Foundations of Reinforcement Learning

Amir-massoud Farahmand

November 18, 2025

Contents

Preface

This book, tentatively titled *Foundations of Reinforcement Learning*, started as lecture notes for a graduate-level Introduction to Reinforcement Learning (RL) course, taught at the Department of Computer Science, University of Toronto, in Spring 2021. Since many students found the lecture notes very useful for learning a solid and modern foundation of RL, I decided to further develop and expand it into a book. What you are reading is its draft, prepared for the second offering of the course at Polytechnique Montréal in Fall 2025. My intention is that this will gradually evolve into a full textbook.

This is an introductory book in the sense that I do not assume prior exposure to RL. It does, however, go beyond providing the high-level intuition and instead tries to mathematically develop the foundation behind many important ideas and concepts in RL. Throughout the book, you and I go through the proof of many basic, or sometimes not so basic, results in RL. We will see formal statements and proofs for many major concepts and algorithms in RL.

This book should be accessible to someone who is mathematically mature and has the knowledge of probability theory, linear algebra, basics of analysis, and statistics and supervised machine learning. As a bonus material, you can find the accompanied video lectures (based on the Spring 2021 course) and the most recent slides based on the book on course webpage.

The book is a work in progress. I intend to expand the content of existing chapters, for example by adding more exercises. I would also like to add several new chapters, including chapters on the important topics of model-based RL and exploration-exploitation. 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 minor 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). Versions with higher number, such as 0.2, show significant revisions compared to the first draft. Hopefully the version of all chapters will eventually converge to 1.

If you find any typos or unclear parts, please send an email to me at amirmassoud.farahmand@gmail.com. I would appreciate your feedback.

Before proceeding, I would like to mention that there are very good textbooks on RL, which I encourage you to consult. Some of them have influenced the content of this book, though they have different styles and emphases. A very well-known textbook is by Sutton and Barto [2018]. It provides a very good intuition on many of the concepts and algorithms that we discuss in this book. Another very good and concise book, which describes a large number of algorithms, is Szepesvári [2010], who happened to be my PhD supervisor! A classic book is Bertsekas and Tsitsiklis [1996]. Even though it is about 30 years old, its mathematical perspective is still relevant and insightful. For a abstract treatment of the dynamic programming, refer to Bertsekas [2018]. Another recent textbook, with a distinct and fresh perspective, is Meyn [2022].

Amir-massoud Farahmand
September 2025

Chapter 1

Introduction

Chapter Introduction

This chapter introduces the main ideas of the book, including the key concepts that will be explored in subsequent chapters.

How should an intelligent system act such that some notion of long-term performance is maximized? This is the Reinforcement Learning (RL) problem, and is the main topic of this book.[1] To make this more clear and concrete, let us consider some examples. **RL Problem**

Consider a robot manipulator in an automobile factory. The robot perceives its workspace through cameras. It also has sensors that measure its joints angles as well as force sensors at the tip of its hand. It can send commands to its motors in order to move the joints to a certain position or velocity, or perhaps exert a certain amount of force on objects. Its goal is to successfully build a car as fast as possible with the minimum cost.

As another example, consider a smart HVAC (Heating, Ventilation, and Air Conditioning) system in a large office building. It can observe the temperature of the room using several thermometers, humidity sensors, and CO2 meters across the office. It may even have infrared cameras that can measure the temperature on the surfaces, hence providing a high-resolution temperature profile of the room. It can act in its world by varying the temperature, humidity, and the airflow rates of the vents distributed across the building. Its goal is to maximize the long-term comfort and health of people working at the office, which are measured through occasional voice feedback (`Too hot!` or `I feel a bit cold!`), or perhaps through a smart

[1] Chapter's Version: 0.15 (2025 March 12).

watch that measures their heart rate and blood oxygen level.[a]

These two were examples of artificial systems. We may also consider an animal, a cat or dog perhaps, that observes its world through its various sensors (eyes, ears, nose, whiskers, etc.) and has a musculoskeletal system that allows it to move and change the world around it. The goal of the animal can be defined at various time scales: in the short term, it is to find food and water for its next meal, which of course can be seen as just a part of the longer plan of maximizing its chance of survival and reproduction.

All these examples can be interpreted as an entity (a robot, an HVAC system, or a animal) being a nexus of causal pathways. Let us relax the definition of an animal from a biological one to include certain artificial systems too. So a robot or even a smart HVAC system are animals too, albeit an artificial ones.

As the animal perceives its environment through its various sensors (cameras; thermometers; eyes and ears), it collects information about its surrounding. It becomes the convergent point, in a non-mathematical sense, of the information around it. The perceived information has a causal power on the animal, as it affects how the animal acts. This action is based on the animal processing its perceived information and making a decision. When the animal acts (moving a joint; blowing hot air; pouncing on a bird), it becomes the point where the causal pathways diverge from the animal and affects the world around it (a screw is picked; a corner and gradually the rest of a room warms up; a bird flies away). Now as time passes, the affected world offers new information to the animal, and this in turn leads to new decisions and actions by the animal.[b]

A central part of this causal convergent and divergent process is how the animal decides on how to act based on what it perceives. The animal should act such that its goals are achieved. Successfully achieving an animal's goals often requires the animal to consider the long-term consequences of its actions. For example, it is XXX

To have such an animal, one needs to design (or evolve) many components and processes. It has various sensors and actuators, whose suitability greatly affects the animal's chance of successfully achieving its long-term goals. These, we ignore. In this book, we solely focus on the decision making aspect of the animal. Specifically, we ask the question of how this animal should act so that some notion of long-term performance is maximized. This is the RL problem. This is admittedly a very general objective. One may argue that the computational aspect of solving the AI problem is equivalent to the RL problem.

RL Methods

It is notable that in addition to the RL problem, we may use RL to refer to a set of computational methods for solving the RL problem. What kind of computation an agent needs to perform 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 have been known as the RL methods. For example, a method such as Q-Learning, which we shall soon encounter as Algorithm 1.1 in Section 1.5, is a well-regarded RL method, but an evolutionary computation method, such as a genetic algorithm, 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 book, we focus on methods that are commonly studied within the "RL Community", though we have a short discussion of some of the evolutionary computation methods later in the book in Section 6.2.3.

So far, our explanation has been high-level, not very precise, and perhaps even a bit philosophical. Next, we discuss the setup of the RL problem in a more concrete way, before formalizing it precisely in the rest of this chapter.

1.1 Setup

In reinforcement learning, 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, which we consider to be discrete, 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

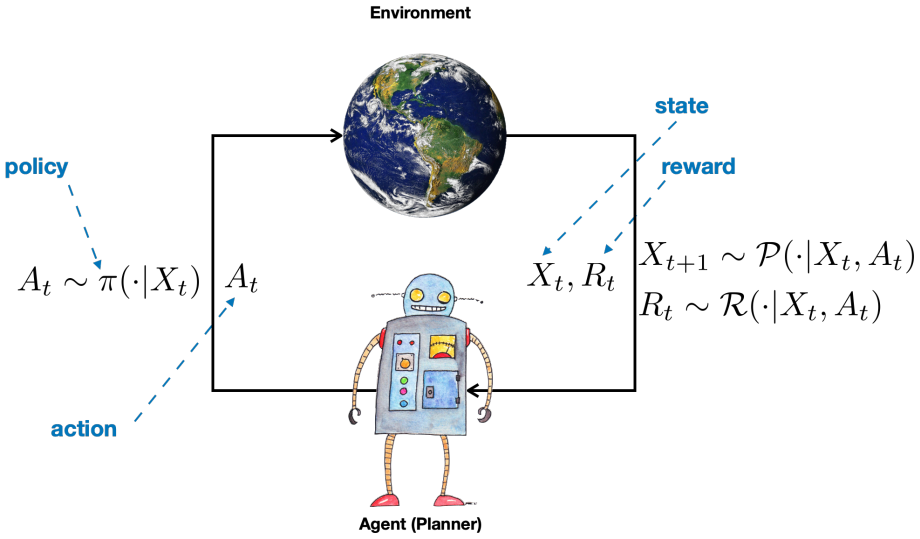

Figure 1.1: Reinforcement Learning Agent

random variable (r.v.) 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 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 simplify our notations. All the developed theory and algorithms also work with the latter form of the reward with minor modifications. The reward is a measure of the performance of the agent at time t. For example, if

our goal is for the 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 zero whenever the robot is not doing anything relevant to the goal. It may 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, $+10$.[c]

This process repeats and as a result the agent receives a sequence of states, 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.

The reward is a measure of immediate (or short-term) performance of the agent. This can be different from the long-term performance. It is possible for an agent to receive some low (or negative) rewards initially before receiving much larger rewards later on. What we often care in the RL is the long-term performance. Next we formalize this description.

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 and the problem that it should solve. In other words, an MDP encodes the decision problem.

In the rest of this book, 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.*[2]

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 $\rho \in \mathcal{M}(\mathcal{X})$.[3] 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 $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$.[4] 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* $a \in \mathcal{A}$*.*

As another example, consider a dynamical system described by the following equation:

$$x_{t+1} = f(x_t, a_t), \tag{1.1}$$

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

$$f(x, a) = Mx + Na,$$

[2] We do not use measure theoretical arguments in this book, but we use quantifiers such as *measurable* in order to make our statements precise and to avoid pathological cases. We can simply think of a measurable space as stating that the space is "nice enough". The finite and countable spaces as well as the usually used subsets of $\mathbb{R}^d$ are nice.

[3] The initial distribution ρ 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.

[4] We could slightly modify the interaction protocol, so that the reward R_t depends on X_t and A_t as well as X_{t+1}, i.e., $R_t \sim \mathcal{R}(\cdot|X_t, A_t, X_{t+1})$. This does not change the formalism.

with $M \in \mathbb{R}^{m\times m}$ and $N \in \mathbb{R}^{m\times n}$, we have a linear (time-invariant) dynamical system. Such a general formulation can represent a wide range of deterministic physical systems. Such dynamical systems are familiar for those with background in control theory. They can be represented in the MDP framework.

Example 1.2 (Deterministic Dynamics)**.** *We can represent deterministic dynamics such as* (1.1) *within the MDP framework. If* $x_{t+1} = f(x_t, a_t)$ *for* $f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$*, 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. This is similar to how Szepesvári [2010] uses it too. These are not the only notation used in the literature, and definitely not the most commonly used one. Sutton and Barto [2018] 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 a nod to 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* $\bar{\pi} = (\pi_1, \pi_2, \ldots)$ *such that for each* t*,*

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

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*

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

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

If π_t is parametrized only by X_t, that is

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

$\bar{\pi}$ 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 categorizes whether the policy is time-dependent or not, whether it uses only the current state X_t or looks at the previous state and action pairs too, and whether it is deterministic or stochastic.

To understand this definition better, let us start from the simplest form of the policy and gradually get to more general cases. The simplest form of a policy is a stationary Markov deterministic policy. This is a function from the state space $\mathcal{X}$ to the action space $\mathcal{A}$, that is $\pi : \mathcal{X} \to \mathcal{A}$, and we use $\pi(x)$ to refer to it. This policy is a function (deterministic property) that does not depend on time (stationary property). It ignores the past states and actions $X_{t-1}, A_{t-1}, X_{t-2}, \cdots$ and only looks at the current state X_t (Markov property). A slightly more complex form is when we allow this policy to be stochastic, instead of deterministic. A stationary Markov stochastic policy is a conditional distribution over the action space depending on the state, that is, $\pi(\cdot|x) \in \mathcal{M}(\mathcal{A})$.

In most of this book, we only focus on stationary Markov policies, and simply use "policy" to refer to a stationary Markov policy $\pi(\cdot|x)$, without any adjectives. It turns out that the class of stationary Markov policies is rich enough to allow the agent make optimal decisions, under the condition that we have access to the actual state of the MDP X_t. Neither non-stationarity nor non-Markovity does not bring any extra performance to the table. As we shall discuss later in Section 1.6, if the agent does not have access to the state of the MDP and only observe some aspects of the state, this is not necessarily true, and the use of a non-Markov policy might be needed for optimal decision making. This means that the policy should look not only at the most recent observation, but at the past observations too.

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*

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

If π *is stochastic, we have*

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

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*[5]

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

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.

The m-step transition probability kernels $(\mathcal{P}^\pi)^m(A|x)$ is the probability that the agent starts at state x, chooses actions according to the policy π for m steps, and falls within the set A. Similarly, $(\mathcal{P}^\pi)^m(B|x, a)$ is the probability of the agent starting from state x, choosing action a at the first step, and for the next $m-1$ steps, chooses actions according to the policy π.

We may sometimes use $\mathcal{P}^\pi(A|x; m)$ and $\mathcal{P}^\pi(B|x, a; m)$ to refer to $(\mathcal{P}^\pi)^m(A|x)$ and $(\mathcal{P}^\pi)^m(B|x, a)$, if having a superscript reduces the clutter.

In case thinking about countable space is more intuitive, the definition $(\mathcal{P}^\pi)^m(A|x)$ for A being a state z ($A = \{z\}$) is

$$(\mathcal{P}^\pi)^m(z|x) \triangleq \sum_{y \in \mathcal{X}} \mathcal{P}(y|x, \pi(x)) (\mathcal{P}^\pi)^{m-1}(z|y).$$

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

If we arrange the probabilities in a matrix, the definition of $(\mathcal{P}^\pi)^m$ takes a perhaps more familiar form. Consider a state space $\mathcal{X} = \{x_1, \ldots, x_{|\mathcal{X}|}\}$. Let us identify $(\mathcal{P}^\pi)(x_j|x_i)$ (the probability of starting from x_i and going to x_j) with an $|\mathcal{X}| \times |\mathcal{X}|$ matrix $P^\pi \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ with its i-th row and j-th

$$[P^\pi]_{i,j} = (\mathcal{P}^\pi)(x_j \mid x_i). \tag{1.2}$$

Consider $(\mathcal{P}^\pi)^2$, the 2-step transition probability kernel (or matrix), and let us calculate the probability of starting from state x_i and moving to state x_j after 2 steps:

$$(\mathcal{P}^\pi)^2(x_j|x_i) = \sum_{y\in\mathcal{X}} \mathcal{P}^\pi(y|x_i)\mathcal{P}^\pi)(x_j|y = \sum_{k\in\{1,\ldots,|\mathcal{X}|\}} P^\pi_{i,k}P^\pi_{k,j} = [P^\pi P^\pi]_{i,j} = [(P^\pi)^2]_{i,j},$$

where the penultimate equality is due to the definition of matrix multiplication. This shows that the 2-step transition probability kernel is the same as taking the matrix P^π and raising it to the power of two. This argument can be performed for any $m \geq 1$ to conclude that for countable state spaces, $(\mathcal{P}^\pi)^m$ can be identified with the matrix P^π raised to the power of m, i.e., $(P^\pi)^m$. In the rest of this notes, we do not use a different font for $\mathcal{P}$ and P, and use $\mathcal{P}$ for both cases.

A useful, and intuitive, property of following a policy π is that if the agent follows it for m_1 steps and then it continues following it for another m_2 steps, from wherever it landed after the first m_1 steps, it is the same as following the agent following π for $m_1 + m_2$ steps. This can be written as

$$(\mathcal{P}^\pi)^{m_1}(\mathcal{P}^\pi)^{m_2} = (\mathcal{P}^\pi)^{m_1+m_2}.$$

We define another notation, which shall be helpful in our discussions.

Definition 1.4. *Given the transition probability kernel $\mathcal{P}$ and a bounded measurable function $f \in \mathcal{B}(\mathcal{X})$, we define $\mathcal{P}f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ as the function*

$$(\mathcal{P}f)(x,a) \triangleq \int_{\mathcal{X}} \mathcal{P}(\mathrm{d}y|x,a)f(y), \qquad \forall (x,a) \in \mathcal{X} \times \mathcal{A}.$$

Likewise, given the transition probability kernel induced by a policy π, we define $\mathcal{P}^\pi f : \mathcal{X} \to \mathbb{R}$ as

$$(\mathcal{P}^\pi f)(x) \triangleq \int_{\mathcal{X}} \mathcal{P}^\pi(\mathrm{d}y|x)f(y), \qquad \forall x \in \mathcal{X}.$$

In other words, $\mathcal{P}^\pi f$ is the function whose value at a state x is the expected value of function f according to the distribution $\mathcal{P}^\pi(\cdot|x)$, that is, $(\mathcal{P}^\pi f)(x) = \mathbb{E}_{X' \sim \mathcal{P}^\pi(\cdot|x)}[f(X')]$. The interpretation of $\mathcal{P}f$ is similar.

For a countable state space $\mathcal{X}$, we have

$$(\mathcal{P}^\pi f)(x) \triangleq \sum_{y \in \mathcal{X}} \mathcal{P}^\pi(y|x) f(y), \qquad \forall x \in \mathcal{X}. \tag{1.3}$$

We may sometimes use $\mathcal{P}(\cdot|x,a)f$ or $\mathcal{P}^\pi(\cdot|x)f$ to refer to the same functions.

Exercise 1.1. *Consider a 2-state MDP with a policy that induces*

$$\mathcal{P}^\pi = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}.$$

Assume that the reward at state x_1 is zero and the reward at state x_2 is equal to 1, that is, $r^\pi = [0; 1]$. Let $\gamma = 1/2$. Answer the following questions assuming that the agent starts at state x_1:

- *What is the immediate reward it receives?*
- *What is the expected reward it receives after moving 1 step in the environment? What about 2? And 10? (You probably need to write one or two lines of code to compute this.)*
- *What is the expected reward it receives after moving infinite steps in the environment?*

Now answer the same questions for when the agent start at state x_2.

Exercise 1.2. *Suppose that $f(x) = c$ for a constant real-valued number $c \in \mathbb{R}$. What is $\mathcal{P}^\pi f$?*

1.2.1 Following a Sequence of Policies (†)

For a sequence of policies $\pi_{1:m} = (\pi_1, \ldots, \pi_m)$, the transition probability kernel of following them in the order of π_1, then π_2, etc., is denoted by $\mathcal{P}^{\pi_1:\pi_m}$ or $\mathcal{P}^{\pi_{1:m}}$ and is

$$\mathcal{P}^{\pi_{1:m}}(A|x) \triangleq \int_{\mathcal{X}} \mathcal{P}^{\pi_1}(\mathrm{d}y|x) \mathcal{P}^{\pi_{2:m}}(A|y),$$
$$\mathcal{P}^{\pi_1:\pi_m}(B|x,a) \triangleq \int_{\mathcal{X}} \mathcal{P}(\mathrm{d}y|x,a) \mathcal{P}^{\pi_2:\pi_m}(A|y),$$

for deterministic policies, and similar for stochastic policies.

The interpretation of $\mathcal{P}^{\pi_{1:m}}(A|x)$ is that this is the probability of starting from a state x, following the sequence of policies $\pi_{1:m}$, and ending up in a set A after exactly m steps (and similar interpretation for $\mathcal{P}^{\pi_1:\pi_m}(B|x)$). When the state space is countable, we can also write it in the matrix form:

$$\mathcal{P}^{\pi_{1:2}}(x_j|x_i) = \sum_{y\in\mathcal{X}} \mathcal{P}^{\pi_1}(y|x_i)\mathcal{P}^{\pi_2}(x_j|y) = \sum_{k\in\{1,\dots,|\mathcal{X}|\}} P^{\pi_1}_{i,k} P^{\pi_2}_{k,j} = [P^{\pi_1} P^{\pi_2}]_{i,j}.$$

As the matrices are not commutative in general $P^{\pi_1}P^{\pi_2} \neq P^{\pi_2}P^{\pi_1}$, which is intuitive, as following a policy π_1 and then π_2 (which induces $\mathcal{P}^{\pi_{1:2}}$) is not the same as following π_2 and then π_1 (which induces $\mathcal{P}^{\pi_{2:1}}$).

The value of function $\mathcal{P}^{\pi_{1:m}} f : \mathcal{X} \to \mathbb{R}$ at state x is the expected value of f at the distribution of an agent that starts at x and follows the policy sequence $\pi_1, \dots, \pi_m$.

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 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 the performance of the agent, one of which is to talk about the *average (expected) reward* that the agent receives within one episode. The 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.

expected reward

Answering the question of how the agent should act to maximize this notion of performance is easy. Let us define *expected reward* as

$$r(x,a) \triangleq \mathbb{E}\left[R|X=x, A=a\right]. \tag{1.4}$$

Here the r.v. R is distributed according to $\mathcal{R}(\cdot|x,a)$. Paying attention that this is the expected reward is important: even if the agent starts at a state x and chooses action a, the actual reward it receives can be higher or lower than $r(x,a)$. In expectation, however, it receives $r(x,a)$.

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 an action that maximizes the average reward $r(x,a)$ at that state. That is,

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

This is the best, or *optimal*, action at state x.[6] 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

$$\pi^*(x) \leftarrow \operatorname*{argmax}_{a\in\mathcal{A}} r(x,a). \tag{1.5}$$

Note that the optimal policy is only a function of the agent's state x. It does not depend on the initial distribution ρ.

Exercise 1.3. *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 1.4. *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 1.5 ($\star\star$)**.** *Come up with a real-world application where the goal is to find an optimal policy maximizing the immediate reward.*

Exercise 1.6 ($\star\star$)**.** *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.

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

1.3.1 Finite Horizon Tasks

The discussion of this section so far has been for when the episode length is $T = 1$. When $T = 1$, as soon as the agent chooses its action A_1, it receives a reward R_1, and then the episode terminates. We can extend the setup to when $T > 1$. In that case, 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 the agent gets to the last state $X_T \sim \mathcal{P}(\cdot|X_{T-1}, A_{T-1})$, chooses action $A_T = \pi(X_T)$ (or $A_T \sim \pi(\cdot|X_T)$ for a stochastic policy), and receives $R_T \sim \mathcal{R}(\cdot|X_T, A_T)$.

Afterward, the agent starts a new episode.[7,8]

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:

$$G^\pi \triangleq R_1 + \ldots + R_T. \tag{1.6}$$

return

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

$$G^\pi \triangleq R_1 + \gamma R_2 + \ldots + \gamma^{T-1} R_T. \tag{1.7}$$

[7] 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 $R_t \sim \mathcal{R}_t(\cdot|X_t, A_t)$. Also the policy might be non-stationary $\bar{\pi} = (\pi_1, \ldots, \pi_T)$, so at each time step t, the action is selected according to π_t. We comment on this further in Remark 1.2 at the end of this section.

[8] If the reward R_t depended on (X_t, A_t, X_{t+1}), as opposed to (X_t, A_t) that we consider here, the agent would get to X_{T+1} and terminates; it would not require to choose an action at the last step.

Whenever $\gamma < 1$, the reward that is received earlier in an episode contributes more to the return. Or similarly, the contribution of later rewards are discounted and contribute less (when $\gamma = 1$, there is no discounting, and all rewards contribute equality). Intuitively, this means that such a definition of return pays more emphasis on earlier rewards. An everyday example is that we prefer to get a cookie today instead of tomorrow, and we prefer a cookie tomorrow to a cookie a week later – assuming that we like cookies after all. How much exactly your preference changes depends on the value of γ. This is an example of a delayed gratification, which has been observed in humans. The Marshmallow test is famous example of it.[d]

The discount factor has a financial interpretation too and is related to the inflation rate.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 $\text{rate}_{\text{inflation}} > 0$, 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 (1.7) (and (1.6) as a special case) is a random variable. To define a performance measure that is not random, we compute its expectation. We define

$$V^{\pi}(x) \triangleq \mathbb{E}\left[\sum_{t=1}^{T} \gamma^{t-1} R_t | X_1 = x\right]. \tag{1.8}$$

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 $\tau \in \{1, \ldots, T\}$ until the end of episode, which is time T, as

$$G_{\tau}^{\pi} \triangleq \sum_{t=\tau}^{T} \gamma^{t-\tau} R_t. \tag{1.9}$$

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

$$V_{\tau}^{\pi}(x) \triangleq \mathbb{E}\left[G_{\tau}^{\pi} | X_{\tau} = x\right]. \tag{1.10}$$

Clearly, V_1^{π} is the same as V^{π} in (1.8).

Comparing the expected reward (1.4) and the value function (1.10) is instructive. We first focus on $T = 1$. We get that

$$V^\pi(x) = \mathbb{E}\left[R_1 | X_1 = x\right].$$

This is similar to $r(x, a) = \mathbb{E}\left[R | X = x, A = a\right]$ with the difference that $r(x, a)$ is conditioned on both x and a, whereas V^π is conditioned on x. The choice of action 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

$$r^\pi(x) \triangleq \mathbb{E}\left[R | X = x\right] \tag{1.11}$$

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 maximizing action, as in (1.5).[9] Finding the optimal policy given V^π may seem 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

$$\Pi = \left\{ \pi \,:\, \pi(\cdot|x) \in \mathcal{M}(\mathcal{A}), \forall x \in \mathcal{X} \right\}, \tag{1.12}$$

we need to find

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

If we find such a π^*, it is an optimal policy. But how can we solve this optimizatoin problem when the search is over the large policy space (1.12)?

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 $x \in \mathcal{X}$,

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

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

$$\sup_{\pi(\cdot|x) \in \mathcal{M}(\mathcal{A})} \int \pi(\mathrm{d}a|x) r(x, a). \tag{1.13}$$

[9] 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.

The maximizing distribution concentrates all its mass at the action a^* that maximizes $r(x, a)$, assuming it exists.[e] 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 next).

Remark 1.2. *In this section, we described the finite horizon task when the action is selected by a stationary policy: $A_t = \pi(X_t)$ or $A_t \sim \pi(\cdot|X_t)$. This is only for simplicity of exposition. More generally, the policy can be non-stationary, so $\bar{\pi} = (\pi_1, \dots, \pi_T)$ (Definition 1.2). The definition of the value function (1.8) would be the same, with the understanding that the selected action at each time step t comes from policy π_t. We may occasionally use $V^{\bar{\pi}}$ to emphasize that we are talking about a non-stationary policy. We also occasionally use $\pi_1 : \pi_T$ or $\pi_{1:T}$ to refer to the policy sequence $(\pi_1, \cdots, \pi_T)$ and $V^{\pi_1:\pi_T}$ or $V^{\pi_{1:T}}$ to its corresponding policy.*

1.3.2 Episodic Tasks

In some scenarios, there is a 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 terminates whenever the agent reaches a certain state x_{terminal} within the state space, that is, it terminates whenever $X_T = x_{\text{terminal}}$.[10] These are called *episodic* tasks. In episodic problems, the length of the episode T is a random variable. We define the

episodic tasks

[10] 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.

return and the value function as before: For $0 \leq \gamma \leq 1$, we have

$$G^\pi \triangleq \sum_{t=1}^{T} \gamma^{t-1} R_t, \tag{1.14}$$

$$V^\pi(x) \triangleq \mathbb{E}\left[G^\pi | X_1 = x\right]. \tag{1.15}$$

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 (think of the case that 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].

