}$ different policies, which is finite.

Even though this is finite, it can be very large even for modest problems. For example, a 10×10 grid world problem with 4 actions at each state has $4^{100} \approx 1.6\times10^{60}$ possible policies. Surely we do not want to perform that many iterations of an algorithm before finding the optimal policy.

In practice, however, we observe that the PI algorithm converges much faster, for example just in a couple of iterations. This suggest that the previous analysis might be very crude. This is indeed the case, as we prove in the next section.

3.4.2 Fast Convergence of Policy Iteration

Here we show that PI converges much faster than $|\mathcal{A}|^{|\mathcal{X}|}$ iterations, as was suggested by the previous analysis. In fact, we can show that the PI terminates in

$$O\left(\frac{|\mathcal{X}||\mathcal{A}|}{1-\gamma}\log\left(\frac{1}{1-\gamma}\right)\right)$$

iterations. This result is relatively new. Variant of it have been proved by Ye [2011]; Hansen et al. [2013]; Scherrer [2016]. I closely follow the results and proofs of Scherrer [2016] in this section.

To prove it, we require a new definitions and series of intermediate results. For the rest of this section, we focus on deterministic policies. As discussed before, there always exists an optimal deterministic policy.

Let π and π' be two policies. We define the advantage of π' w.r.t. π as

$$A_{\pi}^{\pi'} = T^{\pi'}V^{\pi} - V^{\pi}.$$

To understand this definition better, let us expand it for a state x:

$$A_{\pi}^{\pi'}(x) = r(x,\pi'(x)) + \gamma\int \mathcal{P}(\mathrm{d}x'|x,\pi'(x))V^{\pi}(x') - V^{\pi}(x).$$

So at the first step we follow π', and then from the next-step onward, we follow π, and we compare that value with following π from the first step. This is equal to

$$A_{\pi}^{\pi'}(x) = Q^{\pi}(x,\pi'(x)) - V^{\pi}(x). \tag{3.9}$$

Remark 3.3. *There is function called an* advantage *function of a policy π, which is defined as*

$$A^{\pi}(x,a) = Q^{\pi}(x,a) - V^{\pi}(x).$$

The advantage function shows how much the value of an action at a state is higher than the value of the state. We that $A^{\pi}(x,\pi(x)) = 0$ (or $\mathbb{E}[A^{\pi}(x,A)] = 0$ with $A \sim \pi(\cdot|x)$). The advantage function is more commonly used in RL. Of course, $A_{\pi}^{\pi'}$ and A^{π} are related as $A_{\pi}^{\pi'}(x) = A^{\pi}(x,\pi'(x))$.

The following result reveals a relation between V^{π} and $V^{\pi'}$, and their advantage $A_{\pi}^{\pi'}$.

Lemma 3.3. *For all pairs of policies π and π', we have*

$$V^{\pi'} - V^{\pi} = (\mathbf{I} - \gamma\mathcal{P}^{\pi'})^{-1}A_{\pi}^{\pi'}.$$

Proof. The Bellman equation for π' is $V^{\pi'} = r^{\pi'} + \gamma \mathcal{P}^{\pi'} V^{\pi'}$, so by rearranging we get

$$(\mathbf{I} - \gamma \mathcal{P}^{\pi'}) V^{\pi'} = r^{\pi'}.$$

As $\mathcal{P}^{\pi'}$ is a stochastic matrix, its max norm is equal to one, i.e., $\|\mathcal{P}^{\pi'}\|_\infty = 1$. As a result, we have that $\|\gamma \mathcal{P}^{\pi'}\|_\infty = \gamma$. Lemma A.2 in Appendix A.4 shows that the matrix $(\mathbf{I} - \gamma \mathcal{P}^{\pi'})$ is invertible. Thus, we can write

$$V^{\pi'} - V^{\pi} = (\mathbf{I} - \gamma \mathcal{P}^{\pi'})^{-1} r^{\pi'} - V^{\pi}. \tag{3.10}$$

Let us expand V^π as

$$V^{\pi} = \mathbf{I} \cdot V^{\pi} = (\mathbf{I} - \gamma \mathcal{P}^{\pi'})^{-1} (\mathbf{I} - \gamma \mathcal{P}^{\pi'}) V^{\pi}.$$

Substituting this expanded form in (3.10), we get that

$$\begin{aligned}
V^{\pi'} - V^{\pi} &= (\mathbf{I} - \gamma \mathcal{P}^{\pi'})^{-1} \left[r^{\pi'} - (\mathbf{I} - \gamma \mathcal{P}^{\pi'}) V^{\pi} \right] \\
&= (\mathbf{I} - \gamma \mathcal{P}^{\pi'})^{-1} \left[r^{\pi'} + \gamma \mathcal{P}^{\pi'} V^{\pi} - V^{\pi} \right] \\
&= (\mathbf{I} - \gamma \mathcal{P}^{\pi'})^{-1} \left[T^{\pi'} V^{\pi} - V^{\pi} \right] \\
&= (\mathbf{I} - \gamma \mathcal{P}^{\pi'})^{-1} A_{\pi}^{\pi'},
\end{aligned}$$

where in the last step we used the definition of the advantage $A_{\pi}^{\pi'}$. This is the desired result. □

The next result implies that the sequence $(V^{\pi_k})_k$ generated by the PI algorithm gets closer to $V^* = V^{\pi^*}$.

Lemma 3.4. *Given V^π, let $\pi' \leftarrow \pi_g(V^\pi)$, i.e., π' is the greedy policy w.r.t. V^π. We have*

$$\left\| V^{\pi^*} - V^{\pi'} \right\|_\infty \le \gamma \left\| V^{\pi^*} - V^{\pi} \right\|_\infty .$$

Proof. Consider $V^{\pi^*} - V^{\pi'}$ and add and subtract $T^{\pi^*} V^{\pi}$ and $T^{\pi'} V^{\pi}$. After benefitting from the fact that $V^{\pi^*} = T^{\pi^*} V^{\pi^*}$ and $V^{\pi'} = T^{\pi'} V^{\pi'}$, we get

$$V^{\pi^*} - V^{\pi'} = T^{\pi^*} V^{\pi^*} - \left(T^{\pi^*} V^{\pi} - T^{\pi^*} V^{\pi} \right) - \left(T^{\pi'} V^{\pi} - T^{\pi'} V^{\pi} \right) - T^{\pi'} V^{\pi'} \tag{3.11}$$

$$= \left(T^{\pi^*} V^{\pi^*} - T^{\pi^*} V^{\pi} \right) + \left(T^{\pi^*} V^{\pi} - T^{\pi'} V^{\pi} \right) + \left(T^{\pi'} V^{\pi} - T^{\pi'} V^{\pi'} \right). \tag{3.12}$$

Let us consider each term within parentheses separately. For the first term, we have

$$\begin{aligned} T^{\pi^*} V^{\pi^*} - T^{\pi^*} V^{\pi} &= \left(r^{\pi^*} + \gamma \mathcal{P}^{\pi^*} V^{\pi^*} \right) - \left(r^{\pi^*} + \gamma \mathcal{P}^{\pi^*} V^{\pi} \right) \\ &= \gamma \mathcal{P}^{\pi^*} \left(V^{\pi^*} - V^{\pi} \right). \end{aligned} \tag{3.13}$$

Likewise, for the last term, we get

$$T^{\pi'} V^{\pi} - T^{\pi'} V^{\pi'} = \gamma \mathcal{P}^{\pi'} \left(V^{\pi} - V^{\pi'} \right). \tag{3.14}$$

We now show that we can upper bound the second term, $T^{\pi^*} V^{\pi} - T^{\pi'} V^{\pi}$. As π' is the greedy policy w.r.t. V^{π}, it satisfies

$$T^{\pi'} V^{\pi} = T^* V^{\pi}. \tag{3.15}$$

Recall that the Bellman optimality operator is the supremum of the Bellman operators over all policies (see (2.10)), i.e,

$$T^* V = \sup_{\pi \in \Pi} T^{\pi} V.$$

This means that for any policy π'', including π^*, we have

$$T^* V^{\pi} \geq T^{\pi''} V^{\pi}.$$

So with the choice of $\pi'' = \pi^*$, and by substituting $T^* V^{\pi}$ with $T^{\pi'} V^{\pi}$ (3.15), we get that

$$T^{\pi'} V^{\pi} - T^{\pi^*} V^{\pi} = T^* V^{\pi} - T^{\pi^*} V^{\pi} \geq 0. \tag{3.16}$$

Plugging (3.13), (3.14), and (3.16) into (3.11), we get

$$V^{\pi^*} - V^{\pi'} \leq \gamma \mathcal{P}^{\pi^*} \left(V^{\pi^*} - V^{\pi} \right) + \gamma \mathcal{P}^{\pi'} \left(V^{\pi} - V^{\pi'} \right).$$

By the Policy Improvement theorem (Theorem 3.1), we know that $V^{\pi} \leq V^{\pi'}$. As $\mathcal{P}^{\pi'} \geq 0$ (element-wise), the value of $\mathcal{P}^{\pi'} \left(V^{\pi'} - V^{\pi} \right) \geq 0$ too. So the last term in the inequality above is non-positive, which means that

$$V^{\pi^*} - V^{\pi'} \leq \gamma \mathcal{P}^{\pi^*} \left(V^{\pi^*} - V^{\pi} \right).$$

As the value of the optimal policy is greater than or equal to the value of π and π', we have $V^{\pi^*} - V^{\pi'} \geq 0$ and $V^{\pi^*} - V^{\pi} \geq 0$. So we can take the norm from both sides without any change in the direction of the inequality. We take the supremum norm, which can be simplified as

$$\left\| V^{\pi^*} - V^{\pi'} \right\|_\infty \le \left\| \gamma \mathcal{P}^{\pi^*} \left(V^{\pi^*} - V^{\pi} \right) \right\| \le \gamma \underbrace{\left\| \mathcal{P}^{\pi^*} \right\|_\infty}_{=1} \left\| V^{\pi^*} - V^{\pi} \right\|_\infty = \gamma \left\| V^{\pi^*} - V^{\pi} \right\|_\infty .$$

□

This lemma shows that the sequence $(\pi_k)_k$ generated by the PI algorithm satisfies $\left\| V^{\pi^*} - V^{\pi_{k+1}} \right\|_\infty \le \gamma \left\| V^{\pi^*} - V^{\pi_k} \right\|_\infty$, hence

$$\left\| V^{\pi^*} - V^{\pi_k} \right\|_\infty \le \gamma^k \left\| V^{\pi^*} - V^{\pi_0} \right\|_\infty , \tag{3.17}$$

that is, the value of the π_k gets closer to the optimal value function with a geometric rate. A corollary of this result is that we can determine the number of iterations k required to get a policy π_k such that $V^{\pi_k} \ge V^{\pi^*} - \varepsilon$, i.e., following it is at most ε worse than following the optimal policy.

Proposition 3.5. *Assume that for all policies π, $\left\| V^\pi \right\|_\infty \le V_{max}$. Given an $\varepsilon > 0$, we need to run the PI algorithm for at most*

$$\left\lceil \frac{\log(\frac{2V_{max}}{\varepsilon})}{\log(\frac{1}{\gamma})} \right\rceil$$

iterations in order to guarantee that $V^{\pi_k} \ge V^{\pi^*} - \varepsilon$.

Proof. To satisfy $V^{\pi_k} \ge V^{\pi^*} - \varepsilon$, it is sufficient to have $\left\| V^{\pi^*} - V^{\pi_k} \right\|_\infty = \varepsilon$. We have

$$\varepsilon = \left\| V^{\pi^*} - V^{\pi_k} \right\|_\infty \le \gamma^k \left\| V^{\pi^*} - V^{\pi_0} \right\|_\infty \le \gamma^k (2V_{\max}),$$

where we evoked Lemma 3.4 (as we did in (3.17)), and benefitted from the fact that all the value functions are $V_{\max}$-bounded. This is satisfied if

$$\frac{\varepsilon}{2V_{\max}} \le \gamma^k .$$

Solving for k, we get that $\log(\frac{\varepsilon}{2V_{\max}}) \le k \log \gamma$, which after some simple manipulations leads to the desired result. □

This result has a dependency on ε, albeit a very mild one (logarithmic). In contrast with Theorem 3.2, it does not show that the PI algorithm ever terminates. The next result provides such a guarantee.

Theorem 3.6 (Convergence of the Policy Iteration Algorithm – Proposition 2.4.1 of Bertsekas 2018)**.** *The PI algorithm terminates after at most*

$$(|\mathcal{A}|-1)|\mathcal{X}|\left\lceil \frac{1}{1-\gamma}\log\left(\frac{1}{1-\gamma}\right)\right\rceil$$

Proof. Consider the sequence (π_k) generated by the PI algorithm. By Lemma 3.3 with the choice of $\pi = \pi^*$ and $\pi' = \pi_k$, we have

$$-A^{\pi_k}_{\pi^*} = (\mathbf{I} - \gamma\mathcal{P}^{\pi_k})(V^{\pi^*} - V^{\pi_k}).$$

Since $\mathcal{P}^{\pi_k} \geq 0$ and because $\|\mathcal{P}^{\pi_k}\|_\infty = 1$, the mapping $(\mathbf{I} - \gamma\mathcal{P}^{\pi_k}) > 0$. Therefore,

$$-A^{\pi_k}_{\pi^*} \leq (\mathbf{I} - \gamma\mathcal{P}^{\pi_k})(V^{\pi^*} - V^{\pi_k}) \leq V^{\pi^*} - V^{\pi_k}. \tag{3.18}$$

We claim that the function $-A^{\pi_k}_{\pi^*}$ is non-negative. To see this, by its definition, we have

$$-A^{\pi_k}_{\pi^*} = V^{\pi^*} - T^{\pi_k}V^{\pi^*} \geq 0,$$

which is true because V^{π^*} is the optimal value function. This is the same as saying that $V^{\pi^*}(x) - Q^{\pi^*}(x, \pi_k(x)) \geq 0$, which is equivalent to the statement that $Q^{\pi^*}(x, \pi^*(x)) \geq Q^{\pi^*}(x, \pi_k(x)))$. If this wasn't true, we could select action according to π_k at the first step, and then follow π^* and get a higher value, which would contradict the optimality of π^*.[3]

Because of the non-negativity of $-A^{\pi_k}_{\pi^*}$, we can take the norm of both sides of (3.18) and keep the same direction of inequality:

$$\|A^{\pi_k}_{\pi^*}\|_\infty = \|-A^{\pi_k}_{\pi^*}\|_\infty \leq \left\|V^{\pi^*} - V^{\pi_k}\right\|_\infty.$$

By Lemma 3.4 (specifically it consequence (3.17)), Lemma 3.3, and Lemma A.2 in Appendix A.4, we have that

$$\begin{aligned}
\|-A^{\pi_k}_{\pi^*}\|_\infty &\leq \gamma^k \left\|V^{\pi^*} - V^{\pi_0}\right\|_\infty \\
&= \gamma^k \left\|(\mathbf{I} - \gamma\mathcal{P}^{\pi_0})^{-1}(-A^{\pi_0}_{\pi^*})\right\|_\infty \\
&\leq \frac{\gamma^k}{1-\gamma}\|(-A^{\pi_0}_{\pi^*})\|_\infty.
\end{aligned} \tag{3.19}$$

[3]A more formal proof: We prove by contradiction. Suppose that we have $V^{\pi^*} < T^{\pi_k}V^{\pi^*}$ instead. We apply T^{π_k} to both sides, and by the monotonicity property of the Bellman operator, we get $T^{\pi_k}V^{\pi^*} < (T^{\pi_k})^2V^{\pi^*}$, which entails that $V^{\pi^*} < (T^{\pi_k})^2V^{\pi^*}$. Repeating this argument leads to the statement that $V^{\pi^*} < (T^{\pi_k})^mV^{\pi^*}$ for any $m = 1, 2, \ldots$. We now let m goes to ∞, and use the contraction property of the Bellman operator T^{π_k} to obtain $V^{\pi^*} < \lim_{m\to\infty}(T^{\pi_k})^mV^{\pi^*} = V^{\pi_k}$. This is a contradiction, as V^{π^*} is the optimal value function and cannot be smaller than V^{π_k}.

As $(-A^{\pi_0}_{\pi^*})$ is non-negative, $\|-A^{\pi_0}_{\pi^*}\|_\infty = \max_{x\in\mathcal{X}}(-A^{\pi_0}_{\pi^*})(x)$. There is at least one state $x_0 \in \mathcal{X}$ such that the maximum is attained, and its value is $-A^{\pi_0}_{\pi^*}(x_0)$. Let us focus on that state.

The value of $(-A^{\pi_k}_{\pi^*})$ at state x_0 (notice that here we have π_k and not π_0) can be upper bounded by its supremum norm:

$$(-A^{\pi_k}_{\pi^*})(x_0) \leq \|(-A^{\pi_k}_{\pi^*})\|_\infty = \|A^{\pi_k}_{\pi^*}\|_\infty$$

By the upper bound (3.19) of $\|A^{\pi_k}_{\pi^*}\|_\infty$ and $\|A^{\pi_0}_{\pi^*}\|_\infty = \max_{x\in\mathcal{X}}(-A^{\pi_0}_{\pi^*})(x)$, we get that

$$(-A^{\pi_k}_{\pi^*})(x_0) \leq \frac{\gamma^k}{1-\gamma}(-A^{\pi_0}_{\pi^*})(x_0). \tag{3.20}$$

Let k^* be an integer such that $\frac{\gamma^{k^*}}{1-\gamma} < 1$. For all $k \geq k^*$ and $A^{\pi_0}_{\pi^*}(x_0) > 0$, we have the strict inequality

$$-A^{\pi_k}_{\pi^*}(x_0) < -A^{\pi_0}_{\pi^*}(x_0).$$

This entails that the action selected by π_k at state x_0 is different from $\pi_0(x_0)$. To see this more clearly, note that by (3.9), this is equivalent to the statement that $Q^*(x_0, \pi_0(x)) < Q^*(x_0, \pi_k(x))$. This can only hold if $\pi_0(x)$ and $\pi_k(x)$ are different.

If $A^{\pi_0}_{\pi^*}(x_0) = 0$, then $Q^*(x_0, \pi_0(x)) = V^*(x_0)$, so π_0 is optimal at x_0. By (3.20) and the non-negativity of $(-A^{\pi_k}_{\pi^*})$, the advantage of π_k w.r.t. π^* $(A^{\pi_k}_{\pi^*})$ must be zero, hence $\pi_k(x_0)$ is optimal too.

This argument shows that as soon as $k \geq k^*$, the policy π_k never chooses $\pi_0(x_0)$ unless it is optimal already. Therefore, at least one suboptimal action is eliminated after k^* iterations of the PI algorithm. We can repeat this argument (starting from the just obtained π_k as the "new" π_0) to eliminate one suboptimal action after each k^* iterations. Because there are $|\mathcal{X}|(|\mathcal{A}|-1)$ suboptimal actions at most, the PI continues for at most $|\mathcal{X}|(|\mathcal{A}|-1)k^*$ iterations, after which the remaining actions are all optimal.

It remains to calculate k^*. We solve for its value such that $\frac{\gamma^{k^*}}{1-\gamma} = 1$. This leads to

$$k^* \log \gamma = \log(1-\gamma) \Rightarrow k^* = \frac{\log(1-\gamma)}{\log(\gamma)} = \frac{\log(\frac{1}{1-\gamma})}{\log(\frac{1}{\gamma})}.$$

As $\log(\frac{1}{\gamma}) \geq 1-\gamma$, if we choose $k^* = \lceil \frac{\log(\frac{1}{1-\gamma})}{1-\gamma} \rceil$, we get that $\frac{\gamma^{k^*}}{1-\gamma} < 1$. □

3.5 Linear Programming

We can find V^* by solving a Linear Program (LP). This approach is less commonly used compared to VI and PI. Later, we mention an approach to find π^* without computing V^* or Q^* after all, based on the dual formulation of LP.

To begin with, consider the set of all V that satisfy $V \geq T^*V$, i.e.,

$$C = \{ V : V \geq T^*V \} .$$

This set has an interesting property. For any $V \in C$, by the monotonicity of T^*, we have

$$V \geq T^*V \Rightarrow T^*V \geq T^*(T^*V) = (T^*)^2V.$$

Repeating this argument, we get that for any $m \geq 1$,

$$V \geq (T^*)^m V.$$

We take the limit of m going to infinity and use the contraction property of the Bellman optimality operator to conclude

$$V \geq \lim_{m \to \infty} (T^*)^m V = V^*.$$

So any $V \in C$ is a lower bounded by V^*. Or in other words, V^* is the function that is in C, but is less than or equal to any other function in C in the pointwise sense, i.e., $V_1 \leq V_2 \Leftrightarrow V_1(x) \leq V_2(x), \forall x \in \mathcal{X}$.

Based on this observation, we can devise a procedure to find V^*. We find a member of C such that it is less than or equal to any other $V \in C$. We can do this by choosing a strictly positive vector $\mu > 0$ with the dimension of $\mathcal{X}$ (we can think of μ as the probability distribution with a support on $\mathcal{X}$; though at this stage being a distribution is not needed), and solving the following constrained optimization problem:

$$\min_{V \in C} \mu^\top V,$$

which can be written in an expanded form as

$$\begin{array}{ll} \min_V & \mu^\top V, \\ \text{s.t.} & V(x) \geq (T^*V)(x), \qquad \forall x \in \mathcal{X}. \end{array}$$

This has a linear objective and a set of $|\mathcal{X}|$ nonlinear constraints. We can convert each of nonlinear constraints to $|\mathcal{A}|$ linear ones because each

$$V(x) \geq \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \sum_y \mathcal{P}(y|x,a)V(y) \right\}$$

is equivalent to

$$V(x) \geq r(x,a) + \gamma \sum_y \mathcal{P}(y|x,a)V(y), \qquad \forall a \in \mathcal{A}.$$

Therefore, we can solve the following instead:

$$\begin{aligned} \min\nolimits_V \quad & \mu^\top V, \\ \text{s.t.} \quad & V(x) \geq r(x,a) + \gamma \sum_y \mathcal{P}(y|x,a)V(y), \qquad \forall (x,a) \in \mathcal{X} \times \mathcal{A}. \end{aligned}$$

This is a linear program with $|\mathcal{X} \times \mathcal{A}|$ constraints.

The choice of μ, as long as $\mu > 0$, does not matter. To see this, suppose that we find a $\tilde{V} \neq V^*$ as the minimizer. As all $V \in C$ satisfy $V \geq V^*$, this means that for at least a state x', we have that $\tilde{V}(x') > V^*(x')$. But if that is the case, we can decrease the objective from $\mu^\top \tilde{V}$ to $\mu^\top V^*$ by the amount of

$$\underbrace{\mu(x')}_{>0}(\underbrace{\tilde{V}(x') - V^*(x')}_{>0}) > 0.$$

So $\tilde{V}$ cannot be strictly larger than V^*. If $\mu(x') = 0$, however, we could not make this argument anymore.

Exercise 3.9. *What method do we have for solving an LP? What is the computational cost?*

Exercise 3.10. *How can we have an LP-like formulation when the state space is continuous?*

Exercise 3.11. *What if we started our argument from $V \leq T^*V$ and defined $C' = \{ V : V \leq T^*V \}$ instead of C? Would it work? If so, what changes do we need?*

Chapter 4

Learning from a Stream of Data: Value Function Learning

We consider the setting when the MDP model ($\mathcal{P}$ and $\mathcal{R}$) is known in the previous chapter.[1] In the RL setting, however, we do not have access to the model. Instead, we observe data of agent interacting with its environment. The data, in general, is in the form of *data stream*

$$X_1, A_1, R_1, X_2, A_2, R_2,$$

with $A_t \sim \pi(\cdot|X_t)$, $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$ and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$, as we already described in Sections 1.1 and 1.3.

Several important questions are:

- How can we learn a value of policy π?
- How can we learn V^* or Q^* (and consequently, the optimal policy π^*)?

This chapter is about methods for doing these tasks. The methods in this chapter are often feasible for finite MDPs. Similar high-level ideas work for continuous MDPs, with some modifications, which we shall cover later in the lecture notes.

4.1 Online Estimation of the Mean of a Random Variable

Let us start from a simple problem of estimating the mean of a random variable, given samples from it. To be concrete, assume that we are given n real-valued

[1]Chapter's Version: 0.05 (2021 February 23).

r.v. $Z_1, \dots, Z_t$, all drawn i.i.d. from a distribution ν. How can we estimate the expectation $m = \mathbb{E}[Z]$ with $Z \sim \nu$?

Of course, we can use the sample (or empirical) average:

$$m_t \triangleq \frac{1}{t} \sum_{i=1}^{t} Z_i.$$

We know that under mild conditions, by the Law of Large Numbers (LLN), $m_t \to m$, almost surely.[2]

Exercise 4.1 ($\star\star$). *Assume that* $\mathrm{Var}[Z] = \sigma^2$. *What is the variance of m_t?*

Hint: Use the independence assumption between Z_i and Z_j (for $i \neq j$) in your calculation of the variance.

The naive implementation of m_t requires us to store all $Z_1, \dots, Z_t$. This is infeasible when t is large. But we can do it online too. To see it, let us write m_{t+1} in terms of m_t and Z_{t+1}:

$$\begin{aligned} m_{t+1} &= \frac{1}{t+1} \sum_{i=1}^{t+1} Z_i = \frac{1}{t+1} \left[\sum_{i=1}^{t} Z_i + Z_{t+1} \right] = \frac{1}{t+1} \left[t m_t + Z_{t+1} \right] \\ &= \left(1 - \frac{1}{t+1} \right) m_t + \frac{1}{t+1} Z_{t+1}. \end{aligned}$$

Let us define $\alpha_t = \frac{1}{t+1}$. We can write

$$m_{t+1} = (1 - \alpha_t) m_t + \alpha_t Z_t.$$

The variable α_t is called the learning rate or step size.

With this choice of α_t, the estimate m_t converges to m as $t \to \infty$, as this is basically computing the empirical mean, whose convergence is ensured by the LLN. This online procedure is an example of the family of *stochastic approximation* (SA) methods. We shall see more example of it in the rest of this chapter.

We can also choose other α_ts too. In order to avoid confusion with m_t, we use θ_t as our estimate of $m = \mathbb{E}[Z]$, and consider an algorithm in the form of

$$\theta_{t+1} = (1 - \alpha_t) \theta_t + \alpha_t Z_t. \tag{4.1}$$

[2] The condition would be that $\mathbb{E}[|Z|] < \infty$.

Note that θ_t is a random variable. If $\alpha_t = \frac{1}{t+1}$, we get the previous procedure, and we know that $\theta_t \to \mathbb{E}[Z]$ and has a variance that goes to zero (Exercise 4.1). As another choice, we consider a fixed learning rate

$$\alpha_t = \alpha,$$

for a $\alpha > 0$. In that case, the sequence θ_t would be

$$\theta_{t+1} = (1-\alpha)\theta_t + \alpha Z_t.$$

Let us try to understand this sequence better by studying its expectation and variance as a function of time t. Take expectation of both sides to get

$$\begin{aligned}\mathbb{E}[\theta_{t+1}] &= \mathbb{E}[(1-\alpha)\theta_t + \alpha Z_t] \\ &= (1-\alpha)\mathbb{E}[\theta_t] + \alpha\mathbb{E}[Z_t] \\ &= (1-\alpha)\mathbb{E}[\theta_t] + \alpha m.\end{aligned}$$

Denote $\mathbb{E}[\theta_t]$ by $\bar{\theta}_t$ (which is not a r.v. anymore), and write the equation above as

$$\bar{\theta}_{t+1} = (1-\alpha)\bar{\theta}_t + \alpha m.$$

We would like to study the behaviour of $\bar{\theta}_t$ as t increases. Assuming that $\theta_0 = 0$ (so $\bar{\theta}_0 = 0$) and $0 \leq \alpha < 1$, we get that

$$\begin{aligned}&\bar{\theta}_1 = \alpha m, \\ &\bar{\theta}_2 = (1-\alpha)\alpha m + \alpha m, \\ &\bar{\theta}_3 = (1-\alpha)^2\alpha m + (1-\alpha)\alpha m + \alpha m, \\ &\qquad\vdots \\ &\bar{\theta}_t = \alpha\sum_{i=0}^{t-1}(1-\alpha)^i m = \frac{\alpha m(1-(1-\alpha)^t)}{1-(1-\alpha)} = m\left[1-(1-\alpha)^t\right].\end{aligned}$$

Therefore, we can conclude that

$$\lim_{t\to\infty} \bar{\theta}_t = m.$$

So the update rule leads to a sequence $\bar{\theta}_t$ that converges to m in expectation. This is reassuring, but is not enough. It is imaginable that θ_t converges in expectation, but has a large deviation around its mean. Let us compute its variance too.

As Z_t is independent of $Z_1, \dots, Z_{t-1}$ and θ_t is a function of $Z_1, \dots, Z_{t-1}$ only, the random variables θ_t and Z_t are independent too. We use this to get that

$$\mathrm{Var}\left[\theta_{t+1}\right] = \mathrm{Var}\left[(1-\alpha)\theta_t + \alpha Z_t\right] = (1-\alpha)^2 \mathrm{Var}\left[\theta_t\right] + \alpha^2 \mathrm{Var}\left[Z_t\right].$$

As a quick calculation, we have that $\mathrm{Var}\left[\theta_{t+1}\right] \geq \alpha^2 \mathrm{Var}\left[Z_t\right] = \alpha^2\sigma^2$. So for a constant α, the variance of θ_t is not going to converge to zero. In other words, θ_t fluctuates around its mean (in different runs of the data stream; though a similar conclusion would hold within the same sequence (θ_t) too).

To compute the variance exactly, denote $\beta = (1-\alpha)^2$ and $U_t = \mathrm{Var}\left[\theta_t\right]$. Similar to the calculation for the expectation, we have that

$$\begin{aligned}
U_0 &= 0, \\
U_1 &= \alpha^2\sigma^2, \\
U_2 &= \alpha^2\sigma^2(1+\beta) \\
&\vdots \\
U_t &= \alpha^2\sigma^2 \sum_{i=0}^{t-1} \beta^i = \frac{\alpha^2\sigma^2(1-\beta^t)}{1-\beta} = \frac{\alpha\sigma^2\left[1-(1-\alpha)^{2t}\right]}{2-\alpha}.
\end{aligned}$$

So

$$\lim_{t\to\infty} \mathrm{Var}\left[\theta_t\right] = \frac{\alpha\sigma^2}{2-\alpha}. \tag{4.2}$$

To summarize, if we choose $\alpha_t = \frac{1}{t+1}$, the estimate θ_t converges to m almost surely. In finite range of t, the variance of the estimate is $\frac{\sigma^2}{t}$, so it is decreasing as a function of t. On the other hand, if $\alpha_t = \alpha$, the estimate θ_t converges to m only in expectation, but its variance is not going to zero as t grows.

In order to make θ_t actually converge in a sense stronger than expectation, we need $\alpha_t \to 0$ with some schedule. Obviously, $\alpha_t = \frac{1}{t+1}$ works, but it is not the only acceptable one. But any sequence α_t going to zero is not working either. It should not converge to zero too fast, as it would not allow enough adaptation. For example, if $\alpha_t = 0$ for all $t = 1, 2, \dots$, θ_t is not updated at all. Even if α_t becomes zero only after a certain $t_0 > 1$, we would not converge to the expectation.

One can show that the condition for convergence is that the learning rate (or step size) (α_t) should satisfy:

$$\sum_{t=0}^{\infty} \alpha_t = \infty, \tag{4.3}$$

$$\sum_{t=0}^{\infty} \alpha_t^2 < \infty. \tag{4.4}$$

Exercise 4.2 ($\star$). *Does $\alpha_t = \alpha$ (constant) satisfy these conditions?*

Exercise 4.3 ($\star$). *Verify that the sequence $\alpha_t = \frac{1}{t+1}$ satisfies these conditions.*

Exercise 4.4 ($\star$). *Let $\alpha_t = \frac{1}{t^p+1}$. For what range of p these conditions are satisfied?*

4.2 Online Learning of the Reward Function

Recall the immediate reward problem from Section 1.3: At episode t, the agent starts at state $X_t \sim \rho \in \mathcal{M}(\mathcal{X})$, chooses action $A_t \sim \pi(\cdot|X_t)$, and receives a reward of $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$. The agent then starts a new independent episode $t+1$, and the process repeats. The goal is to learn how to act optimally.

When the reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ was known, the optimal policy would be (1.2)

$$\pi^*(x) \leftarrow \operatorname*{argmax}_{a \in \mathcal{A}} r(x, a).$$

What if when we do not know the reward function? In this section, we study this problem in some detail.

We can use SA to estimate $r(x, a)$. This would simply be the extension of how we estimated the mean of a single variable $Z \sim \nu$ to the case when we have many random variables, each for one of the state-action pairs $(x, a) \in \mathcal{X} \times \mathcal{A}$. Each r.v. has a distribution $\mathcal{R}(\cdot|x, a)$ and its mean is $r(x, a) = \mathbb{E}[R|X = x, A = a]$ with $R \sim \mathcal{R}(\cdot|x, a)$ (for all $(x, a) \in \mathcal{X} \times \mathcal{A}$).

Denote $\hat{r}_t : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ as our estimate of r at time t. Let us denote the state-action-indexed sequence $\alpha_t(x, a)$ as the step size for (x, a). At time/episode t, the state-action pair (X_t, A_t) is selected. We update $\hat{r}_t(X_t, A_t)$ as

$$\hat{r}_{t+1}(X_t, A_t) \leftarrow (1 - \alpha_t(X_t, A_t))\hat{r}_t(X_t, A_t) + \alpha_t(X_t, A_t)R_t, \tag{4.5}$$

and do not change our estimate $\hat{r}_{t+1}(x, a)$ from what we had $\hat{r}_t(x, a)$ for all $(x, a) \neq (X_t, A_t)$. This can be written as having $\alpha_t(x, a) = 0$, i.e.,

$$\hat{r}_{t+1}(x, a) \leftarrow (1 - 0)\hat{r}_t(x, a) + 0.R_t,$$

The SA conditions (4.3)-(4.4) apply here, with the difference that it should for each state-action pair, i.e., for any $(x, a) \in \mathcal{X} \times \mathcal{A}$, we need to have

$$\sum_{t=0}^{\infty} \alpha_t(x, a) = \infty,$$
$$\sum_{t=0}^{\infty} \alpha_t^2(x, a) < \infty.$$

To define $\alpha_t(x, a)$, we can use a counter on how many times (x, a) has been picked up to time t. We define

$$n_t(x, a) \triangleq |\{ i : (X_i, A_i) = (x, a), i = 1, \dots, t \}| .$$

We can then choose $\alpha_t(x, a) = \frac{1}{n_t(x,a)}$. This leads to $\hat{r}_t(x, a)$ being a sample mean of all rewards encountered at (x, a).

Note that if the sampling distribution $X_t \sim \rho$ never chooses a particular state x_0 or if the policy $\pi(\cdot|x_0)$ never chooses a particular action a_0 at a certain state x_0, we cannot form any data-dependent estimate of $r(x, a)$. This is in line with what condition $\sum_{t=0}^{\infty} \alpha_t(x_0, a_0) = \infty$ implies. For this condition to be satisfied, the minimum requirement is that the state-action pair (x_0, a_0) is selected infinitely often (if we only select it a finite number of times, the summation would be finite too). Without having infinite number of samples from all state-actions pairs, our estimate $\hat{r}$ would not converge to r.

Is this important? If the goal is to have an accurate estimate of r, the answer is positive. We shall see that we may not care about having an accurate estimate of r, but we only care about choosing the optimal action. We shall see that they are not the same goals.

4.2.1 From Reward Estimation to Action Selection

Recall that by selecting $a \leftarrow \pi_g(x; r) = \text{argmax}_{a \in \mathcal{A}} r(x, a)$, we would choose the optimal action at state x. In lieu of r, we can use $\hat{r}_t : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, estimated using the SA (4.5), and choose the action $A_t = \pi_g(X_t; \hat{r}_t)$ at state X_t.

There is a problem with this approach though. The problem is that if $\hat{r}_t$ is inaccurate estimate of r, the agent may choose a suboptimal action. It is also possible that it gets stuck in choosing that action forever, without any chance to improve its estimate. To see how this might happen, consider a problem where we only have one state x_1 with two actions a_1 and a_2. The reward function is

$$\begin{aligned} r(x_1, a_1) &= 1, \\ r(x_1, a_2) &= 2. \end{aligned}$$

Suppose that the reward is deterministic, so whenever the agent chooses action a_1 at state x_1, it always receive $R = 1$ (and similar in the other case). Suppose that the initial estimate of the reward $\hat{r}(x_1, \cdot) = 0$.

Assume that in the first episode $t = 1$, the agent happen to choose a_1. So its estimates would be

$$\begin{aligned} \hat{r}_2(x_1, a_1) &= (1 - \alpha_1) \times 0 + \alpha_1 \times 1 > 0 \\ \hat{r}_2(x_1, a_2) &= \hat{r}_1(x_1, a_2) = 0. \end{aligned}$$

The next time the agent encounters x_1, if it follows the greedy policy, the selected action would be a_1 again, and $\hat{r}_3(x_1, a_1)$ remains positive. Meanwhile, since a_2 is not selected, the value of $\hat{r}_3(x_1, a_2)$ remains equal to zero. As long as the agent follows the greedy policy, it always chooses action a_1 and never chooses action a_2. This means that the estimate $\hat{r}_t(x_1, a_1)$ becomes ever more accurate (and asymptotically converge to $r(x_1, a_1)$, if the learning rate is selected properly), but $\hat{r}_2(x_1, a_2)$ remains inaccurate. Of course, this is problematic as the optimal action here is a_2.

How can we ensure that the agent learns to eventually choose action a_2? Let us answer a slightly different question: How can we ensure that the agent learns a good estimate of r for all state-action pairs (x, a)? If we have an estimate $\hat{r}$ that is very accurate for all $(x, a) \in \mathcal{X} \times \mathcal{A}$, we can use it instead of r (this is a sufficient condition though).

One solution is to force the agent regularly picks actions other than the one suggested by the greedy policy. If we ensure that all actions are selected infinitely often, the estimate $\hat{r}$ converges to r. Using the ε-greedy policy, which we previously encountered in (1.17), is a possible approach: For $\varepsilon \geq 0$ and a function $\hat{r}$, we define π_ε as

$$\pi_\varepsilon(x; \hat{r}) = \begin{cases} \pi_g(x; \hat{r}) & \text{w.p. } 1 - \varepsilon, \\ \text{Unif}(\mathcal{A}) & \text{w.p. } \varepsilon. \end{cases}$$

Here Unif($\mathcal{A}$) chooses an action from $\mathcal{A}$ uniformly.

If $\varepsilon > 0$, there is non-zero probability of selecting any of the actions, so asymptotically all of them are selected infinitely often. If α_t is selected properly, the SA conditions are satisfied, hence $\hat{r} \to r$ uniformly.

The uniform choice of action in the ε-greedy helps the agent *explore* all actions, even if the action is seemingly suboptimal. This can be contrasted with the greedy part of its action select mechanism that *exploits* the current knowledge about the reward function, and chooses the action that has the highest estimated reward. Exploiting our knowledge, encoded by $\hat{r}$ in this context, is a reasonable choice when our knowledge about the world is accurate. If $\hat{r}$ is exactly equal to r, the optimal decision is to always exploit it and choose the greedy action. When we have uncertainty about the world, however, we should not be overconfident of our knowledge and exploit it all the time, but instead explore other available actions, which might happen to be better.

Exploration-Exploitation Trade-off

The tradeoff between exploration and exploitation is a major topic in RL and is an area of active research. The ε-greedy action selection mechanism is a simple heuristic to balance between them, but it is not the only one, or the optimal one. Another heuristic is to select actions according to the Boltzmann (or Gibbs or softmax) distribution. Given a parameter $\tau > 0$, and the reward function $\hat{r}$, the probability of selecting action a at state x is

$$\pi(a|x;\hat{r}) = \frac{\exp(\frac{\hat{r}(x,a)}{\tau})}{\sum_{a' \in \mathcal{A}} \exp(\frac{\hat{r}(x,a')}{\tau})}.$$

This is for finite action spaces. For continuous action spaces, we replace the summation with an integration.

This distribution assigns more weight to actions with higher estimated value (i.e., reward). When $\tau \to \infty$, the behaviour of this distribution would be the same as the greedy policy, both choose the maximizing action. On the other hand, when $\tau \to 0$, the probability of all actions would be the same, i.e., the distribution becomes close to a uniform distribution.

We discuss how we can tradeoff the exploration and exploitation in a more systematic way in a later chapter.

4.3 Monte Carlo Estimation for Policy Evaluation

The reward learning problem is a special case of value function learning problem when the episode ends in one time step. Let us devise methods to learn (or estimate) the value function V^π and Q^π of a policy, and then the optimal value functions V^* and Q^*. In this section, we focus on the policy evaluation problem. We introduce a technique called Monte Carlo (MC) estimation.

Recall from Chapter 1 that

$$V^\pi(x) = \mathbb{E}\left[G_t^\pi | X_t = x\right],$$

with $G_t^\pi \triangleq \sum_{k \geq t} \gamma^{k-t} R_k$ (1.13). That is, the value function is the conditional expectation of the return. So G_t^π (conditioned on starting from $X_t = x$) plays the same rule as the r.v. Z in estimating $m = \mathbb{E}[Z]$ (Section 4.1), or the reward $R \sim \mathcal{R}(\cdot|x, a)$ in estimating $r(x, a)$ (Section 4.2).

Obtaining a sample from return G^π is easy, at least conceptually. If the agent starts at state x, and follows π, we can draw one sample of r.v. G^π by computing the cumulative average of rewards collected during the episode.[3] Each trajectory is sometimes called a *rollout*. If we repeat this process from the same state, we get another draw of r.v. G^π. Let us call the value of these samples $G^{\pi(1)}(x), G^{\pi(2)}(x), \ldots, G^{\pi(n)}(x)$. We can get an estimate $\hat{V}(x)$ of $V^\pi(x)$ by taking the sample average:

$$\hat{V}^\pi(x) = \frac{1}{n} \sum_{i=1}^{n} G^{\pi(i)}(x).$$

This procedure gives an estimate for a single state x. We can repeat it for all states $x \in \mathcal{X}$ to estimate $\hat{V}^\pi$. If $n \to \infty$, the estimate converges to V^π. In the finite sample regime, the behaviour of the estimate is $\hat{V}^\pi(x) \approx V^\pi(x) + O_P(\frac{1}{\sqrt{n}})$.[4] We can also use a SA update to compute these values. We show a variation of this idea when the initial state X_1 at episode i is selected randomly according to a distribution $\rho \in \mathcal{M}(\mathcal{X})$ in Algorithm 4.1. In this variation, the quality of $\hat{V}_t^\pi(x)$ depends on how often x is samples in the first t time steps.

For the SA to converge, we need to choose the learning rate $(\alpha_t(x))_t$ such that it satisfies the SA conditions, i.e., for all $x \in \mathcal{X}$,

$$\sum_{t=0}^{\infty} \alpha_t(x) = \infty, \qquad \sum_{t=0}^{\infty} \alpha_t^2(x) < \infty. \tag{4.6}$$

[3] If the problem is a continuing task, the episode never ends. So we cannot compute the return this way. That is one of the reasons that I used "conceptual". We can ignore this problem though, as there are ways to either approximate it (see Exercise 3.2) or obtain an unbiased estimate of it. We ignore these issues for now, and assume that the episode terminates in a finite number of time steps.

[4] O_P indicates that this holds with certain probability. To be more accurate, we can show that there exists a $c_1 > 0$ such that for any $0 < \delta < 1$, we have that $|\hat{V}^\pi(x) - V^\pi(x)| \leq c_1 \sqrt{\frac{\log(1/\delta)}{n}}$ with probability at least $1 - \delta$.

Algorithm 4.1 Monte Carlo Estimation (Policy Evaluation) (Initial-State Only)

```
Require: Step size schedule $(\alpha_t(x))_{t \geq 1}$ for all $x \in \mathcal{X}$.
1: Initialize $\hat{V}_1^\pi : \mathcal{X} \to \mathbb{R}$ arbitrary, e.g., $\hat{V}_1^\pi = 0$.
2: for each episode $t$ do
3:     Initialize $X_1^{(t)} \sim \rho$
4:     for each step $t$ of episode do
5:         Follow $\pi$ to obtain $X_1^{(t)}, A_1^{(t)}, R_1^{(t)}, X_2^{(t)}, A_2^{(t)}, R_2^{(t)}, \ldots$.
6:     end for
7:     Compute $G_1^{\pi(t)} = \sum_{k \geq 1} \gamma^{k-1} R_k^{(t)}$.
8:     Update
           $\hat{V}_{t+1}^\pi(X_1^{(t)}) \leftarrow \left(1 - \alpha_t(X_1^{(t)})\right) \hat{V}_t^\pi(X_1^{(t)}) + \alpha_t(X_1^{(t)}) G_1^{\pi(t)}$.
9: end for
```

For example, if we set a counter

$$n_t(x) \triangleq \left| \left\{ X_1^{(i)} = x \; : \; 1 \leq i \leq t \right\} \right|,$$

we can define $\alpha_t(x) = \frac{1}{n_t(x)}$.

The procedure of following π, collecting the rewards, and estimating the value function using the return is called the Monte Carlo (MC) estimation. Here it is used to estimate (or predict) the value of a policy π. As we shall see, we can use the same idea to find the optimal value function too.

A few remarks are in order.

First, we need each initial state x to be selected infinitely often. As discussed in Section 4.2, if $\rho(x) = 0$, we cannot form an estimate for that state, hence $\hat{V}^\pi(x)$ would be inaccurate.

The error $\varepsilon_t(x) = |\hat{V}_t^\pi(x) - V^\pi(x)|$ depends on how many times x has been the initial state by episode t, i.e., $n_t(x)$. For $\alpha_t(x) = \frac{1}{n_t(x)}$ (the mean estimator), we can say that $\varepsilon_i(x) = O_p(\frac{1}{\sqrt{n_t(x)}})$. But note that $n_t(x)$ is a r.v. itself, so it is not a deterministic function of t. Nonetheless, we can say that $n_t(x)$ is concentrated around $t\rho(x)$ with a variation in the order of $\sqrt{t\rho(x)}$.

This version of MC is called *initial-state-only* MC, because we only update the estimation of the value function at the initial state. We shall soon describe two different variations of MC.

Exercise 4.5 ($\star$). *Describe an MC algorithm to estimate Q^π.*

4.3.1 First-Visit and Every-Visit Monte Carlo Estimators

Given a trajectory $X_1^{(t)}, A_1^{(t)}, R_1^{(t)}, X_2^{(t)}, A_2^{(t)}, R_2^{(t)}, \dots, X_m^{(t)}, A_m^{(t)}, R_m^{(t)}, X_m^{(t)}, \dots$, we can compute not only the return $G_1^{\pi\,(t)}$, but also all other returns $G_m^{\pi\,(t)}$ from time step $m = 1, 2, \dots$ (all within the same episode). Each of them would be an unbiased estimate of $V^\pi(X_m^{(t)})$. We can then update not only $\hat{V}^\pi(X_1^{(t)})$, but all $\hat{V}^\pi(X_m^{(t)})$ $(m = 1, 2, \dots)$.

We have to be careful here though. If a trajectory visits a particular states twice (or more) at time steps $m_1, m_2, \dots$ (all within the same episode), the returns ${G_{m_1}^\pi}^{(t)}, {G_{m_2}^\pi}^{(t)}, \dots$ would be *dependent*. To see this, consider that we start from state x at $m = 1$, return to the same x at $m = 2$, and then go to other states afterwards. The returns are

$$
\begin{aligned}
G_1^\pi &= R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots , \\
G_2^\pi &= R_2 + \gamma R_3 + \cdots .
\end{aligned}
$$

We can see that $G_1^\pi = R_1 + \gamma G_2^\pi$. So G_1^π is dependent on G_2^π. Therefore, G_1^π and G_2^π are not two independent samples from the return of following π. With dependent samples, we may lose some of the nice properties of having independent samples. For example, the answer to Exercise 4.1 would be different, and the variance might decrease, as a function of n, slower than when we have independent samples. How the behaviour exactly change depends on how dependent the samples are. It can also be shown that these samples might be biased too.

Being biased does not necessarily mean that the approach is useless. In fact, it can be shown that the estimate is consistent, i.e., the sample mean converges to the true mean, despite dependence and biasedness.

One approach to avoid this issue is to only consider the first visit to each state in updateing the estimation of $\hat{V}^\pi$. That means that if wet to a state x at $m_1(x), m_2(x), \dots$, we update $\hat{V}^\pi(x)$ only based on $G_{m_1(x)}^\pi$, and not $G_{m_j(x)}^\pi$ $(j \geq 1)$. This estimate is unbiased. And there is no issue of dependency either. This variant is called *first-visit* MC, and the former one is called *every-visit*.

Can we say the first-visit MC is a better estimator? Not necessarily! It might be possible that the mean-squared error of every-visit is smaller in some situations, despite the bias of the samples. Let us see an example. XXX This will be added! XXX

4.4 Temporal Difference Learning for Policy Evaluation

MC methods allows us to estimate $V^\pi(x)$ by using returns $G^\pi(x)$. Depending on the method, the samples are either unbiased (initial-state only and first-visit variants) or biased but consistent (every-visit). MC methods, however, do not benefit from the recursive property of the value function, which is codified with the Bellman equation. The MC methods are agnostic to the MDP structure. This can be an advantage if the underlying problem is not an MDP. But if it is, an MC method might be less efficient.

In Chapter 3, we discussed several methods that could be used to compute V^π and V^*. These methods benefitted from the structure of the MDP. Can we use similar methods, even if we do not know $\mathcal{P}$ and $\mathcal{R}$?

Let us focus on VI for PE, i.e., $V_{k+1} \leftarrow T^\pi V_k$. At state x, the procedure is

$$V_{k+1}(x) \leftarrow r^\pi(x) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a|x)V_k(x').$$

If we do not know r^π and $\mathcal{P}$, we cannot compute this. Suppose that we have n samples $A_i \sim \pi(\cdot|x)$, $X'_i \sim \mathcal{P}(\cdot|x, A_i)$, and $R_i \sim \mathcal{R}(\cdot|x, A_i)$. Using these samples and V_k, we compute

$$Y_i = R_i + \gamma V_k(X'_i). \tag{4.7}$$

Now notice that

$$\mathbb{E}\left[R_i|X = x\right] = r^\pi(x),$$

and

$$\mathbb{E}\left[V_k(X'_i)|X = x\right] = \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a|x)V_k(x').$$

So the r.v. Y_i satisfies

$$\mathbb{E}\left[Y_i|X = x\right] = (T^\pi V_k)(x). \tag{4.8}$$

This means that Y_i is an unbiased sample from the effect of T^π on V_k, evaluated at x.