Exercise 1.7. *Describe several real-world applications that are best modelled as an episodic task.*

Exercise 1.8. *Suppose that we want to solve a goal reaching task, as an episodic task with the choice of $\gamma = 1$. We formulate it in two different ways. In the first way, we set the immediate reward -1 whenever we are not at the goal, and 0 whenever we reach the goal, which is the terminal state too. This encourages the agent to get to the goal sooner. In the second way, we add a constant reward $+1$ to the reward function. So, at all states, the reward is 0, except at the goal/terminal state where it is $+1$. This does not encourage getting to the goal state faster, as a later goal reaching has the same value as an earlier one. This sounds like a contradiction. A constant change in the reward function should not change the optimal policy, yet the intuitive argument here indicates that it does. What is happening?*

1.3.3 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 real world lasts forever, even the livable universe itself, 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 $0 \leq \gamma < 1$, we define

the return from time τ forward as

$$G_\tau^\pi \triangleq \sum_{t \geq \tau} \gamma^{t-\tau} R_t. \tag{1.16}$$

We can also define two value functions. One of them is V^π and similar to what we have seen so far. We call it state-value function or simply value function. Another one is called the action-value function. Since these are the value functions that we will use for the rest of the book, we define them formally.

action-value function

Definition 1.5 (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 other value function is called the action-value function, and is very useful in our further developments. 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. Note than

$$\begin{aligned} V^\pi(x) &= \mathbb{E}\left[G_1^\pi | X = x\right], \\ Q^\pi &= \mathbb{E}\left[G_1^\pi | X = x, A = a\right], \end{aligned} \tag{1.17}$$

by definition.

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

$$V^\pi(x) = \mathbb{E}\left[Q^\pi(x, A)\right] = \int \pi(\mathrm{d}a|x) Q^\pi(x, a). \tag{1.18}$$

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 the action-value function of the optimal policy, which shall be defined soon in Section 1.4, 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.,

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

Exercise 1.9. *Describe several real-world applications that are best modelled as a continuing task.*

Exercise 1.10. *Consider the MDP in Exercise 1.1. Compute the value function V^π for the discount factors $\gamma = \{0, 0.5, 0.9\}$.*

Exercise 1.11. *Consider a set of states $x_1, x_2, \ldots, x_N$. The agent has two actions a_{Left} and a_{Right}. Whenever the agent chooses action a_{Right} at state x_i, it goes to state x_{i+1}, unless $i = N$, in which case it stays there. Similarly for a_{Left}, except that it moves to x_{i-1}, unless $i = 1$, in which case it stays there. The reward function is 0 everywhere except at state x_N, in which it is $r(x_N) = +1$, and x_1, in which it is $r(x_1) = -1$. Suppose that $\gamma = 1$.*

- *What are $\mathcal{P}(\cdot|\cdot; a_{\mathsf{Left}})$ and $\mathcal{P}(\cdot|\cdot; a_{\mathsf{Right}})$? The answers should be an $N \times N$ matrix.*
- *Consider π_{Left}, which always chooses action a_{Left}. What are $V^{\pi_{\mathsf{Left}}}$ and $Q^{\pi_{\mathsf{Left}}}$?*
- *Answer the previous question for π_{Right}.*
- *Consider policy π_{Uniform}, which at each state, chooses each action with the same probability of $\frac{1}{2}$. What are $V^{\pi_{\mathsf{Uniform}}}$ and $Q^{\pi_{\mathsf{Uniform}}}$?*

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 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, for example, $A_t \sim \pi(\cdot|X_t, X_{t-1}, X_{t-2}, \ldots)$ 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 $x \in \mathcal{X}$.[11] This is shown in Figure 1.2. We also use a

[11] This is a partial order relationship. It is possible that for two policies, none of them is better than the other one.

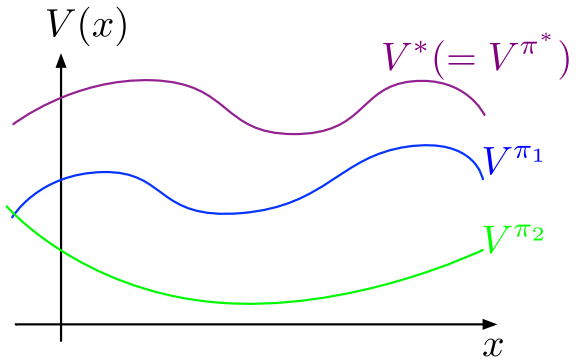

Figure 1.2: For any policy π, we have that $V^{\pi^*} \geq V^{\pi}$. Here the values V^{π_1} and V^{π_2} of two sub-optimal policies π_1 and π_2 are shown.

strict inequality $\pi > \pi'$ if $V^{\pi}(x) \geq V^{\pi'}(x)$ for all states $x \in \mathcal{X}$ and there exists at least a single state $x' \in \mathcal{X}$ such that the inequality is strict, that is $V^{\pi}(x') > V^{\pi'}(x')$.

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, the goal of finding an optimal policy can be written down as the following optimization problem:

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} V^{\pi}, \tag{1.19}$$

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.,

$$\pi^* \leftarrow \operatorname*{argmax}_{\pi \in \Pi} Q^{\pi}. \tag{1.20}$$

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 (1.5) and (1.13). 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 (that is, being better than) 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 $\mathcal{X}$, which perhaps depends on π, but not at all states in $\mathcal{X}$.

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

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

And even more interesting question is how we can *learn* π^*, or a close approximation thereof, without actually knowing the MDP, but only by using samples coming from the interaction of the agent with the MDP. This is the RL problem.

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 continuing 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, and the corresponding methods are called *Planning algorithms*.

planning

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 planning algorithms can be modified to become a learning algorithm. Therefore, it is good to get a good grasp of planning algorithms 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 Chapter 5. [12]

1.5 An Instance of an RL Algorithm: Q-Learning

It takes a while before we get into the detail of any RL algorithm, so it is good to see an example of such an algorithm before starting our excursion into the properties of

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

Algorithm 1.1 Q-Learning (Simplified)

Require: Step size update rule $\alpha \in (0, 1]$

1: Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, except that for x_{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: $Q(X_t, A_t) \leftarrow Q(X_t, A_t) + \alpha \left[R_t + \gamma \max_{a' \in \mathcal{A}} Q(X_{t+1}, a') - Q(X_t, A_t)\right]$. ▷ Q-Learning Update Rule
8: **end for**
9: **end for**

MDPs (Chapter 2) and the planning methods (Chapter 3) until we finally get to RL algorithms in Chapter 4.

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 $0 \leq \varepsilon \leq 1$ 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,

$$A_t = \begin{cases} \text{argmax}_{a \in \mathcal{A}} Q(X_t, a) & \text{w.p. } 1 - \varepsilon \\ \text{uniform}(\mathcal{A}) & \text{w.p. } \varepsilon \end{cases} \tag{1.21}$$

Usually the value of ε is small and may go to zero as the agent learns more about its environment. This occasional random choice of actions ensures that the agent explores its environment. Studying exploration is the subject of Chapter **??**.

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, the Q-Learning algorithm is guaranteed to converge to

the optimal action-value function Q^*.[13] 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.

Exercise 1.12 (Programming)**.** *Implement the Q-Learning algorithm (Algorithm 1.1), and try it for the deterministic MDP described in Exercise 1.11. As that is a continuing task, the episode does not end. You can let the algorithm run for different number of steps, for example,* 1000*,* 10000*, and* 100000*. Answer the following questions for each different number of steps.*

- *Plot* $Q(\cdot, a_{\mathsf{Left}})$ *and* $Q(\cdot, a_{\mathsf{Right}})$*.*
- *For which states* $Q(\cdot, a_{\mathsf{Left}})$ *is larger than* $Q(\cdot, a_{\mathsf{Right}})$*? How do you interpret it?*

1.6 A Few Remarks on the MDP Assumption

Before finishing this chapter, we have several remarks about the MDP assumption. Specifically, we ask the following questions:

- What is the state variable?
- Where does the reward signal come from?

Although most of our focus in this book will be on methods to design an RL agent, assuming that the MDP is given to us, thinking about these questions is nonetheless important for any RL practitioner and researcher.[14]

1.6.1 On State

What is a state variable?

Perhaps the most crucial remark on the MDP assumption 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, that is, its *history*. 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.

[13] For convergence of the Q-Learning algorithm, the step size should gradually converge to zero. We skip these detail here.

[14] And perhaps we expand on these in a future edition of this book.

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 or far from it (as the velocity information cannot be inferred from the position information alone). Or as another example, if the agent can only observe the blood pressure and heart rate at the moment, we cannot know everything that could be known about the patient, for example, whether the heart rate and blood pressure is suddenly spiking up or they have been up for a long time. 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 **history**

$$H_t = (O_1, A_1, R_1, \ldots, O_{t-1}, A_{t-1}, R_{t-1}, O_t).$$

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

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

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.[15] 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

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

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 the book, 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 equation

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

[15] We do not use "compact" in the formal sense used in topology, but in an informal sense meaning being concise.

where $z \in \mathbb{R}^m$, $a \in \mathbb{R}^n$, and $f : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^m$. This is the same dynamics we encountered before (1.1).

Suppose that the observation is $o_t = z_t$. In this case, we do not need to keep $h_t = (z_1, a_1, \ldots, z_{t-1}, a_{t-1}, z_t)$ 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 having a property of 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 (1.1).[16] If we have such a description of the dynamics, the state is often clear as long as we observe the right variable.

Exercise 1.13. *A ball is free falling under the Earth's gravity. The state is the vector described by its location $x(t)$ and 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 1.14. *In Atari games, a single frame is not a state of the agent. Explain why.*

Exercise 1.15. *We just claimed that $\mathbb{P}\{O_{t+1}|O_t, A_t\}$ has more uncertainty than $\mathbb{P}\{O_{t+1}|H_t, A_t\}$.*

- *Formalize this claim. (Hint: Take a look at Appendix A.7.)*
- *Prove it.*

1.6.2 On Reward

Where does reward come from?

How do we determine the reward signal R_t? The reward signal encodes the desire of the problem designer, so it is a part of the problem specification. Therefore, how to come up with a reward signal is a separate question from how to solve the planning or learning problem. In practice, however, when an agent designer wants to design the whole system, they have to both choose the learning algorithm as well as design a

[16] 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.

good reward signal to specify the task. Depending on the task, designing the reward signal can be easy or difficult. Let us briefly comment on it.

In some tasks, the reward function is relatively easy to define. For example, in control engineering, one is often interested in ensuring that the state of a system, described by a dynamical system such as (1.1), reaches a predefined state as quick as possible. For example, if the desired state is state $x = 0$, the reward might be defined as

$$R_t = - \|X_t\|_2^2 .$$

This, however, ignores the cost associated with the choice of actions. To incorporate that cost, we can define the reward as

$$R_t = - \left[\|X_t\|_2^2 + c_a \|A_t\|_2^2 \right] ,$$

for some $c_a > 0$. This is known as the quadratic cost model in control engineering.

For some other problems, the reward might be defined at a higher and more abstract level. For instance, in the robot manipulator in an automobile factor example mentioned earlier in this chapter, the reward might be defined as successfully building a car according to the desired specifications. Whenever such a car is successfully built, the agent receives a reward of $+1$, and at other times, it receives a reward of 0. This is a valid definition for the reward, but since the reward is extremely *sparse*, in the sense that the agent does not receive a non-zero reward very often, the RL agent might have difficulty learning how to the problem. To see this, when the agent has just begun learning, the chance of successfully solving the task, and receiving a non-zero reward, is very slim. When the agent only receives zero reward, it cannot learn much. More concretely, consider what happens to an agent using the Q-Learning algorithm (Algorithm 1.1) when the reward R_t is equal to zero. The update rule for the action-value function would be

$$\begin{aligned} Q(X_t, A_t) \leftarrow & Q(X_t, A_t) + \alpha \left[0 + \gamma \max_{a' \in \mathcal{A}} Q(X_{t+1}, a') - Q(X_t, A_t) \right] \\ & = (1 - \alpha) Q(X_t, A_t) + \alpha \gamma \max_{a' \in \mathcal{A}} Q(X_{t+1}, a'). \end{aligned}$$

If the Q function is initialized as any constant function (zero, for example), as long as the agent has not received any non-zero R_t, the right-hand side (RHS) is equal to the same constant function – the action-value function does not change. No change in the value function shows that the agent does not learn anything.

This is an extreme example to show that the sparse reward might cause difficulty in learning, even though from the goal specification standpoint, the reward is correctly specified.

As another example, consider the smart HVAC system that optimizes the comfort of the occupants. For that problem, the reward might be directly provided by the occupant of the space, given in the form of occasional voice feedback about their level of comfort.

Sometimes we can avoid directly specifying the reward function, but instead try to infer it from other information available to the agent.

One approach is to assume that we have access to an *expert* who knows how to solve the problem, and we can observe their behaviour. The goal is then to find a reward function whose optimal policy leads to a behaviour similar to the expert's. This is called the *inverse reinforcement learning* problem.

The expert data can be in the form of observing the set of states and actions the expert has selected, or only the set of states the behaviour of the expert generated. As a concrete example, consider that the expert has a policy π_E, which is unavailable to the agent. The expert follows policy π_E in an environment with the transition dynamics $\mathcal{P}$. As a result, a sequence of data in the form of $X_1^E, A_1^E, X_2^E, A_2^E, \ldots$ is generated. Assume that the agent can only observe the states of the expert, and not its actions: $X_1^E, X_2^E, \ldots$ (observing the actions is also a possible setup of IRL). The goal of the IRL problem is to find the reward distribution $\mathcal{R}$ such that the optimal policy $\pi^*(\mathcal{P}, \mathcal{R})$ leads to a similar distribution of states as $X_1^E, X_2^E, \ldots$. Different IRL methods use different notions of similarity and take different approaches to compute this unobserved reward distribution. [f]

Instead of providing a scalar reward, we may provide the agent with our preference over its choice of actions or induced trajectory. This is called *preference-based reinforcement learning*. In one approach to this problem, we first learn a scalar reward model that conforming to the preferences, and then use a regular RL algorithm to optimize using this learned reward model instead. The preference-based RL approach in the context of training and aligning the Large Language Models (LLM) with human preferences has attracted much attention under the name of RL from Human Feedback (RLHF) in recent years.[g]

Finally, we would like to remark that in biological animals, the reward signal has not been designed, but has been evolved. Their reward mechanism has been evolved so that the chance of survival and successful reproduction increases. Other than a few exceptions, the RL community does not tend to combine evolution of rewards and a learning-based RL algorithm.[h]

Throughout this course, we assume that the reward signal is given, but we note that much research has been done in how to specify reward.

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 and successful reproduction increases. Throughout this course, we assume that the reward signal is given, but we note that much research has been done in how to specify reward. [i]

1.6.3 On Time

A modelling assumption of the MDP model is that the agent makes decision on its actions at regular time steps $1, 2, \ldots$. What does time actually mean here?

For certain problems, such as board games, the meaning of these decision times is clear: each decision time corresponds to a move of a piece on a chess board. In those problems, the notion of time is abstract and corresponds to the number of steps the game is played.

For an agent living in a physical world such as a robot, or a simulation thereof, the correspondence is usually through a discretization of the continuous flow of time. Each decision time $t = 1, 2, \ldots$ correspond to the physical time $\mathtt{t} = t\Delta t$, where Δt is the discretization resolution. For some tasks, such a control of a robot, the discretization resolution might be in the order of milliseconds, while for a slower process such as the temperature of a room in the smart HVAC problem, the discretization resolution might be in the order of minutes.

1.7 Applications of Reinforcement Learning

Chapter Summary

Summarize the main points that the reader needs to remember from this chapter.

Notes and Remarks

a Relevant citations: Farahmand et al. [2016, 2017]; Pan et al. [2018]. More by others?

b The viewpoint of an animal as the nexus of causal pathways is from Chapter 5 (The Origin of Subjects) of Godfrey-Smith [2020]. I include it because I believe it is a good metaphor on what an intelligent being is, no matter biological or artificial. Of course, it is not a mathematically rigorous definition of what an animal is, and should not be interpreted as such.

c Designing a reward function that reflects our intended goals clearly is not always easy. This is the problem of agent alignment and is a subject of active research in the AI safety community. . If the animal is evolved, however, the problem of designing a proper reward function is taken care of by the evolutionary process: if the reward function does not lead to successful reproduction, the genes that encode that animal, and its reward function, will not survive for long.

d The original Stanford Marshmallow test is conducted by Mischel et al. [1972]. It appears that the discounting model of humans is better modelled by a hyperbolic model, instead of the geometric one. In that model, the influence of reward received after time t is $\frac{R_t}{1+\lambda t}$, instead of $\gamma^t R_t$, which we consider here. It also seems that humans have a relatively stable discounting parameters [Kirby, 2009]. The neural correlates of this discounting has also been studied, for example, by Casey et al. [2011].

e The assumption that the maximizer exists is technical, and may appear when the maximizing action does not belong to the action set. As an example, suppose that the state space is $\mathcal{X} = \mathbb{R}$ and the action set is $\mathcal{A} = (-1, +1)$, an open set. If the reward function is $r(x, a) = -(x - a)^2$, the maximizing action depends on whether $x \in (-1, +1)$ or outside it. If it is inside, the optimal action is to choose a being equal to x. But when x is outside that set, the optimal action is to be as close as $+1$ (for $x > 1$) or as close as -1 (for $x < -1$). But since the maximizing action cannot be realized within the action set, we can choose an action that is arbitrary close to $+1$ or -1.

f Some relevant papers for inverse RL are Russell [1998]; Ng et al. [2000]; Ramachandran and Amir [2007]; Ziebart et al. [2013]; Huang et al. [2015].

g Wirth et al. [2017] survey preference-based RL just before the recent surge of interest in RL from Human Feedback (RLHF). Some related papers directly related to RLHF are Wirth et al. [2017]; Christiano et al. [2017]; Ouyang et al. [2022]; Gheshlaghi Azar et al. [2024].

h What are good papers for this? Singh et al. has "Where Do Rewards Come From?", which seems relevant. And also "Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective" from Singh et al. I did some research about it during my MS (2003 or 2004), but unfortunately I didn't publish.

i There are several approaches to define or design a reward signal. In one approach, we assume that we have access to an expert who knows how to solve the problem, and we can observe their behaviour. The goal is then to find a reward function whose optimal policy leads to a behaviour similar to the expert's. This is called *inverse reinforcement learning* problem [Russell, 1998; Ng et al., 2000; Ziebart et al., 2013; Huang et al., 2015]. An active area of research is to convert the preference of For example, one may try to convert the preference of humans into reward function. , for example, in order to capture human We should mention that there has been work in evolving the reward signal itself.

Chapter 2

Structural Properties of Markov Decision Processes

Chapter Introduction

We get deeper into the MDPs and talk about some of their important aspects and properties. We talk about the Bellman Equations (Section 2.1), which are equations encoding the recursive structure of the value function, and the Bellman Operators (Section 2.3), which are convenient mathematical objects to talk about that recursive structure in a compact way. We discuss two important properties of the Bellman operators in Section 2.4: monotonicity and contraction. These properties allow us to prove that the solution of the Bellman equation is unique and that the stationary policies are enough to represent the optimal policy for an MDP.
By the end of this chapter, at the very least, you need to *remember* Bellman Equation, Bellman Operator, Greedy Policy, Banach Fixed Point theorem. You need to *understand* what do Bellman equation encode and why do we use them? and What do contraction and monotonicity mean?. And you need to be able to comfortably *apply* the contraction property.

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. These properties are important for the planning and learning methods that we will develop

[1]Chapter's Version: 0.15 (2025 August 2).

in later chapters. Here, we focus on the discounted continuing tasks.

2.1 Bellman Equations

The value functions satisfy a certain recursive structure, which is described by the so-called Bellman equations. These equations, derived in this section, regularly appear whenever we talk about value functions.

2.1.1 Bellman Equations for Value Functions of a Policy

Consider a sequence of rewards $R_1, R_2, \ldots$ obtained by following a policy π. Recall that the return G^π (1.16) 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.

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 different A'_t, next-state X'_{t+1}, reward R'_t, etc. The resulting return G'^π_t takes a different value than G_t^π. Its distribution, however, is the same. This means that their expectation, which is the value function at state x (see Definition 1.5 and (1.17)), is the same. By computing the expected value of the return, we can reveal an important recursive property of the value function V^π. For any state $x \in \mathcal{X}$, we have

$$\begin{aligned} V^\pi(x) &= \mathbb{E}\left[G_t^\pi \mid X_t = x\right] \\ &= \mathbb{E}\left[R_t + \gamma G_{t+1}^\pi \mid X_t = x\right] \\ &= \mathbb{E}\left[R(X_t, A_t) \mid X_t = x\right] + \gamma \mathbb{E}\left[G_{t+1}^\pi \mid X_t = x\right] \\ &= r^\pi(x) + \gamma \mathbb{E}\left[V^\pi(X_{t+1}) \mid X_t = x\right], \end{aligned} \tag{2.2}$$

where in the first equality, we simply substituted the definition of the value function (Definition 1.5); 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 (1.11).

This equation is similar to (2.1) with some similarities and differences. In both, a measure of the performance of the agent at two consecutive times steps are related to each other. In (2.1), the performance measure is the return G^π, whereas in (2.2), the performance measure is the value function V^π itself. Another difference is that both sides of $G_t^\pi = R_t + \gamma G_{t+1}^\pi$ are random, but neither sides of $V^\pi(x) = r^\pi(x) + \gamma \mathbb{E}\left[V^\pi(X_{t+1}) \mid X_t = x\right]$ are random.

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

$$V^\pi(x) = r^\pi(x) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a|x)V^\pi(x'). \tag{2.3}$$

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

Bellman equation

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

The Bellman equation allows us to interpret the value function V^π as follows: The value of following a policy π starting from the state x is the reward that a π-following agent receives at that state plus the discounted expected (average) value that the agent receives at the next-state.

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, using the notation of Definition 1.4:

$$V^\pi = r^\pi + \gamma \mathcal{P}^\pi V^\pi. \tag{2.4}$$

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:

$$Q^\pi(x,a) = r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^\pi(x') \tag{2.5}$$

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

or more compactly,

$$Q^\pi = r + \gamma \mathcal{P} V^\pi, \tag{2.7}$$

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 (1.18). 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.

2.1.1.1 Value Function and the Discounted Future-State Distribution

The Bellman equation (2.4) can also be written as $(\mathbf{I} - \gamma \mathcal{P}^\pi) V^\pi = r^\pi$, after rearrangement. The operator (mapping) $(\mathbf{I} - \gamma \mathcal{P}^\pi)$ is invertible, as we shall see in the proof of Proposition 2.8, so we can write

$$V^\pi = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1} r^\pi. \tag{2.8}$$

This means that the value function of a policy π is the effect of the operator $(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}$ applied to the function r^π. The inverse $(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}$ can be expanded as the sequence $\sum_{t \geq 0} \gamma^t (\mathcal{P}^\pi)^t$, using Lemma A.2 in Appendix A.4. Therefore, we get

$$V^\pi = \sum_{t \geq 0} \gamma^t (\mathcal{P}^\pi)^t r^\pi. \tag{2.9}$$

This has an intuitive interpretation. Recall that $(\mathcal{P}^\pi)^t$ is the probability transition kernel of following π for t time steps (Definition 1.3), so $(\mathcal{P}^\pi)^t r^\pi$ (Definition 1.4) is

the expected value of reward r^π after following π for t steps. The RHS of (2.9) is the discounted summation of all these rewards obtained at t-steps. Not surprisingly, this is the same as the definition of the value function $V^\pi(x) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_t | X_1 = x\right]$ (Definition 1.5).

The form (2.8) is a good gateway to introduce the *discounted future-state distribution* ρ_γ^π. For a state $x \in \mathcal{X}$, $\rho_\gamma^\pi(\cdot|x) \in \mathcal{M}(\mathcal{X})$ defines a probability distribution over the state space as

discounted future-state distribution

$$\rho_\gamma^\pi(\cdot|x) = \rho_\gamma(\cdot|x; \mathcal{P}^\pi) \triangleq (1-\gamma) \sum_{t\geq 0} \gamma^t \mathcal{P}^\pi(\cdot|x;t) = (1-\gamma) \sum_{t\geq 0} \gamma^t (\mathcal{P}^\pi)^t(\cdot|t). \quad (2.10)$$

This probability distribution is a mixture distribution consisting of t-step transition kernels $\mathcal{P}^\pi(\cdot|x;t)$, with each term weighted according to $(1-\gamma)\gamma^t$. Using the same $(\mathbf{I} - \gamma\mathcal{P}^\pi)^{-1} = \sum_{t\geq 0} \gamma^t \mathcal{P}^\pi$, which we used to related (2.8) and (2.9), this distribution can also be written compactly as

$$\rho_\gamma^\pi = (1-\gamma)(\mathbf{I} - \mathcal{P}^\pi)^{-1}. \quad (2.11)$$

One interpretation of the distribution $\rho_\gamma^\pi(\cdot|x)$ is that the agent starts from state x and at each time step, it decides to follow π for another step with probability γ or terminates the episode altogether with probability $1-\gamma$. To follow t steps without interruptions, the agent should choose to keep following π for t consecutive steps, hence the proportionality to γ^t weighting. We have the $1-\gamma$ factor to ensure that this probability distribution is properly normalized. One can verify that $\rho_\gamma^\pi(\cdot|x)$ is a valid probability distribution, e.g., $\rho_\gamma^\pi(\mathcal{X}|x) = 1$.

The relevance of this distribution becomes clear by noting that the RHS of (2.9) is $[\sum_{t\geq 0} \gamma^t (\mathcal{P}^\pi)^t] r^\pi = \frac{1}{1-\gamma} \rho_\gamma^\pi r^\pi$. Or in an expanded form, the value function at state x is

$$\begin{aligned} V^\pi(x) &= \mathbb{E}\left[\sum_{t\geq 0} \gamma^t R_t | X_0 = x\right] = \sum_{t\geq 0} \gamma^t \mathbb{E}\left[R_t | X_0 = x\right] = \sum_{t\geq 0} \gamma^t \int \mathcal{P}^\pi(\mathrm{d}x'|x;t) r^\pi(x') \\ &= \frac{1}{1-\gamma} \int \rho_\gamma^\pi(\mathrm{d}x'|x) r^\pi(x') = \frac{1}{1-\gamma} \mathbb{E}_{X' \sim \rho_\gamma^\pi(\cdot|x)}\left[r^\pi(X')\right]. \end{aligned}$$

That is, the value function at a state x is the expected reward when X' is distributed according to $\rho_\gamma^\pi(\cdot|x)$. More compactly, this can be written as

$$V^\pi = \frac{1}{1-\gamma} \rho_\gamma^\pi r^\pi. \quad (2.12)$$

Exercise 2.1. *Describe a situation when the return G_t^π, starting from the same state x, would always be the same.*

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 $\pi : \mathcal{X} \to \mathcal{A}$.*

Exercise 2.5. *Write down the Bellman equation for the MDP of Exercise 1.1.*

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 satisfies (cf. (1.19))

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

Given an optimal policy, the optimal value function would be V^{π^*}.[2]

Does the optimal value function V^{π^*} satisfy a recursive relation similar to the Bellman equation for a policy π (2.3)? It certainly satisfies the Bellman equation for policy π^*, that is,

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

like any other policy. But can we find an equation for V^{π^*} that does not explicitly refer to π^*? It turns out that we actually can. 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

$$V^*(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^*(x') \right\}. \tag{2.13}$$

Bellman optimality equation

This equation is called the *Bellman optimality equation* for the value function. We

[2] 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. We discuss this in detail in Section 2.5.3.1.

prove the existence and uniqueness of V^* later in this chapter (Proposition 2.5 in Section 2.5.1). This claim alone, however, does not show that this V^* is the same as V^{π^*}. They could be different.

The second claim is that V^* is indeed the same as V^{π^*}, the optimal value function when π is restricted to be within the space of stationary policies (Proposition 2.9 in Section 2.5.3). This claim alone, however, does not preclude the possibility that we can find an ever more performant policy by going beyond 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 (Proposition 2.10 in Section 2.5.3.1).

These three claims together show that the Bellman optimality equation (2.13) reveals the recursive structure of the optimal value function $V^* = V^{\pi^*}$. There is no policy, stationary or non-stationary, with a value function better than V^*, for the class of discounted continuing MDPs. This is very reassuring because by finding V^*, through solving (2.13), we can find the value function of the optimal policy.