This should remind us of the problem of estimating $m = \mathbb{E}\left[Z\right]$ using samples $Z_1, Z_2, \ldots$ in Section 4.1. The value of

$$V_{k+1}(x) \triangleq \frac{1}{n}\sum_{i=1}^{n} Y_i$$

Algorithm 4.2 Temporal Difference Learning (Synchronous)

```
Require: Policy $\pi$, step size schedule $(\alpha_k)_{k \geq 1}$.
1: Initialize $V_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, e.g., $V_1(x) = 0$.
2: for iteration $k = 1, 2, \ldots$ do
3:     for each state $x \in \mathcal{X}$ do
4:         Let $A \sim \pi(\cdot|x)$
5:         $X'(x) \sim \mathcal{P}(\cdot|X, A)$ and $R(x) \sim \mathcal{R}(\cdot|x, A)$
6:         Let $(\hat{T}^\pi V_k)(x) \triangleq R(x) + \gamma V_k(X'(x))$
7:     end for
8:     Update
                 $V_{k+1} \leftarrow (1 - \alpha_k) V_k + \alpha_k \hat{T}^\pi V_k$
9: end for
```

converges to $(T^\pi V_k)(x)$ by LLN. And as we have seen, the rate of convergence is $O_P(\frac{1}{\sqrt{m}})$. This is a sample-based version of VI for PE.

Instead of taking the sample average, we can use a SA procedure and update it as

$$V_{k+1,j+1}(x) = (1 - \alpha_j(x))\, V_{k+1,j}(x) + \alpha_j(x) Y_j, \qquad j = 1, 2 \ldots .$$

If we perform this process for all states $x \in \mathcal{X}$ simultaneously, we get an estimate of $T^\pi V_k$, which is contaminated by a zero mean noise. This is summarized in Algorithm 4.2. Note that we defined

$$(\hat{T}^\pi V_k)(x) \triangleq R(x) + \gamma V_k(X'(x)), \tag{4.9}$$

which is the same as Y_i defined above (4.7). We call $\hat{T}^\pi$ the *empirical Bellman operator*. As we shown before (4.8), this r.v. is an unbiased estimate of $(T^\pi V_k)(x)$, and we have

$$\mathbb{E}\left[(\hat{T}^\pi V_k)(x) | X = x\right] = (T^\pi V_k)(x).$$

For a moment assume that we run a VI-like procedure

$$V_{k+1} \leftarrow \hat{T}^\pi V_k,$$

instead of $V_{k+1} \leftarrow T^\pi V_k$ (this corresponds to the choice of $\alpha_k = 0$ in the algorithm above). We can write it down as

$$V_{k+1} \leftarrow \hat{T}^\pi V_k = T^\pi V_k + \underbrace{\left(\hat{T}^\pi V_k - T^\pi V_k\right)}_{\triangleq \varepsilon_k}. \tag{4.10}$$

This is similar to the usual VI with an additional zero-mean noise ε_k. This additional noise, however, does not go zero with this procedure. To see this more clearly, assume that V_k is already the correct value V^π. In this case, the VI stays at the same value, as $V_{k+1} = T^\pi V_k = T^\pi V^\pi = V^\pi$. But for the VI based on the empirical Bellman error, we have

$$V_{k+1} = T^\pi V_k + \varepsilon_k = T^\pi V^\pi + \varepsilon_k = V^\pi + \varepsilon_k.$$

We can continue this to see that this would fluctuate around the true value a non-diminishing fluctuation.

Comparing this with the estimation of the mean using a noisy sample might be instructive. Recall from (4.1) that

$$\begin{aligned} &\theta_{t+1} \leftarrow (1-\alpha_t)\theta_t + \alpha_t Z_t \\ \Leftrightarrow &\theta_{t+1} \leftarrow (1-\alpha_t)\theta_t + \alpha_t m + \alpha_k (Z_t - m) \\ \Leftrightarrow &\theta_{t+1} \leftarrow (1-\alpha_t)\theta_t + \alpha_t m + \alpha_t \varepsilon_t, \end{aligned}$$

with $\varepsilon_t = Z_t - m$ being a zero-mean noise term. When $\alpha_t = 1$, we get a procedure similar to (4.10). We discussed earlier that for θ_t to converge to m, we need α_t to go to zero with a certain rate. With any fixed α, including $\alpha = 1$, the variance (4.2) of the mean estimator does not go to zero.

The problem of finding the fixed point of T^π is not exactly the same as the problem of finding the mean of a r.v., but the condition for convergence is similar. We need the usual SA conditions to be satisfied. For Algorithm 4.2, we need

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \qquad \sum_{k=0}^{\infty} \alpha_k < \infty.$$

The preceding procedure was *synchronously* updating the value of all states. We can also only update one $V(x)$ at any time step, i.e., *asynchronous* update. That is, given a state transition (X, A, R, X'), we update

$$V(X) \leftarrow (1-\alpha)V(X) + \alpha(\hat{T}^\pi V)(X) = V(X) + \alpha[R + \gamma V(X') - V(X)].$$

We have $\mathbb{E}\left[R + \gamma V(X') - V(X) | X\right] = (T^\pi V)(X) - V(X)$, so in expectation $V(X)$ is updated to

$$V(X) + \alpha[(T^\pi V)(X) - V(X)] = (1-\alpha)V(X) + \alpha(T^\pi V)(X).$$

This is the same direction suggested by VI, but we only move the current estimate only proportional to α towards it. Moreover, the update is only at one of the states X,

Algorithm 4.3 Temporal Difference Learning

```
Require: Policy $\pi$, step size schedule $(\alpha_t)_{t \geq 1}$.
1: Initialize $V_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, e.g., $V_1(x) = 0$.
2: Initialize $X_1 \sim \rho$
3: for each step $t = 1, 2, \dots$ do
4:     Let $A_t \sim \pi(\cdot|x)$
5:     Take action $A_t$, observe $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$ and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$
6:     Update
           $$V_{t+1}(x) \leftarrow \begin{cases} V_t(x) + \alpha_t(x)[R_t + \gamma V_t(X_{t+1}) - V_t(X_t)] & x = X_t \\ V_t(x) & x \neq X_t \end{cases}$$
7: end for
```

instead of all states. The hope is that the use of SA gets rid of the noise and we follow the deterministic part, which is VI. The procedure is described in Algorithm 4.3. This asynchronous sample-based variation of the VI algorithm is called the *Temporal Difference* learning algorithm. The update rule could be written in perhaps a simpler, but less precise, form of

$$V(X_t) \leftarrow V(X_t) + \alpha_t(X_t)[R_t + \gamma V(X_{t+1}) - V(X_t)],$$

without showing any explicit dependence of V on time index t.

The term

$$\delta_t \triangleq R_t + \gamma V(X_{t+1}) - V(X_t)$$

is called *temporal difference (TD) error*. This is a noisy measure of how close we are to V^π. To see this more clearly, let us define the dependence on the TD error on its components more explicitly: Given a transition (X, A, R, X') and a value function V, we define

$$\delta(X, R, X'; V) \triangleq R + \gamma V(X') - V(X).$$

We have

$$\mathbb{E}\left[\delta(X, R, X'; V)|X = x\right] = (T^\pi V)(x) - V(x) = \mathrm{BR}(V)(x).$$

So in expectation, the TD error is equal to the Bellman residual of V, evaluated at state x. Recall that the Bellman residual is zero when $V = V^\pi$. The TD error does not become zero, even if we have already found V^π, but it fluctuates around zero when we are there (unless we have a deterministic dynamics and policy).

Remark 4.1. *The error between the MC estimate $G_1^\pi(x)$ of a state x and a value function estimate $V(x)$ is related to the TD error along the trajectory. To see this clearly, consider a trajectory $(X_1, A_1, R_1, \dots)$. The return $G_t^\pi = \sum_{t' \geq t} \gamma^{t'-t} R_{t'}$ and $\delta_t = R_t + \gamma V(X_{t+1}) - V(X_t)$. We have*

$$\begin{aligned} G_1^\pi - V(X_1) &= (R_1 + \gamma G_2^\pi) - V(X_1) \\ &= R_1 + \gamma G_2^\pi - V(X_1) + \gamma V(X_2) - \gamma V(X_2) \\ &= (R_1 + \gamma V(X_2) - V(X_1)) + \gamma (G_2^\pi - V(X_2)) \\ & \delta_1 + \gamma (G_2^\pi - V(X_2)). \end{aligned}$$

Following the same argument, we get that

$$G_1^\pi - V(X_1) = \delta_1 + \gamma \delta_2 + \ldots = \sum_{t \geq 1} \gamma^{t-1} \delta_t.$$

In words, the difference $G_1^\pi - V(X_1)$ is the discounted sum of the TD errors on the trajectory. When $V = V^\pi$, the LHS is a zero-mean noise, as is the RHS (since each $\mathbb{E}[\delta_t | X_1] = \mathbb{E}[\mathbb{E}[\delta_t | X_t] | X_1] = \mathbb{E}[0 | X_1] = 0$.

4.4.1 TD Learning for Action-Value Function

We can use a similar procedure to estimate the action-value function. To evaluate π, we need to have an estimate of $(T^\pi Q)(x, a)$ for all $(x, a) \in \mathcal{X} \times \mathcal{A}$. Suppose that $(X_t, A_t) \sim \mu$ and $X'_t \sim \mathcal{P}(\cdot | X_t, A_t)$ and $R_t \sim \mathcal{R}(\cdot | X_t, A_t)$. The update rule would be

$$Q_{t+1}(X_t, A_t) \leftarrow Q_t(X_t, A_t) + \alpha_t(X_t, A_t) \left[R_t + \gamma Q_t(X'_t, \pi(X'_t)) - Q_t(X_t, A_t) \right],$$

and

$$Q_{t+1}(x, a) \leftarrow Q_t(x, a)$$

for all other $(x, a) \neq (X_t, A_t)$. It is easy to see that

$$\mathbb{E}\left[R_t + \gamma Q_t(X'_t, \pi(X'_t)) | X = x, A = a \right] = (T^\pi Q)(x, a).$$

If we have a stream of data, $X'_t = X_{t+1}$, but this is not necessary.

An interesting observation is that π appears only in $Q_t(X'_t, \pi(X'_t))$ term. The action A_t does not need to be selected by π itself. This entails that the agent can generate the stream of data $X_1, A_1, R_1, X_2, A_2, R_2, \dots$ by following a *behaviour* policy π_b that is different from the policy that we want to evaluate π. When $\pi_b = \pi$, we are in the *on-policy* sampling scenario, in which the agent is evaluating the same policy that it is following. When $\pi_b \neq \pi$, we are in the *off-policy* sampling scenario, in which the agent is evaluating a policy that is different from the one it is following.

On-policy vs. Off-policy

4.4.2 VI vs TD vs MC

Value Iteration, Monte Carlo estimation, and TD learning can all be seen as procedures to estimate V^π. It is instructive to compare the type of approximation each of them are using.

- In MC, we use G^π as the target value. As $V^\pi(x) = \mathbb{E}\left[G^\pi | X = x\right]$, we have a noisy (but unbiased) estimate of V^π. SA allows us to converge to its mean.
- In VI, we update $V_{k+1}(x) \leftarrow (T^\pi V_k)(x) = \mathbb{E}\left[R + \gamma V_k(X') | X = x\right]$. Here we do not know V^π, but use the current approximation V_k instead. Because of the contraction property of the Bellman operator, this converges to V^π.
- In TD learning, the target is $R + \gamma V_k(X')$. This has two sources of approximation: (a) we use V_k instead of V^π, and (b) we use a sample to estimate an expectation.

4.5 Monte Carlo Estimation for Control

So far we have described methods for policy evaluation. We can use similar methods for solving the control problem, i.e., finding the optimal value function and the optimal policy. We describe MC-based methods here, and we get to the TD-based methods in the next section.

The general idea is to use some version of PI. For example, if we run many rollouts from each state-action pair (x, a), we can define $\hat{Q}_t^\pi$ that converges to Q^π. If we wait for an infinite time, $\hat{Q}_\infty^\pi = \lim_{t\to\infty} \hat{Q}_t^\pi = Q^\pi$. We can then choose the new policy $\pi' \leftarrow \pi_g(\hat{Q}_\infty^\pi)$. Because of the Policy Improvement theorem, this new policy would be better than π, unless π is already an optimal policy. This PI can be described by the following sequence of π and Q^π:

$$\pi_0 \xrightarrow{\text{E}} Q^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} Q^{\pi_1} \xrightarrow{\text{I}} \cdots .$$

It turns out that we do not need to have a very accurate estimation of Q^{π_k} before performing the policy improvement step. As a result, we can perform MC for a finite number of rollouts from each state, and then perform the improvement step. In fact, we only need to estimate $Q^{\pi_k}(x, a)$ based on only a single MC estimate. The initial-state-only version of this idea is shown in Algorithm 4.4. We choose the learning rate $(\alpha_k)_k$ to satisfy the standard SA conditions, similar to (4.6).

This version has several features:

Algorithm 4.4 Monte Carlo Control (Initial-State Only)

```
Require: Initial policy $\pi_1$, step size schedule $(\alpha_k)_{k \geq 1}$.
1: Initialize $Q_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, e.g., $Q_1 = 0$.
2: for each iteration $k = 1, 2, \dots$ do
3:     for all $(x, a) \in \mathcal{X} \times \mathcal{A}$ do
4:         Initialize $X_1 = x$ and $A_1 = a$.
5:         Generate an episode from $X_1$ by choosing $A_1$, and then following $\pi_k$ to obtain $X_1, A_1, R_1, X_2, A_2, R_2, \dots$.
6:         Compute $G_1^{\pi_k}(X_1, A_1) = \sum_{t \geq 1} \gamma^{t-1} R_t$.
7:         Update
               $\hat{Q}^{\pi}_{k+1}(X_1, A_1) \leftarrow (1 - \alpha_k(X_1, A_1)) \hat{Q}^{\pi}_k(X_1, A_1) + \alpha_k(X_1, A_1) G_1^{\pi_k}(X_1, A_1)$
8:     end for
9:     Improve policy: $\pi_{k+1} \leftarrow \pi_g(Q_{k+1})$.
10: end for
```

- We start a rollout from all state-action pairs.
- We only use one rollout to obtain a new estimate of Q^{π_k} at each (x, a).
- We do not use the value of $G_t^{\pi_k}$ for all other states encountered on the trajectory.

Before further discussion and improvement of this version, let us state a theoretical result about this procedure.

Proposition 4.1 (Convergence of MC for Control – Proposition 5 of Tsitsiklis 2002)**.** *The sequence Q_k generated by Algorithm 4.4 with the learning rate (α_k) satisfying the SA conditions (4.6) converges to Q^* almost surely.*

Another variation is when at each iteration k, instead of generating rollouts from all state-action pairs, we only choose a single independently selected $(X, A) \sim \text{Unif}(\mathcal{X} \times \mathcal{A})$, follow the policy π_k for a single rollout and update the action-value $Q(X, A)$. This procedure also converges to Q^*, as shown by Tsitsiklis [2002].

If instead of uniform distribution, we select $(X, A) \sim \rho \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$ with $rho > 0$, the number of times that each state-action pair is selected can possibly be vastly different. In order to make this procedure work, we need to define a state-action-dependent step size $\alpha_k(x, a)$, and set it equal to

$$\alpha_k(x, a) = \frac{1}{n_k(x, a)},$$

Algorithm 4.5 Monte Carlo Control (Every-Visit)

```
Require: Initial policy $\pi_1$, step size schedule $(\alpha_k)_{k \geq 1}$, initial distribution $\rho \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$.
 1: Initialize $Q_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, e.g., $Q_1 = 0$.
 2: Initialize $n_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{N}$ with $n_1(x, a) = 0$.
 3: for each iteration $k = 1, 2, \ldots$ do
 4:     for all $(x, a) \in \mathcal{X} \times \mathcal{A}$ do
 5:         Draw $(X_1, A_1) \sim \rho$
 6:         Generate an episode from $X_1$ by choosing $A_1$, and then following $\pi_k$ to obtain $X_1, A_1, R_1, X_2, A_2, R_2, \ldots$ until the end of episode.
 7:         For all $(X_t, A_t)$ visited within the episode, set $n_k(X_t, A_t) \leftarrow n_k(X_t, A_t) + 1$.
 8:         Compute $G_t^{\pi_k}(X_t, A_t) = \sum_{t' \geq t} \gamma^{t'-t} R_{t'}$.
 9:         for all $(X_t, A_t)$ visited in the episode do
10:             Let $\alpha_k(X_t, A_t) = \frac{1}{n_k(X_t, A_t)}$.
11:             Update
                $$\hat{Q}_{k+1}^{\pi}(X_t, A_t) \leftarrow (1 - \alpha_k(X_t, A_t))\, \hat{Q}_k^{\pi}(X_t, A_t) + \alpha_k(X_t, A_t) G_t^{\pi_k}(X_t, A_t)$$
12:         end for
13:         $n_{k+1} \leftarrow n_k$.
14:     end for
15:     Improve policy: $\pi_{k+1} \leftarrow \pi_g(Q_{k+1})$.
16: end for
```

with $n_k(x, a)$ being the number of times (x, a) have been selected up to iteration k. This is essentially the same as averaging all returns starting from (x, a).

Another way we might modify this MC algorithm is to use first-visit or every-visit variations of the MC procedure, as described in Section 4.3.1 for the PE problem. The same idea can be applied, but the convergence guarantee has not been proven so far. We report such an algorithm here as Algorithm 4.5, with the caution that it is known whether it converges or not.

4.6 Temporal Difference Learning for Control: Q-Learning and SARSA Algorithms

We can use TD-like methods for the problem of control too. The same way that we devised the TD learning as the sample-based asynchronous version of VI for the PE problem, we can devise a sample-based asynchronous version of VI for the control problem. Consider any $Q \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$. Let $X' \sim \mathcal{P}(\cdot|X, A)$ and $R \sim \mathcal{R}(\cdot|X, A)$ and define

$$Y = R + \gamma \max_{a' \in \mathcal{A}} Q(X', a'). \tag{4.11}$$

We have

$$\begin{aligned}\mathbb{E}\left[Y | X = x, A = a\right] &= r(x, a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x, a) \max_{a' \in \mathcal{A}} Q(x', a') \\ &= (T^* Q)(x, a).\end{aligned} \tag{4.12}$$

So Y is an unbiased noisy version of $(T^* Q)(x, a)$. The *empirical Bellman optimality operator* is

$$(\hat{T}^* Q)(x, a) \triangleq R + \gamma \max_{a' \in \mathcal{A}} Q(X', a'). \tag{4.13}$$

We can define a noisy version of VI (control), similar to (4.10), as

$$Q_{k+1} \leftarrow \hat{T}^* Q_k,$$

which can be written as

$$Q_{k+1} \leftarrow \hat{T}^* Q_k = T^* Q_k + \underbrace{\left(\hat{T}^* Q_k - T^* Q_k\right)}_{\triangleq \varepsilon_k}. \tag{4.14}$$

In order to diminish the effect of noise, however, we need to use a SA procedure. We can define an online algorithm that updates Q based on (X_t, A_t, R_t, X_{t+1}) as follows:

$$Q_{t+1}(X_t, A_t) \leftarrow (1 - \alpha_t(X_t, A_t)) Q_t(X_t, A_t) + \alpha_t(X_t, A_t) \left[R_t + \gamma \max_{a' \in \mathcal{A}} Q_t(X_{t+1}, a')\right] \tag{4.15}$$

for the observed (X_t, A_t) and

$$Q_{t+1}(x, a) \leftarrow Q_t(x, a)$$

for all other states $(x, a) \neq (X_t, A_t)$. This procedure is called the *Q-Learning* algorithm, which we encountered before as Algorithm 1.1 in Section 1.5. The learning rate $\alpha_t(x, a)$ is state-action-dependent in general, but sometimes might be considered as state-action-independent α_t or even time-independent α.

What is the policy being evaluated by the Q-Learning algorithm? Looking at the update rule, we see that we have $\max_{a' \in \mathcal{A}} Q_t(X_{t+1}, a')$. What is the policy evaluated at state X_{t+1}? We have that

$$\max_{a' \in \mathcal{A}} Q_t(X_{t+1}, a') = Q_t\left(X_{t+1}, \operatorname*{argmax}_{a' \in \mathcal{A}} Q_t(X_t, a')\right) = Q_t\left(X_{t+1}, \pi_g(X_t, Q_t)\right). \quad (4.16)$$

So the Q-Learning algorithm is evaluating the greedy policy w.r.t. the current estimate Q_t. The evaluated policy may change as we update Q_t. It is noticeable that the greedy policy can be different from the policy π that the algorithm follows. This makes the Q-Learning algorithm an example of an algorithm that works under the off-policy sampling scenario. More concisely, we say that Q-Learning is an off-policy algorithm.

Exercise 4.6 ($\star$)**.** *When does the policy π generating data for the Q-Learning algorithm become the same as the policy that it evaluates?*

We can also have a PI-like procedure: Estimate Q^π for a given π, and perform policy improvement to obtain a new π. If we wait for a long time (forever), the TD method produces a Q that converges to Q^π (under usual conditions of the convergence of a TD method). This would be a usual PI procedure in which the policy evaluation part is performed using a TD method. One can, however, improve the policy before Q converges to Q^π. This is called a *generalized policy iteration* or *optimistic policy iteration*.

An example of such an algorithm works as follows: At state X_t, the agent chooses $A_t = \pi_t(X_t)$. It then receives $X_{t+1} \sim \mathcal{P}(\cdot|X_t, A_t)$ and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$. At the time step $t+1$, it chooses $A_{t+1} = \pi_t(X_{t+1})$ and updates the action-value function estimate as

$$Q_{t+1}(X_t, A_t) \leftarrow (1 - \alpha_t(X_t, A_t))Q_t(X_t, A_t) + \alpha_t(X_t, A_t)\left[R_t + \gamma Q_t(X_{t+1}, A_{t+1})\right] \quad (4.17)$$

for the observed (X_t, A_t) and $Q_{t+1}(x, a) \leftarrow Q_t(x, a)$ for all other states $(x, a) \neq (X_t, A_t)$.

The policy π_t is often chosen to be close to a greedy policy $\pi_g(Q_t)$, but with some amount of exploration, e.g., the ε-greedy policy (1.17). The greedy part performs the policy improvement, while the occasional random choice of actions allows the agent to have some exploration.

This algorithm is called *SARSA*. The name comes from the fact that this algorithm only needs the current **S**tate X_t, current **A**ction A_t, **R**eward R_t, and the next **S**tate X_{t+1}, and the next **A**ction A_{t+1} to update Q (the naming would be more obvious if we used S instead of X to refer to an action).[5]

If we compare SARSA's update rule (4.17) to Q-Learning's (4.15), we see that their difference is effectively in the policy they evaluate at the next state. SARSA uses $Q_t(X_{t+1}, A_{t+1})$, which is equal to $Q_t(X_{t+1}, \pi_t(X_{t+1}))$. Hence, SARSA is evaluating π_t, which is the same policy that selects actions. On the other hand, Q-Learning uses $\max_{a' \in \mathcal{A}} Q_t(X_{t+1}, a')$, which evaluates the greedy policy $\pi_g(Q_t)$, as we discussed already (4.16). Whereas Q-Learning is an off-policy algorithm, SARSA is an on-policy algorithm.

Remark 4.2. *We do not need a stream of data for this algorithm (or Q-Learning) to work. Instead of X_{t+1}, we could have $X'_t \sim \mathcal{P}(\cdot|X_t, A_t)$, but do not set X_{t+1} to be X'_t.*

4.7 Stochastic Approximation

We have already encountered the use of stochastic approximation (SA) to estimate the expectation of a random variable. Here we provide a convergence result for a class of SA problems.

Suppose that we want to find the fixed-point of an operator L, i.e., solve the following equation

$$L\theta = \theta,$$

for $\theta \in \mathbb{R}^d$, and $L : \mathbb{R}^d \to \mathbb{R}^d$. Consider the iterative update

$$\theta_{t+1} \leftarrow (1 - \alpha)\theta_t + \alpha L\theta_t. \tag{4.18}$$

If L is c-Lipschitz with $c < 1$ and α is small enough ($0 < \alpha < \frac{2}{1-c}$), this would converge. To see this, notice that the above iteration is equivalent of having $\theta_{t+1} \leftarrow L'\theta_t$ with the new operator

$$L' : \theta \mapsto [(1 - \alpha)\mathbf{I} + \alpha L]\theta.$$

For any $\theta_1, \theta_2 \in \mathbb{R}^d$, this new operator satisfies

$$\|L'\theta_1 - L'\theta_2\| \leq (1 - \alpha) \|\theta_1 - \theta_2\| + \alpha \|L\theta_1 - L\theta_2\| \leq [(1 - \alpha) + \alpha c] \|\theta_1 - \theta_2\| .$$

So if $|(1-\alpha)+\alpha c| < 1$, which is satisfied for $0 < \alpha < \frac{2}{1-c}$, L' is a contraction mapping. By the Banach fixed point theorem (Theorem A.1), the iteration converges.

[5]Based on this naming convention, the Q-Learning algorithm could be called SARS!

Now suppose that we do not have access to $L\theta_t$, but only its noise contaminated $L\theta_t + \eta_t$ with $\eta_t \in \mathbb{R}^d$ being a zero-mean noise. We perform the following instead:

$$\theta_{t+1} \leftarrow (1 - \alpha_t)\theta_t + \alpha_t(L\theta_t + \eta_t). \tag{4.19}$$

This iterative process is similar to (4.1), with the difference that the latter concerns the estimation of a mean given an unbiased noisy value of the mean, while here we are dealing with a noisy evaluation of an operator L being applied to θ_t.

As discussed in Section 4.1 and in particular (4.1), if we choose a constant learning rate α, the variance of the estimate θ_t would not go to zero. That is why we use an iteration-dependent α_t.

Let us consider a more general update than above. The generalization is that we allow some dimensions of θ to be updated while the others remain the same. This asynchronous update occurs in algorithms such as Q-Learning or TD, so we would like to have a model that captures the behaviour of those algorithms.