We do not discuss how to solve the Bellman optimality equation in this chapter, but we develop the foundation needed for solving that equation. Finding the solution itself will be the subject of Chapter 3, in the case the MDP is known (planning), and the chapters afterwards, in the case when the agent can only interact with the environment (reinforcement learning). In the rest of this book, 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^*}.

Bellman Equations for Optimal Action-Value Functions

To define the optimal state-value function, we started from 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 could also 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

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

as we did earlier in (1.20). This can be contrasted with the solution of (1.19), which for now we denote as π^*_V to emphasize its being a maximizer of V^{π}.

What is the relation between π^*_V and π^*_Q? Do they have the same value function? If they do, it shows that it does not matter whether we optimize V^{π} or Q^{π} to find the optimal policy. If they do not, that will undermine the relevance of the action-value function.

Fortunately, it turns out that they lead to the same value function. To understand this better, recall from Section 1.4 that we 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 $(x,a) \in \mathcal{X}$. We use strict inequality $V^\pi > V^{\pi'}$ when there is at least a single state x' where $V^\pi(x) > V^{\pi'}(x)$; similarly, we use $Q^\pi > Q^{\pi'}$ when there is a single state-action pair (x',a') for which $Q^\pi(x',a') > Q^{\pi'}(x',a')$.

The following proposition shows that one of these relations implies the other one.

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

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 (2.5). 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 $Q^\pi(x,a) \geq Q^{\pi'}(x,a)$.

For the strict inequality, the proof is essentially the same. A subtle point worth paying attention is that the strict inequality holds only for those state-action pairs (x,a) such that the support of $\mathcal{P}(\cdot|x,a)$ overlaps with $\mathcal{X}' = \{\, x' \,:\, V^\pi(x') > V^{\pi'}(x') \,\}$. □

The direction of this relationship is only one way: we have $V^\pi \geq V^{\pi'} \Rightarrow Q^\pi \geq Q^{\pi'}$, but $Q^\pi \geq Q^{\pi'} \not\Rightarrow V^\pi \geq V^{\pi'}$. Despite this, when we talk about the optimal policies π^*_V and π^*_Q, which satisfy $V^{\pi^*_V} \geq V^{\pi'}$ and $Q^{\pi^*_Q} \geq Q^{\pi'}$ for any policy π' by the definition of optimality, we are in a favourable situation that they have the same value function, as the next result shows.

Proposition 2.2. *Let $\pi^*_V \leftarrow \mathit{argmax}_{\pi\in\Pi} V^\pi$ and $\pi^*_Q \leftarrow \mathit{argmax}_{\pi\in\Pi} Q^\pi$. We have $V^{\pi^*_V} = V^{\pi^*_Q}$.*

Proof. We prove this result by showing that $V^{\pi^*_V} \geq V^{\pi^*_Q}$ and $V^{\pi^*_V} \leq V^{\pi^*_Q}$, which together imply that $V^{\pi^*_V} = V^{\pi^*_Q}$. We prove two statements:

Statement a) For all $x \in \mathcal{X}$, we have $V^{\pi^*_V}(x) \geq V^{\pi^*_Q}(x)$.

This is the consequence of π^*_V being the maximizer $\pi^*_V \leftarrow \mathrm{argmax}_{\pi\in\Pi} V^\pi$. This means that $V^{\pi^*_V} = \max_{\pi\in\Pi} V^\pi \geq V^{\pi'}$, for any policy π', including $\pi' = \pi^*_Q$. Therefore, $V^{\pi^*_V}(x) \geq V^{\pi^*_Q}(x)$ for any $x \in \mathcal{X}$.

Statement b) For all $x \in \mathcal{X}$, we have $V^{\pi^*_V}(x) \leq V^{\pi^*_Q}(x)$.

First note that by the maximizer proper of π^*_Q, we have $Q^{\pi^*_Q} = \max_{\pi \in \Pi} Q^\pi \geq Q^{\pi'}$ for any policy π', including $\pi' = \pi^*_V$. Therefore,

$$Q^{\pi^*_Q}(x, a) \geq Q^{\pi^*_V}(x, a),$$

for all $(x, a) \in \mathcal{X} \times \mathcal{A}$.

For the moment assume that $V^{\pi^*_V} > V^{\pi^*_Q}$, that is, the value function of the optimal policy π^*_V obtained by maximizing V^π is strictly better than the value function of the optimal policy π^*_Q obtained by maximizing Q^π. We use Proposition 2.1 with the choice of $\pi = \pi^*_V$ and $\pi' = \pi^*_Q$ to get that if $V^{\pi^*_V} > V^{\pi^*_Q}$, we also have $Q^{\pi^*_V} > Q^{\pi^*_Q}$. Under this assumption, there exists an $(x, a) \in \mathcal{X} \times \mathcal{A}$ such that $Q^{\pi^*_V}(x, a) > Q^{\pi^*_Q}(x, a)$. These two statements about the order of $Q^{\pi^*_Q}(x, a)$ and $Q^{\pi^*_V}(x, a)$ show that there exists an $(x, a) \in \mathcal{X} \times \mathcal{A}$ in which

$$Q^{\pi^*_Q}(x, a) \geq Q^{\pi^*_V}(x, a) > Q^{\pi^*_Q}(x, a).$$

This is impossible, so the statement $V^{\pi^*_V} > V^{\pi^*_Q}$ is not true. Hence, there does not exist an $x \in \mathcal{X}$ with $V^{\pi^*_V}(x) > V^{\pi^*_Q}(x)$. This is logically the same as stating that for all $x \in \mathcal{X}$, we have $V^{\pi^*_V}(x) \leq V^{\pi^*_Q}(x)$. This is Statement (b).

The consequence of Statements (a) and (b) is that for all $x \in \mathcal{X}$, we have $V^{\pi^*_V}(x) = V^{\pi^*_Q}(x)$, which is the desired conclusion. □

The result is that we can define the optimal policy as the one that satisfies $\pi^* \leftarrow \operatorname{argmax}_{\pi \in \Pi} Q^\pi$ or the one that satisfies $\pi^* \leftarrow \operatorname{argmax}_{\pi \in \Pi} V^\pi$, and they both have the same value function.

We have the *Bellman optimality equation* for the action-value functions, similar to (2.13). For any $(x, a) \in \mathcal{X} \times \mathcal{A}$,

$$Q^*(x, a) = r(x, a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x, a) \max_{a' \in \mathcal{A}} Q^*(x', a'). \tag{2.14}$$

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 $Q^* = Q^{\pi^*}$.

Exercise 2.6 ($\star\star$)**.** *Show that the other direction of Proposition 2.1 does not hold. That is, if for two policies π and π' we have $Q^\pi \geq Q^{\pi'}$, we may not have $V^\pi \geq V^{\pi'}$.*

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 pick a deterministic optimal policy $\pi^* : \mathcal{X} \to \mathcal{A}$ in the following manner: For any $x \in \mathcal{X}$, the optimal policy is[3]

$$\begin{aligned}\pi^*(x) &= \operatorname*{argmax}_{a\in\mathcal{A}} Q^*(x,a) \\ &= \operatorname*{argmax}_{a\in\mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V^*(x') \right\}. \end{aligned} \tag{2.15}$$

Let us discuss this to gain a better understanding. Suppose that the agent is at state x. To act optimally, it needs to act optimaly both at the current time step (Now) and at any time steps afterwards (Future). If at either Now or Future, it doesn't act optimally, the decision would not be optimal, as 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 gets to the next state $X' \sim \mathcal{P}(\cdot|x,a)$, given its choice of action a at x, it follows the optimal policy π^*. The value of following the optimal policy is $V^*(X') = V^{\pi^*}(X')$. 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 $\int \mathcal{P}(\mathrm{d}x'|x,a)V^*(x')$, the expected value of the optimal value function at the next states. 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 in (2.15). Of course, this is the same action that maximizes the RHS of (2.13).

greedy policy

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 even for any particular policy), the greedy policy $\pi_g : \mathcal{X} \times \mathcal{B}(\mathcal{X}\times\mathcal{A}) \to \mathcal{A}$ is defined as

$$\pi_g(x;Q) = \operatorname*{argmax}_{a\in\mathcal{A}} Q(x,a). \tag{2.16}$$

[3]This is the consequence of Proposition 2.6, which we shall prove.

Likewise, for a value function $V \in \mathcal{B}(\mathcal{X})$, the greedy policy is[4]

$$\pi_g(x; V) = \operatorname*{argmax}_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V(x') \right\}. \tag{2.17}$$

When we do not explicitly state the dependence on x, the symbols $\pi_g(V)$ 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 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 encode the information about the future, so we do not need to explicitly consider all possible futures.

Exercise 2.7. *(Continuation of Exercise 1.1) Consider a 2-state MDP with two actions a_1 and a_2 that have the following probability transition kernels:*

$$\mathcal{P}(\cdot|\cdot, a_1) = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}, \qquad \mathcal{P}(\cdot|\cdot, a_2) = \begin{bmatrix} 0.8 & 0.2 \\ 0.1 & 0.9 \end{bmatrix}.$$

Assume that the reward only depends on the state (so $r(x,a) = r(x)$) and at state x_1 is zero and at state x_2 is equal to 1, that is, $r(x) = [0; 1]$. Let $\gamma = 1/2$. Answer the following questions:

- *Consider $\pi_1 = [a_1, a_1]$ and $\pi_2 = [a_2, a_2]$ and calculate Q^{π_1} and Q^{π_2}.*
- *What are $\pi_g(x; Q^{\pi_1})$ and $\pi_g(x; Q^{\pi_2})$? Are they the same as π_1 and π_2?*
- *If they are different, calculate $Q^{\pi_g(x;Q^{\pi_1})}$ and $Q^{\pi_g(x;Q^{\pi_2})}$?*
- *Suppose Q^{π_1} is perturbed to $Q^{\pi_1} + \Delta$ where $\Delta \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$. Notice that for small perturbations Δ, $\pi_g(x; Q^{\pi_1})$ and $\pi_g(x; Q^{\pi_1} + \Delta)$ are the same. How much Δ can change before these two greedy policies are different? Express the change as a function of supremum norm: $\|Q\|_\infty = \max_{(x,a)} |\Delta(x,a)|$.*[a]

2.3 Bellman Operators

The Bellman equations can be seen as the fixed point equation of certain operators known as the *Bellman operators*. An operator, or a mapping, is a transformation **Bellman operators**

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

from one vector space to another vector space.[5] 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 mappings*

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

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

When the Bellman operator T^π is applied to a function V, the result is another function $T^\pi V$. When we write $(T^\pi V)(x)$, this means that we are evaluating the resulting function $(T^\pi V)$ at the state $x \in \mathcal{X}$, so its value is a real number. 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

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

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

We can write the Bellman operator compactly as the following mappings:

$$T^\pi V : V \mapsto r^\pi + \gamma \mathcal{P}^\pi V,$$
$$T^\pi Q : Q \mapsto r + \gamma \mathcal{P}^\pi Q.$$

Comparing the Bellman equations (2.3) and (2.6) 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, that is,

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

This is a compact form of Bellman equations, as we have seen before in (2.4) and (2.7).

Bellman optimality operator

We can define the *Bellman optimality operators* similarly.

[5] The operators are reviewed in Appendix A.3.1. 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.3.2.

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 mappings*

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

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

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 (2.13) and (2.14), 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 (1.12) 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

$$\Pi_{\text{det}} = \{ \pi \,:\, \pi(x) \in \mathcal{A}, \forall x \in \mathcal{X} \} = \mathcal{A}^{\mathcal{X}}. \tag{2.18}$$

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 function of actions $f \in \mathcal{B}(\mathcal{A})$, 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

$$\sup_{\mu\in\mathcal{M}(\mathcal{A})} \int \mu(\mathrm{d}a) f(a) = \max_{a\in\mathcal{A}} f(a). \tag{2.19}$$

To show this, we first show that the left-hand side (LHS) of (2.19) is greater than or equal to its RHS, and then we show that the RHS is greater than or equal to the LHS. Together they show the equality.

The LHS$\geq$RHS because we can define $\mu = \mu_{\text{Diract}}$ as a Dirac's delta function at one of the maximizers $a^* \leftarrow \text{argmax}_{a\in\mathcal{A}} f(a)$ (so $\mu_{\text{Dirac}}(a) = \delta(a - a^*)$, and the value of the RHS would be the same. Because of the optimizer property, the value of $\sup_{\mu\in\mathcal{M}(\mathcal{A})} \int \mu(\mathrm{d}a) f(a)$ is larger than the value of $\int \mu(\mathrm{d}a) f(a)$ for any μ, including μ_{Dirac}. This shows that

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

The LHS$\leq$LHS because 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 $f(a^*)$. That is, for any distribution $\mu \in \mathcal{M}(\mathcal{A})$, including the maximizer of $\sup_{\mu\in\mathcal{M}(\mathcal{A})} \int \mu(\mathrm{d}a) f(a)$, we have

$$\int \mu(\mathrm{d}a) f(a) \leq \int \mu(\mathrm{d}a) f(a^*) = f(a^*) \int \mu(\mathrm{d}a) = f(a^*) = \max_{a\in\mathcal{A}} f(a),$$

so it is not possible to find a distribution μ such as that the LHS is strictly greater than the RHS.

All these, in summary, show that

$$T^*V = \sup_{\pi\in\Pi_{\text{det}}} T^\pi V = \sup_{\pi\in\Pi} T^\pi V. \tag{2.20}$$

Therefore, the Bellman optimality operator is the supremum of the Bellman operator T^π over all stochastic or deterministic policies. A consequence of this result is that for any policy $\pi \in \Pi$ and any function V, we have

$$T^*V \geq T^\pi V. \tag{2.21}$$

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

Exercise 2.8 ($\star$). *(Continuation of Exercises 1.1 and 2.7) Consider a 2-state and 2-action MDP defined in Exercise 2.7. Let $V = [1; 1]$. Computer $T^{\pi}V$ and $T^{*}V$.*

Exercise 2.9 ($\star$). *Recall the definition of the greedy policy (2.17). Prove that $T^{\pi_g(V)}V = T^{*}V$.*

2.3.1 Sequence of the Bellman Operators (†)

Suppose that we have two policies π_1 and π_2. What is the meaning of the operator $T^{\pi_1}T^{\pi_2}$?[6] To understand what this operator does, consider a value function V, and see what the effect of $T^{\pi_1}T^{\pi_2}$ on V is. Denote $T^{\pi_2}V$ by U. We have

$$(T^{\pi_1}T^{\pi_2}V)(x) = (T^{\pi_1}U)(x) = r^{\pi_1}(x) + \gamma \int \mathcal{P}^{\pi_1}(\mathrm{d}z|x)U(z).$$

The function U at state $z \in \mathcal{X}$ is

$$U(z) = (T^{\pi_2}V)(z) = r^{\pi_2}(z) + \gamma \int \mathcal{P}^{\pi_2}(\mathrm{d}y|z)V(y).$$

Combining these two equations, we get

$$\begin{aligned}
(T^{\pi_1}T^{\pi_2}V)(x) &= r^{\pi_1}(x) + \gamma \int \mathcal{P}^{\pi_1}(\mathrm{d}z|x) \left[r^{\pi_2}(z) + \gamma \int \mathcal{P}^{\pi_2}(\mathrm{d}y|z)V(y) \right] \\
&= r^{\pi_1}(x) + \gamma \int \mathcal{P}^{\pi_1}(\mathrm{d}z|x) r^{\pi_2}(z) + \gamma^2 \int \mathcal{P}^{\pi_1}(\mathrm{d}z|x)\mathcal{P}^{\pi_2}(\mathrm{d}y|z)V(y) \\
&= r^{\pi_1}(x) + \gamma(\mathcal{P}^{\pi_1}r^{\pi_2})(x) + \gamma^2(\mathcal{P}^{\pi_1:\pi_2}V)(x),
\end{aligned}$$

where in the last line we used the definition of $\mathcal{P}^{\pi_1:\pi_2}$ in Section 1.2.1 and the notation of $\mathcal{P}f$ in Definition 1.4. Therefore, the function $T^{\pi_1}T^{\pi_2}V$ is

$$T^{\pi_1}T^{\pi_2}V = r^{\pi_1} + \gamma\mathcal{P}^{\pi_1}r^{\pi_2} + \gamma^2\mathcal{P}^{\pi_1:\pi_2}V,$$

and the operator $T^{\pi_1}T^{\pi_2} : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$ is

$$T^{\pi_1}T^{\pi_2} : V \mapsto r^{\pi_1} + \gamma\mathcal{P}^{\pi_1}r^{\pi_2} + \gamma^2\mathcal{P}^{\pi_1:\pi_2}V.$$

More generally, for a sequence of policies $\pi_1, \ldots, \pi_m$, we have

$$\begin{aligned}
T^{\pi_1}T^{\pi_2}\ldots T^{\pi_m}V &= r^{\pi_1} + \gamma\mathcal{P}^{\pi_1}r^{\pi_2} + \gamma^2\mathcal{P}^{\pi_1:\pi_2}r^{\pi_3} + \cdots + \gamma^{m-1}\mathcal{P}^{\pi_1:\pi_{m-1}}r^{\pi_m} + \gamma^m\mathcal{P}^{\pi_1:\pi_m}V \\
&= \sum_{k=1}^{m} \gamma^{k-1}\mathcal{P}^{\pi_1:\pi_{k-1}}r^{\pi_k} + \gamma^m\mathcal{P}^{\pi_1:\pi_m}V,
\end{aligned}$$

[6] You can skip this section in your first reading without interrupting the flow.

with the understanding that $\mathcal{P}^{\pi_1:\pi_0} = \mathbf{I}$, the identity operator.

This expression has an interesting interpretation: the function $T^{\pi_1}T^{\pi_2}\dots T^{\pi_m}V$ is the value function of following the non-stationary policy $\bar{\pi} = (\pi_1, \dots, \pi_m)$ in a finite horizon MDP with the terminal reward of V. To see this, note that $(\mathcal{P}^{\pi_1:\pi_{k-1}})(\cdot|x)$ is the distribution of following the sequence of policies $\pi_1, \dots, \pi_{k-1}$, starting from the initial state x. The function $\mathcal{P}^{\pi_1:\pi_{k-1}}r^{\pi_k}$ is the expected value of the reward function r^{π_k}, averaged according to the distribution of the agent after following $\pi_1, \dots, \pi_{k-1}$, that is $\mathcal{P}^{\pi_1:\pi_{k-1}}$. The final term $\mathcal{P}^{\pi_1:\pi_m}V$ corresponds to following the sequence of policies $\bar{\pi}$ and receiving the terminal reward of V. The summation takes the discounted sum of these values.

When all the policies are the same and equal to π, that is $\pi_1 = \pi_2 = \dots = \pi_m = \pi$, we have

$$(T^\pi)^{(m)}V = \sum_{k=1}^{m} \gamma^{k-1}(\mathcal{P}^\pi)^{k-1}r^\pi + \gamma^m \mathcal{P}^{\pi_1:\pi_m}V. \tag{2.22}$$

m-step Bellman operator

The effect of this *m-step Bellman operator* on a value function V, when evaluated at state x, is

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

When $V = 0$, the function $(T^\pi)^{(m)}0$ is the same as the value function of a finite horizon MDP with the episode length m, as defined in (1.8). Or likewise, the value function V^π for the m-step finite horizon MDP is $(T^\pi)^{(m-1)}r^\pi$.

We can combine the m-step Bellman operators to get new operators. One way is by linearly combining them. Consider the weight sequence $w = (w_1, w_2, \dots)$ with the property that $\sum_{i\geq 1} w_i = 1$ and $w_i \geq 0$ for all i. We define

$$T_w^\pi \triangleq \sum_{m\geq 1} w_m (T^\pi)^{(m)}. \tag{2.23}$$

The special case of $w_m = (1-\lambda)\lambda^k$ with $\lambda \in [0, 1)$ is commonly used and takes the form of

$$T_\lambda^\pi = (1-\lambda)\sum_{m\geq 1} \lambda^m (T^\pi)^{(m)}. \tag{2.24}$$

Another choice is

$$w_m = \begin{cases} \frac{1}{H} & m = 1, \dots, H \\ 0 & m > H \end{cases}$$

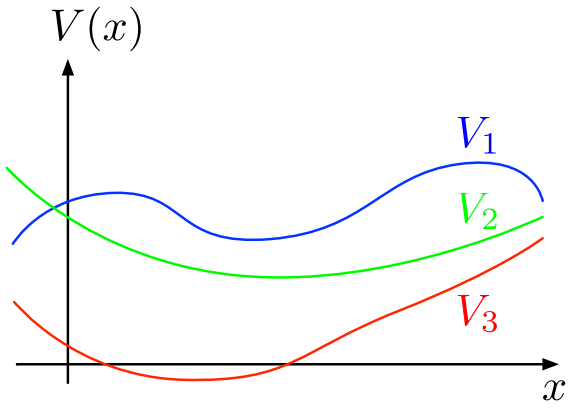

Figure 2.1: A few examples of the order relation between values: $V_3 \leq V_1$ and $V_3 \leq V_2$, but neither $V_2 \leq V_1$, nor $V_1 \leq V_2$.

We can see that V^π is a fixed point of $(T^\pi)^{(m)}$ too, that is, $(T^\pi)^{(m)}V^\pi = V^\pi$. It is a fixed point of T^π_w, including T^π_λ, as well. This can be seen by noting that if two operators L_1 and L_2 have the same fixed points, their weighted addition also has the same fixed point. This observation is interesting because it suggests that if we want to compute the fixed point of the Bellman operator, we may be able to use the fixed point of another operator, for example T^π_λ, instead.

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.

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.

Monotonicity

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

$$\begin{aligned} T^{\pi}V_1 &\leq T^{\pi}V_2, \\ T^*V_1 &\leq T^*V_2. \end{aligned} \tag{2.25}$$

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

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

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, $T^*V_1 \leq T^*V_2$. □

Tips and Tricks

Sometimes expanding the definition of the Bellman operator is all that you need to get started in the proof.

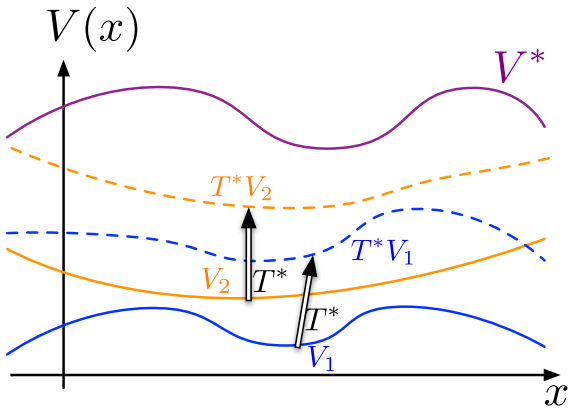

Figure 2.2: When $V_1 \leq V_2$, we also have $T^*V_1 \leq T^*V_2$.

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.[7]

Exercise 2.10 ($\star$)**.** *(Continuation of Exercises 1.1) Consider a 2-state MDP defined in Exercise 1.1. Let* $V_1 = [1; 0]$ *and* $V_2 = [2; 3]$*. Show that* $T^\pi V_1 \leq T^\pi V_2$*.*

Exercise 2.11 ($\star$)**.** *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 2.12 ($\star\star\star$)**.** *Suppose that the Bellman operator is 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?

[7] If it happens that for a value function V, we have $V \leq T^*V$, we can prove that V and T^*V are both smaller or equal to V^*. This is the case in this figure. We shall show such a result, but in the different direction of the inequality, in Section 3.5.

Exercise 2.13 ($\star\star\star$). *Suppose that the Bellman operator is defined as*

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

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.[8] This means that when we apply the Bellman operators to functions V_1 and V_2, the distance between TV_1 and TV_2 will be less than the distance between V_1 and V_2. To make this precise, we need to define a notion of distance between value functions and specify how much they get closer after the application of the Bellman operators.

supremum norm

We often use the *supremum norm* of value functions to define the distance between them (see Example A.5). 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

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

These norms define distances between two value functions V_1 and V_2 too, $d_\infty(V_1, V_2) = \|V_1 - V_2\|_\infty$, and similarly for the action-value functions.

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

Lemma 2.4 (Contraction). *For any π, the Bellman operator T^π is a γ-contraction mapping. Moreover, the Bellman operator T^* is a γ-contraction mapping. That is, for any $V_1, V_2 \in \mathcal{B}(\mathcal{X})$ or ($Q_1, Q_2 \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$) and T being either T^π or T^*, we have*

$$\|TV_1 - TV_2\|_\infty \le \gamma \|V_1 - V_2\|_\infty,$$
$$\|TQ_1 - TQ_2\|_\infty \le \gamma \|Q_1 - Q_2\|_\infty. \tag{2.26}$$

[8] Refer to Appendix A.3, and specifically Appendix A.3.2, 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).

Proof. We show the contraction property 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 $T^* : \mathcal{B}(\mathcal{X} \times \mathcal{A}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A})$. The proof for the value function V is similar.

Consider two action-value functions $Q_1, Q_2 \in \mathcal{B}(\mathcal{X} \times \mathcal{A})$. Let the metric be $d_\infty(Q_1, Q_2) = \|Q_1 - Q_2\|_\infty$. 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

$$\begin{aligned}|(T^\pi Q_1)(x,a) - (T^\pi Q_2)(x,a)| = \Bigg| \left[r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x')Q_1(x',a')\right] - \\ \left[r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x')Q_2(x',a')\right] \Bigg| \\ = \gamma \left| \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x') \left(Q_1(x',a') - Q_2(x',a')\right)\right|. \end{aligned} \tag{2.27}$$

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 $\left|\sum_x P(x) f(x)\right|$ for a countable state space). This can be upper bounded as

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

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

$$\begin{aligned}|(T^\pi Q_1)(x,a) - (T^\pi Q_2)(x,a)| &= \gamma \left| \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x') \left(Q_1(x',a') - Q_2(x',a')\right)\right| \\ &\le \gamma \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x') \left|Q_1(x',a') - Q_2(x',a')\right| \\ &\le \gamma \|Q_1 - Q_2\|_\infty \int \mathcal{P}(\mathrm{d}x'|x,a)\pi(\mathrm{d}a'|x') \\ &= \gamma \|Q_1 - Q_2\|_\infty . \end{aligned} \tag{2.28}$$

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,

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

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

$$\begin{aligned} |(T^* Q_1)(x,a) - (T^* Q_2)(x,a)| &= \gamma \left| \int \mathcal{P}(\mathrm{d}x'|x,a) \left(\max_{a' \in \mathcal{A}} Q_1(x',a') - \max_{a' \in \mathcal{A}} Q_2(x',a') \right) \right| \\ &\leq \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \sup_{x' \in \mathcal{X}} \left| \max_{a' \in \mathcal{A}} Q_1(x',a') - \max_{a' \in \mathcal{A}} Q_2(x',a') \right| . \end{aligned} \tag{2.29}$$

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 (2.29) is upper bounded by

$$\begin{aligned} (2.29) &\leq \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) \sup_{x' \in \mathcal{X}} \max_{a' \in \mathcal{A}} |Q_1(x',a') - Q_2(x',a')| \\ &= \gamma \sup_{(x,a) \in \mathcal{X} \times \mathcal{A}} |Q_1(x,a) - Q_2(x,a)| \int \mathcal{P}(\mathrm{d}x'|x,a) \\ &= \gamma \left\|Q_1 - Q_2\right\|_\infty . \end{aligned}$$

This proves the second claim. □

Remark 2.2. *It is important to note that the Bellman operators are γ-contraction w.r.t. the supremum norm. This may not hold for other norms, such as the $L_1(\nu)$ or $L_2(\nu)$, defined in (A.1). For example, it is not generally true for all MDPs and for any choice of distribution ν that $\|TV_1 - TV_2\|_{2,\nu} \leq \gamma \|V_1 - V_2\|_{2,\nu}$. As we will see later, this has important consequences when we talk about approximating the value function using a function approximators.*

Tips and Tricks

A commonly used proof and analysis strategy when we have terms in the form of $TQ_1 - TQ_2$ (or similar with V) is to get rid of the common terms (r) and try to move Q_1 and Q_2 close together, as is done in (2.27). Depending on the application, we may use some form of upper bounding, such as using Hölder inequality or Cauchy-Schwarz inequality, as is done in (2.28). Some commonly used inequalities are summarized in Appendix (A.8).