The algorithm model is as follows: Assume that at time t, the i-th component of θ_t is updated as

$$\theta_{t+1}(i) \leftarrow (1 - \alpha_t(i))\theta_t(i) + \alpha_t(i)\left[(L\theta_t)(i) + \eta_t(i)\right], \tag{4.20}$$

with the understanding that $\alpha_t(j) = 0$ for $j \neq i$ (components that are not updated).

We denote the history of the algorithm up to time t by F_t:[6]

$$F_t = \{\theta_0, \theta_1, \ldots, \theta_t\} \cup \{\eta_0, \eta_1, \ldots, \eta_{t-1}\} \cup \{\alpha_0, \alpha_1, \ldots, \alpha_t\}.$$

Note that η_t is not included because it has not happened just before performing this step.

We need to make some assumptions on the noise.

Assumption A1

(a) For every i and t, we have $\mathbb{E}\left[\eta_t(i) | F_t\right] = 0$.

(b) Given any norm $\|\cdot\|$ on $\mathbb{R}^d$, there exist constants c_1, c_2 such that for all i and t, we have

$$\mathbb{E}\left[|\eta_t(i)|^2 | F_t\right] \leq c_1 + c_2 \|\theta_t\|^2.$$

[6]The σ-algebras $(\sigma(F_t))_t$ is called filtration in stochastic process or martingale theory.

The first assumption simply indicates that the noise η_t should be zero-mean, conditioned on the information available up to time t. The second assumption is the requirement that the variance of the noise is not too much. The variance is allowed to depend on parameter θ_t though.

The following result provides the condition for the convergence of the SA iteration to its fixed point.

Theorem 4.2 (Convergence of the Stochastic Approximation – Proposition 4.4 of Bertsekas and Tsitsiklis 1996)**.** *Let (θ_t) be the sequence generated by (4.20). Assume that*

1. *(Step Size) The step sizes $\alpha_t(i)$ (for $i = 1, \dots, d$) are non-negative and satisfy*

$$\sum_{t=0}^{\infty} \alpha_t(i) = \infty, \qquad \sum_{t=0}^{\infty} \alpha_t^2(i) < \infty.$$

2. *(Noise) The noise $\eta_t(i)$ satisfies Assumption A1.*

3. *The mapping L is a contraction w.r.t. $\|\cdot\|_\infty$ with a fixed point of θ^*.*

Then θ_t converges to θ^* almost surely.

Remark 4.3. *Proposition 4.4 of Bertsekas and Tsitsiklis [1996] is slightly more general than what we have here as it allows L to be a weighted maximum norm pseudo-contraction. We do not need that generality here.*

4.8 Convergence of Q-Learning

We use Theorem 4.2 to prove the convergence of the Q-Learning algorithm for finite state-action MDPs. The Q-Learning update rule (4.15) has the same form as the SA update rule (4.20), if we identify θ with $Q \in \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$, and the operator L with the Bellman optimality operator T^*. The index i in the SA update plays the role of the selected (X_t, A_t). And the noise term $\eta_t(i)$ would be the difference between $(T^*Q_t)(X_t, A_t)$ and the sample-based version $R_t + \gamma \max_{a' \in \mathcal{A}} Q_t(X_{t+1}, a')$. In order to prove the convergence of the Q-Learning, we have to verify the conditions of Theorem 4.2.

Theorem 4.3. *Suppose that for all $(x, a) \in \mathcal{X} \times \mathcal{A}$, the step sizes $\alpha_t(x, a)$ satisfy*

$$\sum_{t=0}^{\infty} \alpha_t(x, a) = \infty, \qquad \sum_{t=0}^{\infty} \alpha_t^2(x, a) < \infty.$$

Furthermore, assume that the reward is of bounded variance. Then, Q_t converges to Q^* almost surely.

Proof. Suppose that at time t, the agent is at state X_t, takes action A_t, gets to $X'_t \sim \mathcal{P}(\cdot|X_t, A_t)$ and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$. The update rule of the Q-Learning algorithm can be written as

$$Q_{t+1}(X_t, A_t) \leftarrow (1-\alpha_t(X_t, A_t))Q_t(X_t, A_t) + \alpha_t(X_t, A_t)\left[(T^*Q_t)(X_t, A_t) + \eta_t(X_t, A_t)\right],$$

with $\eta_t(X_t, A_t) = (R_t + \gamma \max_{a' \in \mathcal{A}} Q_t(X'_t, a')) - (T^*Q_t)(X_t, A_t)$, and

$$Q_{t+1}(x, a) \leftarrow Q_t(x, a) \qquad (x, a) \notin (X_t, A_t).$$

We have already established that T^* is a γ-contraction mapping, so condition (3) of the theorem is satisfied. Condition (1) is assumed too. So it remains to verify the conditions (2) on noise η_t, which are conditions (a) and (b) of Assumption A1.

Let F_t be the history of algorithm up to and including when the step size $\alpha_t(X_t, A_t)$ is chosen, but just before X'_t and R_t are revealed. We have, similar to what we had seen before in (4.12),

$$\mathbb{E}\left[\eta_t(X_t, A_t)|F_t\right] = \mathbb{E}\left[R_t + \gamma \max_{a' \in \mathcal{A}} Q_t(X'_t, a') \mid F_t\right] - (T^*Q_t)(X_t, A_t) = 0.$$

This verifies condition (a).

To verify (b), we provide an upper bound on $\mathbb{E}\left[\eta_t^2(X_t, A_t)|F_t\right]$:

$$\begin{aligned}\mathbb{E}\left[\eta_t^2(X_t, A_t) \mid F_t\right] = \mathbb{E}\Bigg[\Big|(R_t - r(X_t, A_t)) + \\ \gamma\left(\max_{a' \in \mathcal{A}} Q_t(X'_t, a') - \int \mathcal{P}(\mathrm{d}x'|X_t, A_t) \max_{a' \in \mathcal{A}} Q_t(x', a')\right)\Big|^2 \mid F_t\Bigg] \\ \leq 2\mathrm{Var}\left[R_t \mid X_t, A_t\right] + 2\gamma^2 \mathrm{Var}\left[\max_{a' \in \mathcal{A}} Q_t(X', a') \mid X_t, A_t\right].\end{aligned}$$

We have

$$\begin{aligned}\mathrm{Var}\left[\max_{a' \in \mathcal{A}} Q_t(X', a') \mid X_t, A_t\right] &\leq \mathbb{E}\left[\left|\max_{a' \in \mathcal{A}} Q_t(X', a')\right|^2 \mid X_t, A_t\right] \\ &\leq \max_{x,a} |Q_t(x, a)|^2 \\ &\leq \sum_{x,a} |Q_t(x, a)|^2 = \|Q_t\|_2^2.\end{aligned}$$

If we denote the maximum variance of the reward distribution over the state-action space $\max_{(x,a)\in\mathcal{X}\times\mathcal{A}} \mathrm{Var}\left[R(x,a)\right]$ by σ_R^2, which is assumed to be bounded, we have

$$\mathbb{E}\left[\eta_t^2(X_t,A_t) \mid F_t\right] \le 2(\sigma_R^2 + \gamma^2 \left\|Q_t\right\|_2^2).$$

Therefore, we can choose $c_1 = 2\sigma_R^2$ and $c_2 = 2\gamma^2$ in condition b.

All conditions of Theorem 4.2 are satisfied, so Q_t converges to Q^* (a.s.). □

A few remarks are in order. The step size condition is state-action dependent, and it should be satisfied for all state-action pairs. If there is a state-action pair that is not selected at all or only a finite number of times, the condition cannot be satisfied. We need each state-action pair to be visited infinitely often. This is only satisfied if the mechanism generating (X_t, A_t) is exploratory enough.

The state-action-dependence of the step size might be different from how the Q-Learning algorithm is sometimes presented (e.g., Algorithm 1.1 in Chapter 1), in which a single learning rate α_t is used for all state-action pairs. A single learning rate suffices if the agent happens to visit all $(x,a) \in \mathcal{X}\times\mathcal{A}$ frequent enough, for example every $M < \infty$ steps. To see this, suppose we visit (x_1, a_1) every M steps. When it is visited, its effective learning rate is $\alpha'_t(x_1,a_1) = \alpha_t$, and when it is not visited, it is $\alpha'_t(x_1,a_1) = 0$, as it is not updated. So $\sum_t \alpha'_t(x_1,a_1)$ has the form of

$$\cdots + \alpha_t + \underbrace{0 + 0 + \cdots + 0}_{M-1 \text{ times}} + \alpha_{t+M} + \cdots .$$

If $\alpha_t = \frac{1}{t+1}$, which satisfied the SA conditions, this would be $\sum_t \alpha'_t(x_1,a_1) = \frac{1}{M} + \frac{1}{2M} + \ldots = \frac{1}{M}\sum \frac{1}{t}$, which has the same convergence behaviour as $\sum_{t\ge 1}\frac{1}{t} = \infty$. The same is true for $\sum_t {\alpha'_t}^2(x_1,a_1)$ in comparison with $\sum \frac{1}{t^2}$.

Another remark is that this result only provides an asymptotic guarantee, but it does not show anything about the convergence rate, i.e., how fast Q_t converges to Q^*. There are some results for this.

Chapter 5

Value Function Approximation

In many real-world problems, the state-action space $\mathcal{X} \times \mathcal{A}$ is so large that we cannot represent quantities such as the value function or policy exactly.[1] An example is when $\mathcal{X} \subset \mathbb{R}^d$ with $d \geq 1$. Exact representation of an arbitrary function on $\mathbb{R}^d$, or even on $\mathbb{R}$, on a computer is infeasible as these sets have an uncountable number of members. Instead, we need to approximate those functions using a representation that is feasible to manipulate on a computer. This is called *function approximation (FA)*. This chapter is about approximation of the value function. In a later chapter, we focus on approximating the policy.

Function approximation is studied in several fields, including the approximation theory, machine learning, and statistics, each with slightly different goals. In the context of RL, the use of FA means that we would like to compute a value function that is approximately the same as the true value function (i.e., $\hat{V}^\pi \approx V^\pi$ or $\hat{V}^* \approx V^*$), or a policy that is $\hat{\pi}^* \approx \pi^*$.

These function approximations should be easily represented on a computer. As an example, we may use a linear function approximator defined based on a set of basis functions, i.e.,

$$\hat{V}(x) = \phi(x)^\top w = \sum_{i=1}^{p} \phi_i(x) w_i,$$

with $w \in \mathbb{R}^p$ and $\phi : \mathcal{X} \to \mathbb{R}^p$. So any $\hat{V}$ belongs to the space of functions $\mathcal{F}$

$$\mathcal{F} = \left\{ x \mapsto \phi(x)^\top w \, : \, w \in \mathbb{R}^p \right\}. \tag{5.1}$$

The function space $\mathcal{F}$ is called the value function space. In this example, it is a span of a set of features. We simply call it a linear function space. Note that the linearity is in the parameters w and not in the state x.

[1]Chapter's Version: 0.04 (2021 March 18). Some results need to be typed.

There are many other ways to represent the value function approximation $\hat{V}$ (and effectively, $\mathcal{F}$). Some examples are

- Deep neural networks (DNN)
- reproducing kernel Hilbert spaces (RKHS), e.g., often used along Support Vector Machines (SVM) and other kernel methods.
- Decision trees and random forests
- Local methods such as smoothing kernels, k-nearest neighbours, etc.

5.0.1 The Choice of Value Function Approximator

How should we choose the value function approximation? Let us briefly talk about two competing tradeoffs, without going into any detail or practical advice.

We want the FA to be expressive enough, so that a large range of possible functions can be represented. In other words, we would like $\mathcal{F}$ to be such that for any V^π (or V^*), we can find a $\hat{V} \in \mathcal{F}$ that is close to V^π (or V^*) w.r.t. some distance function, e.g., $\|\hat{V} - V^\pi\|_p$ is small.

Consider the value function depicted in Figure 5.1, which has a subset of $\mathbb{R}$ as its domain. One way to represent this function is to discretize its domain with the resolution of ε, and then use a piecewise constant function to represent it. This can be represented as a linear function approximator: Assume that the domain is $[-b, +b]$, we can define ϕ_i (for $i = 0, 1, \ldots, \lceil \frac{2b}{\varepsilon} \rceil$) as

$$\phi_i(x) = \mathbb{I}\{x \in [-b + i\varepsilon, -b + (i+1)\varepsilon)\}.$$

Any function V can be approximated by a $\hat{V}(x) = \hat{V}(x; w) = \phi(x)^\top w$ with $w \in \mathbb{R}^{\lceil \frac{2b}{\varepsilon} \rceil + 1}$. So we need $O(\frac{1}{\varepsilon})$ parameters to describe such a function. Let us denote such a function space by $\mathcal{F}_\varepsilon$.

The approximation quality depends on the regularity or structure of the value function V. If we allow V to change arbitrary, we cannot hope to have a good approximation. But if it has some regularity, for example being an L-Lipschitz function, we can see that there is always a piecewise $\hat{V}$ of the form just described that is $O(L\varepsilon)$ close to it. In other words, we have that for any V that is L-Lipschitz,

$$\inf_{\hat{V} \in \mathcal{F}_\varepsilon} \left\| \hat{V} - V \right\|_\infty \leq L\varepsilon.$$

This is called the *approximation error* or *bias* of this function space. It quantifies the expressivity of the function space. Note that the approximation error depends on

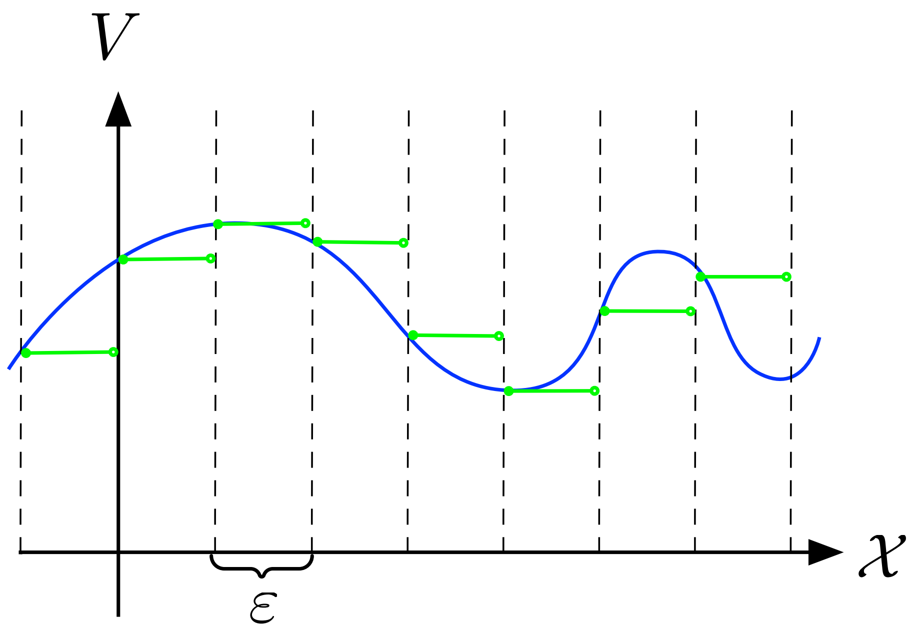

Figure 5.1: Approximating a function (blue) with a piecewise constant function (green) based on an ε-discretized state space

the structure of the function approximator (e.g., piecewise constant) as well as the class of functions that is being approximated, e.g., L-Lipschitz functions here. If we decrease ε, the function space $\mathcal{F}_\varepsilon$ becomes more expressive, and its approximation error decreases.

If the domain was $\mathcal{X} = [-1, +1]^d$ for $d \geq 1$, we would need $O(\frac{1}{\varepsilon^d})$ parameters to describe such a function. This increases exponentially fast as a function of d. For example, if $d = 1$, we need a 20-dimensional θ to represent such a function; if $d = 2$, we need a 400-dimensional vector, and if $d = 10$, we need a $\approx 10^{13}$-dimensional vector. This latter one is clearly infeasible to store on most computers. Note that $d = 10$ is not a very large state space for many real-world applications. This exponential growth of the number of parameters required to represent a high-dimensional function is an instance of the *curse of dimensionality*.[2]

Other than the expressivity of the function space $\mathcal{F}$, we also need to pay attention

[2]The curse of dimensionality has other manifestations too. For example, computing integrals in high-dimensions requires an exponential increase in computation. We shall see another example of it later.

to the statistical aspect of estimating a function within this function space using a finite number of data points. The estimation accuracy depends on some notion of complexity or size of $\mathcal{F}$. Quantifying this requires some further development, which we shall do later in a simplified setting, but roughly speaking, the statistical error behaves as $O(\sqrt{\frac{\log |\mathcal{F}|}{n}})$, where n is the number of data points used in the estimation. This is called *estimation error* or *variance*.

In the choice of function approximator, we have to consider both the approximation capability of $\mathcal{F}$ and the statistical complexity of estimation. This balance between the approximation vs. estimation error (or bias vs. variance) is a well-known tradeoff in supervised learning and statistics. We also have a very similar problem in the RL context too. We shall quantify it in analyzing one of the algorithms.

5.1 Value Function Computation with a Function Approximator

Let us develop some general approaches for the value function computation when we are restricted to use functions from a value function space $\mathcal{F}$. Most of the approaches are based on defining a loss function that should be minimized in order to find an approximate value function. The presentation focuses on the population version of these loss functions, when we have access to the model, akin to the setup of Chapter 3 when we knew $\mathcal{P}$ and $\mathcal{R}$. In the next sections, we describe how the data can be used to compute the value functions. Many of those methods are essentially use the empirical loss function instead of the population one.

5.1.1 Approximation Given the Value Function

The simplest approach is perhaps when we happen to know V^π (or Q^π, V^*, Q^*), and we want to represent it with a function $V \in \mathcal{F}$. Our goal can then be expressed as finding $V \in \mathcal{F}$ such that

$$V \approx V^\pi.$$

To make the approximate symbol $\approx$ quantified, we have to pick a distance function between function V and V^π, i.e., $d : \mathcal{B}(\mathcal{X}) \times \mathcal{B}(\mathcal{X}) \to \mathbb{R}$. Given such a distance function, we can express our goal as

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} d(V, V^\pi).$$

A commonly used family of distances are based on the L_p-norm w.r.t. a (probability) measure $\mu \in \mathcal{M}(\mathcal{X})$ (A.1) in Appendix A.2. We can then have

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - V^\pi\|_{p,\mu} . \tag{5.2}$$

A common choice is the L_2-norm, which should remind us of the mean squared loss function commonly used in regression.

A reasonable question at this point is that how can we even have access to V^π? If we do know it, what is the reason for approximating it after all? We recall from them MC estimation (Section 4.3) that we can indeed estimate V^π at a state x by following π, and the estimate is unbiased (for initial-state and first-visit variants). So even though we may not have $V^\pi(x)$, we can still have $V^\pi(x)+\varepsilon(x)$ with $\mathbb{E}[\varepsilon(x)] = 0$. When the state space is large (e.g., continuous), we cannot run MC for all states, but only a finite number of them. The role of FA is then to help us generalize from a finite number of noisy data points to the whole state space.

We can also design approaches that benefit from the structural properties of the value function, which we studied in Chapter 2. We used those properties to design algorithms such as VI, PI, and LP in Chapter 3. All these methods have variants that work with FA. They are generally called *Approximate VI, VP, LP* and are simply referred to as AVI, API, ALP. These methods are also sometimes called *approximate dynamic programming*.

5.1.2 Approximate Value Iteration (Population Version)

Recall that VI (Section 3.3) is defined by

$$V_{k+1} \leftarrow TV_k,$$

with T being either T^π or T^*. One way to develop its approximate version is to perform each step only approximately, i.e., find $V_{k+1} \in \mathcal{F}$ such that

$$V_{k+1} \approx TV_k.$$

We can think of different distance function, as before. A commonly used one is based on the L_p-norm, and most commonly the L_2-norm. In that case, we start from a $V_0 \in \mathcal{F}$, and then at each iteration k of AVI we solve

$$V_{k+1} \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - TV_k\|_{p,\mu} . \tag{5.3}$$

The procedure for the action-value function is similar with obvious modifications.

It is notable that even though $V_k \in \mathcal{F}$, after the application of the Bellman operator T on it, it may not be within $\mathcal{F}$ anymore. Therefore, we may have some function approximation error at each iteration of AVI. The amount of this error depends on how expressive $\mathcal{F}$ is and how much T can push a function within $\mathcal{F}$ outside that space.

5.1.3 Bellman Error (or Residual) Minimization (Population Version)

We can cast the goal of approximating V^π as finding a $V \in \mathcal{F}$ such that the Bellman equation is approximately satisfied. We know that if we find a V such that $V = T^\pi V$, that function must be equal to V^π. Under FA, we may not achieve this exact equality, but instead have

$$V \approx T^\pi V, \tag{5.4}$$

for some $V \in \mathcal{F}$. We can think of different ways to quantify the quality of approximation. The L_p-norm w.r.t. a distribution μ is a common choice:

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - T^\pi V\|_{p,\mu} = \|\mathrm{BR}(V)\|_{p,\mu}\,. \tag{5.5}$$

The value of p is often selected to be 2. This procedure is called the Bellman Residual Minimization (BRM). The same procedure works for the action-value function Q with obvious change. This procedure is different from AVI in that we do not mimic the iterative process of VI (which is convergent in the exact case without any FA), but instead directly go for the solution of the fixed-point equation.

A geometric viewpoint might be insightful. Consider the space of all functions $\mathcal{B}(\mathcal{X})$, and $\mathcal{F}$ as its subset. When $\mathcal{F}$ is the set of linear functions (5.1), its geometry is the subspace spanned by ϕ. The subspace can be visualized as a plane. The following argument, however, is not specialized to linear FA.

Given $V \in \mathcal{F}$, we apply T^π to it in order to get $T^\pi V$. In general, $T^\pi V$ is not within $\mathcal{F}$, so we visualize it with a point outside the plane. Figure 5.2 shows a few Vs and their corresponding Bellman residuals. BRM minimizes the distance $\|V - T^\pi V\|_{2,\mu}$ among all functions in $V \in \mathcal{F}$.

If it happens that there exists a $V \in \mathcal{F}$ that makes $\|V - T^\pi V\|_{2,\mu} = 0$, and if we assume that $\mu(x) > 0$ for all $x \in \mathcal{X}$, we can conclude that $V(x) = (T^\pi V)(x)$ for $x \in \mathcal{X}$ (a.s.). This is the Bellman equation, so its solution is $V = V^\pi$. But if it is not, which is the general case, the minimizer V of (5.5) is not the value function V^π. Nevertheless, it still has some good approximation properties.

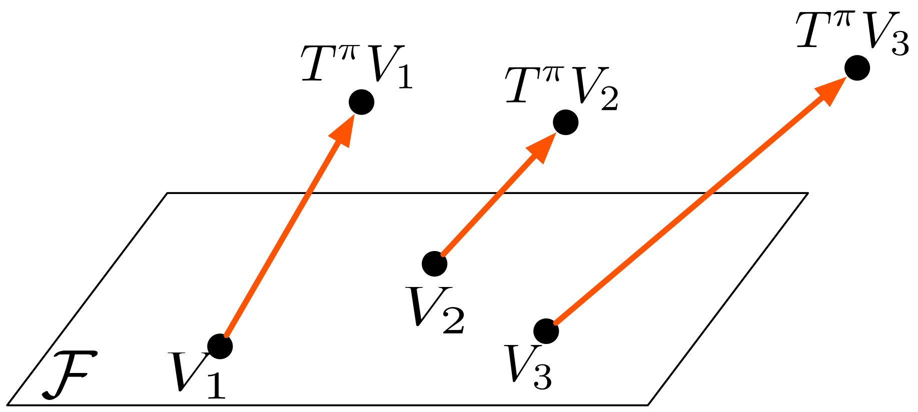

Figure 5.2: The Bellman operator may take value functions from $\mathcal{F}$ to a point outside that space. BRM finds $V \in \mathcal{F}$ with the minimal distance.

Recall from Proposition 2.7 that

$$\|V - V^\pi\|_\infty \leq \frac{\|V - T^\pi V\|_\infty}{1-\gamma}.$$

This shows that if we find a good solution V to BRM w.r.t. the supremum norm, the error of $\|V - V^\pi\|_\infty$ is also small (amplified by a factor of $\frac{1}{1-\gamma}$ though, which can be large when γ is close to 1). We do not need the knowledge of V^π to find this approximation, as opposed to (5.2). Minimizing the Bellman error is enough.

The result of Proposition 2.7, however, is for the supremum norm, and not the $L_p(\mu)$-norm of (5.5). We could define the optimization problem as minimizing $\|V - T^\pi V\|_\infty$. Even though we could in principle solve the minimizer of $\|V - T^\pi V\|_\infty$, working with an L_p-norm, especially the L_2-norm, is more common in ML, both because of algorithmic reasons and theoretical ones.[3] Instead of changing the optimization problem, we provide a similar error bound guarantee w.r.t. an L_p-norm. Here we pick μ to be the stationary distribution of π, instead of any general distribution, which we denote by ρ^π.

5.1.3.1 Stationary Distribution of Policy π

The stationary (or invariant) distribution of a policy π is the distribution that does not change as we follow π. To be more clear, assume that we initiate the agent at

[3]Providing a guarantee on the supremum of the Bellman residual requires extra conditions on sampling distribution, e.g., it should be bounded away from zero.