The m-step Bellman operator (2.22) is also a contraction, but with the contraction factor of γ^m. We prove it using more compact notations:

$$\begin{aligned}(T^\pi)^{(m)}V_1 - (T^\pi)^{(m)}V_2 = &\left[\sum_{k=1}^{m}\gamma^{k-1}(\mathcal{P}^\pi)^{k-1}r^\pi + \gamma^m\mathcal{P}^{\pi_1:\pi_m}V_1\right] - \\ &\left[\sum_{k=1}^{m}\gamma^{k-1}(\mathcal{P}^\pi)^{k-1}r^\pi + \gamma^m\mathcal{P}^{\pi_1:\pi_m}V_2\right] \\ &= \gamma^m\mathcal{P}^{\pi_1:\pi_m}(V_1 - V2).\end{aligned}$$

Taking the supremum norm from both sides and noticing that $\mathcal{P}^{\pi_1:\pi_m}$ is a stochastic kernel, thus its supremum norm is 1, get us to

$$\left\|(T^\pi)^{(m)}V_1 - (T^\pi)^{(m)}V_2\right\|_\infty \leq \gamma^m \left\|V_1 - V_2\right\|_\infty .$$

Exercise 2.14 ($\star$). *Write the proof of Lemma 2.4 for the state value function, that is, show that* $\|TV_1 - TV_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$ *for* $T \in \{T^*, T^*\}$.

Exercise 2.15 (Both $\star$ and $\star\star$). *Consider Exercise 2.10, in which* $\mathcal{P}^\pi = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$ *and* $r^\pi = [0; 1]$, *with the choice of discount factor* $\gamma = 0.9$.
Part I) ($\star$) *Calculate* $\|T^\pi V_1 - T^\pi V_2\|_\infty$ *and* $\|V_1 - V_2\|_\infty$ *for each of these cases, compute the empirical ratio of contraction (i.e.,* $\frac{\|T^\pi V_1 - T^\pi V_2\|_\infty}{\|V_1 - V_2\|_\infty}$*) and compare it with the* γ *of the* γ*-contraction property of the Bellman operator.*

- $V_1 = [1; 0]$ *and* $V_2 = [2; 3]$.
- $V_1 = [1; 1]$ *and* $V_2 = [-1; -1]$.
- $V_1 = [-1; 2]$ *and* $V_2 = [1; -2]$

Part II) *(⋆⋆) Why are the empirical rates different? Justify it using the property of $\mathcal{P}^\pi$.*

Exercise 2.16 (⋆⋆)**.** *Prove that for two functions $f_1, f_2 : \mathcal{A} \to \mathbb{R}$, we have*

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

Exercise 2.17 (⋆ ⋆)**.** *The Bellman operator is γ-contractive w.r.t. the supremum norm. Show that this may not hold for other norms, such as the L_1-norm. To prove this for T^π, show that*

$$\|T^\pi V_1 - T^\pi V_2\|_1 \leq C \|V_1 - V_2\|_1 ,$$

for some C that might be larger than 1. Tip: For simplicity, assume the state space $\mathcal{X}$ is finite.

Exercise 2.18 (⋆ ⋆)**.** *Consider the Bellman operator T^π_λ (2.24). Assume that $\lambda \in [0, 1)$. What is its contraction factor? Is it smaller or larger than γ?*

Exercise 2.19 (⋆ ⋆)**.** *Consider the Bellman operator T_{mult} defined in Exercise 2.12. Assume the assumption that makes the operator monotonic. Is it also a contraction operator? What conditions do you need for that to hold? And what is the contraction factor?*

2.5 Some Consequences of Monotonicity and Contraction

We have shown that the Bellman operators for a discounted MDP are both monotonic (Lemma 2.3) and γ-contraction w.r.t. the supremum norm (Lemma 2.4). These properties have important consequences, some of which we study here. One of the most important ones is that each of these operators has a unique fixed point, which in turn implies that each Bellman equations has a unique solution.

Table 2.1 briefly summarizes all the results of this section and indicates whether monotonicity or contraction properties of the Bellman operators have been used in the proofs of those results.

Result	Monotonicity	Contraction
Uniqueness of Fixed Points (Theorem 2.5)	✗	✓
Error Upper Bounds (Propositions 2.7 & 2.8)	✗	✓
Fixed point of T^* is Optimal Value (Proposition 2.9)	✓	✓
Stationary Policies are All You Need (Proposition 2.10)	✓	✓

Table 2.1: The use of Monotonicity and Contraction Properties in the proof of various results

2.5.1 Uniqueness of Fixed Points

Theorem 2.5 (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.,*

$$V^\pi = T^\pi V^\pi; \quad Q^\pi = T^\pi Q^\pi,$$
$$V^* = T^* V^*; \quad Q^* = T^* Q^*.$$

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

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

for $k = 0, 1, \ldots$. 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 supremum norm, i.e., $d_\infty(V_1, V_2) = \|V_1 - V_2\|_\infty = \sup_{x\in\mathcal{X}} |V_1(x) - V_2(x)|$. The space $(\mathcal{B}(\mathcal{X}), d_\infty)$ is a complete metric space.

Lemma 2.4 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), each of these operators has a unique fixed point. Moreover, any sequence (V_k) with $V_0 \in \mathcal{B}(\mathcal{X})$ and $V_{k+1} \leftarrow T^\pi V_k$ $(k = 0, 1, \ldots)$ 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 is an important result not only because it establishes the uniqueness of the fixed points, but also because it suggests a way to compute V^π and V^* by the

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.

Remark 2.3. *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). The completeness of $\mathcal{B}(\mathcal{X})$ is the direct implication of Theorem 43.6 of Munkres [2018], under $\mathcal{X}$ being a topological space and $\mathbb{R}$ being a complete space, which it is.*

In this proof, we only used the contraction property of the Bellman operators, and not their monotonicity. We will use the monotonicity later.

Exercise 2.20 ($\star$)**.** *Consider the MDP defined in Exercise 1.1 and Exercise 2.7. Compute V_1, V_2, V_3 (and if you like, a few more steps) of Theorem 2.5, for both T^π (based on Exercise 1.1) and T^* (based on Exercise 2.7).*

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

Proof. We first prove $T^\pi V^* = T^* V^* \Rightarrow V^\pi = V^*$. 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 (Theorem 2.5) 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 ($V^\pi = V^* \Rightarrow T^\pi V^* = T^* V^*$), 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 2.21 ($\star$). *Prove that $T^\pi Q^* = T^* Q^*$ if and only if $Q^\pi = Q^*$.*

This proposition shows that 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 in Section 2.5.3.

To see the connection to the greedy policy (2.17) 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

$$(T^{\pi_g(V)} V)(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V(x') \right\}. \tag{2.30}$$

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

$$(T^* V)(x) = \max_{a \in \mathcal{A}} \left\{ r(x,a) + \gamma \int \mathcal{P}(\mathrm{d}x'|x,a) V(x') \right\}. \tag{2.31}$$

Comparing the RHS of (2.30) and (2.31), we observe that both are indeed the same. This shows that

$$T^{\pi_g(V)} V = T^* V. \tag{2.32}$$

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. This is the basis of many algorithms in Planning and RL, which are sometimes referred to as the *value-based* methods, in which they aim to find $\hat{V}^*$ that is either exactly equal to V^*, or a close approximation thereof, and then use the greedy policy $\pi_g(\hat{V}^*)$, or a perturbation of it, as the policy of the agent. We start about the methods in Chapter 3. We next discuss the consequence of $\hat{V}^*$ being only an approximation.

Exercise 2.22 ($\star$). *Show that for any $\pi \in \Pi$ and any value function V, we have*

$$T^\pi V \leq T^{\pi_g(V)} V.$$

Hint: Look at (2.21).

Exercise 2.23 ($\star\star$). *State and prove similar results to Theorem 2.5 for T^π_{mult} (Exercise 2.12) and T^π_{min} (Exercise 2.13).*

2.5.2 Error Bounds

A consequence of the uniqueness property of the Bellman operators is that if we find a function V that satisfies the Bellman equation, that is $V = T^\pi V$ (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? 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 this question is important because, in practice, we may not be able to solve the Bellman equations exactly, but only approximately. There are several reasons for the solution to be only an approximation. They can broadly be categorized as computational, representational, and statistical. The finiteness of computational budget prevents our algorithms to be run forever, and for many algorithms, that finiteness limits how close they can get to the solution of the Bellman equations. Moreover, the use of a function approximator to represent the value function may introduce an error on how well we approximate the true value function. At the very least, if the true value function does not belong to the space of functions representable by the function approximator, we cannot hope to learn the true value function precisely. Furthermore, whenever we use data to compute the value function, as in the RL setting, we face statistical errors. We discuss these in more detail in later chapters. For now, we only assume that $V \approx T^\pi V$ (or $V \approx T^* V$) and ask how close the approximate value function V is to V^π (or V^*).

It turns out that we can relate the closeness of V to V^π based on the size of $\mathrm{BR}^\pi(V) \triangleq T^\pi V - V$ or $\mathrm{BR}^*(V) \triangleq T^* V - V$. The functions $\mathrm{BR}^\pi(V) \in \mathcal{B}(\mathcal{X})$ and $\mathrm{BR}^*(V) \in \mathcal{B}(\mathcal{X})$ are called the *Bellman Residuals*. The next result shows an upper bound on the error in approximating V^* by V based on the norm of the Bellman residual $\mathrm{BR}^*(V)$.

Bellman residual

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

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

Proof. We prove the result for V. We add and subtract T^*V to $V - V^*$, take the supremum norm, and use the triangle inequality as follows:

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

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. □

Bellman error

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 upper bound, as a crucial step in the proof is 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, as you hopefully showed in Exercise 2.17.

A closer look at both sides of the inequality in this result is worthwhile. Computing the LHS and RHS of the inequality requires different information or sources of compute. If we are only given V and we know the MDP, which means that we know the transition kernel $\mathcal{P}$, the reward kernel $\mathcal{R}$, and the discount factor γ, we can in principle compute T^*V by computing $(T^*V)(x) = \max_a \{r(x, a) + \gamma \mathcal{P}(\cdot|x, a)V\}$ for all states $x \in \mathcal{X}$. It is clear that this requires querying r and $\mathcal{P}$ and compute certain integrals. As a result, we can also compute $\|V - T^*V\|_\infty$. This is all the information and computation that we need. On the other hand, for computing the LHS, we need V^* too. The function V^* is not available to us as a part of the MDP itself. By spending some computation, we can solve the MDP and obtain V^*. Solving an MDP is the subject of Chapter 3 (and the chapters thereafter, for the approximate case) and is sometimes feasible, but it is often much more expensive than computing T^*V.

These bounds allow us to approximately compute $\|V - V^*\|_\infty$, which indicates how much V is different from the optimal value function V^*, without computing V^*, and with only using V and the ability to query the MDP in order to compute T^*V. The price we pay is that we only get an upper bound, which may not be tight for all choices of the value function V.[9]

Tips and Tricks

Three high-level ideas are used in this proof that sometimes appear in the proof of these kinds of upper bounds involving value functions and the Bellman operator.
The first is to add and subtract equal terms ($\pm T^*V$ in this proof) in order to get terms in familiar or intended form ($\mathrm{BR}^*(V) \triangleq T^*V - V$ here). The reason we do this is as follows: since we intended to have an upper bound in terms of $T^*V - V$ but we have $V - V^*$ on the LHS, we subtract T^*V to get the term $V - T^*V$ on the LHS, which is what we want to see on the RHS. Of course, subtracting T^*V changes the LHS, so in order not to change the LHS, we have to add T^*V too, so that overall we add $0 = -T^*V + T^*V$ to the LHS. The addition of T^*V means that we are now left with the term $T^*V - V^*$, which has to be dealt with.
The second idea is to benefit from the fixed-point property of the Bellman operator, which states that $V^* = T^*V^*$. This helps us to convert $T^*V - V^*$, which is not "balanced" in terms of the application of the Bellman operator, to $T^*V - T^*V^*$, which is balanced.
The third idea is to benefit from the contraction property of the Bellman operator, which is applicable because we now have $T^*V - T^*V^*$.

Exercise 2.24 ($\star$)**.** *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.*

We also have an essentially the same result for the Policy Evaluation case. We can upper bound the supremum norm of $V - T^\pi V$ by a function of the norm of the Bellman residual $\mathrm{BR}^\pi(V)$.

[9] Computing T^*V itself, even if we know r and $\mathcal{P}$, may not be computationally cheap or even feasible. This is usually an issue when the state and action spaces are large. Yet, it is still cheaper than computing V^*.

Proposition 2.8. *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*

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

Proof. This result can be proven essentially the same way as in the proof of Proposition 2.7. Here, we prove it differently.

We add and subtract $T^\pi V$, use the fact that V^π is the fixed point of the Bellman operator ($V^\pi = T^\pi V^\pi$), and rearrange both sides to get that

$$\begin{aligned} V - V^\pi &= V - T^\pi V + T^\pi V - V^\pi \\ &= (V - T^\pi V) + (T^\pi V - T^\pi V^\pi) \\ &= (V - T^\pi V) + \gamma \mathcal{P}^\pi (V - V^\pi) \\ \Rightarrow (\mathbf{I} - \gamma \mathcal{P}^\pi)(V - V^\pi) &= V - T^\pi V. \end{aligned} \tag{2.33}$$

As $\|\gamma \mathcal{P}^\pi\|_\infty = \gamma < 1$ is smaller than 1, Lemma A.2 in Appendix A.4 shows that the matrix $\mathbf{I} - \gamma \mathcal{P}^\pi$ is non-singular, hence it is invertible. Therefore, we can write the error in value function $V - V^\pi$ as a linear transformation of the Bellman residual $V - T^\pi V$:

$$V - V^\pi = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}(V - T^\pi V). \tag{2.34}$$

By taking the supremum norm of both sides, we get

$$\|V - V^\pi\|_\infty = \left\|(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}(V - T^\pi V)\right\|_\infty \leq \left\|(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}\right\|_\infty \|V - T^\pi V\|_\infty .$$

The inequality is because of the sub-multiplicativity of vector-induced matrix (or operator) norm: $\|AB\|_p \leq \|A\|_p \|B\|_p$ (see (A.12) in Appendix A.4; or (A.8) in Appendix A.3.1). We use the other part of Lemma A.2 to conclude that

$$\left\|(\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}\right\|_\infty \leq \frac{1}{1 - \|\gamma \mathcal{P}^\pi\|_\infty} = \frac{1}{1-\gamma}.$$

These together show that

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

as desired. □

Remark 2.4. *We can write the equality* $V - V^\pi = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}(V - T^\pi V)$ (2.34) *in a more compact form of*

$$V - V^\pi = (\mathbf{I} - \gamma \mathcal{P}^\pi)^{-1}(V - T^\pi V) = \frac{1}{1-\gamma} \rho_\gamma^\pi B R^\pi(V), \tag{2.35}$$

where we used the definition of the discounted future-state distribution (2.11).

Exercise 2.25 ($\star$)**.** *Prove Proposition 2.8 using a similar technique to Proposition 2.7.*

Are these upper bounds tight? They are tight in the sense that for some choices of the MDP and the value function V, the inequalities in Proposition 2.7 and 2.8 become equalities. This means that for that choice of the MDP and those particular value functions, the RHS in each of those results is exactly the same as the LHS, and there is no gap between them. This does not, however, mean that for a particular MDP and a particular function V, if we compute the RHS, the LHS would also be exactly the same, or just slightly smaller. The RHS can indeed be significantly larger than the LHS. How much larger can it be? We do not characterize the size of this gap, but we have an exercise for you to investigate it.[b]

Let us define the error bound tightness coefficient as the ratio of the upper bound in these results to the actual value of the quantity of interest (LHS) for a particular value of V:

$$\text{EB-subopt}^\pi(V) = \frac{\frac{\|V - T^\pi V\|_\infty}{1-\gamma}}{\|V - V^\pi\|_\infty}. \tag{2.36}$$

We also define

$$\text{EB-subopt}^\pi = \sup_{V \in \mathcal{B}(\mathcal{X})} \text{EB-subopt}^\pi(V). \tag{2.37}$$

The definitions of EB-subopt$^*(V)$ and EB-subopt* (cf. Proposition 2.7) are similar.

Exercise 2.26 (Error Bound Tightness)**.** *The goal of this exercise to study the tightness of the error bound in Proposition 2.8.*

Part I) ($\star$) *Consider Exercise 2.10, in which* $\mathcal{P}^\pi = \begin{bmatrix} 0.9 & 0.1 \\ 0.2 & 0.8 \end{bmatrix}$ *and* $r^\pi = [0; 1]$, *with the choice of discount factor* $\gamma = 0.9$. *As the reference, the value function for this problem is* $V^\pi = [2.432, 5.135]$. *Compute EB-subopt*$^\pi(V)$ *for the following value functions:*

- $V_1 = [1; 0]$
- $V_2 = [0; 1]$
- $V_3 = [1; 1]$
- $V_4 = V^\pi + [1; 0]$
- $V_5 = V^\pi + [0; 1]$
- $V_6 = V^\pi + [1; 1]$
- *Can you find a V with a larger EB-subopt$^\pi(V)$ than any of the previous examples?*

Part II) *($\star$) Choose $\mathcal{P}^\pi = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$ and repeat the same exercise.*

Part III) *($\star$) Choose $\mathcal{P}^\pi = \begin{bmatrix} 0.01 & 0.99 \\ 0.99 & 0.01 \end{bmatrix}$ and repeat the same exercise.*

Part IV) *($\star\star\star$) As the previous parts show, the value of EB-subopt$^\pi(V)$ depends on both V and the MDP. Show that there exists a value function V such that*

$$\textit{EB-subopt}^\pi(V) \geq \frac{1 - \gamma\lambda_{min}}{1 - \gamma},$$

where λ_{min} is the smallest eigenvalue of $\mathcal{P}^\pi$. Specify such a V explicitly.

Part V) *($\star\star\star$)* ***(or maybe*** *($\star\star\star\star$)* ***)*** *Is the lower bound on EB-subopt$^\pi(V)$ of the previous part tight? If it is, prove it. If it is not, can you come up with a tighter lower bound, or even better, have an equality expression for EB-subopt$^\pi$.*

Part VI) *($\star\star\star$)* ***(or maybe*** *($\star\star\star\star$)****)*** *Answer the same question for EB-subopt$^*(V)$ and EB-subopt*.*

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. We use the monotonicity of T^*, in addition to its contraction property, to prove this result.

Proposition 2.9 (Proposition 2.1.2 of Bertsekas 2018)**.** *Let V^* be the fixed point of T^*, i.e., $V^* = T^*V^*$. The function V^* is the optimal value function in the space of stationary policies, that is,*

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

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

For the first direction, we use Proposition 2.8 with the choice of $V = V^*$. We get that for any $\pi \in \Pi$,

$$\|V^* - V^\pi\|_\infty \le \frac{\|V^* - T^\pi V^*\|_\infty}{1-\gamma}. \tag{2.38}$$

Let $\varepsilon > 0$. Choose a policy a $\pi_\varepsilon \in \Pi$ such that

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

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, (2.38) shows that

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

This means that

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

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

$$V^*(x) \le \lim_{\varepsilon\to 0} \{V^{\pi_\varepsilon}(x) + \varepsilon\} \le \lim_{\varepsilon\to 0} \left\{ \sup_{\pi\in\Pi} V^\pi(x) + \varepsilon \right\} = \sup_{\pi\in\Pi} V^\pi(x) \qquad \forall x \in \mathcal{X}. \tag{2.39}$$

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 $x \in \mathcal{X}$,[10]

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

[10] If this is not clear, see the discussion around (2.19). This is (2.21).

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

$$T^\pi V^* \leq V^*. \tag{2.40}$$

We use the monotonicity of T^π (Lemma 2.3) to conclude that

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

which by (2.40) 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 $k \to \infty$, Theorem 2.5 shows that $(T^\pi)^k V^*$ converges to V^π (the choice of V^* is indeed 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

$$\sup_{\pi \in \Pi} V^\pi \leq V^*. \tag{2.41}$$

The inequalities (2.39) and (2.41) together show that

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

which is the desired result. □

Exercise 2.27 ($\star\star\star$). *Recall the Bellman operator T^π_{mult} of Exercise 2.12, defined for a specific policy π, and extend it to its optimality version $T^*_{mult} : \mathcal{B}(\mathcal{X}) \to \mathcal{B}(\mathcal{X})$, defined as*

$$(T^*_{mult} V)(x, a) \triangleq \max_a \{ r(x, a) \int \mathcal{P}(\mathrm{d}x' | x, a) V(x') \},$$

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

(a) Specify the conditions that makes this operator monotonic and contractive.

(b) Is the fixed point V^* of T^*_{mult} the optimal value function in the space of stationary policies, that is,

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

2.5.3.1 Optimality Over the Space of Non-stationary Policies (†)

Consider 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. Following $\bar{\pi}$ means that at the first time step, the agent chooses action from π_1, and at the second time step, the agent chooses action from π_2, and so on. Is it possible that following such a non-stationary policy leads to higher expected return compared to following any single stationary Markov policy π, including the optimal one $\pi^* \leftarrow \operatorname{argmax}_{\pi \in \Pi} V^\pi$ (1.19) within the space of stationary Markov policies Π? In this section, we show that for discounted MDPs, it suffices to search in the space of stationary policies, as there is no non-stationary policy with a higher expected return.

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, so Π is a (proper) subset of $\bar{\Pi}$.

The expected return of following $(\pi_1, \pi_2, \dots, \pi_k)$ and terminating at time k with the terminal reward of V_0, which we denote by $V^{\pi_1:\pi_k}$ (cf. Remark 1.2), is

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

as argued in Section 2.3.1. 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 the terminal reward V_0 is discounted by $\lim_{k\to\infty} \gamma^k = 0$.

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

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

The following result shows that the optimal value function within the space of stationary policies, V^*, is the same as the optimal value function within the space of non-stationary policies, V^+.[c]

Proposition 2.10. *We have* $V^* = V^+$.

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

$$V^* = \sup_{\pi \in \Pi} V^\pi \leq \sup_{\bar{\pi} \in \bar{\Pi}} V^{\bar{\pi}} = V^+, \tag{2.42}$$

where the equality of V^* and $\sup_{\pi \in \Pi} V^\pi$ is because of Proposition 2.9.

Now consider an arbitrary sequence $(\pi_1, \ldots, \pi_k)$, and any value function V_0. By the optimality property of T^*, cf. (2.21), we have that

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

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

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

Together, they show that

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

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 Theorem 2.5 (thanks to the contraction property of the Bellman operator), 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

$$\sup_{\bar{\pi} \in \bar{\Pi}} V^{\bar{\pi}} \leq V^*. \tag{2.43}$$

By (2.42) and (2.43) we get the desired result. □

Result	Formula
Value Function (2.12)	$V^\pi = \frac{1}{1-\gamma}\rho_\gamma^\pi r^\pi$
Value Error (2.35)	$V - V^\pi = \frac{1}{1-\gamma}\rho_\gamma^\pi \mathrm{BR}^\pi(V).$
Performance Difference (3.3)	$V^{\pi'} - V^\pi = \frac{1}{1-\gamma}\rho_\gamma^{\pi'} A_\pi^{\pi'}$
Policy Gradient (6.29)	$\nabla_\theta V^{\pi_\theta}(x) = \frac{1}{1-\gamma}\mathbb{E}_{X\sim\rho_\gamma^{\pi_\theta}(x), A\sim\pi_\theta(\cdot\|X)}\left[\nabla_\theta \log \pi_\theta(A\|X) Q^{\pi_\theta}(X,A)\right]$

Table 2.2: Relation between the discounted future-state distribution and some value function-related quantities. Bellman Residual: $\mathrm{BR}^\pi(V) = T^\pi V - V$. Advantage of π' w.r.t. π: $A_\pi^{\pi'}(x) = A^\pi(x, \pi'(x)) = Q^\pi(x, \pi'(x)) - V^\pi(x)$.

2.6 Discounted Future-State Distribution and Value-Related Quantities

The discounted future-state distribution ρ_γ^π naturally appears in several results related to the value function. So far, we have encountered its use in representing the value function of a policy π as the product of ρ_γ^π and the reward function of the policy r^π (2.12) in Section 2.1.1.1, and how the error between V and V^π is related to the Bellman error $BR^\pi(V)$ in Section 2.5.2. We shall see some other results in the rest of this book. Table 2.2 summarizes those results.

Chapter Summary

Summarize the main points that the reader needs to remember from this chapter.

Notes and Remarks

a This robustness of the greedy (and optimal) policy to the changes in the value function is called *action-gap phenomenon* and is formally studied by Farahmand [2011a].

b We do not know if there has been many, or in fact any, studies to characterize the tightness/looseness of these error bounds.

c This result is from the discussion in Section 2.1 of Bertsekas [2018].

Chapter 3

Planning with a Known Model

Chapter Introduction

This chapter introduces the main Planning algorithms: Value Iteration, Policy Iteration, and Linear Programming. These algorithms assume that the MDP is known, and compute the (optimal) value function and the optimal policy. By the end of this chapter, at the very least, you need to *remember* the Value Iteration and Policy Iteration algorithms. You need to *understand* why the Value Iteration, Policy Iteration, and Linear Programming algorithms work. And you need to be able to apply the Value Iteration and Policy Iteration algorithms to solve an MDP.

Our goal is to design an RL agent act optimally (or close to optimally) in an environment.[1] So far we have defined the value function, policy, and studied some 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 both $\mathcal{R}$ and $\mathcal{P}$. We cal this the *Planning* setting, as you may recall from Section 1.4.

The assumption of knowing the MDP does not hold in the RL setting, in which the agent has only access to data coming from interaction with the environment, but not $\mathcal{R}$ and $\mathcal{P}$. Designing methods for finding the optimal policy given that knowledge, however, provides the foundation for developing methods for the RL setting. We shall see that many methods that can handle the RL setting, which we shall start

[1] Chapter's Version: 0.12 (2025 September 21). Partially revised up to the beginning of Section 3.4.2.

developing from Chapter 4, 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 value function. As we discussed in Section 2.2 and formally proved in Proposition 2.6, this is a sensible approach.

The methods in the second category directly search in the space of policies without explicitly constructing the optimal value function. The boundaries between these two categories are not clear cut, however, as some of the direct policy search methods implicitly use a quantity that has the same interpretation as the value function.

And then there are hybrid methods that use the explicitly constructed value function to guide an explicit search in the policy space.

In this chapter, our focus will be on the value-based methods. We talk about policy search ones in Chapter 6.

dynamic programming

Most of methods described in this chapter are considered as *dynamic programming (DP)* 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.[a]

3.1 Policy Evaluation vs. Control Problems

We make 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^*.[b]

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, compute V^π or Q^π.