$X_1 \sim \rho \in \mathcal{M}(\mathcal{X})$. The agent follows π and gets to $X_2 \sim \mathcal{P}^\pi(\cdot|X_1)$. The probability distribution of X_2 being in a (measurable) set B is

$$\mathbb{P}\left\{X_2 \in B\right\} = \int \rho(\mathrm{d}x)\mathcal{P}^\pi(B|x),$$

or for countable state space, the probability of being in state y is

$$\mathbb{P}\left\{X_2 = y\right\} = \sum_{x \in \mathcal{X}} \rho(x)\mathcal{P}^\pi(y|x).$$

If the distribution of X_2 is the same as the distribution of X_1, which is ρ, we say that ρ is the stationary distribution induced by π. We denote this distribution by ρ^π to emphasize its dependence on π. By induction, it would be the distribution of $X_3, X_4, \ldots$ too.

This distribution satisfies $\mathbb{P}\left\{X_1 = y\right\} = \mathbb{P}\left\{X_2 = y\right\}$, which means that

$$\rho^\pi(y) = \sum_{x \in \mathcal{X}} \mathcal{P}^\pi(y|x)\rho^\pi(x), \tag{5.6}$$

or

$$\rho^\pi(B) = \int \rho(\mathrm{d}x)\mathcal{P}^\pi(B|x). \tag{5.7}$$

For countable state spaces, we can write it in the matrix form too. If we denote $\mathcal{P}^\pi$ by an $n \times n$ matrix with $[\mathcal{P}^\pi]_{xy} = \mathcal{P}^\pi(y|x)$, we have

$$\rho^\pi(y) = \sum_x \mathcal{P}^\pi_{xy}\rho^\pi_x, \qquad \forall y \in \mathcal{X}$$

so

$$\rho^{\pi\top} = \rho^{\pi\top}\mathcal{P}^\pi. \tag{5.8}$$

Note that ρ^π is the left eigenvector corresponding to eigenvalue with value 1 of matrix $\mathcal{P}^\pi$ (or likewise, it would be the right eigenvector of $\mathcal{P}^{\pi\top}$).

An important property of the stationary distribution is that the Markov chain induced by π converges to the stationary distribution ρ^π, under certain conditions, even if the initial distribution is not ρ^π. That is, for any $\mu \in \mathcal{M}(\mathcal{X})$, we have that

$$\mu(\mathcal{P}^\pi)^k \to \rho^\pi.$$

We can show that the Bellman operator T^π is a γ-contraction w.r.t. the $L_2(\rho^\pi)$. This is a special property of the stationary distribution as T^π is not generally a contraction w.r.t. $L_2(\mu)$ for a distribution μ (as opposed to its being contraction for the supremum norm). This property is going to be crucially important in designing TD-like algorithms with FA.

Lemma 5.1. *The Bellman operator T^π is a γ-contraction w.r.t. $\|\cdot\|_{2,\rho^\pi}$.*

Proof. For any $V_1, V_2 \in \mathcal{B}(\mathcal{X})$, we have that

$$
\begin{aligned}
&\|T^\pi V_1 - T^\pi V_2\|_{2,\rho^\pi}^2 = \\
&\int \rho^\pi(\mathrm{d}x) \left| \left(r^\pi(x) + \gamma \int \mathcal{P}^\pi(\mathrm{d}x'|x) V_1(x') \right) - \left(r^\pi(x) + \gamma \int \mathcal{P}^\pi(\mathrm{d}x'|x) V_2(x') \right) \right|^2 = \\
&\int \rho^\pi(\mathrm{d}x) \left| \gamma \int \mathcal{P}^\pi(\mathrm{d}x'|x)(V_1(x') - V_2(x') \right|^2 \leq \\
&\gamma^2 \int \rho^\pi(\mathrm{d}x) \mathcal{P}^\pi(\mathrm{d}x'|x) \left| V_1(x') - V_2(x') \right|^2 = \\
&\gamma^2 \int \rho^\pi(\mathrm{d}x') \left| V_1(x') - V_2(x') \right|^2 = \gamma^2 \left\| V_1 - V_2 \right\|_{2,\rho^\pi}^2 ,
\end{aligned}
$$

where we used the Jensen's inequality to get the inequality and the definition of the stationary distribution in the penultimate equality. □

5.1.3.2 Stationary Distribution Weighted Error Bound for BEM

We are ready to prove the following error bound. This is similar to Proposition 2.7 with the difference that it upper bounds the L_1-norm of the value approximation error, weighted according to the stationary distribution ρ^π, to the $L_p(\rho^\pi)$ norm of the Bellman residual, instead of the same quantities measured according to their supremum norms.

Proposition 5.2. *Let ρ^π be the stationary distribution of $\mathcal{P}^\pi$. For any $V \in \mathcal{B}(\mathcal{X})$ and $p \geq 1$, we have*

$$
\|V - V^\pi\|_{1,\rho^\pi} \leq \frac{\|V - T^\pi V\|_{p,\rho^\pi}}{1-\gamma}.
$$

Proof. For any V, we have that

$$
\begin{aligned}
V - V^\pi &= V - T^\pi V + T^\pi V - V^\pi \\
&= (V - T^\pi V) + (T^\pi V - T^\pi V^\pi). \qquad (5.9)
\end{aligned}
$$

The second term, evaluated at a state x, is

$$(T^\pi V)(x) - (T^\pi V^\pi)(x) = \gamma \int \mathcal{P}^\pi(\mathrm{d}y|x)(V(y) - V^\pi(y)).$$

We take the absolute values of both sides of (5.9), use the obtained form of the second term, and integrate w.r.t. ρ^π, to get that

$$\int |V(x) - V^\pi(x)| \,\mathrm{d}\rho^\pi(x) \leq \int |V(x) - (T^\pi V)(x)| \mathrm{d}\rho^\pi(x) + \\ \gamma \int \mathrm{d}\rho^\pi(x) \left| \int \mathcal{P}^\pi(\mathrm{d}y|x)(V(y) - V^\pi(y)) \right|.$$

By Jensen's inequality, we have

$$\int |V(x) - V^\pi(x)| \,\mathrm{d}\rho^\pi(x) \leq \int |V(x) - (T^\pi V)(x)| \mathrm{d}\rho^\pi(x) + \\ \gamma \int \mathrm{d}\rho^\pi(x) \mathcal{P}^\pi(\mathrm{d}y|x) \left| V(y) - V^\pi(y) \right|.$$

Because ρ^π is the stationary distribution, by (5.7) the second integral in the RHS can be simplified as

$$\int \mathrm{d}\rho^\pi(x) \mathcal{P}^\pi(\mathrm{d}y|x) \left| V(y) - V^\pi(y) \right| = \int \mathrm{d}\rho^\pi(x) \left| V(x) - V^\pi(x) \right|.$$

So

$$\left\| V - V^\pi \right\|_{1,\rho^\pi} \leq \left\| V - T^\pi V \right\|_{1,\rho^\pi} + \gamma \left\| V - V^\pi \right\|_{1,\rho^\pi}.$$

After re-arranging, we get the result for $p = 1$. By Jensen's inequality, we have that $\left\| V - T^\pi V \right\|_{1,\rho^\pi} \leq \left\| V - T^\pi V \right\|_{p,\rho^\pi}$, for any $p \geq 1$. □

5.1.4 Projected Bellman Error (Population Version)

Another loss function is defined based on the idea that the distance between a value function $V \in \mathcal{F}$ and the projection of $T^\pi V$ onto $\mathcal{F}$ should be made small. That is, we find a $V \in \mathcal{F}$ such that

$$V = \Pi_{\mathcal{F}} T^\pi V, \tag{5.10}$$

where $\Pi_{\mathcal{F}}$ is the projection operator onto $\mathcal{F}$. This should be compared with (5.4), where there is no projection back to $\mathcal{F}$.

Let us formally define the projection operator before we proceed. The projection operator $\Pi_{\mathcal{F},\mu}$ is a linear operator that takes $V \in \mathcal{B}(\mathcal{X})$ and maps it to closest point on $\mathcal{F}$, measured according to its $L_2(\mu)$ norm. That is,

$$\Pi_{\mathcal{F},\mu} V \triangleq \operatorname*{argmin}_{V' \in \mathcal{F}} \|V' - V\|_{2,\mu}.$$

If the choice of distribution μ is clear from the context, we may omit it. By definition, $\Pi_{\mathcal{F},\mu} V \in \mathcal{F}$. Moreover, the projection of any point on $\mathcal{F}$ onto $\mathcal{F}$ is itself, i.e., if $V \in \mathcal{F}$, we have $\Pi_{\mathcal{F},\mu} V = V$. We later show that if $\mathcal{F}$ is a subspace, the projection operator is a non-expansion.

We can define a loss function based on (5.10). We can use different norms. A common choice is to use the $L_2(\mu)$-norm:

$$\|V - \Pi_{\mathcal{F}} T^\pi V\|_{2,\mu}. \tag{5.11}$$

This is called *Projected Bellman Error* or *Mean Square Projected Bellman Error* (MSPBE).

We find the value function by solving the following optimization problem:

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \|V - \Pi_{\mathcal{F}} T^\pi V\|_{2,\mu}. \tag{5.12}$$

Note that as $V \in \mathcal{F}$, we can write

$$V - \Pi_{\mathcal{F},\mu} T^\pi V = \Pi_{\mathcal{F},\mu} V - \Pi_{\mathcal{F},\mu} T^\pi V = \Pi_{\mathcal{F},\mu}(V - T^\pi V) = -\Pi_{\mathcal{F},\mu}(\mathrm{BR}(V)).$$

So the loss $\|V - \Pi_{\mathcal{F}} T^\pi V\|_{2,\mu}$ can also be written as $\|\Pi_{\mathcal{F},\mu}(\mathrm{BR}(V))\|_{2,\mu}$, the norm of the projection of the Bellman residual onto $\mathcal{F}$.

Comparing the projected Bellman error with the Bellman residual minimization (5.5), we observe that the difference is that the latter does not compute the distance between the projected T^π, but instead, computes the distance of V with $T^\pi V$. Figure 5.3 visualizes this difference. This figure is a good mental image on what each of these algorithms try to achieve.

Since the projection itself is an optimization problem (i.e., finding a function with a minimal distance to a function space), we can think of the PBE as simultaneously solving these two coupled (or nested) optimization problems:

$$\begin{aligned} V &\leftarrow \operatorname*{argmin}_{V' \in \mathcal{F}} \left\| V' - \tilde{V}(V') \right\|_{2,\mu}^2, \\ \tilde{V}(V') &\leftarrow \operatorname*{argmin}_{V'' \in \mathcal{F}} \|V'' - T^\pi V'\|_{2,\mu}^2. \end{aligned} \tag{5.13}$$

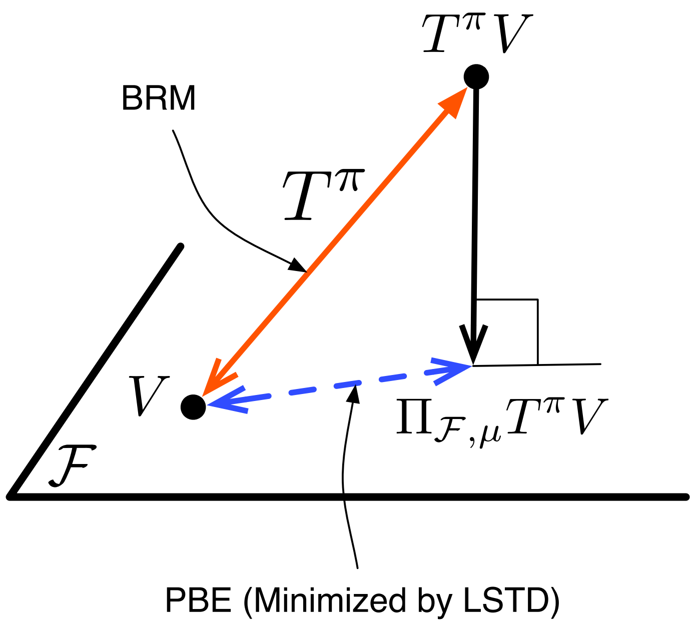

Figure 5.3: The Bellman operator may take value functions from $\mathcal{F}$ to a point outside that space. BRM finds $V \in \mathcal{F}$ with the minimal distance.

The second optimization problem finds the projection $\tilde{V}(V')$ of $T^\pi V'$ onto $\mathcal{F}$; the first optimization problem finds a $V' \in \mathcal{F}$ that is closest to $\tilde{V}(V')$, the projected function.

When $\mathcal{F}$ is a linear function space, the projection has a closed-form solution, and we can "substitute" the closed-form solution of $\tilde{V}(V')$ in the first one. For more general spaces, however, the solution may not be simple.

We remark is passing that we have regularized variants of the objectives too, which are suitable for avoiding overfitting when $\mathcal{F}$ is a very large function space.

There are different approaches to solve (5.13), some of which may not appear to be related as first glance. In the next section, we describe the main idea behind a few of them discussing the abstract problem of solving a linear system of equation. Afterwards, we get back to the PBE minimization problem (5.12) and suggest a few approaches (at the population level). These approaches would be a basis for several data-driven methods, as we shall see in Section XXX.

5.1.4.1 Solving $Ax \approx b$: A Few Approaches

Suppose that we want to solve a linear system of equations

$$Ax \approx b, \tag{5.14}$$

with $A \in \mathbb{R}^{N \times d}$, $x \in \mathbb{R}^d$, and $b \in \mathbb{R}^N$ ($N \geq d$). When $N > d$, this is an overdetermined system so the equality may not be satisfied. Nonetheless, there are several approaches to solve this overdetermined linear system of equations, at least approximately.

One is to formulate it as an optimization problem:

$$x^* \leftarrow \operatorname*{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 = (Ax - b)^\top (Ax - b). \tag{5.15}$$

We can use our favourite numerical optimizer to solve it, e.g., Gradient Descent (GD). As the gradient of $(Ax - b)^\top (Ax - b)$ is

$$A^\top (Ax - b),$$

the GD procedure would be

$$x_{k+1} \leftarrow x_k - \alpha A^\top (Ax_k - b).$$

We can use more advanced optimization techniques too. This approach finds a x^* that minimizes the squared error loss function.

We can also solve for the zero of the gradient:

$$A^\top Ax = A^\top b \Rightarrow x^* = (A^\top A)^{-1} A^\top b, \tag{5.16}$$

assuming the invertibility of $A^\top A$. For this approach, we need to have a method to invert the matrix $A^\top A$.

If $N = d$, another approach is to use a fixed-point iteration algorithm. We can rewrite $Ax = b$ as

$$(\mathbf{I} - A)x + b = x.$$

This is of the form of a fixed-point equation $Lx = x$ with $L : \mathbb{R}^d \to \mathbb{R}^d$ being the mapping $x \mapsto (\mathbf{I} - A)x + b$. If it happens that L is a contraction mapping, which is not always the case, the Banach fixed point theorem (Theorem A.1) shows that the iterative procedure

$$x_{k+1} \leftarrow Lx_k = (\mathbf{I} - A)x_k + b \tag{5.17}$$

converges to x^*, the solution of $Ax^* = b$.

It is also possible to define a slightly modified procedure of

$$x_{k+1} \leftarrow (1-\alpha)x_k + \alpha L x_k. \tag{5.18}$$

This makes it similar to the iterative update rule (4.18), from which we got the synchronous SA update rule (4.19) and later the asynchronous SA update rule (4.20) in Section 4.7.

Comparing the linear system of equations (5.14) with having $V \approx \Pi_{\mathcal{F}} T^\pi V$ (**??**), which was our starting point to obtain the PBE, suggests a few ways to find a value function. The PBE minimization (5.12) is more like the optimization-based approach (5.15). We may wonder what happens if we use a direct solution similar to (5.16) or an iterative procedure similar to (5.18). We explore these possibilities next.

Exercise 5.1. *Express the condition required for convergence of* (5.17) *as a property of A.*

Exercise 5.2. *Can you think of other ways to solve $Ax = b$ using an iterative approach?*

5.1.4.2 Least Square Temporal Difference Learning (LSTD) (Population Version)

Instead of minimizing $\|V - \Pi_{\mathcal{F}} T^\pi V\|_{2,\mu}$ over value functions $V \in \mathcal{F}$, we provide a direct solution similar to (5.16). For the rest of this section, we consider $\mathcal{F}$ to be a linear FA with basis functions (or features) $\phi_1, \dots, \phi_p$. So the value function space is $\mathcal{F} = \left\{ x \mapsto \phi(x)^\top w \, : \, w \in \mathbb{R}^p \right\}$ (5.1).

Our objective in this section is to find a value function that satisfies

$$V(x) = \left(\Pi_{\mathcal{F},\mu} T^\pi V\right)(x), \qquad \forall x \in \mathcal{X}, \tag{5.19}$$

where V is restricted to be in $\mathcal{F}$.

To make our treatment simpler, we assume that $\mathcal{X}$ is finite and has N states, potentially much larger than p. Hence, each ϕ_i is an N-dimensional vector. We denote $\Phi \in \mathbb{R}^{N \times p}$ as the matrix of concatenating all features:

$$\Phi = \left[\begin{array}{ccc} \phi_1 & \cdots & \phi_p \end{array} \right].$$

The value function corresponding to a weight $w \in \mathbb{R}^p$ is then $V_{N\times 1} = \Phi_{N\times p} w_p$ (we occasionally specify the dimension of vectors and matrices through subscripts).

Solving (5.19) when $V = V(w) = \Phi w \in \mathcal{F}$ means that we have to find a $w \in \mathbb{R}$ such that

$$\Phi w = \Pi_{\mathcal{F},\mu} T^\pi \Phi w. \tag{5.20}$$

First, we re-write this in a matrix form, and then solve it using linear algebraic manipulations.

In order to provide an explicit form for the projection operator, let us first note that the μ-weighted inner product between $V_1, V_2 \in \mathbb{R}^N$ is

$$\langle V_1 \, , \, V_2 \rangle_\mu = \sum_{x \in \mathcal{X}} V_1(x) V_2(x) \mu(x) = V_1^\top M V_2, \tag{5.21}$$

with $M = \operatorname{diag}(\mu)$. The $L_2(\mu)$-norm $\|V\|_{2,\mu}$ of $V \in \mathbb{R}^N$ can be written in the matrix form as well:

$$\|V\|_{2,\mu}^2 = \langle V \, , \, V \rangle_\mu = \sum_{x \in \mathcal{X}} |V(x)|^2 \mu(x) = V^\top M V.$$

The projection operator onto a linear $\mathcal{F}$ would be

$$\begin{aligned} \Pi_{\mathcal{F},\mu} V &= \operatorname*{argmin}_{V' \in \mathcal{F}} \|V' - V\|_{2,\mu}^2 \\ &= \operatorname*{argmin}_{w \in \mathbb{R}^p} \|\Phi w - V\|_{2,\mu}^2 \\ &= \operatorname*{argmin}_{w \in \mathbb{R}^p} (\Phi w - V)^\top M (\Phi w - V). \end{aligned}$$

Taking the derivative and setting it to zero, we get that

$$\Phi^\top M (\Phi w - V) = 0 \Rightarrow w = (\Phi^\top M \Phi)^{-1} \Phi^\top M V,$$

assuming that $\Phi^\top M \Phi$ is invertible. Therefore, the projected function is Φw, i.e.,

$$\Pi_{\mathcal{F},\mu} V = \Phi (\Phi^\top M \Phi)^{-1} \Phi^\top M V. \tag{5.22}$$

We also have

$$(T^\pi \Phi w)_{N \times 1} = r^\pi_{N \times 1} + \gamma \mathcal{P}^\pi_{N \times N} \Phi_{N \times p} w_p.$$

Combining all these, we get that (5.19) in the matrix form is

$$\Phi w = \left[\Phi (\Phi^\top M \Phi)^{-1} \Phi^\top M \right] \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w \right]. \tag{5.23}$$

Two approaches, parallel to (5.16) and (5.18), are to either solve this directly or use an iterative solver based on the fixed-point iteration procedure.

Let us start with the direct solution approach.

To simplify the equation, we use the following result.

Proposition 5.3. *If $\Phi A x^* = \Phi b$ and $\Phi^\top M \Phi$ is invertible, we have $A x^* = b$.*

Proof. Multiply both sides of $\Phi A x^* = \Phi b$ by $\Phi^\top M$ to get

$$\Phi^\top M \Phi A x^* = \Phi^\top M \Phi b.$$

As $\Phi^\top M \Phi$ is assumed to be invertible, we have

$$A x^* = (\Phi^\top M \Phi)^{-1} \Phi^\top M \Phi b = b.$$

□

Based on this proposition, we multiply both sides of (5.23) by $\Phi^\top M$ and simplify as follows:

$$\begin{aligned} \Phi^\top M \Phi w &= \Phi^\top M \Phi (\Phi^\top M \Phi)^{-1} \Phi^\top M \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w \right] \\ &= \Phi^\top M \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w \right]. \\ \Rightarrow & \Phi^\top M \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w - \Phi w \right] = 0. \end{aligned} \tag{5.24}$$

We re-arrange this to

$$\left[\Phi^\top M \Phi - \gamma \Phi^\top M \mathcal{P}^\pi \Phi \right] w = \Phi^\top M r^\pi.$$

Solving for w, we have

$$w = \left[\Phi^\top M (\Phi - \gamma \mathcal{P}^\pi \Phi) \right]^{-1} \Phi^\top M r^\pi. \tag{5.25}$$

This fixed-point equation shall be used to define the *Least Square Temporal Difference (LSTD)* method. This solution can be seen as the population version of it, so with some non-standard naming convention, we refer to it as LSTD (Population) as well.

Equation (5.24) provides a geometric viewpoint to what LSTD does. Comparing with (5.21), this equation requires us to have

$$\langle \phi_i \, , T^\pi V(w) - V(w) \rangle_\mu = 0, \qquad \forall i = 1, \dots, p.$$

As $\text{BR}(V(w)) = T^\pi V(w) - V(w)$, this is also equivalent to

$$\langle \phi_i \, , \text{BR}(V(w)) \rangle_\mu = 0, \qquad \forall i = 1, \dots, p.$$

So we are aiming to find a w such that the Bellman Residual $\text{BR}(V(w))$ is orthogonal to the basis of $\mathcal{F}$, when the orthogonality is measured according to the distribution μ.

5.1.4.3 Fixed Point Iteration for Projected Bellman Operator

Let us design two iterative approaches for finding the fixed-point of $\Pi_{\mathcal{F},\mu}T^{\pi}$, see (5.10). We attempt to design methods that look like an SA iteration, so when we deal with the samples, instead of the true model, they can handle the noise. We specifically focus on the case when the distribution μ is the stationary distribution ρ^{π} of π.

Approach #1 Consider

$$\hat{V}_{k+1} \leftarrow (1-\alpha)\hat{V}_k + \alpha \Pi_{\mathcal{F},\rho^{\pi}} T^{\pi} \hat{V}_k, \tag{5.26}$$

with an $0 < \alpha \leq 1$. This can be seen as a fixed-point iterative method with the operator

$$L : V \mapsto \left[(1-\alpha)\mathbf{I} + \alpha \Pi_{\mathcal{F},\rho^{\pi}} T^{\pi}\right] V.$$

This operator is a contraction w.r.t. $L_2(\rho^{\pi})$. To see this, by the triangle inequality and the linearity of the projection operator $\Pi_{\mathcal{F},\rho^{\pi}}$ and the Bellman operator T^{π}, we have

$$\left\| LV_1 - LV_2 \right\|_{2,\rho^{\pi}} \leq (1-\alpha) \left\| V_1 - V_2 \right\|_{2,\rho^{\pi}} + \alpha \left\| \Pi_{\mathcal{F},\rho^{\pi}} T^{\pi} (V_1 - V_2) \right\|_{2,\rho^{\pi}}. \tag{5.27}$$

The norm in the second term on the RHS can be upper bounded by noticing that the projection operator $\Pi_{\mathcal{F},\rho^{\pi}}$ is non-expansive w.r.t. the $L_2(\rho^{\pi})$ and the Bellman operator T^{π} is γ-contraction w.r.t. the same norm (Lemma 5.1):

$$\begin{aligned} \left\| \Pi_{\mathcal{F},\rho^{\pi}} T^{\pi} (V_1 - V_2) \right\|_{2,\rho^{\pi}} &\leq \left\| T^{\pi} (V_1 - V_2) \right\|_{2,\rho^{\pi}} \\ &\leq \gamma \left\| V_1 - V_2 \right\|_{2,\rho^{\pi}}. \end{aligned} \tag{5.28}$$

This along with (5.27) shows that

$$\left\| LV_1 - LV_2 \right\|_{2,\rho^{\pi}} \leq \left[(1-\alpha) + \alpha\gamma\right] \left\| V_1 - V_2 \right\|_{2,\rho^{\pi}}.$$

If $0 < \alpha \leq 1$, L is a contraction.

Therefore, the iterative method (5.26) is going to be convergent. Note that its projection operator is w.r.t. $\left\| \cdot \right\|_{2,\rho^{\pi}}$, the L_2-norm induced by the stationary distribution of π. The convergence property may not hold for other $\mu \neq \rho^{\pi}$.

Let us use a linear FA, defined by the set of features ϕ to write $\hat{V}_k = \Phi w_k$. With a linear FA, we can use the explicit formula (5.22) for the projection operator $\Pi_{\mathcal{F},\rho^{\pi}}$.

We use $D = \text{diag}(\rho^\pi)$, instead of M, in order to emphasize the dependence on ρ^π. The iteration (5.26) can be written as

$$\hat{V}_{k+1} = \Phi w_{k+1} \leftarrow (1-\alpha)\Phi w_k + \alpha \Phi (\Phi^\top D^\pi \Phi)^{-1} \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k \right].$$

Multiply both sides by $\Phi^\top D^\pi$, we get

$$(\Phi^\top D^\pi \Phi) w_{k+1} \leftarrow (1-\alpha)(\Phi^\top D^\pi \Phi) w_k + \alpha (\Phi^\top D^\pi \Phi)(\Phi^\top D^\pi \Phi)^{-1} \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k \right].$$

Assuming that $\Phi^\top D^\pi \Phi$ is invertible, we can write the dynamics of (w_k) as

$$w_{k+1} \leftarrow (1-\alpha) w_k + \alpha (\Phi^\top D^\pi \Phi)^{-1} \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k \right]. \tag{5.29}$$

This is a convergent iteration and converges to the fixed point of $\Phi w = \Pi_{\mathcal{F},\mu} T^\pi \Phi w$ (5.19). This is the same as the LSTD's solution (5.25).

A potential challenge with this approach is that it requires a one-time inversion of a $p \times p$ matrix $(\Phi^\top D^\pi \Phi) = \sum_x \rho^\top(x) \phi^\top(x) \phi(x)$, which is $O(p^3)$ operation[4] (a naive approach). When we move to the online setting, where this matrix itself is updated as every new data point arrives, a naive approach of updating the matrix and re-computing its inverse, would be costly. There are ways to improve the efficiency of the computation of the inversion, which we shall discuss later.

Exercise 5.3. *Why didn't we use the supremum norm, instead of the $L_2(\rho^\pi)$ in (5.28), in showing that $\Pi_{\mathcal{F},\rho^\pi} T^\pi$ is a contraction? We know that T^π is a contraction w.r.t. the supremum norm after all.*

Approach #2 (First Order). Another iterative method can be obtained from (5.24):

$$\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w - \Phi w \right] = 0. \tag{5.30}$$

A solution to this is the same as the LSTD solution. As mentioned earlier, this is aiming to find w such that the Bellman residual is orthogonal to each ϕ_i, weighted according to ρ^π.

To define an iterative procedure, note that if $Lw = 0$, we also have $\alpha L w = 0$. Adding an identity to both sides does not change the equation, so we have

$$w + \alpha L w = w.$$

[4]This complexity is not optimal. The Strassen algorithm has the computational complexity of $O(n^{2.807})$. The Coppersmith-Winograd algorithm has the complexity of $O(n^{2.375})$, though it is not a practical algorithm.

This is in the form of a fixed-point equation for a new operator $L' : w \mapsto (\mathbf{I} + \alpha L)w$. The fixed point of L' is the same as the solution of $Lw = 0$. So we may apply $w_{k+1} \leftarrow L'w_k = (\mathbf{I} + \alpha L)w_k$, assuming L' is a contraction.

If we choose $L : w \mapsto \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w - \Phi w \right]$, we get the following iterative procedure, which is somewhat similar to (5.18):

$$\begin{aligned} w_{k+1} \leftarrow & w_k + \alpha D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k - \Phi w_k \right] \\ =& (\mathbf{I} - \alpha A) w_k + \alpha \Phi^\top D^\pi r^\pi, \end{aligned} \tag{5.31}$$

with

$$A = \Phi^\top D^\pi (\mathbf{I} - \gamma \mathcal{P}^\pi) \Phi.$$

This iterative procedure is not a convex combination of $\Pi_{\mathcal{F},\rho^\pi} T^\pi$ with the identity matrix, as (5.26) was, so the condition for convergence does not follow from what we had before. Despite that, we can show that for small enough value of α, it is convergent.

5.1.4.4 Convergence of the Fixed Point Iteration (5.31)

Proposition 5.4. *Assume that $\rho^\pi > 0$. There exists $\alpha_0 > 0$ such that for any step size $\alpha < \alpha_0$, the iterative procedure (5.31) is convergent to the fixed point of $\Pi_{\mathcal{F},\rho^\pi} T^\pi$.*

Proof. For the dynamical system (5.31) to be convergent (or stable), we need to have all the eigenvalues of $\mathbf{I} - \alpha A$ to be within the unit circle (in the complex plane), i.e.,

$$|\lambda(\mathbf{I} - \alpha A)) < 1.$$

The eigenvalues λ_i of $\mathbf{I} - \alpha A$ are located at $1 - \alpha \lambda_i(A)$, where $\lambda_i(A)$ is the corresponding eigenvalue of A. If A has any negative eigenvalue, the value of $1 - \alpha \lambda_i(A)$ would be outside the unit circle (for any $\alpha > 0$). Therefore, all eigenvalues of A should be positive. Moreover, α should be small enough such that $|1 - \alpha \lambda_i(A)| < 1$ for all i.

Let us establish that A is a positive definite matrix, in the sense that for any $y \in \mathbb{R}^p$,

$$y^\top A y > 0.$$

This entails the positiveness of the eigenvalues of A.

Instead of showing

$$y^\top A y > 0 = y^\top \Phi D^\pi (\mathbf{I} - \gamma \mathcal{P}^\pi) \Phi y > 0, \tag{5.32}$$

we can show that for any $z \in \mathbb{R}^N$,

$$z^\top D^\pi (\mathbf{I} - \gamma \mathcal{P}^\pi) z > 0. \tag{5.33}$$

If the latter is true for any z, it is also true for $z = \Phi y$, which guarantees (5.32).

For the first term, as D^π is a diagonal matrix $\mathrm{diag}(\rho^\pi)$, we have

$$z^\top D^\pi z = \sum_{i,j} z_i D^\pi_{i,j} z_j = \sum_i z_i^2 \rho_i^\pi = \|z\|^2_{2,\rho^\pi} > 0. \tag{5.34}$$

For the negative term $-\gamma z^\top D^\pi \mathcal{P}^\pi z$, we show that its size is not too large (and in fact, smaller than $\|z\|^2_{2,\rho^\pi}$), so the whole summation remains positive. Consider $z^\top D^\pi \mathcal{P}^\pi z$. We have that

$$[D^\pi \mathcal{P}]_{ij} = \sum_{i,k} D^\pi_{ik} \mathcal{P}^\pi_{kj} = \rho_i^\pi \mathcal{P}^\pi_{ij}.$$

So

$$z^\top D^\pi \mathcal{P}^\pi z = \sum_{i,j} z_i [D^\pi \mathcal{P}]_{ij} z_j = \sum_{i,j} z_i \rho_i^\pi \mathcal{P}^\pi_{ij} z_j. \tag{5.35}$$

We upper bound this using the Cauchy-Schwarz inequality:[5]

$$\sum_{i,j} z_i \rho_i^\pi \mathcal{P}^\pi_{ij} z_j = \sum_{i,j} z_i \sqrt{\rho_i^\pi} \sqrt{\rho_i^\pi} \mathcal{P}^\pi_{ij} z_j = \sum_i \underbrace{z_i \sqrt{\rho_i^\pi}}_{\triangleq a_i} \underbrace{\sqrt{\rho_i^\pi} \sum_j \mathcal{P}^\pi_{ij} z_j}_{\triangleq b_j}$$

$$\leq \sqrt{\sum_i z_i^2 \rho_i^\pi} \sqrt{\sum_i \rho_i^\pi \left(\sum_j \mathcal{P}^\pi_{ij} z_j \right)^2}. \tag{5.36}$$

We upper bound the second term in the RHS. By Jensen's inequality and the stationarity of ρ^π (which entails that $\rho_j^\pi = \sum_i \rho_i^\pi \mathcal{P}^\pi_{ij}$), we obtain

$$\sum_i \rho_i^\pi \left(\sum_j \mathcal{P}^\pi_{ij} z_j \right)^2 \leq \sum_i \rho_i^\pi \sum_j \mathcal{P}^\pi_{ij} z_j^2 = \sum_j z_j^2 \sum_i \rho_i^\pi \mathcal{P}^\pi_{ij} = \sum_j z_j^2 \rho_j^\pi = \|z\|^2_{2,\rho^\pi}.$$

This, along with (5.36), allows us to upper bound (5.35) by

$$z^\top D^\pi \mathcal{P}^\pi z \leq \|z\|^2_{2,\rho^\pi}$$

[5]For summation, the Cauchy-Schwarz inequality state that $\sum_i a_i b_j \leq \sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}$.

Plugging this inequality and (5.34) in (5.33), we get that for any $z \neq 0$ and $\rho^\pi > 0$,

$$z^\top D^\pi(\mathbf{I} - \gamma \mathcal{P}^\pi) z \geq (1-\gamma) \left\| z \right\|_{2,\rho^\pi}^2 > 0.$$

This also shows that $y^\top A y > 0$, as argued earlier. The positive definiteness of A entails that all of its eigenvalues have positive real components.[6]

We can find a step size α such that the eigenvalues of $\mathbf{I} - \alpha A$ are within unit circle. To see this, suppose that $\lambda_i(A) = a_i + j b_i$ with $a_i > 0$. We need to have $|1 - \alpha \lambda_i(A)| < 1$ for all $i = 1, \dots, p$. We expand

$$\left|1 - \alpha \lambda_i(A)\right|^2 < 1 \Leftrightarrow |1 - \alpha a_i|^2 + |\alpha b|^2 = 1 + (\alpha a_i)^2 - 2\alpha a_i + (\alpha b_i)^2 < 1.$$

After simplifications, we get that the inequality is ensured if

$$\alpha < \frac{2a_i}{a_i^2 + b_i^2} = \frac{2a_i}{|\lambda_i(A)|^2}.$$

As $a_i > 0$, we can always pick an α to satisfy these conditions (for $i = 1, \dots, p$) by selecting it such that

$$\alpha < \alpha_0 = \min_{i=1,\dots,p} \frac{2a_i}{a_i^2 + b_i^2} = \frac{2\mathbf{Re}[\lambda_i(A)]}{|\lambda_i(A)|^2}.$$

□

5.1.4.5 Error Bound on the LSTD Solution

Suppose that we find a value function V that is the fixed point of the projected Bellman error w.r.t. ρ^π, i.e.,

$$V = \Pi_{\mathcal{F},\rho^\pi} T^\pi V.$$

For the linear FA, the LSTD method (population) (5.25) and the fixed point iterations (5.26) and (5.31) find this solution. Let us call this the TD solution V_{TD}. How close is this value function to the true value function V^π?

If the value function space $\mathcal{F}$ cannot represent V^π precisely, which is often the case under function approximation, we cannot expect to have a small error. The smallest error we can hope is $\|\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi\|$, the distance between V^π and its projection. The TD solution is not as close to V^π as the projection of V^π onto $\mathcal{F}$, but it can be close to that. The next result provides an upper bound for the quality of the TD solution.

[6]Refer to `https://math.stackexchange.com/a/325412` for a proof.

Proposition 5.5. *If ρ^π is the stationary distribution of π, we have*

$$\left\| V_{TD} - V^\pi \right\|_{2,\rho^\pi} \leq \frac{\left\| \Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi \right\|_{2,\rho^\pi}}{\sqrt{1-\gamma^2}}.$$

Proof. As $V_{\text{TD}} = \Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}}$, we have

$$\begin{aligned} V_{\text{TD}} - V^\pi &= V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi + \Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi \\ &= \Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi + \Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi. \end{aligned}$$

Take the absolute values of both sides, square it, and take the integral w.r.t. ρ^π to get that

$$\begin{aligned} \left\| V_{\text{TD}} - V^\pi \right\|_{2,\rho^\pi}^2 = &\left\| \Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi \right\|_{2,\rho^\pi}^2 + \left\| \Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi \right\|_{2,\rho^\pi}^2 + \\ &2 \left\langle \Pi_{\mathcal{F},\rho^\pi} V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi \, , \, \Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi \right\rangle_{\rho^\pi} . \end{aligned} \tag{5.37}$$

Consider the inner product term. The vector $\Pi_{\mathcal{F},\rho^\pi} V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi$ is a member of $\mathcal{F}$. The vector $\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi$ is the difference between V^π and its projection (or the orthogonal complement of V^π), so it is orthogonal to to $\mathcal{F}$. Therefore, their inner product is zero.

We provide an upper bound on $\left\| \Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi \right\|_{2,\rho^\pi}^2$. We replace V^π with $T^\pi V^\pi$, and benefit from the non-expansiveness of $\Pi_{\mathcal{F},\rho^\pi}$ and the γ-contraction of T^π (Lemma 5.1) and (both w.r.t. $L_2(\rho^\pi)$) to get that

$$\begin{aligned} \left\| \Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi \right\|_{2,\rho^\pi}^2 &= \left\| \Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} T^\pi V^\pi \right\|_{2,\rho^\pi}^2 \\ &\leq \left\| T^\pi V_{\text{TD}} - T^\pi V^\pi \right\|_{2,\rho^\pi}^2 \\ &\leq \gamma^2 \left\| V_{\text{TD}} - V^\pi \right\|_{2,\rho^\pi}^2 . \end{aligned}$$

Therefore, we can upper bound (5.37) as

$$\left\| V_{\text{TD}} - V^\pi \right\|_{2,\rho^\pi}^2 \leq \gamma^2 \left\| V_{\text{TD}} - V^\pi \right\|_{2,\rho^\pi}^2 + \left\| \Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi \right\|_{2,\rho^\pi}^2 ,$$

which leads to the desired result after a re-arrangement. □

5.2 Batch RL Methods

We use ideas developed in the previous section to develop RL algorithms that work with function approximators. The key step is to find an empirical version of the relevant quantities and estimate them using data. For example, many of the aforementioned methods require the computation of TV. If the model is not known, this cannot be computed. We have to come up with a procedure that estimate TV based only on data.

In this section, we consider the *batch* data setting. Here the data is already collected, and we are interested in using it to estimate quantities such as Q^π or Q^*. To be concrete, suppose that we have

$$\mathcal{D}_n = \{(X_i, A_i, R_i, X'_i)\}_{i=1}^n, \tag{5.38}$$

with $(X_i, A_i) \sim \mu \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$, and $X'_i \sim \mathcal{P}(\cdot|X_i, A_i)$ and $R_i \sim \mathcal{R}(\cdot|X_i, A_i)$. The data could be generated by following a behaviour policy π_b and having trajectories in the form of $(X_1, A_1, R_1, X_2, A_2, R_2, \dots)$. In this case, $X'_t = X_{t+1}$.

In the batch setting, the agent does not interact with the environment while it is computing Q^π, Q^*, etc. This can be contrasted with the online method such as TD or Q-Learning, where the agent updates its estimate of the value function as each data point arrives.

Of course, the boundary between the batch and online methods is blurry. We might have methods that collect a batch of data, process them, and then collect a new batch of data, possibly based on a policy resulted from the previous batch processing computation.

In this section, we develop several batch RL methods based on what we learned in the previous section. In the next section, we develop some online RL methods.

5.2.1 Value Function Approximation Given the Monte Carlo Estimates

Suppose that we are given a batch of data in the form of

$$\mathcal{D}_n = \{(X_i, A_i, G^\pi(X_i, A_i))\}_{i=1}^n,$$

with $G^\pi(X_i, A_i)$ being a return of being at state X_i, taking action A_i, and following the policy π afterwards. Here we suppose that the distribution of $(X_i, A_i) \sim \mu$. The return can be obtained using the initial-state only MC by selecting $(X_i, A_i) \sim \mu$ and then following π until the end of episode (in the episodic case). Or we can extract this information from the trajectory obtained by following the policy π using the first-visit MC (Section 4.3).

We encountered the following population loss function in Section 5.1.1 (but for V instead of Q):

$$Q \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - Q^{\pi}\|_{2,\mu} . \tag{5.39}$$

There are two differences with the current setup. The first is that we do not have a direct access to the distribution μ and only have samples from it. The second is that we do not know Q^{π} itself and only we have unbiased estimate G^{π} at a finite number of data points.

Let us focus on the latter issue. We show that having access to unbiased noisy estimate of Q^{π} does not change the solution of the minimization problem. For any Q, we can decompose the expectation $\mathbb{E}\left[|Q(X,A) - G^{\pi}(X,A)|^2\right]$ (with $(X,A) \sim \mu$), as

$$\begin{aligned}
\mathbb{E}\left[|Q(X,A) - G^{\pi}(X,A)|^2\right] &= \mathbb{E}\left[|Q(X,A) - Q^{\pi}(X,A) + Q^{\pi}(X,A) - G^{\pi}(X,A)|^2\right] \\
&= \mathbb{E}\left[|Q(X,A) - Q^{\pi}(X,A)|^2\right] + \\
&\quad \mathbb{E}\left[|Q^{\pi}(X,A) - G^{\pi}(X,A)|^2\right] + \\
&\quad 2\mathbb{E}\left[(Q(X,A) - Q^{\pi}(X,A))\left(Q^{\pi}(X,A) - G^{\pi}(X,A)\right)\right] .
\end{aligned}$$

The first term is $\|Q - Q^{\pi}\|_{2,\mu}$, the same as the population loss (5.39). The second term is

$$\mathbb{E}\left[\operatorname{Var}\left[G^{\pi}(X,A) \mid X, A\right]\right],$$

the average (w.r.t. μ) of the (conditional) variance of the return. Note that this is not a function of Q. Let us consider the inner product term. We take the conditional expectation w.r.t. (X,A) and use the fact that $Q(X,A)$ and $Q^{\pi}(X,A)$ are measurable functions of (X,A) to get[7]

$$\begin{aligned}
&\mathbb{E}\left[(Q(X,A) - Q^{\pi}(X,A))\left(Q^{\pi}(X,A) - G^{\pi}(X,A)\right)\right] = \\
&\mathbb{E}\left[\mathbb{E}\left[(Q(X,A) - Q^{\pi}(X,A))\left(Q^{\pi}(X,A) - G^{\pi}(X,A)\right) \mid X, A\right]\right] = \\
&\mathbb{E}\left[(Q(X,A) - Q^{\pi}(X,A))\left(Q^{\pi}(X,A) - \mathbb{E}\left[G^{\pi}(X,A) \mid X, A\right]\right)\right] .
\end{aligned}$$

As $\mathbb{E}\left[G^{\pi}(X,A) \mid X, A\right]$ is equal to $Q^{\pi}(X,A)$, the inside of second parenthesis in the last equality is zero. So the value of this whole term is zero.

[7]Here we use the property of the conditional expectation that if f is an X-measurable function, $\mathbb{E}\left[f(X)Z|X\right] = f(X)\mathbb{E}\left[Z|X\right]$. In words, conditioned on X, there is no randomness in $f(X)$, so it can be taken out of the expectation.

Therefore, we have that

$$\begin{aligned}
&\operatorname*{argmin}_{Q \in \mathcal{F}} \mathbb{E}\left[|Q(X,A) - G^{\pi}(X,A)|^2 \right] = \\
&\operatorname*{argmin}_{Q \in \mathcal{F}} \left\{ \mathbb{E}\left[|Q(X,A) - Q^{\pi}(X,A)|^2 \right] + \mathbb{E}\left[\mathrm{Var}\left[G^{\pi}(X,A) \mid X,A \right] \right] \right\} = \\
&\operatorname*{argmin}_{Q \in \mathcal{F}} \left\| Q - Q^{\pi} \right\|_{2,\mu}^2 ,
\end{aligned}$$

as the variance term $\mathbb{E}\left[\mathrm{Var}\left[G^{\pi}(X,A) \mid X,A\right]\right]$ is not a function of Q, so it does not change the minimizer. If we could find the minimizer of $\mathbb{E}\left[|Q(X,A) - G^{\pi}(X,A)|^2\right]$, the solution would be the same as the minimizer of (5.39).

Nonetheless, we cannot compute the expectation because we do not know μ, as already mentioned. We only have samples from it. A common solution in ML to address this issue is to use the *empirical risk minimization (ERM)*, which prescribes that we solve

$$\hat{Q} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} |Q(X_i, A_i) - G^{\pi}(X_i, A_i)|^2 . \tag{5.40}$$

This is indeed a regression problem with the squared error loss.

This loss function can be seen as using the *empirical measure* μ_n instead of μ. The empirical measure, given data $\{(X_i, A_i)\}_{i=1}^n$, is defined as the probability distribution that assigns the following probability to any (measurable) set $B \subset \mathcal{X} \times \mathcal{A}$:

$$\mu_n(B) \triangleq \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{(X_i, A_i) \in B\}.$$

With this notation, the optimization above can be written as

$$\hat{Q} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \left\| Q - G^{\pi} \right\|_{2,\mu_n}^2 .$$

We occasionally may use $\|Q - G^{\pi}\|_{2,\mathcal{D}_n}^2$ or $\|Q - G^{\pi}\|_{2,n}^2$ to denote this norm (see Appendix A.2).

How close is this $\hat{Q}$ to the minimizer of $\|Q - Q^{\pi}\|_{2,\mu}$? And how close is it going to be from Q^{π}? What is the effect of the number of samples n? And what about the effect of the function space $\mathcal{F}$?

Studying such questions is the topic of *statistical learning theory (SLT)*. In this particular case, when the problem reduces to a conventional regression problem, we can benefit from the standard results in SLT. We only provide a simplified analysis, which do not require much previous background. You can refer to Györfi et al. [2002] for detailed study of several *nonparametric* regression methods, van de Geer [2000] for thorough development of the *empirical processes*, required for the analysis of SLT methods, and [Steinwart and Christmann, 2008] for the analysis of SVM-style algorithms and RKHS function spaces.

Assumption A2 We assume that

(a) $Z_i = (X_i, A_i)$ $(i = 1, \dots, n)$ are i.i.d. samples from distribution $\mu \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$.

(b) The reward distribution $\mathcal{R}(\cdot|x, a)$ is $R_{\max}$-bounded for any $(x, a) \in \mathcal{X} \times \mathcal{A}$.

(c) The functions in $\mathcal{F}$ are $Q_{\max} = \frac{R_{\max}}{1-\gamma}$ bounded.

(d) The function space has a finite number of members $|\mathcal{F}| < \infty$.

Among all these assumptions, Assumption A2(d) is not realistic, as the function space that we deal with have uncountably infinite number of members. This is, however, only for the simplification of analysis, and allows us to prove a result without using tools from the empirical process theory. This simplification is not completely useless though. In fact, some of the results that can deal with an uncountably infinite function space $\mathcal{F}$ are based on first covering the original function space by finite-sized $\mathcal{F}_\varepsilon$ and then showing that the finite covering incurs a controlled amount of error $O(\varepsilon)$. This is called the *covering argument*.

Theorem 5.6 (Regression)**.** *Consider the solution $\hat{Q}$ returned by solving (5.40). Suppose that Assumption A2 holds. For any $\delta > 0$, we then have*

$$\left\| \hat{Q} - Q^\pi \right\|_{2,\mu}^2 \leq \inf_{Q \in \mathcal{F}} \| Q - Q^\pi \|_{2,\mu}^2 + 8 Q_{max}^2 \sqrt{\frac{2(\ln(6|\mathcal{F}|) + 2\ln(1/\delta)}{n}},$$

with probability at least $1 - \delta$.

Proof. □

5.2.2 Approximate Value Iteration

Designing an empirical version of AVI (Section 5.1.2) is easy. The iteration of (5.3),

$$Q_{k+1} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \| Q - TQ_k \|_{2,\mu} = \int |Q(x, a) - (TQ)(x, a)|^2 \, \mathrm{d}\mu(x, a), \qquad (5.41)$$

cannot be computed as (1) μ is not known, and (2) TQ_k cannot be computed as $\mathcal{P}$ is not available under the batch RL setting (5.38). We can use samples though, similar to how we converted the population-level loss function (5.39) to the empirical one (5.40) in Section 5.1.1. Let us do it in a few steps.

First, note that if we are only given tuples in the form of (X, A, R, X'), we cannot compute

$$(T^\pi Q)(X, A) = r(X, A) + \gamma \int \mathcal{P}(\mathrm{d}x'|X, A) Q(x', \pi(x')),$$

or similar for $(T^* Q)(X, A)$. We can, however, form an unbiased estimate of them. We use the empirical Bellman operator applied to Q (4.9) and (4.13), which would be

$$\begin{aligned}(\hat{T}^\pi Q)(X, A) &= R + \gamma Q(X', \pi(X')),\\ (\hat{T}^* Q)(X, A) &= R + \gamma \max_{a' \in \mathcal{A}} Q(X', a').\end{aligned}$$

For any integrable Q, they satisfy

$$\mathbb{E}\left[(\hat{T} Q)(X, A)|X, A\right] = (TQ)(X, A),$$

for any (X, A).

Similar to Section 5.2.1, replacing TQ_k with $\hat{T} Q_k$ does not change the optimizer. Given any $Z = (X, A)$, we have

$$\begin{aligned}\mathbb{E}\left[\left|Q(Z) - (\hat{T} Q_k)(Z)\right|^2 \mid Z\right] &= \mathbb{E}\left[\left|Q(Z) - (TQ_k)(Z) + (TQ_k)(Z) - (\hat{T} Q_k)(Z)\right|^2 \mid Z\right]\\ &= \mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2 \mid Z\right] +\\ &\quad \mathbb{E}\left[\left|(TQ_k)(Z) - (\hat{T} Q_k)(Z)\right|^2 \mid Z\right] +\\ &\quad 2\mathbb{E}\left[(Q(Z) - (TQ_k)(Z))\left((TQ_k)(Z) - (\hat{T} Q_k)(Z)\right) \mid Z\right].\end{aligned}$$

As $\mathbb{E}\left[(\hat{T} Q_k)(Z) \mid Z\right] = TQ_k(Z)$, the last term is zero. Also conditioned on Z, the function $Q(Z) - (TQ_k)(Z)$ is not random, so $\mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2 \mid Z\right] = |Q(Z) - (TQ_k)(Z)|^2$. Therefore, we get that

$$\begin{aligned}\mathbb{E}\left[\left|Q(Z) - (\hat{T} Q_k)(Z)\right|^2 \mid Z\right] &= |Q(Z) - (TQ_k)(Z)|^2 + \mathbb{E}\left[\left|(TQ_k)(Z) - (\hat{T} Q_k)(Z)\right|^2 \mid Z\right]\\ &= |Q(Z) - (TQ_k)(Z)|^2 + \mathrm{Var}\left[(\hat{T} Q_k)(Z) \mid Z\right]. \quad (5.42)\end{aligned}$$

Taking expectation over $Z \sim \mu$, we have that

$$\begin{aligned}\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2\right] &= \mathbb{E}\left[\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2 \mid Z\right]\right] \\ &= \mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2\right] + \mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q_k)(Z) \mid Z\right]\right].\end{aligned}$$

The term $\mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2\right]$ is $\|Q - TQ_k\|_{2,\mu}^2$. So we get that

$$\operatorname*{argmin}_{Q \in \mathcal{F}} \mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2\right] = \tag{5.43}$$

$$\operatorname*{argmin}_{Q \in \mathcal{F}} \left\{\|Q - TQ_k\|_{2,\mu}^2 + \mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q_k)(Z) \mid Z\right]\right]\right\} = \tag{5.44}$$

$$\operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - TQ_k\|_{2,\mu}^2 , \tag{5.45}$$

as the term $\mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q_k)(Z) \mid Z\right]\right]$ is not a function of Q, hence it does not change the minimizer. So instead of (5.41), we can minimize $\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2\right]$.