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*

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

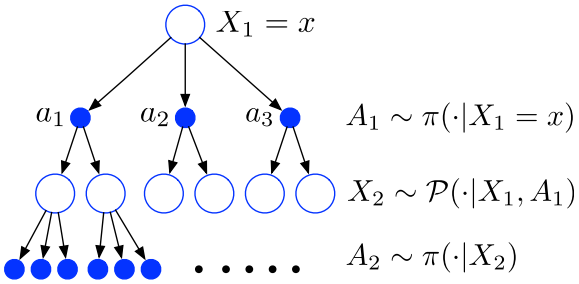

Figure 3.1: All future possibilities starting from state $X_1 = x$ up to depth 2.

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

Exercise 3.3 ($\star\star$). *Write a program that takes an MDP and compute* V_H^π *for various values of* $H = 1, 2, \ldots$ *by expanding the tree of all future possibilities. Consider the 2-state MDP of Exercise 1.1 as a concrete example.*

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 $(V(x_1), \ldots, V(x_n))$ and has 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 (Theorem 2.5), 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

$$V(x) - \gamma \sum_{x' \in \mathcal{X}} \mathcal{P}^\pi(x'|x) V(x') = r^\pi(x), \tag{3.1}$$

or more compactly in the matrix form as

$$(\mathbf{I} - \gamma \mathcal{P}^\pi) V = r^\pi, \tag{3.2}$$

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 $[\mathcal{P}^\pi]_{i,j} = \mathcal{P}^\pi(x_j|x_i)$ (1.2). 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 very large (say, a few thousands or so), we can easily solve this linear system of equations by computing the inverse of A and writing it as $x = A^{-1}b$, that is, we solve $V = (\mathbf{I} - \gamma\mathcal{P}^\pi)^{-1} r^\pi$.

When the state space is very large, computing the matrix inverse does not scale very well. We can use linear solvers, especially those 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 3.4 ($\star$). *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 3.5 ($\star$). *Write down the linear system of equations needed for finding Q^π. How many equations do we have?*

Exercise 3.6 ($\star\star$). *The inversion $(\mathbf{I}-\gamma\mathcal{P}^\pi)^{-1}$ needed in $V = (\mathbf{I}-\gamma\mathcal{P}^\pi)^{-1} r^\pi$ requires $(\mathbf{I} - \gamma\mathcal{P}^\pi)$ to be invertible. Prove that it is.*

Exercise 3.7 ($\star\star$). *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 3.8 ($\star\star$). *Write a program that takes an MDP and compute V^π by solving $(\mathbf{I} - \gamma\mathcal{P}^\pi)V = r^\pi$ through*

[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 an example of a linear solver itself.

- *inversion of* $(\mathbf{I} - \gamma \mathcal{P}^\pi)$ *and multiplying it to* r^π
- *using a standard linear system of equations solver (specify what solver you have used).*

As a concrete example, consider the 2-state MDP of Exercise 1.1. Also study how the size of the MDP affects the time to solve the equation. To get a meaningful clock time results, you need a much larger MDP.

Exercise 3.9 $(\star\star\star)$**.** *(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 Theorem 2.5, 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 operator. The VI can be used for both PE and Control problems. We start with its description for the PE problem.

Starting from $V_0 \in \mathcal{B}(\mathcal{X})$, a bounded function over the state space, we compute the sequence of functions $(V_k)_{k \geq 0}$ by

$$V_{k+1} \leftarrow T^\pi V_k. \tag{3.3}$$

Theorem 2.5 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.

Exercise 3.10 $(\star)$**.** *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 (3.3), with the difference that we apply the Bellman optimality operator instead, i.e.,

$$\begin{aligned} V_{k+1} &\leftarrow T^* V_k, \\ Q_{k+1} &\leftarrow T^* Q_k. \end{aligned} \tag{3.4}$$

Theorem 2.5 guarantees that $V_k \to V^*$ (or $Q_k \to Q^*$).

The value functions V_k or Q_k generated throughout VI procedure have corresponding greedy policies $\pi_g(\cdot; V_k)$ (2.17) or $\pi_g(\cdot; Q_k)$ (2.16). As these value functions converge to the optimal value functions, their greedy policies also converge to an optimal policy. This is because the greedy policy of the optimal value function is an optimal policy (Proposition 2.6). This suggests that we can use V_k to obtain a policy that is close to an optimal policy, and we can hope that as k grows, the greedy becomes closer to an optimal policy. We shall show this formally later.

We make a few remarks.

The update rule of the VI method (3.3)-(3.4) uses the current approximation of the value function, V_k, to compute a better approximation of the true value function (V^π for PE; V^* for Control). We say that VI bootstraps the current approximation to get a better one. The use of an existing approximation to get better ones, especially in this specific form where a Bellman operator or an approximation thereof is involved, is called *bootstrapping* in the RL literature. **bootstrapping**

We could also define the VI based on the m-step Bellman operator (2.22), or more generally, (2.23) with its special case (2.24). For example, we can have

$$V_{k+1} \leftarrow (T^\pi)^{(m)} / T^\pi_w / T^\pi_\lambda \; V_k.$$

Remark 3.1. *Sutton and Barto [2018] call the iteration (3.3) used for solving the PE problem "Iterative Policy Evaluation", and reserve Value Iteration for the procedure used to compute V^* or Q^* (3.4). 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 does 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 3.11 ($\star$)**.** *Write a program that accepts an MDP and performs Value Iteration.*

Remark 3.2. *If each step of VI is not performed exactly, we have an* approximate value iteration. *This means that instead of performing $V_{k+1} \leftarrow TV_k$, we find a V_{k+1} such that $V_{k+1} \approx TV_k$. This sequence V_k may not converge to the true value function (V^π or V^*) anymore. We discuss this further in Section 5.2.2.*

3.3.2 Value Iteration as an Operator Splitting (†)

In this section we show that the Value Iteration algorithm for PE can be interpreted as an instance of a matrix/operator splitting method. This is not only a new perspective on what VI is, but can also allow us to design more efficient variants of VI, which we do later in Chapter **??**.

Let us first describe the matrix splitting approach. Consider the linear system

$$Az = b,$$

with a nonsingular $A \in \mathbb{R}^{d\times d}$ and $z, b \in \mathbb{R}^d$. We split A to

$$A = M - N$$

for some choices of $M, N \in \mathbb{R}^{d\times d}$ (assume that M is nonsingular). As a simple example, consider $M = (1-\lambda)\mathbf{I}$ and $N = \mathbf{I} - \lambda A$ with $\lambda > 0$. We shall give some more concrete examples soon. This is called matrix (or operator) splitting.[3] The vector z satisfying $Az = b$ also satisfies

matrix/operator splitting

$$Mz = Nz + b. \tag{3.5}$$

To find a z that solves this equation, the *matrix splitting-based iterative* method, commonly used in numerical linear algebra, computes the new approximation z_{k+1} given the current z_k by solving

$$Mz_{k+1} = Nz_k + b,$$

or equivalently,

$$z_{k+1} = M^{-1}(Nz_k + b).$$

If the matrix (operator) M is $(1-\lambda)\mathbf{I}$, the inversion M^{-1} is very easy to compute. For other choices, it may or may not be. This iteration can converge to the solution z of the linear system of equations, with a rate that depends on M and N.

[3] We describe the idea using finite dimensional matrices and vectors, but more generally, A, M, and N can be linear operators and z a general infinite dimensional vector (a function). That is why we can call the method *operator* splitting too.

How does the choice of M and N affect the convergence rate? Let us analyze its convergence behaviour. Denote the error by $e_k \triangleq z_k - z$. As the solution of (3.5) satisfies

$$z = M^{-1}(Nz + b),$$

we have

$$e_{k+1} = z_{k+1} - z = M^{-1}(Nz_k + b) - M^{-1}(Nz + b) = M^{-1}N(z_k - z) = M^{-1}Ne_k,$$

so the dynamics of the error is

$$e_k = M^{-1}Ne_{k-1} = (M^{-1}N)^2 e_{k-2} = \cdots = (M^{-1}N)^k e_0. \tag{3.6}$$

Let $G \triangleq M^{-1}N$. The norm of the sequence $(e_k)_{k\geq 1}$ can be upper bounded as

$$\|e_k\| = \left\|G^k e_0\right\| \leq \left\|G^k\right\| \|e_0\| \leq \|G\|^k \|e_0\|. \tag{3.7}$$

The errors are (norm-) convergent if

$$\|G\| = \|M^{-1}N\| < 1,$$

for some choice of norm. More generally, the necessary and sufficient condition for convergence is that the spectral radius $\rho(G)$, the maximum absolute value of eigenvalues of G, is smaller than 1, see e.g., Theorem 4.1 of Saad [2003] or Theorem 11.2.1 of Golub and Van Loan [2013].[4] The convergence is faster if the spectral radius (or norm) is closer to zero.

The success of this iterative procedure as a way to solve the linear system of equation depends on two factors

- How we choose M and N such that the norm (or spectral radius) is as small as possible.
- How to choose M such that solving the equation $Mz_{k+1} = Nz_k + b$ is not very expensive.

The first factor affects how many iterations we need in order to get close to the solution. If the spectral radius is smaller, the number of iterations is smaller too. The second factor determines the computational cost of each iteration. Overall, they determine the total computational cost of solving the problem.

[4] For any matrix norm, we have $\rho(G) \leq \|G\|$, so the condition on the norm is sufficient, but not necessary. Our analysis will be norm-based.

Let us consider two cases in terms of the computation needed to compute M^{-1} and the spectral radius of $M^{-1}N$. The first one is when M is an identity matrix $\mathbf{I}$ and $N = \mathbf{I} - A$. In this case, the iteration becomes $z_{k+1} = (\mathbf{I} - A)z_k + b$. The computation of $M^{-1} = \mathbf{I}$ is clearly very cheap – no computation is needed. The convergence rate depends on the spectral radius of $M^{-1}N = N = (\mathbf{I} - A)$. So if $\rho(\mathbf{I} - A) < 1$, the iterative process is convergent. As another example on the other side of the tradeoff between these factors, consider $M = A$ and $N = 0$. One iteration of this procedure is enough to find the solution z. This can be seen by the spectral radius $\rho(M^{-1}N) = \rho(0) = 0$ too. The cost of computing the inverse of $M = A$, however, is expensive. Solving one iteration of this procedure cost essentially the same as solving the original problem. There are other choices between these two extremes.

For example, if we decompose A by its diagonal part D, its strictly lower triangular part $-L$, and its strictly upper triangular part $-U$ (so $A = D - L - U$), then the choice of $M = D$ and $N = L + U$ leads to what is known as the Jacobi iteration. Clearly, the computation of $M^{-1} = D^{-1}$ is easy. If we select $M = D - L$ and $N = U$ (or $M = D - U$ and $N = U$), we get the forward (or backward) Gauss-Seidel iteration. In all these cases, solving $Mz_k = Nz_{k-1} + b$ is easy. The convergence of these methods can be established too. For instance, if A is strictly diagonally dominated, the Jacobi iteration is convergent (Theorem 11.2.2 of Golub and Van Loan [2013]). These examples, as well as other choices available in the numerical linear algebra literature such as the Successive Over Relaxation method, show that there are multiple ways to split A to M and N, each with their own convergence properties.

After this review of matrix splitting, we are ready to make the connection between splitting-based iterative methods and VI for PE. By setting

$$A = \mathbf{I} - \gamma \mathcal{P}^\pi,$$

we see that equation $AV^\pi = r^\pi$ is indeed the Bellman equation for policy π (3.2). The VI for PE, which is

$$V_{k+1} = \gamma \mathcal{P}^\pi V_k + r^\pi = (\underbrace{\mathbf{I}}_{M})^{-1} \left[\underbrace{(\mathbf{I} - A)}_{N} V_k + r^\pi \right]$$

corresponds to the choice of $M = \mathbf{I}$ and $N = \gamma \mathcal{P}^\pi$. The convergence rate of this procedure is determined by

$$\rho(M^{-1}N) = \rho(\gamma \mathcal{P}^\pi) = \gamma < 1,$$

where we used the fact that the largest eigenvalue of a stochastic matrix/operator $\mathcal{P}^\pi$ is 1. Of course, this is the same VI convergence rate that we have obtained before.

This discussion shows that VI belongs to the family of splitting-based iterative methods. And it brings up the question of whether it is possible to split A differently so that the resulting VI-like procedure has better convergence properties. We give a particular answer to this question in Chapter **??**.[c]

3.3.3 Value Iteration as a Dynamical System (†)

3.4 Policy Iteration

A different approach to compute the optimal policy π^* 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 VI (PE) to compute the value of a policy π_k.

To perform the *policy improvement* step, we have to find a policy that is better than the current one. A 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 an improved 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

$$Q^{\pi_k}(x, \pi_g(x; Q^{\pi_k})) = \max_{a \in \mathcal{A}} Q^{\pi_k}(x, a) \geq V^{\pi_k}(x). \tag{3.8}$$

Algorithm 3.1 Policy Iteration

1: Initialize π_0 arbitrarily
2: $k \leftarrow 0$
3: **repeat**
4: $\quad Q^{\pi_k} \leftarrow$ solution of $Q = T^{\pi_k} Q$ ▷ Policy Evaluation: compute Q^{π_k}
5: $\quad \pi_{k+1} \leftarrow$ policy s.t. $T^{\pi_{k+1}} Q^{\pi_k} = T^* Q^{\pi_k}$ ▷ Policy Improvement
6: $\quad k \leftarrow k + 1$
7: **until** $\pi_k = \pi_{k-1}$

So this new policy is equal to or 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 the new policy π_{k+1} to be such that its value is higher than the previous policy's, that is, $V^{\pi_{k+1}} \geq V^{\pi_k}$. The *Policy Iteration* (PI) algorithm refers to the specific case that we pick the new policy π_{k+1} as the greedy policy $\pi_g(Q^{\pi_k})$.

Policy Iteration

As the value function of π_k is the unique fixed point of T^{π_k} (Theorem 2.5), and the greedy policy $\pi_g(Q^{\pi_k})$ satisfies $T^{\pi_{k+1}} Q^{\pi_k} = T^* Q^{\pi_k}$ (see (2.32)), we can summarize each iteration of the Policy Iteration algorithm as

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

Algorithm 3.1 summarizes the high-level steps of the PI algorithm. This algorithm has a loop over k. An important question is whether this loop ever stops, and if it stops, what the output policy would be. We get to these questions next.

Remark 3.3. *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.*

Remark 3.4. *The policy improvement step does not need to be done through computing the greedy policy $\pi_{k+1} \leftarrow \pi_g(Q_k)$. There are other mechanisms that generate an improved policy, or a sequence of policies, and still converge to an improved and even optimal policy. We will visit a family of them in Chapter 6.*

Exercise 3.12 ($\star$). *Write down the Policy Iteration algorithm based on V^{π_k}.*

Exercise 3.13 ($\star$). *Write a program that accepts an MDP and performs Policy Iteration.*

3.4.1 Convergence of Policy Iteration

The PI algorithm converges to π^*, and in practice, the convergence is often fast. In this section, we establish its convergence and show that for finite MDPs, the algorithm stops in a finite time. Our proof, however, only shows the eventuality of convergence, and not how fast PI converges. That is the topic of the next section.

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 (3.8).

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 (3.8).

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 first show that $T^{\pi'}Q^\pi \geq Q^\pi$. To see this, first notice that $T^{\pi'}Q^\pi = T^*Q^\pi$, by the assumption. We also have $T^*Q^\pi \geq T^\pi Q^\pi$, as for any $(x, a) \in \mathcal{X} \times \mathcal{A}$, it holds that

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

This shows that $T^{\pi'}Q^\pi = T^*Q^\pi \geq T^\pi Q^\pi$. As $T^\pi Q^\pi = Q^\pi$, we conclude that $T^{\pi'}Q^\pi \geq Q^\pi$.

The second step is to use the just proved $T^{\pi'}Q^\pi \geq Q^\pi$ to show that $Q^{\pi'} \geq Q^\pi$. 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 2.3) to get

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

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

$$(T^{\pi'})^m Q^\pi \geq T^*Q^\pi \geq Q^\pi. \tag{3.9}$$

Take the limit of $m \to \infty$. Because of the contraction property of the Bellman operator $T^{\pi'}$ (Theorem 2.5), we get that

$$\lim_{m\to\infty} (T^{\pi'})^m Q^\pi = Q^{\pi'}. \tag{3.10}$$

By combining (3.9) and (3.10), we get that

$$Q^{\pi'} = \lim_{m\to\infty} (T^{\pi'})^m Q^\pi \geq T^* Q^\pi \geq Q^\pi, \tag{3.11}$$

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 show that the PI algorithm converges to an optimal policy. And if the size of the state-action space is finite ($|\mathcal{X} \times \mathcal{A}| < \infty$), 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 $V^{\pi_k} = V^*$.*
- $\lim_{k\to\infty} \|V^{\pi_k} - V^*\|_\infty = 0$.
- *If the set of policies is finite, the PI algorithm converges in a finite number of iterations.*

Proof. This theorem states several related results. We prove them consecutively.

The first claim is that $V^{\pi_{k+1}} \geq V^{\pi_k}$, with equality if and only if $V^{\pi_k} = V^*$. By Theorem 3.1, we already know that $V^{\pi_{k+1}} \geq V^{\pi_k}$. So we need to prove that the equality holds ($V^{\pi_{k+1}} = V^{\pi_k}$) if and only if $V^{\pi_k} = V^*$. We consider each direction of the implication separately. First, we consider that the value of two policies π_k and π_{k+1} are the same: $V^{\pi_k} = V^{\pi_{k+1}}$. Apply $T^{\pi_{k+1}}$ on both sides to 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 the Bellman equation $T^{\pi_{k+1}} V^{\pi_{k+1}} = V^{\pi_{k+1}}$. 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 Theorem 2.5, so we must have that $V^{\pi_{k+1}} = V^*$.

Let us now prove the other direction: If $V^{\pi_k} = V^*$, then π_k is an optimal policy. The greedy policy of $V^{\pi_k} = V^*$, which is π_{k+1}, is still an optimal policy, hence $V^{\pi_{k+1}} = V^* = V^{\pi_k}$.

To prove the convergence $\lim_{k\to\infty} \|V^{\pi_k} - V^*\|_\infty = 0$, recall from (3.11) that we have

$$Q^{\pi_{k+1}} \geq T^* Q^{\pi_k} \geq Q^{\pi_k}. \tag{3.12}$$

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 policies 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.,

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

This entails that $\|Q^{\pi_{k+1}} - Q^*\|_\infty \leq \left\|(T^*)^k Q^{\pi_0} - Q^*\right\|_\infty$.[5] We show that the RHS converges to zero. By the contraction property of the Bellman optimality operator, we have that

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

Therefore, we get 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 (3.12) can be a strict inequality is going to be finite too. □

[5] This can be seen by noting that if for three real numbers a, b, c, we have $c \leq b \leq a$, we also have $|a - b| \leq |a - c|$.

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.

Exercise 3.14 ($\star$). *In the proof of Theorem 3.2, it is claimed that if $V^{\pi_k} = V^*$, its greedy policy π_{k+1} is still an optimal policy. Justify this!*

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 PI terminates in

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

iterations, which is a significantly stronger result compared to Theorem 3.2.[d]

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

$$A_\pi^{\pi'} = T^{\pi'}V^\pi - V^\pi. \tag{3.13}$$

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

$$A_\pi^{\pi'}(x) = Q^\pi(x,\pi'(x)) - V^\pi(x). \tag{3.14}$$

tage **func-**

Remark 3.5. *There is function called the* 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 a particular action at a state differs from the value of the state. We have that $A^\pi(x,\pi(x)) = 0$ (or $\mathbb{E}\left[A^\pi(x,A)\right] = 0$ with $A \sim \pi(\cdot|x)$). The notation of A^π is commonly used in the RL literature whereas $A_\pi^{\pi'}$ is not. 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 (Performance Difference Lemma)**.** *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

$$V^{\pi'} - V^\pi = (\mathbf{I} - \gamma\mathcal{P}^{\pi'})^{-1} r^{\pi'} - V^\pi. \tag{3.15}$$

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 (3.15), 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 $A_\pi^{\pi'}$ (3.13). This is the desired result. □

This result is sometimes known as the *performance difference lemma*. Let us explain it. Recalling the definition of the discounted future-state distribution (6.26), this lemma can be written as

$$V^{\pi'} - V^{\pi} = \frac{1}{1-\gamma} \rho_{\gamma}^{\pi'} A_{\pi}^{\pi'}.$$

It states that the difference between the value functions of policy π' and π is equal to the expectation of the advantage $A_{\pi}^{\pi'}$ w.r.t. the discounted future-state distribution induced by policy π'. Because $A_{\pi}^{\pi'}(x) = A^{\pi}(x, \pi'(x)) = Q^{\pi}(x, \pi'(x)) - V^{\pi}(x)$, this is the expected value of the difference between "the value of choosing action according to π' for one step and then following π afterwards" and "the value of following π from the first step onwards", with the expectation being taken w.r.t. the discounted future-state distribution of a π'-following agent. As a sanity check, if both policies have the same value, the advantage is zero, and both LHS and RHS are zero.

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*

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

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

$$\begin{aligned} V^{\pi^*} - V^{\pi'} &= T^{\pi^*} V^{\pi^*} - \left(T^{\pi^*} V^{\pi} - T^{\pi^*} V^{\pi}\right) - \left(T^{\pi'} V^{\pi} - T^{\pi'} V^{\pi}\right) - T^{\pi'} V^{\pi'} \qquad (3.16) \\ &= \left(T^{\pi^*} V^{\pi^*} - T^{\pi^*} V^{\pi}\right) + \left(T^{\pi^*} V^{\pi} - T^{\pi'} V^{\pi}\right) + \left(T^{\pi'} V^{\pi} - T^{\pi'} V^{\pi'}\right). \qquad (3.17) \end{aligned}$$

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

$$\begin{aligned} T^{\pi^*} V^{\pi^*} - T^{\pi^*} V^{\pi} &= \left(r^{\pi^*} + \gamma \mathcal{P}^{\pi^*} V^{\pi^*}\right) - \left(r^{\pi^*} + \gamma \mathcal{P}^{\pi^*} V^{\pi}\right) \\ &= \gamma \mathcal{P}^{\pi^*} \left(V^{\pi^*} - V^{\pi}\right). \qquad (3.18) \end{aligned}$$

Likewise, for the last term, we get

$$T^{\pi'} V^{\pi} - T^{\pi'} V^{\pi'} = \gamma \mathcal{P}^{\pi'} \left(V^{\pi} - V^{\pi'}\right). \qquad (3.19)$$

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

$$T^{\pi'}V^\pi = T^*V^\pi. \tag{3.20}$$

Recall that the Bellman optimality operator is the supremum of the Bellman operators over all policies (see (2.20)), 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^π with $T^{\pi'}V^\pi$ (3.20), we get that

$$T^{\pi'}V^\pi - T^{\pi^*}V^\pi = T^*V^\pi - T^{\pi^*}V^\pi \geq 0. \tag{3.21}$$

Plugging (3.18), (3.19), and (3.21) into (3.16), 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

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

Here in the second inequality, we used the property that the vector norm of an operator (or matrix) transformed vector (i.e., $\|Lz\|$, for an operator L and vector z) is upper bounded by the operator norm $\|L\|$ multiplied by the vector norm $\|z\|$ (see (A.7) in Appendix A.3.1; or (A.11) in Appendix A.4) and the fact that the supremum norm of a stochastic operator (matrix) is 1 (A.10). □

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

$$\left\|V^{\pi^*} - V^{\pi_k}\right\|_\infty \leq \gamma^k \left\|V^{\pi^*} - V^{\pi_0}\right\|_\infty, \tag{3.22}$$

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 $V^{\pi_k} \geq V^{\pi^*} - \varepsilon$, i.e., following it is at most ε worse than following the optimal policy.

Proposition 3.5. *Assume that for all policies π, $\|V^\pi\|_\infty \leq V_{max}$. 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 $V^{\pi_k} \geq V^{\pi^*} - \varepsilon$.

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

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

where we evoked Lemma 3.4 (as we did in (3.22)), and benefitted from the fact that all the value functions are $V_{\max}$-bounded. This is satisfied if

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

Solving for k, we get that $\log(\frac{\varepsilon}{2V_{\max}}) \leq k \log \gamma$, 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*

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

iterations.

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

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

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,

$$-A_{\pi^*}^{\pi_k} \leq (\mathbf{I} - \gamma \mathcal{P}^{\pi_k})(V^{\pi^*} - V^{\pi_k}) \leq V^{\pi^*} - V^{\pi_k}. \tag{3.23}$$

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

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

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^*.[6]

Because of the non-negativity of $-A_{\pi^*}^{\pi_k}$, we can take the norm of both sides of (3.23) and keep the same direction of inequality:

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

By Lemma 3.4 (specifically it consequence (3.22)), Lemma 3.3, and Lemma A.2 in Appendix A.4, we have that

$$\begin{aligned} \|-A_{\pi^*}^{\pi_k}\|_\infty &\leq \gamma^k \left\|V^{\pi^*} - V^{\pi_0}\right\|_\infty \\ &= \gamma^k \left\|(\mathbf{I} - \gamma \mathcal{P}^{\pi_0})^{-1}(-A_{\pi^*}^{\pi_0})\right\|_\infty \\ &\leq \frac{\gamma^k}{1-\gamma} \left\|(-A_{\pi^*}^{\pi_0})\right\|_\infty. \end{aligned} \tag{3.24}$$

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

[6] 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 $T^{\pi_k} V^{\pi^*} < (T^{\pi_k})^2 V^{\pi^*}$, which entails that $V^{\pi^*} < (T^{\pi_k})^2 V^{\pi^*}$. Repeating this argument leads to the statement that $V^{\pi^*} < (T^{\pi_k})^m V^{\pi^*}$ 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 $V^{\pi^*} < \lim_{m \to \infty} (T^{\pi_k})^m V^{\pi^*} = V^{\pi_k}$. This is a contradiction, as V^{π^*} is the optimal value function and cannot be smaller than V^{π_k}.

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 (3.24) 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

$$(-A^{\pi_k}_{\pi^*})(x_0) \leq \frac{\gamma^k}{1-\gamma}(-A^{\pi_0}_{\pi^*})(x_0). \tag{3.25}$$

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 (3.14), 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 (3.25) 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 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

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

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:

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

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

Exercise 3.16. *How can we have an LP-like formulation when the state space is continuous?*

Exercise 3.17. *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 Summary

Summarize the main points that the reader needs to remember from this chapter.

Notes and Remarks

a What exactly refers to the dynamic programming is not very clear. Bertsekas and Tsitsiklis [1996] define it as "the mathematical formalization of the tradeoff

between the immediate and the future cost" (p2), and mention that "the objective of DP is to calculate numerically the optimal cost-to-go function J^*" (p3), which is essentially the same as V^* and Q^* in this book. Bertsekas [2018] considers that "DP is the principal method for analysis of a large and diverse class of sequential decision problems." (p2).

Sutton and Barto [2018] states that "The term dynamic programming (DP) refers to a collection of algorithms that can be used to compute optimal policies given a perfect model of the environment as a Markov decision process (MDP)." (Chapter 4, p73). Therefore, their emphasis is on whether the perfect model of the environment is available or not.

Szepesvári [2010] does not define what DP is, but introduces Value Iteration and Policy Iteration as DP algorithms.

Richard Bellman, who coined the term DP, began the preface of his book on DP [Bellman, 1957] by saying that "The purpose of this work is to provide an introduction to the mathematical theory of multi-stage decision processes. Since these constitute a somewhat formidable set of terms we have coined the term "dynamic programming" to describe the subject matter." Since the initial intention was to have an umbrella term for many concepts and methods, it is not surprising that coming up with a clear and precise definition is not easy.