We do not have μ though. As before we can use samples and form the empirical loss function. The result is the following ERM problem:

$$\hat{Q} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left|Q(X_i, A_i) - (\hat{T}Q_k)(X_i, A_i)\right|^2 = \left\|Q - \hat{T}Q_k\right\|_{2,\mathcal{D}_n}^2 . \tag{5.46}$$

This is the AVI procedure, also known as the *Fitted Value Iteration (FVI)* or *Fitted Q Iteration (FQI)* algorithm. This is the basis of the Deep Q-Network (DQN) algorithm, where one uses a DNN to represent the value function space.

5.2.3 Bellman Residual Minimization

We may use data to define the empirical version of BRM (Section 5.1.3). Instead of

$$Q \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - T^{\pi}Q\|_{2,\mu}^2 , \tag{5.47}$$

we can solve

$$Q \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left|Q(X_i, A_i) - (\hat{T}^{\pi}Q)(X_i, A_i)\right|^2 = \left\|Q - \hat{T}^{\pi}Q\right\|_{2,\mathcal{D}_n}^2 . \tag{5.48}$$

Here we used data $\mathcal{D}_n$ to both convert the integration w.r.t. μ to an integration w.r.t. μ_n (or summation), and substitute $T^\pi Q$ to its empirical version $\hat{T}^\pi Q$. Note that we have Q in both terms inside the norm, as opposed to only the first term in FQI (5.46). This appearance of Q in both terms causes an issue though: the minimizer of $\|Q - T^\pi Q\|_{2,\mu}^2$ and $\|Q - \hat{T}^\pi Q\|_{2,\mu}^2$ are not necessarily the same for stochastic dynamics. To see this, we compute $\mathbb{E}\left[|Q(Z) - (\hat{T}^\pi Q)(Z)|^2 \mid Z\right]$ for any Q and $Z = (X, A)$ (by conditioning on Z, the randomness in the expectation is over the next-state X', and not the choice of Z) as follows:

$$
\begin{aligned}
\mathbb{E}\left[\left|Q(Z) - (\hat{T}^\pi Q)(Z)\right|^2 \mid Z\right] &= \mathbb{E}\left[\left|Q(Z) - (T^\pi Q)(Z) + (T^\pi Q)(Z) - (\hat{T}^\pi Q)(Z)\right|^2 \mid Z\right] \\
&= \mathbb{E}\left[|Q(Z) - (T^\pi Q)(Z)|^2 \mid Z\right] + \\
&\quad \mathbb{E}\left[\left|(T^\pi Q)(Z) - (\hat{T}^\pi Q)(Z)\right|^2 \mid Z\right] + \\
&\quad 2\mathbb{E}\left[(Q(Z) - (T^\pi Q)(Z))\left((T^\pi Q)(Z) - (\hat{T}^\pi Q)(Z)\right) \mid Z\right].
\end{aligned}
$$

Given Z, there is no randomness in $|Q(Z) - (T^\pi Q)(Z)|^2$, so $\mathbb{E}\left[|Q(Z) - (T^\pi Q)(Z)|^2 \mid Z\right] = |Q(Z) - (T^\pi Q)(Z)|^2$. Moreover, the inner product term is zero because it is equal to

$$
\begin{aligned}
&(Q(Z) - (T^\pi Q)(Z))\left((T^\pi Q)(Z) - \mathbb{E}\left[(\hat{T}^\pi Q)(Z) \mid Z\right]\right) = \\
&(Q(Z) - (T^\pi Q)(Z))\left((T^\pi Q)(Z) - (T^\pi Q)(Z)\right) = 0.
\end{aligned}
$$

Therefore,

$$
\mathbb{E}\left[\left|Q(Z) - (\hat{T}^\pi Q)(Z)\right|^2 \mid Z\right] = |Q(Z) - (T^\pi Q)(Z)|^2 + \mathrm{Var}\left[(\hat{T}^\pi Q)(Z) \mid Z\right].
$$

In contrast with AVI/FQI (5.42), the second term is a function of Q too. By taking the expectation w.r.t. $Z \sim \mu$, we get that

$$
\begin{aligned}
\operatorname*{argmin}_{Q \in \mathcal{F}} \mathbb{E}\left[\left|Q(Z) - (\hat{T}Q)(Z)\right|^2\right] &= \operatorname*{argmin}_{Q \in \mathcal{F}} \left\{\|Q - TQ\|_{2,\mu}^2 + \mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q)(Z) \mid Z\right]\right]\right\} \\
&\neq \operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - TQ\|_{2,\mu}^2 .
\end{aligned}
\tag{5.49}
$$

For stochastic dynamical systems, the variance term is non-zero, whereas for deterministic ones, it is zero. By replacing $T^\pi Q$ with $\hat{T}^\pi Q$ in BRM in stochastic dynamics, we obtain a solution that is not the same as minimizer of the Bellman error within the function space $\mathcal{F}$.

Let us take a closer look at the variance term. For simplicity, assume that the reward is deterministic, so $R_i = r(X_i, A_i)$. In that case

$$\begin{aligned}
&\operatorname{Var}\left[(\hat{T}^\pi Q)(Z) \mid Z\right] = \\
&\mathbb{E}\left[\left|(r(Z) + \gamma Q(X', \pi(X'))) - \left(r(Z) + \gamma \int \mathcal{P}(\mathrm{d}x'|Z) Q(x', \pi(x'))\right)\right|^2 \mid Z\right] = \\
&\gamma^2 \mathbb{E}\left[\left|Q(X', \pi(X') - \int \mathcal{P}(\mathrm{d}x'|Z) Q(x', \pi(x')\right|^2 \mid Z\right].
\end{aligned}$$

This is the variance of Q at the next-state X'. Having this variance term in optimization (5.49) encourages finding Q that has small next-state variance. For example, if Q is constant, this term would be zero. If Q is varying slowly as a function of state x (i.e., a smooth function), it is going to be small. This induced smoothness is, however, not desirable because it is not a smoothness that is natural to the problem, but imposed by the biased objective. Moreover, it is not controllable in the sense that we can change its amount by a hyperparameter. If it was controllable, we could use it as a way to regularize the value function estimate, which would be useful, for example, to avoid overfitting.

5.2.4 LSTD and Least Square Policy Iteration (LSPI)

Starting from the PBE, we got several approaches to approximate V^π. One of them was based on solving $V = (\Pi_{\mathcal{F},\mu} T^\pi V)$ (5.19) with V being a linear FA with $V_N = \Phi_{N \times p} w_p$ (Section 5.1.4.2). We showed that the solution to this equation is

$$w_{\mathrm{TD}} = A^{-1}_{p \times p} b_{p \times 1}$$

with

$$\begin{aligned}
A &= \Phi^\top M (\Phi - \gamma \mathcal{P}^\pi \Phi), \\
b &= \Phi^\top M r^\pi.
\end{aligned}$$

We need to use data $\mathcal{D}_n$ in order to estimate these. The way becomes more clear if we expand A and b in terms of summation (or integration, more generally). As M

is a diagonal matrix, we have

$$A_{ij} = \left[\Phi^\top M(\Phi - \gamma \mathcal{P}^\pi \Phi)\right]_{ij} = \sum_{m=1}^{N} \Phi^\top_{im} \mu(m) \left(\Phi - \gamma \mathcal{P}^\pi \Phi\right)_{mj},$$

$$b_i = \sum_{m=1}^{N} \Phi^\top_{im} \mu(m) r^\pi_m$$

Or expanded more explicitly in terms of state x and next-state x',

$$A = \sum_{x \in \mathcal{X}} \mu(x) \phi(x) \left(\phi(x) - \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^\pi(x'|x) \phi(x') \right)^\top,$$

$$b = \sum_{x \in \mathcal{X}} \mu(x) \phi(x) r^\pi(x).$$

These forms suggest that we can use data to estimate A and b. Given $\mathcal{D}_n = \{(X_i, R_i, X'_i)\}_{i=1}^n$ with $X_i \sim \mu \in \mathcal{M}(\mathcal{X})$, and $X'_i \sim \mathcal{P}^\pi(\cdot|X_i)$ and $R_i \sim \mathcal{R}^\pi(\cdot|X_i)$ (cf. (5.38)), we define the empirical estimates $\hat{A}_n$ and $\hat{b}_n$ as

$$\hat{A}_n = \frac{1}{n} \sum_{i=1}^{n} \phi(X_i) \left(\phi(X_i) - \gamma \phi(X'_i)\right)^\top,$$

$$\hat{b}_n = \frac{1}{n} \sum_{i=1}^{n} \phi(X_i) R_i.$$

We have

$$\begin{aligned}
&\mathbb{E}\left[\phi(X_i) \left(\phi(X_i) - \gamma \phi(X'_i)\right)^\top\right] = \\
&\mathbb{E}\left[\mathbb{E}\left[\phi(X_i) \left(\phi(X_i) - \gamma \phi(X'_i)\right)^\top\right] \mid X_i\right] = \\
&\mathbb{E}\left[\phi(X_i) \left(\phi(X_i) - \gamma \mathbb{E}\left[\phi(X'_i) \mid X_i\right]\right)^\top\right] = \\
&\mathbb{E}\left[\phi(X_i) \left(\phi(X_i) - \gamma (\mathcal{P}^\pi \Phi)(X_i)\right)^\top\right] = A.
\end{aligned}$$

This shows that $\hat{A}_n$ is an unbiased estimate of A. If $X_i \sim \mu$ are selected independently, by the LLN, we get that $\hat{A}_n \to A$ (a.s.) (assuming the bounded fourth moment on the features). Similar guarantee holds when X_i are not independent, but comes from a Markov chain induced by following a policy.[8] Showing that $\hat{b}_n$ is an unbiased estimate of b is similar.

[8]For instance, we can show that if the chain is positive Harris, LLN holds for functions that belong to $L_1(\rho^\pi)$ (Theorem 17.1.7 of Meyn and Tweedie 2009).

Exercise 5.4. *Show that $\hat{b}_n$ is an unbiased estimate of b.*

We can also have the LSTD estimation procedure for the action-value function. We briefly show how relevant quantities can be represented and what the LSTD solution looks like.

We represent $Q(x,a) = \phi^\top(x,a)w$ in the matrix form as

$$Q_{N\times 1} = \Phi_{N\times p} w_p$$

with $N = |\mathcal{X} \times \mathcal{A}|$ and

$$\Phi = \begin{bmatrix} \phi_1 & \cdots & \phi_p \end{bmatrix}.$$

with

$$\phi_i = \begin{bmatrix} \phi_i(x_1, a_1) \\ \vdots \\ \phi_i(x_1, a_{|\mathcal{A}|}) \\ \phi_i(x_2, a_1) \\ \vdots \\ \phi_i(x_{|\mathcal{X}|}, a_{|\mathcal{A}|}) \end{bmatrix},$$

and P as the $|\mathcal{X} \times \mathcal{A}| \times |\mathcal{X}|$ matrix

$$[\mathcal{P}]_{(x,a),x'} = \mathcal{P}(x'|x,a),$$

and Π_π as $|\mathcal{X}| \times |\mathcal{X} \times \mathcal{A}|$ matrix with

$$[\Pi_p i]_{x,(x,a)} = \pi(a|x).$$

Then, the Bellman equation can be written as

$$\left(\mathbf{I}_{N\times N} - \gamma \mathcal{P}_{N\times|\mathcal{X}|} \Pi_{\pi|\mathcal{X}|\times N}\right) Q^\pi_{N\times 1} = r_{N\times 1}.$$

Let $\mu \in \mathcal{M}(\mathcal{X} \times \mathcal{A})$. We define $M_{N\times N} = \text{diag}(\mu)$. The solution of the LSTD solution (population version) is

$$w = \left[\Phi^\top M(\Phi - \gamma \mathcal{P}\Pi_\pi \Phi)\right]^{-1} \Phi^\top M r = A^{-1} b. \tag{5.50}$$

Obtaining the empirical version of A and b is similar to what we already discussed. This is also known as LSTDQ, but we use LSTD to refer to solution for either V or Q, as the principle is the same.

Algorithm 5.1 LSPI

```
Require: $\mathcal{D}_n = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n$ and initial policy $\pi_1$.
1: for $k = 1, 2, \ldots, K$ do
2:     $\hat{Q}^{\pi_k} \leftarrow \text{LSTD}(\mathcal{D}_n, \pi_k)$          ▷ Policy Evaluation
3:     $\pi_{k+1} \leftarrow \pi_g(\hat{Q}^{\pi_k})$          ▷ Policy Improvement
4: end for
5: return $\hat{Q}^{\pi_K}$ and $\pi_{K+1}$.
```

We can use LSTD to define an an approximate PI (API) procedure to obtain a close to optimal policy as shown in Algorithm 5.1. This is a policy iteration algorithm that uses LSTD to evaluate a policy. It is approximate because of the use of a function approximation and a finite number of data point.

We may also collect more data during LSPI. For example, as we obtain a new policy, we can follow it to collect new data points. Note that LSTD is an off-policy algorithm because it can evaluate a policy π that is different from the one collecting data.

LSTD and LSPI are considered as sample efficient algorithms. They are, however, not computationally cheap. The matrix inversion $\hat{A}_{p\times p}$ is $O(p^3)$, if computed naively. If we want to perform it in an online fashion, as new samples arrive, the computational cost can be costly: $O(np^3)$. Note that we may use Sherman-Morrison formula (or the matrix inversion lemma) to compute $\hat{A}_n^{-1}$ incrementally based on the previous inverted matrix $\hat{A}_{n-1}^{-1}$.

5.3 Online RL Methods

In the online setting, the agent updates the value function as it interacts with the environment. We can use the update rules derived in Section 5.1.4.3 in order to design a SA procedure.

First, we consider the weight update rule (5.29):

$$w_{k+1} \leftarrow (1-\alpha)w_k + \alpha(\Phi^\top D^\pi \Phi)^{-1}\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k\right].$$

In order to convert this to a SA procedure, we need to empirically estimate $\Phi^\top D^\pi \Phi$ and $\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k\right]$. We have

$$\begin{aligned}(\Phi^\top D^\pi \Phi)_{p\times p} &= \sum_{x\in\mathcal{X}} \rho^\pi(x)\phi(x)\phi^\top(x) \\ &= \mathbb{E}\left[\phi(X)\phi^\top(X)\right], \qquad X \sim \rho^\pi.\end{aligned}$$

So if we have t data points $X_1, \dots X_t$ with $X_i \sim \rho^\pi$, the stationary distribution of π, we can estimate it by a matrix $\hat{A}_t$

$$\hat{A}_t = \frac{1}{t} \sum_{i=1}^{t} \phi(X_i)\phi^\top(X_i).$$

This matrix is an unbiased estimate of $(\Phi^\top D^\pi \Phi)$, and converges to it under usual conditions of LLN.

We also have

$$\Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k\right] = \sum_{x \in \mathcal{X}} \rho^\pi(x)\phi(x) \left(r^\pi(x) + \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^\pi(x'|x)\phi^\top(x')w_k \right). \quad (5.51)$$

If $X_t \sim \rho^\pi$, $X'_t \sim \mathcal{P}^\pi(\cdot|X_t)$, and $R_t \sim \mathcal{R}^\pi(\cdot|X_t)$, the r.v.

$$\phi(X_t)\left(R_t + \gamma\phi^\top(X'_t)w_t\right)$$

is an unbiased estimate of (5.51).

These suggest a SA procedure that at each time step t, after observing X_t, R_t, X'_t, updates the weight w_t to w_{t+1} by

$$w_{t+1} \leftarrow (1 - \alpha_t)w_t + \alpha_t \hat{A}_t^{-1}\phi(X_t)\left(R_t + \gamma\phi^\top(X'_t)w_t\right) \quad (5.52)$$

with

$$\begin{aligned} \hat{A}_t &= \frac{1}{t}\left[(t-1)\hat{A}_{t-1} + \phi(X_t)\phi^\top(X_t)\right] \\ &= \left(1 - \frac{1}{t}\right)\hat{A}_{t-1} + \frac{1}{t}\phi(X_t)\phi^\top(X_t). \end{aligned}$$

The inversion of $\hat{A}_t$ is expensive, if done naively. We can use Sherman-Morrison formula to incrementally update it (Section A.5) as

$$(\hat{A}_t)^{-1} = \hat{A}_{t-1}^{-1} - \frac{\hat{A}_{t-1}^{-1}\phi(X_t)\phi^\top(X_t)\hat{A}_{t-1}^{-1}}{1 + \phi^\top(X_t)\hat{A}_{t-1}^{-1}\phi(X_t)}.$$

This requires a matrix-vector multiplication and is $O(p^2)$. The per-sample computational cost of (5.52) is then $O(p^2)$. This is significantly higher than the $O(1)$ computational cost of the TD update for a problem with finite state-action spaces for which the value function can be represented exactly in a lookup table (for example, see Algorithm 4.3).

This comparison, however, may not be completely fair. The computational cost of evaluating $V(x)$ at any x for a finite state problem with an exact representation was $O(1)$ itself, but the computational cost of evaluating the value function with a linear FA with p features (i.e., $V(x;w) = \phi^\top(x)w$) is $O(p)$. A better baseline is perhaps to compare the cost of update per time step with the cost of computation of V for a single state. We can then compute

$$\frac{\text{computational cost of update per sample}}{\text{computational cost of computing the value of a single state}}.$$

For TD with a finite state(-action) space with the exact representation, the ratio is $O(1)$. For the method (5.52), the ratio is $O(p)$. This shows a more graceful dependence on p, but it still scales linearly with the number of features.

We can have a better computational cost using the other update rule (5.31) we derived in Section 5.1.4.3. The population version is

$$w_{k+1} \leftarrow w_k + \alpha D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k - \Phi w_k \right].$$

If $X_t \sim \rho^\pi$, $X'_t \sim \mathcal{P}^\pi(\cdot|X_i)$, and $R_t \sim \mathcal{R}^\pi(\cdot|X_i)$, we use the r.v.

$$\phi(X_t) \left(R_t + \gamma \phi^\top(X'_t) w_t - \phi(X_t) w_t \right) = \phi(X_t)\delta_t,$$

with $\delta_t = R_t + \gamma \phi^\top(X'_t) w_t - \phi(X_t) w_t$, the TD error, is an unbiased estimate of $D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k - \Phi w_k \right]$. Therefore, the SA update rule would be

$$w_{k+1} \leftarrow w_k + \alpha_t \phi(X_t) \delta_t. \tag{5.53}$$

This is the TD Learning with linear FA.

We have shown that the population version of this update rule under $X \sim \rho^\pi$ converges in Proposition 5.4. We do not show it for the SA version, but we might suspect that it does because it follows a noise contaminated version of a stable/convergent dynamical system. With proper choice of the step size sequence (α_t), we can expect convergence. This indeed true, as shown by Tsitsiklis and Van Roy [1997].

It is worth mentioning that this convergence holds only when $X_t \sim \rho^\pi$, the stationary distribution of π. If its distribution is not the same, the TD with linear FA might diverge. This is contrast with the TD for finite state problems where the conditions of convergence were much easier and we did not have divergence.

The same method works for learning an action-value function Q^π of policy π using an approximation

$$Q(x,a) = Q(x,a;w) = \phi(x,a)^\top w.$$

For $X_t \sim \rho^\pi$, $A_t = \pi(X_t)$, $X'_t \sim \mathcal{P}(\cdot|X_t, A_t)$, and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$, we can update the weights as

$$w_{k+1} \leftarrow w_k + \alpha_t \phi(X_t, A_t)\delta_t, \tag{5.54}$$

with the TD error

$$\delta_t = R_t + \gamma\phi(X'_t, \pi(X'_t))^\top w_t - \phi(X_t, A_t)^\top w_t.$$

We may use a similar procedure for the control problem and define SARSA-like and Q-Learning-like algorithms with linear FA. For SARSA, the update uses the tuple $(X_t, A_t, R_t, X'_t, A'_t)$ with $A_t \sim \pi(\cdot|X_t)$ and $A'_t \sim \pi(\cdot|X'_t)$, and π being a policy that is close to being greedy w.r.t. Q_t, e.g., an ε-greedy policy $\pi_\varepsilon(Q_t)$. The update would be the same with the difference that the TD error would be

$$\delta_t = R_t + \gamma\phi(X'_t, A'_t)^\top w_t - \phi(X_t, A_t)^\top w_t.$$

We may also form a Q-Learning-like algorithm by having

$$\delta_t = R_t + \gamma \max_{a' \in \mathcal{A}} \phi(X'_t, a')^\top w_t - \phi(X_t, A_t)^\top w_t$$

Even though the agent may be following a policy π and have samples $X_t \sim \rho^\pi$ and $A_t \sim \pi(\cdot|X_t)$ (or similar for the deterministic policy), the policy being evaluated is the greedy policy $\pi_g(\cdot; Q_t)$. The evaluated policy may not be the same as π. This is an off-policy samplings scenario. The convergence guarantee for TD with linear FA, shown by Tsitsiklis and Van Roy [1997], does not hold here. In fact, Q-Learning with linear FA might divergence.

5.3.1 Semi-Gradient Viewpoint

We motivated the TD method with linear FA by starting from $V = \Pi_{\mathcal{F},\rho^\pi} T^\pi V$ with $V = \Phi w$, and devised an iterative SA procedure for its computation. One may also see it as an SGD-like procedure, with some modifications, as we explain here. It is that approach followed by Sutton and Barto [2019].

Suppose that we know the true value function V^π, and we want to find an approximation $\hat{V}$, parameterized by w (the same setup as we had earlier in Section 5.1.1). The population loss is

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \frac{1}{2} \left\| V^\pi - \hat{V}(w) \right\|^2_{2,\mu}. \tag{5.55}$$

Using samples $X_t \sim \mu$, we can define an SGD procedure that updates w_t as follows:

$$\begin{aligned} w_{t+1} \leftarrow & w_t - \alpha_t \nabla_w \left[\frac{1}{2} \left| V^\pi(X_t) - \hat{V}(X_t; w_t) \right|^2 \right] \\ = & w_t + \alpha_t \left(V^\pi(X_t) - \hat{V}(X_t; w_t) \right) \nabla_w \hat{V}(X_t; w_t). \end{aligned}$$

If the step size α_t is selected properly, the SGD converges to the stationary point if the objective of (5.55). If we use a linear FA to represent $\hat{V}$, the objective would be convex, so w_t converges to the global minimum of the objective. In this formulation, $V^\pi(X_t)$ acts as the target, in the supervised learning sense.

When we do not know V^π, we may use a bootstrapped estimate instead:

$$(\hat{T}^\pi V_t)(X_t) = R_t + \gamma V_t(X'_t) = R_t + \gamma \hat{V}(X_t; w_t).$$

With this substitution, the update rule would be

$$w_{t+1} \leftarrow w_t + \alpha_t \left(R_t + \gamma \hat{V}(X'_t; w_t) - \hat{V}(X_t; w_t) \right) \nabla_w \hat{V}(X_t; w_t).$$

For linear FA, we have $\hat{V}(x; w) = \phi^\top(x) w$, and we get the update rule

$$\begin{aligned} w_{t+1} \leftarrow & w_t + \alpha_t \left(R_t + \gamma \hat{V}(X'_t; w_t) - \hat{V}(X_t; w_t) \right) \phi(X_t) \\ = & w_t + \alpha_t \delta_t \phi(X_t). \end{aligned}$$

This is the same update rule that we had before for TD with linear FA (5.53).

The substitution of $V^\pi(X_t)$ with $(\hat{T}^\pi V_t)(X_t)$ does not follow from the SGD of any loss function. The TD update is not a true SGD update.

One may wonder why we do not perform SGD on

$$\frac{1}{2} \left| \hat{V}(X_t; w_t) - (\hat{T}^\pi \hat{V}(w_t))(X_t) \right|^2$$

instead. Certainly, we can write

$$\begin{aligned} w_{t+1} \leftarrow & w_t - \alpha_t \nabla_w \left[\frac{1}{2} \left| \hat{V}(X_t; w_t) - (\hat{T}^\pi \hat{V}(w_t))(X_t) \right|^2 \right] \\ = & w_t - \alpha_t \left(\hat{V}(X_t; w_t) - (\hat{T}^\pi \hat{V}(w_t))(X_t) \right) \left(\nabla_w \hat{V}(X_t; w_t) - \nabla_w (\hat{T}^\pi \hat{V}(w_t))(X_t) \right) \\ = & w_t - \alpha_t \left(\hat{V}(X_t; w_t) - (\hat{T}^\pi \hat{V}(w_t))(X_t) \right) \left(\nabla_w \hat{V}(X_t; w_t) - \gamma \nabla_w \hat{V}(X'_t; w_t) \right) \end{aligned}$$

With linear FA, this becomes

$$\begin{aligned}
w_{t+1} \leftarrow & w_t - \alpha_t \left(\phi(X_t)^\top w_t - (R_t + \gamma \phi(X'_t)^\top w_t \right) (\phi(X_t) - \gamma \phi(X'_t)) \\
= & w_t - \alpha_t \delta_t \cdot (\phi(X_t) - \gamma \phi(X'_t))
\end{aligned}$$

This is similar to the TD update, with the difference that instead of $\phi(X_t)$, we have $\phi(X_t) - \gamma\phi(X'_t)$. The issue, however, is that this empirical loss function $\frac{1}{2}|\hat{V}(X_t; w_t) - (\hat{T}^\pi \hat{V}(w_t))(X_t)|^2$ is biased, as explained in Section 5.2.3. Minimizing it does not lead to the minimizer of the Bellman error.