Interestingly, Bellman mentions in his autobiography [Bellman, 1984] that since an influential administrator hated the term research, and presumably mathematics, he tried to shield his work from them by using a new term. He was initially "interested in planning, in decision-making, in thinking", but he decided on using "programming", and to get across that "this was dynamic, this was multi-stage, this was time-varying", he chose the adjective "dynamic". This is why he came up with the term dynamic programming. If not for that administrator, we could perhaps have multi-stage planning or even dynamic thinking.

In this book, we use DP to refer to methods that benefit from the special recursive structure of the sequential decision-making problem, specifically, the MDP, to solve it. We reserve it for the methods that assume the knowledge of the problem, specifically $\mathcal{P}$ and $\mathcal{R}$. One can argue that the intention behind the definition was broad enough to encompass many RL methods as well. We make a distinction between DP and RL, however, as this is commonly accepted in the community. To be specific and concrete, we consider Value Iteration (Section 3.3) and Policy Iteration (Section 3.4), and their non-sample-based variants, as the quintessential DP algorithms. We consider methods such as Temporal Difference (Sections 1.5, 4.4, 4.6) or Approximate Value Iteration (Sections 5.2.2 and 5.3.2) as approximate DP methods that are better be thought of as RL methods, with the understanding of their close

relations. Or perhaps we could consider all of them as RL methods, as they are methods for solving RL problems after all (see the beginning of Chapter 1), albeit with different requirement for the information available to them (full knowledge of the MDP or only samples coming from the environment).

b Referring the problem of finding the optimal value function or policy as the problem of "control" can be confusing, especially for a reader with a background in control theory/engineering. Here we use Policy Evaluation vs. Control to emphasize whether we are interested in computing V^π or Q^π vs. V^* or Q^* or π^*, or approximation thereof.

c The idea of using matrix splitting to analyze or design variants of Value Iteration have been explored in the literature.

Kushner and Kleinman [1971] is one of the earliest papers that mentions the Jacobi and Gauss-Seidel procedures for computing the value function. Porteus [1975] propose several transformations to the reward and the probability transition matrix with the goal of improving the computational cost of solving the transformed MDP. One of the transformations, called *pre-inverse transform*, has some similarities with the matrix splitting.

Bacon and Precup [2016] and Chapter 4 of Bacon [2018] provide a matrix splitting perspective on solving the PE problem given a set of options, which are temporally-extended actions. They show that the computation of the value function using a given set of options can be interpreted as a particular choice of matrix splitting.

Discussion of this section follows from our own work [Rakhsha et al., 2022], where we introduce a new operator splitting approach that leads to accelerated computation of the value function, for both PE and Control problems. We get back to the accelerated algorithm when we discuss model-based RL algorithms in Chapter **??**. The goal of acceleration using matrix/operator splitting was also pursued by **?**, who propose using matrix splitting, alongside another technique called matrix deflation, to effectively remove (deflate) the top dominant eigen-structure of the transition matrix $\mathcal{P}$. This leads to an accelerated algorithm called *Deflated Dynamics Value Iteration*.

d This result is relatively new, compared to when the PI algorithm was introduced [Howard, 1960]. Variants of it have been proven by Ye [2011]; Hansen et al. [2013]; Scherrer [2016]. We closely follow the results and proofs of Scherrer [2016].

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 book.

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. $Z_1, \ldots, Z_t$, 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 $Z_1, \ldots, Z_t$. 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 $0 < \alpha < 1$, we get that

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

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 $Z_1, \ldots, Z_{t-1}$ and θ_t is a function of $Z_1, \ldots, Z_{t-1}$ 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 $t = 1, 2, \ldots$, 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 (1.5)

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

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.,

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

The SA conditions (4.3)-(4.4) apply here, with the difference that it should be for each state-action pairs, 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

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

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

$$r(x_1, a_1) = 1,$$
$$r(x_1, a_2) = 2.$$

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

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

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 (1.21), 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 $\text{Unif}(\mathcal{A})$ 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 Tradeoff

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 $\tau \to 0$, the behaviour of this distribution would be the same as the greedy policy, i.e., both choose the maximizing action. On the other hand, when $\tau \to \infty$, 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 $G_t^\pi \triangleq \sum_{k \geq t} \gamma^{k-t} R_k$ (1.16). 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 sampled 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 k of the episode t **do**
5: Follow π 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 $X_1^{(t)}, A_1^{(t)}, R_1^{(t)}, X_2^{(t)}, A_2^{(t)}, R_2^{(t)}, \ldots, X_m^{(t)}, A_m^{(t)}, R_m^{(t)}, X_m^{(t)}, \ldots$, we can compute not only the return $G_1^{\pi\,(t)}$, but also all other returns $G_m^{\pi\,(t)}$ from time step $m = 1, 2, \ldots$ (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 $\hat{V}^\pi(X_m^{(t)})$ $(m = 1, 2, \ldots)$.

We have to be careful here though. If a trajectory visits a particular states twice (or more) at time steps $m_1, m_2, \ldots$ (all within the same episode), the returns $G_{m_1}^{\pi\,(t)}, G_{m_2}^{\pi\,(t)}, \ldots$ 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

$$G_1^\pi = R_1 + \gamma R_2 + \gamma^2 R_3 + \cdots,$$
$$G_2^\pi = R_2 + \gamma R_3 + \cdots.$$

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 changes 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 updating the estimation of $\hat{V}^\pi$. That means that if we get to a state x at $m_1(x), m_2(x), \ldots$, 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.

4.4 Temporal Difference Learning for Policy Evaluation

MC methods allow 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 π, step size schedule $(\alpha_k)_{k\geq 1}$.

Initialize $V_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, e.g., $V_1(x) = 0$.
for iteration $k = 1, 2, \ldots$ **do**
  for each state $x \in \mathcal{X}$ **do**
    Let $A \sim \pi(\cdot|x)$
    $X'(x) \sim \mathcal{P}(\cdot|X, A)$ and $R(x) \sim \mathcal{R}(\cdot|x, A)$
    Let $(\hat{T}^\pi V_k)(x) \triangleq R(x) + \gamma V_k(X'(x))$
  end for
  Update

$$V_{k+1} \leftarrow (1 - \alpha_k)V_k + \alpha_k \hat{T}^\pi V_k$$

end for

converges to $(T^\pi V_k)(x)$ by LLN. And as we have seen, the rate of convergence is $O_P(\frac{1}{\sqrt{n}})$. 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 π, 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 $t = 1, 2, \ldots$ **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

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

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

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

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 $V^\pi(x) = \mathbb{E}[G^\pi | X = x]$, we have a noisy (but unbiased) estimate of V^π. SA allows us to converge to its mean.
- In VI, we update $V_{k+1}(x) \leftarrow (T^\pi V_k)(x) = \mathbb{E}[R + \gamma V_k(X') | X = x]$. 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 $\hat{Q}^\pi_t$ that converges to Q^π. If we wait for an infinite time, $\hat{Q}^\pi_\infty = \lim_{t\to\infty} \hat{Q}^\pi_t = Q^\pi$. We can then choose the new policy $\pi' \leftarrow \pi_g(\hat{Q}^\pi_\infty)$. 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 π_1, step size schedule $(\alpha_k)_{k\geq 1}$.

Initialize $Q_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, e.g., $Q_1 = 0$.
for each iteration $k = 1, 2, \ldots$ **do**
  for all $(x, a) \in \mathcal{X} \times \mathcal{A}$ **do**
    Initialize $X_1 = x$ and $A_1 = a$.
    Generate an episode from X_1 by choosing A_1, and then following π_k to obtain $X_1, A_1, R_1, X_2, A_2, R_2, \ldots$.
    Compute $G_1^{\pi_k}(X_1, A_1) = \sum_{t\geq 1} \gamma^{t-1} R_t$.
    Update

$$\hat{Q}^{\pi}_{k+1}(X_1, A_1) \leftarrow (1 - \alpha_k(X_1, A_1))\, \hat{Q}^{\pi}_{k}(X_1, A_1) + \alpha_k(X_1, A_1) G_1^{\pi_k}(X_1, A_1)$$

  end for
  Improve policy: $\pi_{k+1} \leftarrow \pi_g(Q_{k+1})$.
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 π_1, step size schedule $(\alpha_k)_{k\geq 1}$, initial distribution $\rho \in \mathcal{M}(\mathcal{X}\times\mathcal{A})$.

Initialize $Q_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrary, e.g., $Q_1 = 0$.
Initialize $n_1 : \mathcal{X} \times \mathcal{A} \to \mathbb{N}$ with $n_1(x, a) = 0$.
for each iteration $k = 1, 2, \ldots$ **do**
  for all $(x, a) \in \mathcal{X} \times \mathcal{A}$ **do**
    Draw $(X_1, A_1) \sim \rho$
    Generate an episode from X_1 by choosing A_1, and then following π_k to obtain $X_1, A_1, R_1, X_2, A_2, R_2, \ldots$ until the end of episode.
    For all (X_t, A_t) visited within the episode, set $n_k(X_t, A_t) \leftarrow n_k(X_t, A_t)+1$.
    Compute $G_t^{\pi_k}(X_t, A_t) = \sum_{t'\geq t} \gamma^{t'-t} R_{t'}$.
    for all (X_t, A_t) visited in the episode **do**
      Let $\alpha_k(X_t, A_t) = \frac{1}{n_k(X_t,A_t)}$.
      Update

$$\hat{Q}^{\pi}_{k+1}(X_t, A_t) \leftarrow (1 - \alpha_k(X_t, A_t))\, \hat{Q}^{\pi}_k(X_t, A_t) + \alpha_k(X_t, A_t) G_t^{\pi_k}(X_t, A_t)$$

    end for
    $n_{k+1} \leftarrow n_k$.
  end for
  Improve policy: $\pi_{k+1} \leftarrow \pi_g(Q_{k+1})$.
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 not 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 (1.21). 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 $L\theta_t$ itself, but only to its noise contaminated counterpart $L\theta_t+\eta_t$ 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, where we only update the value function at the current state(-action), 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 $\mathbb{E}\left[\eta_t^2(X_t, A_t)|F_t\right]$:

$$\begin{aligned}
\mathbb{E}\left[\eta_t^2(X_t, A_t) \mid F_t\right] &= \mathbb{E}\Bigg[\Big|(R_t - r(X_t, A_t))+ \\
&\qquad \gamma\left(\max_{a' \in \mathcal{A}} Q_t(X'_t, a') - \int \mathcal{P}(\mathrm{d}x'|X_t, A_t) \max_{a' \in \mathcal{A}} Q_t(x', a')\right)\Big|^2 \mid F_t\Bigg] \\
&\leq 2\mathrm{Var}\left[R_t \mid X_t, A_t\right] + 2\gamma^2 \mathrm{Var}\left[\max_{a' \in \mathcal{A}} Q_t(X', a') \mid X_t, A_t\right].
\end{aligned}$$

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

$$\mathbb{E}\left[\eta_t^2(X_t, A_t) \mid F_t\right] \leq 2(\sigma_R^2 + \gamma^2 \left\|Q_t\right\|_2^2).$$

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 $\sum_{t\geq 1} \frac{1}{t} = \infty$. 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.

5.1 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,

$$\inf_{\hat{V} \in \mathcal{F}_\varepsilon} \left\| \hat{V} - V \right\|_\infty \leq L\varepsilon.$$

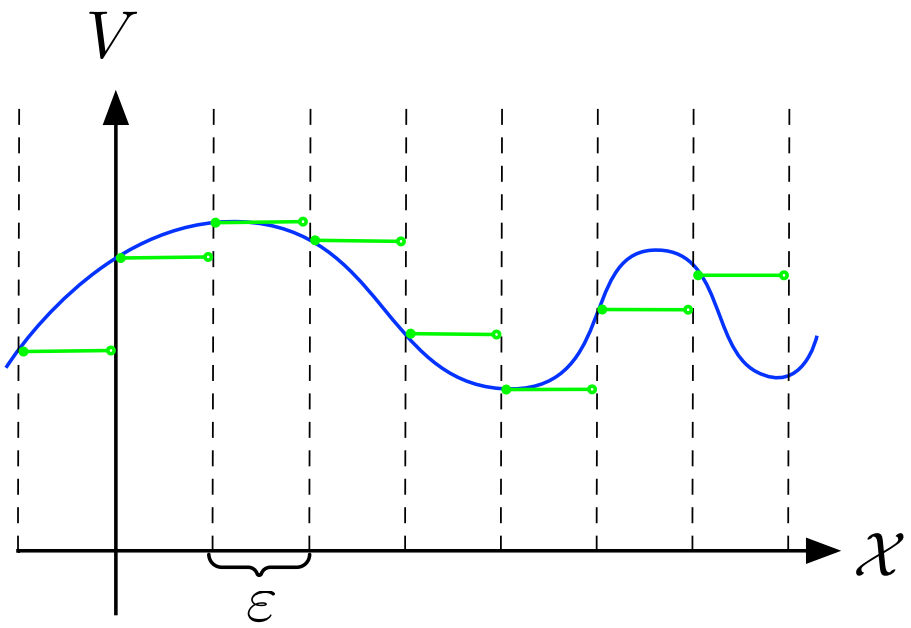

Figure 5.1: Approximating a function (blue) with a piecewise constant function (green) based on an ε-discretized state space

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 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]

[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

Other than the expressivity of the function space $\mathcal{F}$, we also need to pay attention 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.

5.2 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.

5.2.1 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

of it later.

function, we can express our goal as

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} d(V, V^\pi).$$

We can use the L_p-norm w.r.t. a (probability) measure $\mu \in \mathcal{M}(\mathcal{X})$ (see (A.1) in Appendix A.2) to define distances between functions by letting $d(V_1, V_2) = \|V_1 - V_2\|_{p,\mu}$. We 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 the 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*.

5.2.2 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 functions, 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.

5.2.3 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 changes. 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 $V \in \mathcal{F}$.

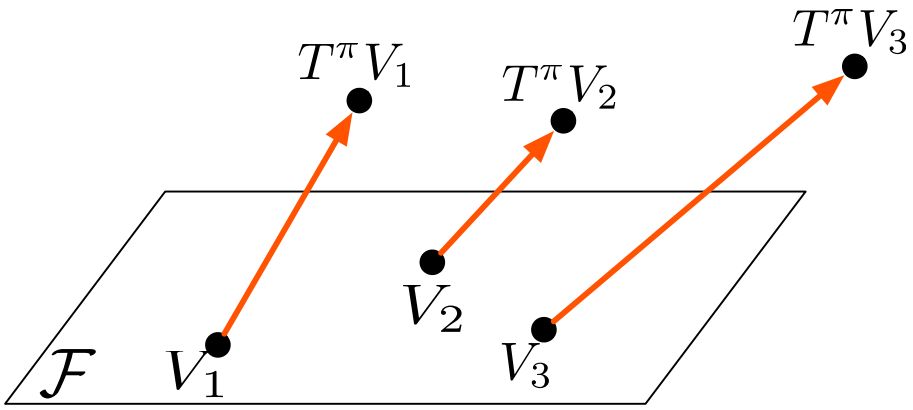

Figure 5.2: The Bellman operator may take value functions from $\mathcal{F}$ to a point outside that space. BRM finds $V \in \mathcal{F}$ with the minimal distance.

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.

Recall from Proposition 2.8 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 2.8, 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.

[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.

5.2.3.1 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 $X_1 \sim \rho \in \mathcal{M}(\mathcal{X})$. 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

$$\mathbb{P}\{X_2 \in B\} = \int \rho(\mathrm{d}x)\mathcal{P}^\pi(B|x),$$

or for countable state space, the probability of being in state y is

$$\mathbb{P}\{X_2 = y\} = \sum_{x\in\mathcal{X}} \rho(x)\mathcal{P}^\pi(y|x).$$

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 $\mathbb{P}\{X_1 = y\} = \mathbb{P}\{X_2 = y\}$, 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. □

5.2.3.2 Stationary Distribution Weighted Error Bound for BEM

We are ready to prove the following error bound. This is similar to Proposition 2.8 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

$$\int |V(x) - V^\pi(x)| \, \mathrm{d}\rho^\pi(x) \le \int |V(x) - (T^\pi V)(x)| \mathrm{d}\rho^\pi(x) + \gamma \int \mathrm{d}\rho^\pi(x) \left| \int \mathcal{P}^\pi(\mathrm{d}y|x)(V(y) - V^\pi(y)) \right|.$$

By Jensen's inequality, we have

$$\int |V(x) - V^\pi(x)| \, \mathrm{d}\rho^\pi(x) \le \int |V(x) - (T^\pi V)(x)| \mathrm{d}\rho^\pi(x) + \gamma \int \mathrm{d}\rho^\pi(x) \mathcal{P}^\pi(\mathrm{d}y|x) \, |V(y) - V^\pi(y)| \, .$$

Because ρ^π is the stationary distribution, by (5.7) the second integral in the RHS can be simplified as

$$\int \mathrm{d}\rho^\pi(x) \mathcal{P}^\pi(\mathrm{d}y|x) \, |V(y) - V^\pi(y)| = \int \mathrm{d}\rho^\pi(x) \, |V(x) - V^\pi(x)| \, .$$

So

$$\|V - V^\pi\|_{1,\rho^\pi} \le \|V - T^\pi V\|_{1,\rho^\pi} + \gamma \, \|V - V^\pi\|_{1,\rho^\pi} \, .$$

After re-arranging, we get the result for $p = 1$. By Jensen's inequality, we have that $\|V - T^\pi V\|_{1,\rho^\pi} \le \|V - T^\pi V\|_{p,\rho^\pi}$, for any $p \ge 1$. □

5.2.4 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:

$$\begin{aligned} V &\leftarrow \operatorname*{argmin}_{V' \in \mathcal{F}} \left\| V' - \tilde{V}(V') \right\|_{2,\mu}^2 , \\ \tilde{V}(V') &\leftarrow \operatorname*{argmin}_{V'' \in \mathcal{F}} \|V'' - T^\pi V'\|_{2,\mu}^2 . \end{aligned} \tag{5.13}$$

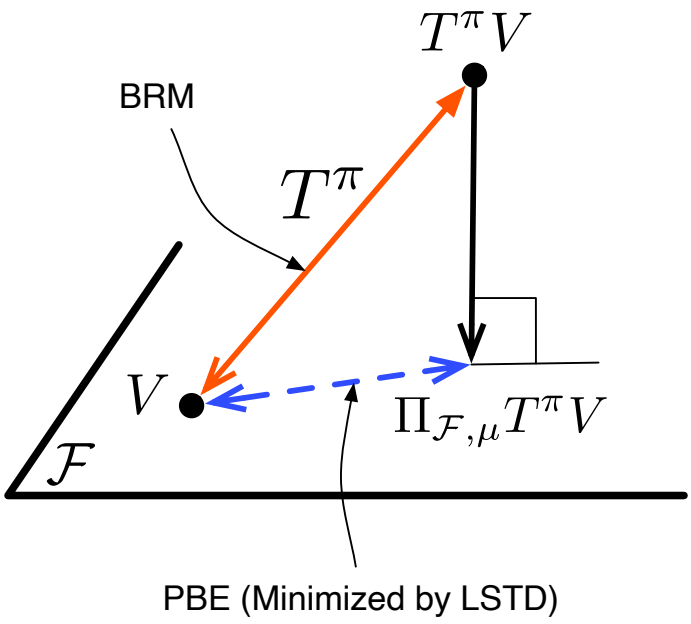

Figure 5.3: The Bellman operator may take value functions from $\mathcal{F}$ to a point outside that space. BRM finds $V \in \mathcal{F}$ with the minimal distance.

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 Sections 5.3 and 5.4.

5.2.4.1 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:

$$x^* \leftarrow \underset{x \in \mathbb{R}^d}{\operatorname{argmin}} \|Ax - b\|_2^2 = (Ax - b)^\top (Ax - b). \tag{5.15}$$

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 $V \approx \Pi_{\mathcal{F}} T^\pi V$ (5.10), 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?*

5.2.4.2 Least Squares 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) $\phi_1, \ldots, \phi_p$. 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:

$$\Phi = \begin{bmatrix} \phi_1 & \cdots & \phi_p \end{bmatrix}.$$

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 $M = \mathrm{diag}(\mu)$. 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

$$\Pi_{\mathcal{F},\mu} V = \operatorname*{argmin}_{V' \in \mathcal{F}} \|V' - V\|_{2,\mu}^2 .$$

We can identifying the projected function $V = \Phi w^+$ with the weight w^+ minimizing the following:

$$\operatorname*{argmin}_{w \in \mathbb{R}^p} \|\Phi w - V\|_{2,\mu}^2 = \operatorname*{argmin}_{w \in \mathbb{R}^p} (\Phi w - V)^\top M (\Phi w - V).$$

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 $Ax^* = b$.*

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

$$Ax^* = (\Phi^\top M \Phi)^{-1} \Phi^\top M \Phi b = b.$$

□

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 weight vector, and its corresponding value function, is the solution of the fixed-point equation (5.20). It shall be used to define the *Least Squares 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

$$\langle \, \phi_i \, , \, T^\pi V(w) - V(w) \, \rangle_\mu = 0, \qquad \forall i = 1, \ldots, p.$$

As $\mathrm{BR}(V(w)) = T^\pi V(w) - V(w)$, this is also equivalent to

$$\langle \, \phi_i \, , \, \mathrm{BR}(V(w)) \, \rangle_\mu = 0, \qquad \forall i = 1, \ldots, p.$$

So we are aiming to find a w such that the Bellman Residual $\mathrm{BR}(V(w))$ is orthogonal to the basis of $\mathcal{F}$, when the orthogonality is measured according to the distribution μ.

5.2.4.3 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

$$L : V \mapsto [(1-\alpha)\mathbf{I} + \alpha\Pi_{\mathcal{F},\rho^\pi}T^\pi]\, V.$$

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

$$\|LV_1 - LV_2\|_{2,\rho^\pi} \leq (1-\alpha)\,\|V_1 - V_2\|_{2,\rho^\pi} + \alpha\,\|\Pi_{\mathcal{F},\rho^\pi}T^\pi(V_1 - V_2)\|_{2,\rho^\pi}\,. \tag{5.27}$$

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):

$$\begin{aligned}\|\Pi_{\mathcal{F},\rho^\pi}T^\pi(V_1 - V_2)\|_{2,\rho^\pi} &\leq \|T^\pi(V_1 - V_2)\|_{2,\rho^\pi} \\ &\leq \gamma\,\|V_1 - V_2\|_{2,\rho^\pi}\,.\end{aligned} \tag{5.28}$$

This along with (5.27) shows that

$$\|LV_1 - LV_2\|_{2,\rho^\pi} \leq [(1-\alpha) + \alpha\gamma]\,\|V_1 - V_2\|_{2,\rho^\pi}\,.$$

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. $\|\cdot\|_{2,\rho^\pi}$, 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 $\alpha Lw = 0$. 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):

$$\begin{aligned} w_{k+1} \leftarrow & w_k + \alpha \Phi^\top D^\pi \left[r^\pi + \gamma \mathcal{P}^\pi \Phi w_k - \Phi w_k\right] \\ = & (\mathbf{I} - \alpha A) w_k + \alpha \Phi^\top D^\pi r^\pi, \end{aligned} \tag{5.31}$$

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.

5.2.4.4 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 $\text{diag}(\rho^\pi)$, 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]

$$\begin{aligned}\sum_{i,j} z_i \rho_i^\pi \mathcal{P}^\pi_{ij} z_j &= \sum_{i,j} z_i \sqrt{\rho_i^\pi} \sqrt{\rho_i^\pi} \mathcal{P}^\pi_{ij} z_j = \sum_i \underbrace{z_i \sqrt{\rho_i^\pi}}_{\triangleq a_i} \underbrace{\sqrt{\rho_i^\pi} \sum_j \mathcal{P}^\pi_{ij} z_j}_{\triangleq b_j} \\ &\leq \sqrt{\sum_i z_i^2 \rho_i^\pi} \sqrt{\sum_i \rho_i^\pi \left(\sum_j \mathcal{P}^\pi_{ij} z_j \right)^2}. \end{aligned} \tag{5.36}$$

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

$$\sum_i \rho_i^\pi \left(\sum_j \mathcal{P}^\pi_{ij} z_j \right)^2 \leq \sum_i \rho_i^\pi \sum_j \mathcal{P}^\pi_{ij} z_j^2 = \sum_j z_j^2 \sum_i \rho_i^\pi \mathcal{P}^\pi_{ij} = \sum_j z_j^2 \rho_j^\pi = \|z\|^2_{2,\rho^\pi}.$$

[5] For summation, the Cauchy-Schwarz inequality states that $\sum_i a_i b_j \leq \sqrt{\sum_i a_i^2} \sqrt{\sum_i b_i^2}$. See Appendix A.8.

This, along with (5.36), allows us to upper bound (5.35) by

$$z^\top D^\pi \mathcal{P}^\pi z \leq \|z\|^2_{2,\rho^\pi}$$

Plugging this inequality and (5.34) in (5.33), we get that for any $z \neq 0$ and $\rho^\pi > 0$,

$$z^\top D^\pi (\mathbf{I} - \gamma \mathcal{P}^\pi) z \geq (1-\gamma) \|z\|^2_{2,\rho^\pi} > 0.$$

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 $\lambda_i(A) = a_i + jb_i$ with $a_i > 0$. We need to have $|1 - \alpha\lambda_i(A)| < 1$ for all $i = 1, \ldots, p$. We expand

$$|1 - \alpha\lambda_i(A)|^2 < 1 \Leftrightarrow |1 - \alpha a_i|^2 + |\alpha b|^2 = 1 + (\alpha a_i)^2 - 2\alpha a_i + (\alpha b_i)^2 < 1.$$

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 $i = 1, \ldots, p$) by selecting it such that

$$\alpha < \alpha_0 = \min_{i=1,\ldots,p} \frac{2a_i}{a_i^2 + b_i^2} = \frac{2\mathbf{Re}[\lambda_i(A)]}{|\lambda_i(A)|^2}.$$

□

5.2.4.5 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?

[6] Refer to https://math.stackexchange.com/a/325412 for a proof.

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.

Proposition 5.5. *If ρ^π is the stationary distribution of π, we have*

$$\|V_{TD} - V^\pi\|_{2,\rho^\pi} \leq \frac{\|\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi\|_{2,\rho^\pi}}{\sqrt{1-\gamma^2}}.$$

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

$$\begin{aligned} \|V_{\text{TD}} - V^\pi\|^2_{2,\rho^\pi} = &\|\Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi\|^2_{2,\rho^\pi} + \|\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi\|^2_{2,\rho^\pi} + \\ &2 \left\langle \Pi_{\mathcal{F},\rho^\pi} V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi \,,\, \Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi \right\rangle_{\rho^\pi}. \end{aligned} \tag{5.37}$$

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 $\mathcal{F}$. Therefore, their inner product is zero.

We provide an upper bound on $\|\Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi\|^2_{2,\rho^\pi}$. 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

$$\begin{aligned} \|\Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} V^\pi\|^2_{2,\rho^\pi} &= \|\Pi_{\mathcal{F},\rho^\pi} T^\pi V_{\text{TD}} - \Pi_{\mathcal{F},\rho^\pi} T^\pi V^\pi\|^2_{2,\rho^\pi} \\ &\leq \|T^\pi V_{\text{TD}} - T^\pi V^\pi\|^2_{2,\rho^\pi} \\ &\leq \gamma^2 \|V_{\text{TD}} - V^\pi\|^2_{2,\rho^\pi}. \end{aligned}$$

Therefore, we can upper bound (5.37) as

$$\|V_{\text{TD}} - V^\pi\|^2_{2,\rho^\pi} \leq \gamma^2 \|V_{\text{TD}} - V^\pi\|^2_{2,\rho^\pi} + \|\Pi_{\mathcal{F},\rho^\pi} V^\pi - V^\pi\|^2_{2,\rho^\pi},$$

which leads to the desired result after a re-arrangement. □

5.3 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 estimates 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.