Appendix A

Mathematical Background

A.1 Probability Space

For a space Ω, with σ-algebra σ_Ω, we define $\mathcal{M}(\Omega)$ as the set of all probability measures over σ_Ω. Further, we let $\mathcal{B}(\Omega)$ denote the space of bounded measurable functions w.r.t. (with respect to) σ_Ω and we denote $\mathcal{B}(\Omega, L)$ as the space of bounded measurable functions with bound $0 < L < \infty$.

We write $\nu_1 \ll \nu_2$ if $\nu_2(A) = 0$ implies that $\nu_1(A) = 0$ as well. For two σ-finite measures ν_1 and ν_2 on some measurable space (Ω, σ_Ω), ν_1 is *absolutely continuous* w.r.t. ν_2 if there is a non-negative measurable function $f : \Omega \to \mathbb{R}$ such that $\mu_1(A) = \int f d\nu_2$ for all $A \in \sigma_\Omega$. It is known that ν_1 is absolutely continuous w.r.t. ν_2 if and only if $\nu_1 \ll \nu_2$. We write $\frac{d\nu_1}{d\nu_2} = f$ and call it the *Radon-Nikodym* derivative of ν_1 w.r.t. ν_2 [Rosenthal, 2006, Chapter 12].

A.2 Norms and Function Spaces

We use $\mathcal{F} : \mathcal{X} \to \mathbb{R}$ to denote a subset of measurable functions.[1] The exact specification of this space should be clear from the context. We usually denote $\mathcal{F}$ as the space of value functions.

For a probability distribution $\nu \in \mathcal{M}(\mathcal{X})$, and a measurable function $V \in \mathcal{F}$, we define the $L_p(\nu)$-norm of V with $1 \leq p < \infty$ as

$$\|V\|_{p,\nu}^p \triangleq \int_{\mathcal{X}} |V(x)|^p d\nu(x). \tag{A.1}$$

[1]This section is quoted almost verbatim from Section 2.1 of Farahmand [2011].

When $p = 2$, this is the norm induced by the inner product $\langle \cdot , \cdot \rangle : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$: For any V_1, V_2, we have

$$\langle V_1 , V_2 \rangle_\nu = \int_{\mathcal{X}} V_1(x) V_2(x) \mathrm{d}\nu(x). \tag{A.2}$$

It is clear that $\|V\|_{2,\nu}^2 = \langle V , V \rangle_\nu$.

The $L_\infty(\mathcal{X})$-norm is defined as

$$\|V\|_\infty \triangleq \sup_{x \in \mathcal{X}} |V(x)|. \tag{A.3}$$

If we want to emphasize that the probability distribution is defined on the state space $\mathcal{X}$, we use $\nu_{\mathcal{X}}$ and $\|V\|_{p,\nu_{\mathcal{X}}}$.

We define $\mathcal{F}^{|\mathcal{A}|} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^{|\mathcal{A}|}$ as a subset of vector-valued measurable functions with the following identification:

$$\mathcal{F}^{|\mathcal{A}|} = \left\{ (Q_1, \dots, Q_{|\mathcal{A}|}) \, : \, Q_i \in \mathcal{F}, \; i = 1, \dots, |\mathcal{A}| \right\}.$$

We use $Q_j(x) = Q(x, j) \, (j = 1, \dots, |\mathcal{A}|)$ to refer to the j^{th} component of $Q \in \mathcal{F}^{|\mathcal{A}|}$. We often denote $\mathcal{F}^{|\mathcal{A}|}$ as a space of action-value functions. If there is no chance of ambiguity, we may use $\mathcal{F} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^{|\mathcal{A}|}$ for a space of action-value functions though.

Let $z_{1:n}$ denote the $\mathcal{Z}$-valued sequence $(z_1, \dots, z_n)$. We define the empirical measure as the measure that assigns the following probability to any (measurable) set $B \subset \mathcal{Z}$:

$$\nu_n(B) \triangleq \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{Z_i \in B\}.$$

The empirical norm of function $f : \mathcal{Z} \to \mathbb{R}$ is then

$$\|f\|_{p,z_{1:n}}^p = \|f\|_{p,\mathcal{D}_n}^p \triangleq \|f\|_{p,\nu_n} = \frac{1}{n} \sum_{i=1}^{n} |f(z_i)|^p. \tag{A.4}$$

When there is no chance of confusion about $\mathcal{D}_n$, we may simply use $\|f\|_{p,n}^p$. Based on this definition, one may define $\|V\|_n$ (with $\mathcal{Z} = \mathcal{X}$) and $\|Q\|_n$ (with $\mathcal{Z} = \mathcal{X} \times \mathcal{A}$).

If $\mathcal{D}_n = Z_{1:n}$ is random with $Z_i \sim \nu$, the empirical norm is random as well. For any fixed function f, we have $\mathbb{E}\left[\|f\|_{p,n} \right] = \|f\|_{p,\nu}$.

We sometimes use the shorthand notation of $\nu|Q|^p = \|Q\|_{p,\nu}^p$ (similar for $\nu_{\mathcal{X}}$ and other probability distributions). In this book, most results are stated for $p = 1$ or $p = 2$. The symbols $\|\cdot\|_\nu$ and $\|\cdot\|_n$ refers to an L_2-norm.

Finally, define the projection operator $\Pi_{\mathcal{F}^{|\mathcal{A}|},\nu} : B(\mathcal{X} \times \mathcal{A}) \to B(\mathcal{X} \times \mathcal{A})$ as

$$\Pi_{\mathcal{F},\nu} Q \triangleq \operatorname*{argmin}_{Q' \in \mathcal{F}^{|\mathcal{A}|}} \|Q' - Q\|_\nu^2$$

for $Q \in B(\mathcal{X} \times \mathcal{A})$. The definition of $\Pi_{\mathcal{F},\nu_{\mathcal{X}}} : B(\mathcal{X}) \to B(\mathcal{X})$ is similar. If the distribution $\nu_{\mathcal{X}}$ or ν are clear from the context, we may simply write $\Pi_{\mathcal{F}}$ and $\Pi_{\mathcal{F}^{|\mathcal{A}|}}$ instead.

A.3 Contraction Mapping

The contraction mapping (or operator) is a mapping that maps points (i.e., vectors, functions) closer to each other. As the Bellman operators for discounted tasks are contraction mapping, it is useful to have a good understanding on what such a mapping is, and what their properties have. Our discussion here freely borrows from Hunter and Nachtergaele [2001].

First, let us recall the definition of a *metric space.* Let $\mathcal{Z}$ be an arbitrary non-empty set.

Definition A.1 (Metric – Definition 1.1 of Hunter and Nachtergaele 2001)**.** *A metric or a distance function on $\mathcal{Z}$ is a function $d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ with the following properties:*

- *$d(x, y) \geq 0$ for all $x, y \in \mathcal{Z}$; and $d(x, y) = 0$ if and only if $x = y$.*
- *$d(x, y) = d(y, x)$ for all $x, y \in \mathcal{Z}$ (symmetry).*
- *$d(x, y) \leq d(x, z) + d(z, y)$ for all $x, y, z \in \mathcal{Z}$ (triangle inequality).*

Given a metric, we can define a metric space.

Definition A.2 (Metric Space)**.** *A metric space $(\mathcal{Z}, d)$ is a set $\mathcal{Z}$ equipped with a metric d.*

Example A.1. *Let $\mathcal{Z} = \mathbb{R}$ and $d(x, y) = |x - y|$. These together define a metric space $(\mathbb{R}, d)$.*

Example A.2. *Let $\mathcal{Z}$ be a discrete set and define*

$$d(x, y) = \begin{cases} 0 & x = y, \\ 1 & x \neq y. \end{cases}$$

We often work with *linear vector spaces* and *norms* in this course. Let us define them.

Definition A.3 (Linear Space – Definition 1.7 of Hunter and Nachtergaele 2001). *A linear space $\mathcal{Z}$ over the scalar field $\mathbb{R}$ (or $\mathbb{C}$) is a set of points (or vectors), on which operations of vector additions and scalar multiplications with the following properties are defined:*

(a) The set $\mathcal{Z}$ is a commutative group with the operation of $+$ of vector addition, that is,

- $x + y = y + x.$
- $x + (y + z) = (x + y) + z$
- *There exists an element $0 \in \mathcal{Z}$ such that for any $x \in \mathcal{Z}$, we have $x + 0 = x$.*
- *For each $x \in \mathcal{Z}$, there exists a unique vector $-x \in \mathcal{Z}$ such that $x+(-x) = 0$.*

(b) For all $x, y \in \mathcal{Z}$ and $a, b \in \mathbb{R}$ (or $\mathbb{C}$), we have

- $1.x = x.$
- $(a + b)x = ax + bx.$
- $a(bx) = (ab)x.$
- $a(x + y) = ax + by.$

Next we define a notion of the length or size of a vector.

Definition A.4 (Norm – Definition 1.8 of Hunter and Nachtergaele 2001). *A norm on a linear space $\mathcal{Z}$ is a function $\|\cdot\| : \mathcal{Z} \to \mathbb{R}$ with the following properties:*

(a) (non-negative) For all $x \in \mathcal{Z}$, $\|x\| \geq 0$.

(a) (homogenous) For all $x \in \mathcal{Z}$ and $\lambda \in \mathbb{R}$ (or $\mathbb{C}$), $\|\lambda x\| = |\lambda| \, \|x\|$.

(a) (triangle inequality) For all $x, y \in \mathcal{Z}$, $\|x + y\| \leq \|x\| + \|y\|$.

(a) (strictly positive) If for a $x \in \mathcal{Z}$, we have that $\|x\| = 0$, it implies that $x = 0$.

The same way that we used a metric space given a metric, we can define a normed linear space given a norm.

Definition A.5 (Normed Linear Space). *A normed linear space $(\mathcal{Z}, \|\cdot\|)$ is a linear space $\mathcal{Z}$ equipped with a norm $\|\cdot\|$.*

We can use a norm to define a distance between two points in a linear space $\mathcal{Z}$, simply by defining $d(x, y) = \|x - y\|$. This gives us a metric space $(\mathcal{Z}, d)$.

Example A.3. *Let $\mathcal{Z} = \mathbb{R}^d$ $(d \geq 1)$. The following norms are often used:*

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{d} |x_i|^p}, \qquad 1 \leq p < \infty,$$
$$\|x\|_\infty = \max_{i=1,\dots,d} |x_i|.$$

Example A.4. *Consider the space of continuous functions with domain $[0, 1]$. It is denoted by $\mathcal{C}([0, 1])$. This plays the rule of $\mathcal{Z}$. We define the following norm for a function $f \in \mathcal{C}([0, 1])$:*

$$\|f\|_\infty = \sup_{x \in [0,1]} |f(x)|.$$

This is called the supremum *or* uniform *norm. Given this norm, $(\mathcal{C}([0, 1]), \|\cdot\|_\infty)$ would be a normed linear space.*

This norm is similar to $\|x\|_\infty$ with $x \in \mathbb{R}^d$ (previous example), but it is for the space of continuous functions.

We often use the supremum norm of value functions. For $V \in \mathcal{B}(\mathcal{X})$ and $Q \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$, their supremum norms are

$$\|V\|_\infty = \sup_{x \in \mathcal{X}} |V(x)|,$$
$$\|Q\|_\infty = \sup_{(x,a) \in \mathcal{X} \times \mathcal{A}} |Q(x, a)|,$$

We are ready to define the contraction mapping formally.

Definition A.6 (Contraction Mapping – Definition 3.1 of Hunter and Nachtergaele 2001)**.** *Let $(\mathcal{Z}, d)$ be a metric space. A mapping $L : \mathcal{Z} \to \mathcal{Z}$ is a contraction mapping (or contraction) if there exists a constant $0 \leq a < 1$ such that for all $z_1, z_2 \in \mathcal{Z}$, we have*[2]

$$d(L(z_1), L(z_2)) \leq a d(z_1, z_2).$$

This is visualized in Figure A.1.

[2]Sometimes the condition of having $a < 1$ is called strict contraction [Berinde, 2007], and the condition that $d(L(z_1), L(z_2)) < d(z_1, z_2)$ is called contractive.

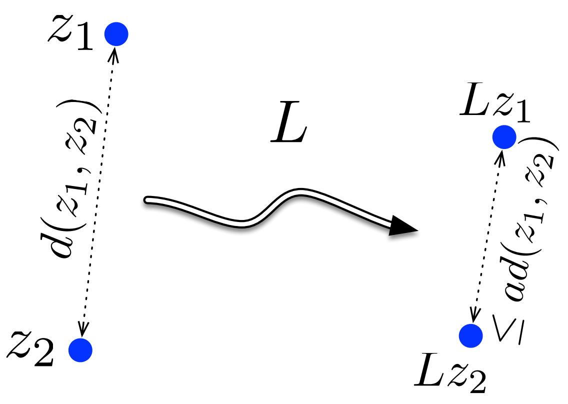

Figure A.1: Visualization of an a-contraction mapping L.

Example A.5. *Let $\mathcal{Z} = \mathbb{R}$ and $d(z_1, z_2) = |z_1 - z_2|$. Consider the mapping $L : z \mapsto az$ for $a \in \mathbb{R}$. We have Let us see if/when this mapping is a contraction or not.*

For any $z_1, z_2 \in \mathbb{R}$, we have

$$d(L(z_1), L(z_2)) = |L(z_1) - L(z_2)| = |az_1 - az_2| = |a||z_1 - z_2| = |a|d(z_1, z_2).$$

So if $|a| < 1$, this is a contraction mapping.

Exercise A.1 ($\star$). *Consider the same $(\mathbb{R}, |\cdot|)$ as before, but let the mapping be $L : z \mapsto az + b$ for $a, b \in \mathbb{R}$. What is condition on a and b for this mapping to be a contraction.*

Exercise A.2 ($\star\star$). *Consider the same $(\mathbb{R}, |\cdot|)$ as before, and let $L : z \mapsto az^2 + b$ for $a, b \in \mathbb{R}$. Is this a contraction mapping for some choice of a and b? If yes, specify a and b. If not, can you consider another space $\mathcal{Z}$ (a subset of $\mathbb{R}$) that makes this a contraction (possibly with an appropriate choice of a and b)?*

Exercise A.3 ($\star\star$). *Consider $\mathcal{Z} = \mathbb{R}^d$. Given a matrix $A \in \mathbb{R}^{d\times d}$ and a vector $b \in \mathbb{R}^d$, define the mapping $L : z \mapsto Az + b$. Using the vector norm $\|\cdot\|_p$ ($1 \leq p \leq \infty$, define the metric $d_p(z_1, z_2) = \|z_1 - z_2\|_p$. Then, $d_p(L(z_1), L(z_2)) = \|Az_1 - Az_2\|_p$.*

What is the condition that L is a contraction? Note that this depends on the choice of p.

Exercise A.4 ($\star\star$). *Consider $(\mathbb{R}, |\cdot|)$. Let $r \geq 0$ and define the mapping*

$$L : z \mapsto rz(1 - z).$$

When is this a contraction mapping?

Why do we care about a contraction mapping? We have two reasons in mind.

The first is that we can describe the behaviour of a dynamical system depending on whether the mapping describing it is a contraction or not. To be concrete, let $z_0 \in \mathcal{Z}$ and consider a mapping $L : z \mapsto az$ for some $a \in \mathbb{R}$. Define the dynamical system

$$z_{k+1} = Lz_k, \qquad k = 0, 1, \dots .$$

The dynamical system described by this mapping generates

$$\begin{aligned} &z_0 \\ &z_1 = az_0 \\ &z_2 = az_1 = a^2 z_0 \\ &\quad\vdots \\ &z_k = az_{k-1} = a^k z_0. \end{aligned}$$

If $|a| < 1$, z_k converges to zero, no matter what z_0 is. If $a = 1$, we have $z_k = z_0$. So depending on z_0, it converges to different points. For $a = -1$, the sequence would oscillate between $+z_0$ and $-z_0$. And if $|a| > 1$, the sequence diverges (unless $z_0 = 0$).

An interesting observation is that the case of converge is the same as the case of L being a contraction map (see Example A.5). This is not an isolated example, as we shall see. A dynamical system defined based on a contraction mapping converges. We call such a system *stable*.[3]

The second reason we care about contraction is that we can sometimes use it to solve equations. We can convert an equation that we want to solve (think of solving $f(z) = 0$) as the fixed point equation, as we shall see. If it happens that the underlying mapping is contraction, we can define an algorithm based on a dynamical system in order to solve the equation. Let us make this idea more concrete.

Definition A.7 (Fixed Point). *If $L : \mathcal{Z} \to \mathcal{Z}$, then a point $z \in \mathcal{Z}$ such that*

$$Lz = z$$

is called a fixed point of L.

In general, a mapping may have more one, many, or no fixed point.

Given an equation $f(z) = 0$, we can convert it to a fixed point equation $Lz = z$ by defining $L : z \mapsto f(z) + z$. Then, if $Lz^* = z^*$ for a z^*, we get that $f(z^*) = 0$, i.e., the fixed point of L is the same as the solution of $f(z) = 0$.

[3] There are various notions of stability in control theory. What we consider as stable is the same as globally exponentially stable.

Example A.6. *Suppose that we want to solve $cz + b = 0$ for $z \in \mathbb{R}$ and constants $c, b \in \mathbb{R}$. We can choose $L : z \mapsto (c + 1)z + b$. The mapping L is a contraction if $|c+1| < 1$ (or $-2 < c < 0$). As a numerical example, if we want to solve $-0.5z+1 = 0$ (which has $z^* = 2$), we can write it as $L : z \mapsto 0.5z + 1$. If we start from $z_0 = 0$, we get the sequence of $(z_0, z_1, \dots) = (1, 1.5, 1.75, 1.875, 1.9375, 1.96875, \dots)$.*

Of course, this is a very simple example, and we may not use such an iterative method to solve that equation.

The next theorem formalizes what we discussed about the convergence property of a contraction mapping. This is a simple, yet very important, result. It is known as the *contraction mapping* or *Banach fixed point* theorem.

Theorem A.1 (Banach Fixed Point Theorem – Theorem 3.2 of Hunter and Nachtergaele 2001)**.** *If $L : \mathcal{Z} \to \mathcal{Z}$ is a contraction mapping on a complete metric space (Z, d), then there exists a unique $z^* \in \mathcal{Z}$ such that $Lz^* = z^*$.*

Furthermore, the point z^* can be found by choosing an arbitrary $z_0 \in \mathcal{Z}$ and defining $z_{k+1} = Lz_k$. We have $z_k \to z^*$.

Note that the convergence is in norm, and it means that $\lim_{k\to\infty} d(z_k, z^*) = 0$.

There are extensions of this result, for example, when L is not a contraction per se, but is non-expansion, i.e., $d(L(z_1), L(z_2)) \leq d(z_1, z_2)$. With a relaxed assumption on the contraction property, we may lose some of the properties (e.g., uniqueness of the fixed point) or we may need extra conditions on the space, e.g., its compactness.[4]

A.4 Matrix Norm and Some Properties of Inverses

Let us recall some results from linear algebra regarding the matrix norm, and the inverse of $\mathbf{I} - A$ and its matrix norm. The material here is mostly from Section 2.3 of Golub and Van Loan [2013].

The vector induced p-norm of a matrix $A \in \mathbb{R}^{d\times d}$ is defined as

$$\left\| A \right\|_p = \sup_{\left\| x \right\|_p = 1} \left\| Ax \right\|_p .$$

[4]As an example, we quote Theorem 3.1 of Berinde [2007]: Let $\mathcal{Z}$ be a closed bounded convex subset of the Hilbert space $\mathcal{H}$ and $L : \mathcal{Z} \to \mathcal{Z}$ be a non-expansion mapping. Then L has at least one fixed point. This does not, however, mean that we can find it by an iterative application of L.

The intuition is that we find a unit vector $x \in \mathbb{R}^d$ (according to the ℓ_p-norm) that maximizes the sizes of the mapped vector $Ax \in \mathbb{R}^d$, measured according to the same ℓ_p-norm. We could generalize this definition to have different dimensions of domain and range ($\mathbb{R}^{d_1}$ and $\mathbb{R}_{d_2}$) and use different vector norms to measure the length of the vectors before and after mapping. As we do not use them such results, we do not present them.

We have the following identities for the matrix norms:

- $\|A\|_1 = \max_{1 \le j \le d} \sum_{i=1}^{d} |a_{i,j}|$ (maximum of the sum over rows)
- $\|A\|_\infty = \max_{1 \le i \le d} \sum_{j=1}^{d} |a_{i,j}|$ (maximum of the sum over columns)
- $\|A\|_2 = \sqrt{\lambda_{\max}(A^\top A)}$ (the maximum eigenvalue of $A^\top A$).

If A is a stochastic matrix, the sum over columns (next state) is equal to one. So

$$\|\mathcal{P}^\pi\|_\infty = 1.$$

The following result shows that if a matrix A has a norm that is smaller than 1, the inverse of $\mathbf{I} - A$ exists, it has a Neumann expansion, and we can provide a bound on its norm.

Lemma A.2 (Lemma 2.3.3 of Golub and Van Loan 2013)**.** *If $A \in \mathbb{R}^{d \times d}$ and $\|A\|_p < 1$, then $\mathbf{I} - A$ is non-singular, and*

$$(\mathbf{I} - A)^{-1} = \sum_{k=0}^{\infty} A^k.$$

We also have

$$\left\|(\mathbf{I} - A)^{-1}\right\|_p \le \frac{1}{1 - \|A\|_p}.$$

The consequence of this result for us is that we can write

$$(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1} = \sum_{k \ge 0} (\gamma \mathcal{P}^\pi)^k,$$

and conclude that

$$\left\|(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}\right\|_\infty \le \frac{1}{1 - \gamma}.$$

A.5 Incremental Matrix Inversion

There are some formulae that allow us to incrementally update a matrix inversion (Section 2.1.4 of Golub and Van Loan 2013).

The Sherman-Morrison-Woodbury formula states that for a matrix $A_{d\times d}$ and two $d \times k$ matrices U and V, we have

$$(A + UV^\top)^{-1} = A^{-1} - A^{-1}U(\mathbf{I} + V^\top A^{-1}U)^{-1}V^\top A^{-1},$$

assuming that A and $(\mathbf{I} + V^\top A^{-1}U)$ are invertible. As $UV^\top$ is a $k \times k$ matrix (so of rank at most k), $A + UV^\top$ can be thought of as a rank-k update of the matrix A. The update of its inverse requires the computation of the inverse of $k \times k$ matrix $(\mathbf{I} + V^\top A^{-1}U)$, which can be much cheaper that directly inverting the new $d \times d$ matrix $A + UV^\top$ when k is smaller than d.

A special case of this formula is known as the Sherman-Morrison formula. It states that for an invertible matrix $A_{d\times d}$ and vectors $u, v \in \mathbb{R}^d$, the matrix $A + uv^\top$ is invertible if and only if $1 + v^\top A^{-1}u \neq 0$. And if it is invertible, we can compute it as

$$\left(A + uv^\top\right)^{-1} = A^{-1} - \frac{A^{-1}uv^\top A^{-1}}{1 + v^\top A^{-1}u}.$$

Note that the denominator is a scalar.

Bibliography

Vasile Berinde. *Iterative approximation of fixed points*, volume 1912. Springer, 2007. 127, 130

Dimitri P. Bertsekas. *Abstract dynamic programming*. Athena Scientific Belmont, 2nd edition, 2018. 5, 34, 37, 49, 55

Dimitri P. Bertsekas and Steven E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Academic Press, 1978. 5

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996. 5, 12, 82

Amir-massoud Farahmand. *Regularization in Reinforcement Learning*. PhD thesis, University of Alberta, 2011. 123

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 4th edition, 2013. 130, 131, 132

László Györfi, Michael Kohler, Adam Krzyżak, and Harro Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer Verlag, New York, 2002. 110

Thomas Dueholm Hansen, Peter Bro Miltersen, and Uri Zwick. Strategy iteration is strongly polynomial for 2-player turn-based stochastic games with a constant discount factor. *Journal of the ACM (JACM)*, 60(1):1–16, 2013. 51

John K. Hunter and Bruno Nachtergaele. *Applied analysis*. World Scientific Publishing Company, 2001. 125, 126, 127, 130

Sean Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, New York, NY, USA, 2009. 115

James R. Munkres. *Topology*. Pearson Modern Classic, 2nd edition, 2018. 33

Jeffrey S. Rosenthal. *A Fist Look at Rigorous Probability Theory*. World Scientific Publishing, 2nd edition, 2006. 123

Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. *Mathematics of Operations Research*, 41(3):758–774, 2016. 51

Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer, 2008. 110

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988. 16

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2019. 1, 5, 46, 120

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010. 5

John N. Tsitsiklis. On the convergence of optimistic policy iteration. *Journal of Machine Learning Research (JMLR)*, 3(1):59–72, 2002. 76

John N. Tsitsiklis and Benjamin Van Roy. An analysis of temporal difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42:674–690, 1997. 119, 120

Sara A. van de Geer. *Empirical Processes in M-Estimation*. Cambridge University Press, 2000. 110

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, University of Cambride, 1989. 16

Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011. 51