5.3.1 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 5.2.1 (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

$$\begin{aligned}
\mathbb{E}\left[|Q(X,A) - G^\pi(X,A)|^2\right] &= \mathbb{E}\left[|Q(X,A) - Q^\pi(X,A) + Q^\pi(X,A) - G^\pi(X,A)|^2\right] \\
&= \mathbb{E}\left[|Q(X,A) - Q^\pi(X,A)|^2\right] + \\
&\quad \mathbb{E}\left[|Q^\pi(X,A) - G^\pi(X,A)|^2\right] + \\
&\quad 2\mathbb{E}\left[\left(Q(X,A) - Q^\pi(X,A)\right)\left(Q^\pi(X,A) - G^\pi(X,A)\right)\right] .
\end{aligned}$$

The first term is $\|Q - Q^\pi\|_{2,\mu}$, the same as the population loss (5.39). The second term is

$$\mathbb{E}\left[\mathrm{Var}\left[G^\pi(X,A) \mid X, A\right]\right] ,$$

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]

$$\begin{aligned}
&\mathbb{E}\left[\left(Q(X,A) - Q^\pi(X,A)\right)\left(Q^\pi(X,A) - G^\pi(X,A)\right)\right] = \\
&\mathbb{E}\left[\mathbb{E}\left[\left(Q(X,A) - Q^\pi(X,A)\right)\left(Q^\pi(X,A) - G^\pi(X,A)\right) \mid X, A\right]\right] = \\
&\mathbb{E}\left[\left(Q(X,A) - Q^\pi(X,A)\right)\left(Q^\pi(X,A) - \mathbb{E}\left[G^\pi(X,A) \mid X, A\right]\right)\right] .
\end{aligned}$$

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 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

$$\begin{aligned}
&\operatorname*{argmin}_{Q\in\mathcal{F}} \mathbb{E}\left[|Q(X,A) - G^\pi(X,A)|^2\right] = \\
&\operatorname*{argmin}_{Q\in\mathcal{F}} \left\{\mathbb{E}\left[|Q(X,A) - Q^\pi(X,A)|^2\right] + \mathbb{E}\left[\mathrm{Var}\left[G^\pi(X,A) \mid X, A\right]\right]\right\} = \\
&\operatorname*{argmin}_{Q\in\mathcal{F}} \|Q - Q^\pi\|_{2,\mu}^2 ,
\end{aligned}$$

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

$$\hat{Q} \leftarrow \operatorname*{argmin}_{Q\in\mathcal{F}} \|Q - G^\pi\|_{2,\mu_n}^2 .$$

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 to 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 spaces that we often 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. This will be completed. □

5.3.2 Approximate Value Iteration

Designing an empirical version of AVI (Section 5.2.2) is easy. The iteration of (5.3),

$$Q_{k+1} \leftarrow \operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - TQ_k\|_{2,\mu} = \int |Q(x, a) - (TQ)(x, a)|^2 \, \mathrm{d}\mu(x, a), \tag{5.41}$$

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 5.2.1. 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 5.3.1, 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

$$\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2\right] = \mathbb{E}\left[\mathbb{E}\left[\left|Q(Z) - (\hat{T}Q_k)(Z)\right|^2 \mid Z\right]\right]$$
$$= \mathbb{E}\left[|Q(Z) - (TQ_k)(Z)|^2\right] + \mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q_k)(Z) \mid Z\right]\right].$$

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:

$$\hat{Q}_{k+1} \leftarrow \operatorname*{argmin}_{Q\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^{n}\left|Q(X_i, A_i) - (\hat{T}Q_k)(X_i, A_i)\right|^2 = \left\|Q - \hat{T}Q_k\right\|_{2,\mathcal{D}_n}^2. \tag{5.46}$$

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.

5.3.3 Bellman Residual Minimization

We may use data to define the empirical version of BRM (Section 5.2.3). 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

$$\begin{aligned}
\operatorname*{argmin}_{Q \in \mathcal{F}} \mathbb{E}\left[\left|Q(Z) - (\hat{T}Q)(Z)\right|^2\right] &= \operatorname*{argmin}_{Q \in \mathcal{F}} \left\{\|Q - TQ\|_{2,\mu}^2 + \mathbb{E}\left[\mathrm{Var}\left[(\hat{T}Q)(Z) \mid Z\right]\right]\right\} \\
&\neq \operatorname*{argmin}_{Q \in \mathcal{F}} \|Q - TQ\|_{2,\mu}^2. \qquad (5.49)
\end{aligned}$$

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

$$\begin{aligned}
&\mathrm{Var}\left[(\hat{T}^\pi Q)(Z) \mid Z\right] = \\
&\mathbb{E}\left[\left|[r(Z) + \gamma Q(X', \pi(X'))] - \left[r(Z) + \gamma \int \mathcal{P}(\mathrm{d}x'|Z) Q(x', \pi(x'))\right]\right|^2 \mid Z\right] = \\
&\gamma^2 \mathbb{E}\left[\left|Q(X', \pi(X')) - \int \mathcal{P}(\mathrm{d}x'|Z) Q(x', \pi(x'))\right|^2 \mid Z\right].
\end{aligned}$$

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.

5.3.4 LSTD and Least Squares 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 5.2.4.2). 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

$$A_{ij} = \left[\Phi^\top M(\Phi - \gamma \mathcal{P}^\pi \Phi)\right]_{ij} = \sum_{m=1}^{N} \Phi_{im}^\top \mu(m) \left(\Phi - \gamma \mathcal{P}^\pi \Phi\right)_{mj},$$

$$b_i = \sum_{m=1}^{N} \Phi_{im}^\top \mu(m) r_m^\pi$$

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

$$\mathbb{E}\left[\phi(X_i) \left(\phi(X_i) - \gamma\phi(X_i')\right)^\top\right] =$$

$$\mathbb{E}\left[\mathbb{E}\left[\phi(X_i) \left(\phi(X_i) - \gamma\phi(X_i')\right)^\top\right] \mid X_i\right] =$$

$$\mathbb{E}\left[\phi(X_i) \left(\phi(X_i) - \gamma\mathbb{E}\left[\phi(X_i') \mid X_i\right]\right)^\top\right] =$$

$$\mathbb{E}\left[\phi(X_i) \left(\phi(X_i) - \gamma(\mathcal{P}^\pi\Phi)(X_i)\right)^\top\right] = A.$$

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 $\mathcal{P}$ 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

$$[\Pi_\pi]_{x,(x,a)} = \pi(a|x).$$

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 $M_{N\times N} = \mathrm{diag}(\mu)$. The solution of the LSTD solution (population version) is

$$w = \left[\Phi^\top M(\Phi - \gamma\mathcal{P}\Pi_\pi\Phi)\right]^{-1}\Phi^\top Mr = A^{-1}b. \tag{5.50}$$

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.

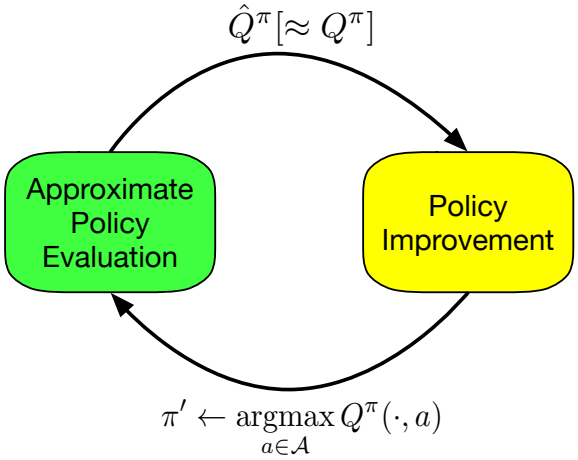

Figure 5.4: Approximate Policy Iteration

Algorithm 5.1 LSPI

Require: $\mathcal{D}_n = \{(X_i, A_i, R_i, X_i')\}_{i=1}^n$ and initial policy π_1.

1: **for** $k = 1, 2, \ldots, K$ **do**
2: $\quad \hat{Q}^{\pi_k} \leftarrow \text{LSTD}(\mathcal{D}_n, \pi_k)$ ▷ Policy Evaluation
3: $\quad \pi_{k+1} \leftarrow \pi_g(\hat{Q}^{\pi_k})$ ▷ Policy Improvement
4: **end for**
5: **return** $\hat{Q}^{\pi_K}$ and π_{K+1}.

We can use LSTD to define an an approximate PI (API) procedure (Figure 5.4) 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, and is known as the *Least Squares Policy Iteration (LSPI)* algorithm. 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 $\hat{A}_{n-1}^{-1}$ (Appendix A.5).

5.4 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 5.2.4.3 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 $X_1, \ldots X_t$ 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 do better though. Consider the summation $\hat{S}_t = \sum_{i=1}^{t} \phi(X_i)\phi^\top(X_i)$. This means that $\hat{A}_t^{-1} = t\hat{S}_t^{-1}$. We can use Sherman-Morrison formula (Section A.5) to incrementally update $\hat{S}_t^{-1}$, and as a result $\hat{A}_t^{-1}$, (Section A.5) as

$$(\hat{S}_t)^{-1} = \hat{S}_{t-1}^{-1} - \frac{\hat{S}_{t-1}^{-1}\phi(X_t)\phi^\top(X_t)\hat{S}_{t-1}^{-1}}{1 + \phi^\top(X_t)\hat{S}_{t-1}^{-1}\phi(X_t)}.$$

This requires a matrix-vector multiplication and is $O(p^2)$. The update rule (5.52) becomes

$$w_{t+1} \leftarrow (1-\alpha_t)w_t + \alpha_t t\hat{S}_t^{-1}\phi(X_t)\left(R_t + \gamma\phi^\top(X_t')w_t\right). \tag{5.53}$$

The per-sample computational cost of (5.52) (and (5.53)) 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 5.2.4.3. The population version is

$$w_{k+1} \leftarrow w_k + \alpha\Phi^\top D^\pi\left[r^\pi + \gamma\mathcal{P}^\pi\Phi w_k - \Phi w_k\right].$$

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 $\Phi^\top D^\pi\left[r^\pi + \gamma\mathcal{P}^\pi\Phi w_k - \Phi w_k\right]$. Therefore, the SA update rule would be

$$w_{k+1} \leftarrow w_k + \alpha_t\phi(X_t)\delta_t. \tag{5.54}$$

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 may diverge. This is in contrast with the TD for finite state problems where the conditions of convergence were much more relaxed and we did not face 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

$$w_{k+1} \leftarrow w_k + \alpha_t\phi(X_t,A_t)\delta_t, \tag{5.55}$$

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 may diverge.

5.4.1 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. This is the approach followed by Sutton and Barto [2018].

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 5.2.1). The population loss is

$$V \leftarrow \operatorname*{argmin}_{V \in \mathcal{F}} \frac{1}{2} \left\| V^\pi - \hat{V}(w) \right\|_{2,\mu}^2 . \tag{5.56}$$

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 (5.56). 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 (5.54).

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 5.3.3. Minimizing it does not lead to the minimizer of the Bellman error.

Chapter 6

Policy Search Methods

6.1 Introduction

So far we have described methods for computing the optimal policy, either exactly or inexactly, that are based on the computation of the value function.[1] Given Q^*, we can compute the optimal policy by computing its greedy policy. In these methods, only the value function was explicitly represented, and the policy could be computed based on it.

There are also methods that are based on the explicit representation of the policy, as opposed to value function, and optimizing the performance of the agent by searching in the space of policies. We call these methods *policy search* algorithms. These are the focus of this chapter. There are also hybrid methods that use the explicit representation of both value and policy.

6.1.1 Policy Parametrization

Let us start from policy parametrization. Consider a stochastic policy $\pi_\theta : \mathcal{X} \to \mathcal{M}(\mathcal{A})$ that is parameterized by a $\theta \in \Theta$. Here the set Θ is the parameter space, e.g., a subset of $\mathbb{R}^p$. We also denote the space of all policy parameterized this way by Π_Θ:

$$\Pi_\Theta = \{ \pi_\theta : \mathcal{X} \to \mathcal{M}(\mathcal{A}) \ : \ \theta \in \Theta \} . \tag{6.1}$$

This space depends on the mapping π_θ and the space of parameters Θ.

There are many choices for how we can parameterize a policy π_θ. A generic example is based on the Boltzmann (or softmax) distribution. Given a function

[1] Chapter's Version: 0.04 (2021 April 2). Some results need to be typed.

$f_\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ (e.g., a DNN or decision tree parameterized by θ), the density of choosing action a at state x is

$$\pi_\theta(a|x) = \frac{\exp(f_\theta(x,a))}{\int \exp(f_\theta(x,a'))\mathrm{d}a'}.$$

A special case would be when $f_\theta(x,a) = \phi(x,a)^\top \theta$ for some features $\phi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^p$ and $\theta \in \mathbb{R}^p$, in which case we get

$$\pi_\theta(a|x) = \frac{\exp(\phi(x,a)^\top \theta)}{\int \exp(\phi(x,a')^\top \theta)\mathrm{d}a'}.$$

When the action space $\mathcal{A}$ is discrete, $\pi_\theta(a|x)$ denotes the probability of choosing action a at state x (instead of it density), and for the linear parameterization of f_θ, we have

$$\pi_\theta(a|x) = \frac{\exp(\phi(x,a)^\top \theta)}{\sum_{a' \in \mathcal{A}} \exp(\phi(x,a')^\top \theta)}.$$

Another example is when $\pi_\theta(\cdot|x)$ defines a Normal distribution over action space with θ parameterization its mean and covariance:

$$\pi_\theta(\cdot|x) = \mathcal{N}\left(\mu_\theta(x), \Sigma_\theta(x)\right).$$

If the action space is $d_{\mathcal{A}}$-dimensional, the mean is $\mu_\theta : \mathcal{X} \to \mathbb{R}^{d_{\mathcal{A}}}$ and the covariance is $\Sigma_\theta : \mathcal{X} \to S_+^{d_{\mathcal{A}}}$. Here $S_+^{d_{\mathcal{A}}}$ refers to the set of $d_{\mathcal{A}} \times d_{\mathcal{A}}$ positive semi-definite matrices.

This latest example shows one of the reasons that the explicit parameterization of the policy and the policy search methods is appealing: Working with continuous action spaces is easy. By simply parametrizing the mean and covariance of a Normal distribution, we can easily select a continuous action from π_θ. If we can come up with methods to optimize a policy parametrized this way, this would be a viable approach for working with MDPs continuous action spaces, potentially of high dimensions. In comparison, continuous action space causes challenge for value-based methods such as VI or PI that need to compute $\max_{a \in \mathcal{A}} Q(x,a)$ (or a similar quantity). This is an optimization in the action space, which is difficult when $\mathcal{A}$ is a high-dimensional continuous space.

6.1.2 Performance Measure

The performance can be measured in various ways. Here we focus on the expected return of following π_θ, the value function. Our goal, as before, is to find a policy

that maximizes this performance measure. We are, however, restricted to choosing policies within Π_Θ (6.1).

For the moment, assume that we only care about the performance at state $x \in \mathcal{X}$. In that case, the goal of policy search would be

$$\operatorname*{argmax}_{\pi \in \Pi_\Theta} V^\pi(x) = \operatorname*{argmax}_{\theta \in \Theta} V^{\pi_\theta}(x). \tag{6.2}$$

The interpretation is that we are interested in finding a policy that if the agent starts at this particular state x, its performance, measured according to its expected return, is maximized.

Of course, if we find $\pi^* \in \Pi_\Theta$, not only it maximizes the value function at this particular x, but also at any other $x' \in \mathcal{X}$. But if $\pi^* \notin \Pi_\Theta$, we will not be able to find a policy that maximizes the value at all states. In that case, we may want to find a policy that is only good at our starting state x, and ignore the performance at other states. In that case, the obtained policy is going to be initial-state-dependent. That is, if we change x to another state $x' \neq x$, the optimal policy within Π_Θ might change.

More general than this extreme case of having a single initial state x is when the initial state of the agent is distributed according to some distribution $\rho \in \mathcal{M}(\mathcal{X})$. The performance measure would then be the average of following π_θ with the initial state $X_1 \sim \rho$. We define

$$J(\pi_\theta) = J_\rho(\pi_\theta) \triangleq \int V^{\pi_\theta}(x) \mathrm{d}\rho(x) = \mathbb{E}_{X \sim \rho}\left[V^{\pi_\theta}(X)\right]. \tag{6.3}$$

The optimal policy maximizes the performance measure J_ρ and we have $J_\rho(\pi^*) \geq J_\rho(\pi_\theta)$ for any $\pi_\theta \in \Pi_\Theta$. If $\pi^* \notin \Pi_\Theta$, which is the case in general, the inequality is strict. In policy search methods, however, we aim to find the maximizes of the performance measure within Π_Θ. That is,

$$\bar{\pi} \leftarrow \operatorname*{argmax}_{\pi_\theta \in \Pi_\Theta} J_\rho(\pi_\theta). \tag{6.4}$$

The corresponding policy is denoted by $\bar{\theta}$, i.e., $\bar{\pi} = \pi_{\bar{\theta}}$.

For different ρ, we may get different optimizers. If we want to emphasize the dependence of the maximizer on ρ, we use $\bar{\pi}_\rho$. We may sometimes denote $J(\pi_\theta)$ or $J_\rho(\pi_\theta)$ simply by $J_\rho(\theta)$.

6.1.3 Policy Search as an Optimization Problem

How can we solve the optimization problem (6.4) to find π_θ that maximizes the performance measure J_ρ? In principle, this is an optimization problem, so we can

benefit from the arsenal of optimization algorithms. Being an RL problem, however, brings both challenges and opportunities. As an example of a challenge, the value of J_ρ is not readily available, and has to be estimated through interaction with the environment. The special structure of the RL problem, such as the value function satisfying the Bellman equation, provides an opportunity to design more elegant algorithms than one would obtain if we merely consider the problem as a blackbox optimization one. We discuss these in some detail and describe a few methods on how the optimization problem can be solved.

Optimization methods, broadly speaking, can be categorized based on the information they need about their objective. Zero-order methods only use the value of the objective at various query points. For example, they compute $J_\rho(\theta)$ at various θs in order to guide the optimization process. First-order methods use the derivative of the objective instead of, or in addition to, the value of the objective. They use $\nabla_\theta J_\rho(\theta)$ in order to guide the search. The quintessential first-order optimization method is the gradient descent (and its stochastic variant), which has a counterpart for the RL problems too.

6.2 Zero-Order Methods

We first consider the case when the policy parameter space Θ is finite. This helps us understand some of the challenges. We then extend our discussion to case when Θ is continuous function space.

6.2.1 Finite Policy Parameter Space

Assume that we are given a finite $\Theta = \{\theta_1, \ldots, \theta_m\}$ policy parameters. This defines the finite policy space $\Pi_\Theta = \{\pi_\theta : \theta \in \Theta\}$. We want to find the policy $\pi_\theta \in \Pi_\Theta$ such that $J_\rho(\pi_\theta)$ is maximized (6.4).

If we can easily compute $J_\rho(\pi_\theta)$ for each $\theta \in \Theta$, this is an easy problem, at least in principle.[2] So the main issue is to compute $J_\rho(\pi_\theta)$.

The performance measure $J_\rho(\pi_\theta)$ is the expectation of $V^{\pi_\theta}(X)$ w.r.t. $X \sim \rho$. We can try to compute $V^{\pi_\theta}(x)$ for all $x \in \mathcal{X}$, using any of the PE methods that we have developed (either in the Planning or RL scenarios), and then take the weighted

[2] If $m = |\Theta|$ is a very large finite number, say 10^{40}, this may not be feasible on a computer, but that is another issue, which we ignore in this discussion.

average according to ρ. If $\mathcal{X}$ is discrete, this would be

$$J_\rho(\pi_\theta) = \sum_{x \in \mathcal{X}} \rho(x) V^{\pi_\theta}(x).$$

If $\mathcal{X}$ is large, however, computing V^{π_θ} itself is not going to be easy, no matter whether we know the model or not. Moreover, computing the integral $\int V^{\pi_\theta}(x) \mathrm{d}\rho(x)$ is going to be challenging too, even if we know V^{π_θ} exactly.

An alternative is computing an unbiased estimate of $J_\rho(\pi_\theta)$ instead using MC estimation. To see how it works, let us derive such an estimator in two steps.

In the first step, assume that we know V^{π_θ} and we want to estimate $J_\rho(\pi_\theta)$. If we sample $X \sim \rho$, we have that $V^{\pi_\theta}(X)$ is an unbiased estimate of $J_\rho(\pi_\theta)$ as

$$\mathbb{E}\left[V^{\pi_\theta}(X)\right] = \int V^{\pi_\theta}(x) \mathrm{d}\rho(x) = J_\rho(\pi_\theta),$$

by the definition of $J_\rho(\pi_\theta)$ (6.3).

If we draw n independent samples $X_1, \ldots, X_n \sim \rho$, the estimator

$$\frac{1}{n} \sum_{i=1}^{n} V^{\pi_\theta}(X_i)$$

would be an unbiased as well, with a variance of

$$\frac{\mathrm{Var}\left[V^{\pi_\theta}(X)\right]}{n}.$$

This variance goes to 0 as n increases. So by increasing the number of samples, we can have a more accurate estimate of $J_\rho(\pi_\theta)$.

The variance $\mathrm{Var}\left[V^{\pi_\theta}(X)\right]$ is a measure of dispersion of the value function across states samples according to ρ. For example, if the value function is constant, it will be zero, and if it is changing slowly as a function of the state, it would be small. On the other hand, if the value function varies greatly, the variance is large. Note that this variance is a function of the policy π_θ, so for each $\pi_\theta \in \Pi_\Theta$, we get a different variance.

The second step is to replace $V^{\pi_\theta}(x)$ with the return $G^{\pi_\theta}(x)$, which is an unbiased estimate of the value itself. Computation of $G^{\pi_\theta}(x)$ requires starting the agent from state x and following π_θ (i.e., performing a rollout from x) until the end of episode for episodic tasks, or until infinity for continual tasks (see Exercise 3.2 and the comment in Section 4.3).

If $X \sim \rho$, $G^{\pi_\theta}(X)$ is an unbiased estimate of $J_\rho(\pi_\theta)$ as

$$\mathbb{E}_{X\sim\rho}\left[G^{\pi_\theta}(X)\right] = \mathbb{E}_{X\sim\rho}\left[\mathbb{E}\left[G^{\pi_\theta}(X) \mid X\right]\right] = \mathbb{E}_{X\sim\rho}\left[V^{\pi_\theta}(X)\right] = J_\rho(\pi_\theta).$$

If we draw n independently selected $X_1, \ldots, X_n \sim \rho$, we can form

$$\hat{J}_n(\pi_\theta) = \frac{1}{n}\sum_{i=1}^{n} G^{\pi_\theta}(X_i), \tag{6.5}$$

as an unbiased estimate of $J_\rho(\pi_\theta)$. The next result provides the variance of this estimate.

Proposition 6.1. *The estimator $\hat{J}_n(\pi_\theta)$ (6.5) is an unbiased estimator for $J_\rho(\pi_\theta)$ and has the variance of*

$$\mathrm{Var}\left[\hat{J}_n(\pi_\theta)\right] = \frac{1}{n}\left(\mathbb{E}\left[\mathrm{Var}\left[G^{\pi_\theta}(X) \mid X\right]\right] + \mathrm{Var}\left[V^{\pi_\theta}(X)\right]\right).$$

Proof. The estimator is unbiased because

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} G^{\pi_\theta}(X_i)\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\mathbb{E}\left[G^{\pi_\theta}(X_i) \mid X_i\right]\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[V^{\pi_\theta}(X_i) \mid X_i\right] = \frac{n}{n}J_\rho(\pi_\theta),$$

as had already been established before.

As X_is are i.i.d., we have

$$\mathrm{Var}\left[\frac{1}{n}\sum_{i=1}^{n} G^{\pi_\theta}(X_i)\right] = \frac{1}{n}\mathrm{Var}\left[G^{\pi_\theta}(X_1)\right]. \tag{6.6}$$

We could use the law of total variance to decompose $\mathrm{Var}\left[G^{\pi_\theta}(X)\right]$ (with $X \sim \rho$). Instead of using that result, we prove it directly. We have

$$\mathrm{Var}\left[G^{\pi_\theta}(X)\right] = \mathbb{E}\left[G^{\pi_\theta}(X)^2\right] - \left(\mathbb{E}\left[G^{\pi_\theta}(X)\right]\right)^2. \tag{6.7}$$

We expand each term. For the term $\mathbb{E}\left[G^{\pi_\theta}(X)^2\right]$, we have

$$\mathbb{E}\left[G^{\pi_\theta}(X)^2\right] = \mathbb{E}\left[\mathbb{E}\left[G^{\pi_\theta}(X)^2 \mid X\right]\right].$$

Its inner expectation is

$$\begin{aligned}\mathbb{E}\left[G^{\pi_\theta}(X)^2 \mid X\right] &= \mathrm{Var}\left[G^{\pi_\theta}(X) \mid X\right] + \left(\mathbb{E}\left[G^{\pi_\theta}(X) \mid X\right]\right)^2 \\ &= \mathrm{Var}\left[G^{\pi_\theta}(X) \mid X\right] + V^{\pi_\theta}(X)^2,\end{aligned}$$

as $V^{\pi_\theta}(x) = \mathbb{E}\left[G^{\pi_\theta}(x) \mid x\right]$ by definition. Therefore,

$$\mathbb{E}\left[G^{\pi_\theta}(X)^2\right] = \mathbb{E}\left[\mathrm{Var}\left[G^{\pi_\theta}(X) \mid X\right]\right] + \mathbb{E}\left[V^{\pi_\theta}(X)^2\right]. \tag{6.8}$$

For the second term $(\mathbb{E}\left[G^{\pi_\theta}(X)\right])^2$, notice that we have

$$\mathbb{E}\left[G^{\pi_\theta}(X)\right] = \mathbb{E}\left[\mathbb{E}\left[G^{\pi_\theta}(X) \mid X\right]\right] = \mathbb{E}\left[V^{\pi_\theta}(X)\right]. \tag{6.9}$$

Substituting (6.8) and (6.9) in (6.7), we get

$$\begin{aligned}\mathrm{Var}\left[G^{\pi_\theta}(X)\right] &= \mathbb{E}\left[\mathrm{Var}\left[G^{\pi_\theta} \mid X\right]\right] + \left(\mathbb{E}\left[V^{\pi_\theta}(X)^2\right] - \left(\mathbb{E}\left[V^{\pi_\theta}(X)\right]\right)^2\right) \\ &= \mathbb{E}\left[\mathrm{Var}\left[G^{\pi_\theta} \mid X\right]\right] + \mathrm{Var}\left[V^{\pi_\theta}(X)\right].\end{aligned}$$

This along with (6.6) get us to the stated result. □

This result shows that if we have a finite number of parameters in Θ, we can estimate $J_\rho(\pi_{\theta_i}) \approx \hat{J}_n(\pi_{\theta_i}) \pm O_P(\frac{1}{\sqrt{n}})$ for each $\theta_i \in \Theta$. We can use these estimates to select the best among them:

$$\hat{\pi} = \pi_{\hat{\theta}} \leftarrow \operatorname*{argmax}_{\theta \in \Theta} \hat{J}_n(\pi_\theta). \tag{6.10}$$

As there is an $O_P(\frac{1}{\sqrt{n}})$ error in estimation of each $J_\rho(\pi_\theta)$, the selected policy $\hat{\pi}$ may not be the same as the maximizer $\bar{\pi}$ of (6.4). The error in selection can happen if $J_\rho(\hat{\pi})$ and $J_\rho(\bar{\pi})$ are within $O_P(\frac{1}{\sqrt{n}})$ of each other, so their ranking of their empirical estimate would be different than the ranking of their true value. That is, $\hat{J}_n(\hat{\pi}) > \hat{J}_n(\bar{\pi})$ (which leads to preferring $\hat{\pi}$ to $\bar{\pi}$ according to the empirical performance measure) even though $J_\rho(\hat{\pi}) < J_\rho(\bar{\pi})$. Even if we make an error in selecting the best policy, the gap in their performance is within $O_P(\frac{1}{\sqrt{n}})$. The next result and its proof make this statement and argument precise. Note that as we increase n, the error in estimating $J_\rho(\pi_\theta)$ decreases and the probability of selecting an optimal policy increases. This increased accuracy, however, is at the cost of increased sample and computational complexity, which would be $n|\Theta|$ rollouts.

Proposition 6.2. *Consider the maximizer of (6.10). Assume that the returns $G^{\pi_\theta}(x)$ are all Q_{max}-bounded for any $\theta \in \Theta$ and $x \in \mathcal{X}$. Furthermore, suppose that $|\Theta| < \infty$. For any $\delta > 0$, we have that*

$$J_\rho(\hat{\theta}) \geq \max_{\theta \in \Theta} J_\rho(\theta) - 2Q_{max}\sqrt{\frac{2\ln\left(\frac{2|\Theta|}{\delta}\right)}{n}},$$

with probability at least $1 - \delta$.

Proof. We bound the difference between $J_\rho(\theta)$ and $\hat{J}_n(\theta)$, for any $\theta \in \Theta$, using the union bound and the Hoeffding's inequality (Lemma A.3 in Appendix A). For any $\theta' \in \Theta$, we have

$$\begin{aligned}\mathbb{P}\left\{\left|J_\rho(\theta') - \hat{J}_n(\theta')\right| > \varepsilon\right\} &\leq \mathbb{P}\left\{\bigcup_{\theta\in\Theta}\left\{\left|J_\rho(\theta) - \hat{J}_n(\theta)\right| > \varepsilon\right\}\right\} \\ &\leq \sum_{\theta\in\Theta}\mathbb{P}\left\{\left|J_\rho(\theta) - \hat{J}_n(\theta)\right| > \varepsilon\right\} \\ &\leq 2|\Theta|\exp\left(-\frac{n\varepsilon^2}{2Q_{\max}^2}\right).\end{aligned}$$

Assigning the LHS to $\delta > 0$, and solving for ε, we get that

$$\varepsilon \leq Q_{\max}\sqrt{\frac{2\ln\left(\frac{2|\Theta|}{\delta}\right)}{n}}.$$

Therefore, for any $\theta' \in \Theta$, including $\hat{\theta}$ and $\bar{\theta}$, we have

$$\left|J_\rho(\theta') - \hat{J}_n(\theta')\right| \leq Q_{\max}\sqrt{\frac{2\ln\left(\frac{2|\Theta|}{\delta}\right)}{n}}, \tag{6.11}$$

with probability at least $1-\delta$. For conciseness, let us denote the RHS by $u(n,\delta)$.

As $\hat{J}_n(\hat{\theta}) = \hat{J}_n(\pi_{\hat{\theta}})$ is the maximizer of (6.10), we have

$$\hat{J}_n(\hat{\theta}) \geq \hat{J}_n(\theta), \qquad \forall\theta \in \Theta.$$

This includes $\theta = \tilde{\theta} \in \Theta$, the parameter of the best policy within Θ. So $\hat{J}_n(\hat{\theta}) \geq \hat{J}_n(\bar{\theta})$. This along with two applications of (6.11) show that for any $\delta > 0$,

$$\begin{aligned}J_\rho(\hat{\theta}) &\geq \hat{J}_n(\hat{\theta}) - u(n,\delta) \\ &\geq \hat{J}_n(\bar{\theta}) - u(n,\delta) \\ &\geq (J_\rho(\bar{\theta}) - u(n,\delta)) - u(n,\delta),\end{aligned}$$

with probability at least $1-\delta$. □

Assigning the same number n of MC rollouts to all $\theta \in \Theta$ may not be the best strategy in selecting the optimal parameter in terms of the total number of required rollouts. To see why, assume that for $\theta_1 \in \Theta$, the performance measure $J_\rho(\pi_{\theta_1})$ is much smaller than $J_\rho(\pi_{\theta_2})$ for another $\theta_2 \in \Theta$. In this case, we can perhaps eliminate θ_1 from further consideration without spending many samples on accurately estimate its $J_\rho(\pi_{\theta_1})$. More concretely, given n_1 rollouts assigned to θ_1, we can form the confidence interval for $J_\rho(\pi_{\theta_1})$. For example, if we assume $Q_{\max}$-bounded returns, we can use the Hoeffding's inequality to obtain that for any $\delta > 0$,

$$J_\rho(\pi_{\theta_1}) \in \left(\hat{J}_{n_1}(\pi_{\theta_1}) - Q_{\max}\sqrt{\frac{2\ln(2/\delta)}{n_1}}, \hat{J}_{n_1}(\pi_{\theta_1}) + Q_{\max}\sqrt{\frac{2\ln(2/\delta)}{n_1}} \right),$$

with probability at least $1-\delta$. We have a similar result for $J_\rho(\pi_{\theta_2})$ with n_2 rollout assigned to it:

$$J_\rho(\pi_{\theta_2}) \in \left(\hat{J}_{n_2}(\pi_{\theta_2}) - Q_{\max}\sqrt{\frac{2\ln(2/\delta)}{n_2}}, \hat{J}_{n_2}(\pi_{\theta_2}) + Q_{\max}\sqrt{\frac{2\ln(2/\delta)}{n_2}} \right),$$

with probability at least $1-\delta$. If it happens that

$$\hat{J}_{n_1}(\pi_{\theta_1}) + Q_{\max}\sqrt{\frac{2\ln(2/\delta)}{n_1}} < \hat{J}_{n_2}(\pi_{\theta_2}) - Q_{\max}\sqrt{\frac{2\ln(2/\delta)}{n_2}},$$

we conclude that the best statistically possible performance of π_{θ_1} is worse than the worst statistically possible performance of π_{θ_2}, with probability at least $1-2\delta$. Hence, we can say that with high probability, π_{θ_2} is a better policy than π_{θ_1}. In that case, there is no need to improve our estimate of π_{θ_1}, as there is a better policy already π_{θ_2} identified.

This idea can be fully developed into an algorithm that adaptively adjusts the number of rollouts n_i assigned to each policy $\pi_i \in \Pi_\Theta$. Such an algorithm can be more sample efficient in identifying a good policy than assigning the same number of rollouts to all policies in the policy class. We do not develop it any further here though, and refer an interested reader to relevant references.

6.2.2 Random Search

If Θ is not finite, we cannot evaluate $\hat{J}_n(\pi_\theta)$ for all $\theta \in \Theta$. There are several generic methods for searching in a large parameter space, including random search, simulated

Algorithm 6.1 Random Search

Require: Distribution $\nu \in \mathcal{M}(\Theta)$; Number of rollouts n; Maximum number of iterations K

Draw a parameter $\theta_1 = \theta'_1 \sim \nu$
Evaluate $\hat{J}_n(\pi_{\theta_1})$
for $k = 2, 3, \ldots, K$ **do**
    Draw a parameter $\theta'_k \sim \nu$
    Evaluate $\hat{J}_n(\pi_{\theta'_k})$
    if $\hat{J}_n(\pi_{\theta'_k}) > \hat{J}_n(\pi_{\theta_{k-1}})$ **then**
        $\theta_k \leftarrow \theta'_k$
    else
        $\theta_k \leftarrow \theta_{k-1}$
    end if
end for
return π_{θ_K}

annealing, and various evolutionary algorithms. We briefly discuss random search in this section and a particular type of evolutionary algorithm in the next one.

In random search (RS), we randomly pick m policy parameters $\theta_1, \ldots, \theta_m \in \Theta$, evaluate $\hat{J}_n(\pi_{\theta_i})$, and pick the one with the highest value. The intuition of why this works is that with large enough m, one of θ_i might hit close to the optimal

$$\hat{\theta} \leftarrow \operatorname*{argmax}_{\theta \in \Theta} \hat{J}_n(\pi_\theta).$$

Moreover, if n is large enough, the difference between $\hat{J}_n(\theta)$ and $J_\rho(\theta)$ would be small for all randomly selected θ. To be more concrete, if we denote the set of randomly selected parameters by $\Theta_m = \{\theta_1, \ldots, \theta_m\}$, we may use a result similar to Proposition 6.2 to show that the gap between the true performance $J_\rho(\hat{\theta}_m)$ of the selected parameter

$$\hat{\theta}_m \leftarrow \operatorname*{argmax}_{\theta \in \Theta_m} \hat{J}_n(\pi_\theta)$$

and $\max_{\theta \in \Theta_m} J_\rho(\pi_\theta)$ is small.[3] We do not prove this result in its complete form. But we show that RS asymptotically chooses a θ whose $\hat{J}_n$ is arbitrary close to $\hat{J}_n(\hat{\theta})$.

To show that RS hits close to the optimum, let us first specify the procedure more precisely in Algorithm 6.1. The distribution $\nu \in \mathcal{M}(\Theta)$, which is an input to

[3] Proposition 6.2 assume that $|\Theta| < \infty$. If we allow $m \to \infty$ in a random search, we need to modify that result.

the algorithm, generates samples in the policy parameter space. We assume that it assigns a positive probability to any (measurable) set $A \subset \Theta$. This is required because otherwise we may not draw any sample from the regions close to the optimal point $\hat{\theta}$ after all. We define the set of points close to being optimal. For $\varepsilon > 0$, denote the set of points in Θ that are ε-optimal (according to $\hat{J}_n$) by

$$M_\varepsilon = \left\{ \theta \,:\, \hat{J}_n(\theta) > \hat{J}_n(\hat{\theta}) - \varepsilon \right\}.$$

Proposition 6.3. *Consider the sequence (θ_k) generated by Algorithm 6.1. Assume that the distribution $\nu \in \mathcal{M}(\Theta)$ is such that for any (measurable) set $A \subset \Theta$, the probability assigned to it is strictly positive, i.e., $\nu(A) > 0$. Then, for any $\varepsilon > 0$,*

$$\lim_{k\to\infty} \mathbb{P}\{\theta_k \in M_\varepsilon\} = 1.$$

Proof. By the k-th iteration of the algorithm, the probability of θ_k not being in M_ε is the same as the probability of none of $\theta'_1, \theta'_2, \ldots, \theta'_k$ being within M_ε. Therefore,

$$\begin{aligned}
\mathbb{P}\{\theta_k \notin M_\varepsilon\} &= \mathbb{P}\{\theta'_1 \notin M_\varepsilon, \theta'_2 \notin M_\varepsilon, \ldots, \theta'_k \notin M_\varepsilon\} \\
&= \prod_{i=1}^{k} \mathbb{P}\{\theta'_i \notin M_\varepsilon\} \\
&= \prod_{i=1}^{k} (1 - \mathbb{P}\{\theta'_i \in M_\varepsilon\}) \\
&= \prod_{i=1}^{k} (1 - \nu(M_\varepsilon)) = (1 - \nu(M_\varepsilon))^k .
\end{aligned}$$

By assumption, $\nu(M_\varepsilon) > 0$, so $(1 - \nu(M_\varepsilon)) = p < 1$. So, at the limit of $k \to \infty$,

$$\mathbb{P}\{\theta_k \notin M_\varepsilon\} = \lim_{k\to\infty} p^k = 0.$$

Therefore, as $\lim_{k\to\infty} \mathbb{P}\{\theta_k \in M_\varepsilon\} = 1$. □

Despite this asymptotic guarantee, RS is not the most efficient way to search a parameter space. The way it is presented here does not benefit from all previous evaluation of the function when suggesting a new θ'_k. That knowledge can be useful, for example, by helping us focus on more promising regions of the search space, instead of blindly sampling from the same distribution ν. This can be achieved by adaptively changing the distribution ν_k to be a function of previous evaluations. Next, we study one such example.

Algorithm 6.2 Evolutionary Strategy (ES)$(1+1)$

Require: Initial point $\theta_0 \in \Theta$; Number of rollouts n; Maximum number of iterations K
Require: Initial standard deviation of mutation operator: $\sigma_1 > 0$
Require: Adaptation parameters: $c_+ > 0$ and $c_- < 0$.
Evaluate $\hat{J}_n(\pi_{\theta_0})$
for $k = 1, 2, \ldots, K$ **do**
  Draw a perturbation $\eta \sim \mathcal{N}(0, \mathbf{I})$
  $\theta'_k \leftarrow \theta_k + \sigma_k \eta$ ▷ Mutation
  Evaluate $\hat{J}_n(\pi_{\theta'_k})$
  if $\hat{J}_n(\pi_{\theta'_k}) > \hat{J}_n(\pi_{\theta_k})$ **then** ▷ Selection
    $\theta_{k+1} \leftarrow \theta'_k$
    $\sigma_{k+1} \leftarrow \sigma_k e^{c_+}$
  else
    $\theta_{k+1} \leftarrow \theta_{k-1}$
    $\sigma_{k+1} \leftarrow \sigma_k e^{c_-}$
  end if
end for
return π_{θ_K}

6.2.3 Evolutionary Algorithms

A large class of optimization methods are inspired by the process of evolution, in which heritable characteristics of individuals in a population change over generations due to processes such as natural selection. The evolution leads to the adaptation of individuals, which means that they become better to live in their habitat.

By identifying a solution to an optimization problem as an individual in a population and the value of the function to be optimized for a particular solution as the fitness of that individual, we can form an evolutionary process to optimize the function. There are many variations in how we can do this. For example, there are family of methods called *Genetic Algorithms*, *Genetic Programming*, *Evolutionary Strategy*, etc.

Let us consider a simple algorithm called Evolutionary Strategy (ES)$(1 + 1)$, which is similar to RS, which is presented as Algorithm 6.2. Compared to RS, this algorithm proposes a new θ'_k by perturbing θ_k by a Gaussian noise around the current θ_k. The magnitude of perturbation is adaptive. If the new θ'_k is better than θ_k, the standard deviation increases, and vice versa.

To see why this is an evolutionary algorithm, let us identify its elements with the

usual components of an evolutionary process.

Each iteration k acts as a *generation* index. The parameter θ_k is the only individual in the population. It acts as a parent for the next generation. The parameter θ'_k is the offspring of θ_k, generated through an asexual process. The addition of noise η that perturbs θ_k in order to get θ'_k is interpreted as the mutation, in which the offspring's genetic code (θ'_k) is similar to its parent (θ_k) with the addition of some mutation. The mutation in ES is a random vector. In this particular variant, it is an isometric Gaussian with a standard deviation of σ_k.

The selection process is based on the competition between parent θ_k and its offspring θ'_k (so one can say that at some point we have two individuals in the population). Only one of them, however, has the chance of forming the next generation, and it can be either the parent or offspring.

A modification of this algorithm is called ES$(1, \lambda)$ with $\lambda > 1$ being an integer number. In that modification, the parent θ_k generates λ offsprings:

$$\theta'_{k,j} = \theta_k + \sigma_k \eta_j, \qquad j = 1, \dots, \lambda.$$

The competition would only be between the offsprings $\{\theta'_{k,j}\}_{j=1}^{\lambda}$, and not with the parent. Only one of the λ offsprings gets to the next generation. The ES$(1+\lambda)$ would be similar with the difference that the parent is also in the selection process, and if it is better than all its offsprings, it can form the next generation. More generally, we have ES(μ, λ) and ES$(\mu + \lambda)$ variants of ES, in which the population size is $\mu \in \mathbb{N}$ and there is a competition between either λ offsprings (ES(μ, λ)) or $\lambda + \mu$ offsprings and parents to form the next generation of μ individuals. There are various ways to decide how one should pick the best μ out of them.

ES does not have any sexual reproduction. This means that each offspring is not reproduced based on two (or more) parents, but is the result of a mutation of the genetic code of a single one. There are algorithms that have a recombination procedure that mimics the sexual reproduction, in which two parents (think of $\theta_{k,i}$ and $\theta_{k,j}$ from the same population) contribute to generation of an offspring. One such class of an algorithms is called Genetic Algorithms (GA). An example of such a recombination would be to select two parents $\theta_{k,i}$ and $\theta_{k,j}$, and reproduce the l-th offspring by

$$\theta'_{k+1,l} \leftarrow a_l \theta_{k,i} + (1 - a_l)\theta_{k,j} + \eta_l,$$

with $a_l \in (0, 1)$, possibly selected randomly. This is a convex combination of the parents, perturbed using a noise η_l.

Evolutionary algorithms can be quite complicated algorithms. Their performance is often evaluated only empirically. Analyzing them theoretically can be quite com-

plicated, and the current available results are limited to simple algorithms, such as ES$(1+1)$, which may not be the best performing ones in practice.

Studying evolutionary algorithms to solve RL problems is a niche area in the RL community, although with a growing interest in recent years. Sometimes they are not considered as a part of the RL research proper. Despite that, I briefly mentioned them because I consider it useful for an RL researcher to be familiar with the basic ideas in the evolutionary algorithms. Both evolution and learning have been crucial adaptation mechanisms to get to the point where we have smart enough species. Building AI agents with comparable capabilities to animals may require borrowing ideas from both learning and evolution, each performing at different time scale (within a life span of the agent vs. across generations of the agents).

6.3 First-Order Methods and the Policy Gradient

The availability of the gradient of $J_\rho(\pi_\theta)$ w.r.t. θ allows us to design first-order optimization methods that are potentially more efficient in finding an optimum of the performance compared to zero-order methods. It is not obvious, however, that the gradient computation is easy to achieve as the performance $J_\rho(\pi_\theta)$ depends on V^{π_θ}, which is not a simple function of π_θ. The value function is a complicated function of the policy, reward distribution $\mathcal{R}$ and the transition dynamics $\mathcal{P}$.

6.3.1 Finite Difference Approximation of the Policy Gradient

One may resort to a Finite Difference (FD) approximation of the policy gradient, which can be computed using the value of the performance objective itself.

Recall that given a function $f : \mathbb{R} \to \mathbb{R}$, the FD approximation of the derivative $f'(x) = \frac{\mathrm{d}f}{\mathrm{d}x}(x)$ is

$$f'_{\mathrm{FD}}(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x}, \tag{6.12}$$

where Δx is a small number. This is called the *forward difference* approximation. This is reasonable procedure. To see why, note that by the Taylor's theorem, assuming twice differentiability, we have

$$f(x+\Delta x) = f(x) + f'(x)\Delta x + f''(z)|_{x<z<x+\Delta x}\frac{(\Delta x)^2}{2!}.$$

Therefore,

$$f'(x) = \frac{f(x+\Delta x) - f(x)}{\Delta x} - f''(z)|_{x<z<x+\Delta x} \frac{(\Delta x)^2}{2!}.$$

Therefore, the error between the FD approximation (6.12) and $f'(x)$ is

$$\left| f''(z)\Big|_{x<z<x+\Delta x} \frac{(\Delta x)^2}{2!} \right|,$$

that is, $O((\Delta x)^2)$.

More accurate result can be obtained using the *central difference* approximation:

$$f'_{\text{FD}}(x) = \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}.$$

Exercise 6.1. *Show that the error of the central difference approximation is* $O((\Delta x)^3)$.

To compute the gradient of $J_\rho(\pi_\theta)$ w.r.t. $\theta \in \mathbb{R}^p$, we need to compute $2p$ evaluations of J_ρ:

$$\nabla_\theta J_\rho(\pi_\theta) \approx \nabla_\theta^{(\text{FD})} J_\rho(\pi_\theta) = \begin{bmatrix} \frac{J_\rho(\theta+\varepsilon e_1) - J_\rho(\theta-\varepsilon e_1)}{2\varepsilon} \\ \vdots \\ \frac{J_\rho(\theta+\varepsilon e_i) - J_\rho(\theta-\varepsilon e_i)}{2\varepsilon} \\ \vdots \\ \frac{J_\rho(\theta+\varepsilon e_p) - J_\rho(\theta-\varepsilon e_p)}{2\varepsilon} \end{bmatrix},$$

where e_i is a unit vector along dimension i of $\mathbb{R}^p$.

As we have seen before, we cannot directly compute $J_\rho(\pi_\theta)$. We can only compute $\hat{J}_n(\pi_\theta)$. So we need to replace each J_ρ above with their corresponding $\hat{J}_n$. This requires $2pn$ rollouts in total.

Given the approximated gradient, which has error caused by both the FD approximation and using $\hat{J}_n$ instead of J_ρ, we may use the gradient ascent to move towards higher value of $J_\rho(\pi_\theta)$:

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \nabla_\theta^{(\text{FD})} \hat{J}_n(\pi_{\theta_k}). \tag{6.13}$$

Even though this is a feasible approach, as we see next, we can compute the gradient more elegantly.

6.3.2 Policy Gradient for the Immediate Reward Problem

Suppose that we want to find a policy $\pi_\theta : \mathcal{X} \to \mathcal{M}(\mathcal{A})$ with $\theta \in \mathbb{R}^p$ that maximizes the performance for the immediate reward problem (see Sections 1.3 and 4.2). Recall that the interaction protocol is

- At episode t, $X_t \sim \rho \sim \mathcal{M}(\mathcal{X})$
- The agent chooses action $A_t \sim \pi_\theta(\cdot|X_t)$
- The agent receives reward $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$.
- The agent starts the new (independent) episode $t+1$.

This is an RL setting as ρ and $\mathcal{R}$ are not directly available to the agent, but only through samples.

The performance measure is

$$J_\rho(\pi_\theta) = \int V^{\pi_\theta}(x)\mathrm{d}\rho(x) = \int r^{\pi_\theta}(x)\mathrm{d}\rho(x) = \int r(x,a)\pi_\theta(a|x)\mathrm{d}\rho(x)\mathrm{d}a,$$

as the value function V^{π_θ} for the immediate reward problem is the same as r^{π_θ}. Here we consider the action space to be continuous and we assume that $\pi_\theta(\cdot|x)$ provides a density over the state space. If we had a discrete action space, we would instead have

$$\int_{\mathcal{X}} \mathrm{d}\rho(x) \sum_{a\in\mathcal{A}} r(x,a)\pi_\theta(a|x).$$

We may switch back and forth between continuous and discrete action spaces, if one of them provides better intuition in a specific context.

Let us compute the gradient of $J_\rho(\pi_\theta)$ w.r.t. θ:

$$\begin{aligned}\nabla_\theta J_\rho(\pi_\theta) &= \int r(x,a)\nabla_\theta\pi_\theta(a|x)\mathrm{d}\rho(x)\mathrm{d}a = \int \mathrm{d}\rho(x)\int r(x,a)\nabla_\theta\pi_\theta(a|x)\mathrm{d}a \\ &= \mathbb{E}_{X\sim\rho}\left[\int r(X,a)\nabla_\theta\pi_\theta(a|X)\mathrm{d}a\right]. \qquad (6.14)\end{aligned}$$

For discrete action spaces, the inner integral should be replaced $\sum_{a\in\mathcal{A}} r(x,a)\nabla_\theta\pi_\theta(a|x)$. We call $\nabla_\theta J_\rho(\pi_\theta)$ the *Policy Gradient* (PG).

If we can compute PG, we can update the policy parameters, for example, using a gradient ascent method:

$$\theta_{k+1} \leftarrow \theta_k + \alpha_k \nabla_\theta J_\rho(\pi_{\theta_k}), \qquad (6.15)$$

similar to what we have done using the FD approximation (6.13).

How can we compute this gradient? We build this gradually in several steps. At each step, we relax some assumptions until we get to a procedure that can use the data available by the interaction protocol above.

If we know ρ and r, we have all the necessary information for computing the gradient. For each $x \in \mathcal{X}$, we compute the summation (or integral) over all $a \in \mathcal{A}$ of $r(x,a)\nabla_\theta \pi_\theta(a|x)$. And then, we weigh that term proportional to $\rho(x)$ and take average over all x.[4] But this is not the RL setting described as the interaction protocol at the beginning of the section, as we do not know ρ or r, but only have sampled data from them.

If we assume that r is known, but ρ can only be sampled, we can approximately solve this problem by sampling $X_i \sim \rho$ $(i = 1, \dots, n)$ and computing

$$\frac{1}{n}\sum_{i=1}^{n}\left(\sum_{a\in\mathcal{A}} r(X_i,a)\nabla_\theta \pi_\theta(a|X_i)\right). \tag{6.16}$$

or $\frac{1}{n}\sum_{i=1}^{n}(\int r(X_i,a)\nabla_\theta \pi_\theta(a|X_i)\mathrm{d}a)$ (continuous). As $X_i \sim \rho$, this is an unbiased estimate of

$$\nabla_\theta J_\rho(\pi_\theta) = \mathbb{E}_{X\sim\rho}\left[\sum_{a\in\mathcal{A}} r(X,a)\nabla_\theta \pi_\theta(a|X)\right]$$

or $\mathbb{E}_{X\sim\rho}\left[\int r(x,a)\nabla_\theta \pi_\theta(a|x)\mathrm{d}a\right]$ (continuous).

This is still not feasible if r is unknown in our interaction protocol, which specifies that when the agent is initialized at state x, it has to choose its action according to $A \sim \pi_\theta(\cdot|x)$.

The term $\sum_{a\in\mathcal{A}} r(x,a)\nabla_\theta \pi_\theta(a|x)$ (or $\int r(x,a)\nabla_\theta \pi_\theta(a|x)\mathrm{d}a$ for continuous action space) can be interpreted as the expectation of $r(x,A)\nabla_\theta \pi_\theta(A|x)$ when A is coming from a uniform distribution with $q(a) = \frac{1}{|\mathcal{A}|}$ (for $a \in \mathcal{A}$) as

$$\sum_{a\in\mathcal{A}} r(x,a)\nabla_\theta \pi_\theta(a|x) = |\mathcal{A}|\sum_{a\in\mathcal{A}} q(a)r(x,a)\nabla_\theta \pi_\theta(a|x). \tag{6.17}$$

For the continuous case, we have

$$\int r(x,a)\nabla_\theta \pi_\theta(a|x) = \mathrm{Vol}(\mathcal{A})\int q(a)r(x,a)\nabla_\theta \pi_\theta(a|x)\mathrm{d}a, \tag{6.18}$$

[4] We are ignoring the computational challenge of doing this procedure if $\mathcal{X}$ or $\mathcal{A}$ are large (e.g., continuous) spaces. The emphasis is on what can be computed given the available information.

with $q(a) = \frac{1}{\mathrm{Vol}(\mathcal{A})}$.

If the actions were coming from a uniform distribution, we could easily form an empirical estimate of these terms. But the actions in the interaction protocol comes from distribution $\pi_\theta(\cdot|x)$, which in general is different distribution than a uniform one. Therefore, we have some form of off-policy sampling scenario in the distribution of actions.

There are a few ways to handle this though. One way is to estimate $r(x,a)$ by $\hat{r}(x,a)$ for all $(x,a) \in \mathcal{X} \times \mathcal{A}$, and use the estimate instead in the computation of the gradient. If $|\mathcal{X} \times \mathcal{A}| < \infty$, we can form an estimate of r as follows. Given $\mathcal{D}_n = (X_t, A_t, R_t)_{t=1}^n$ with $X_t \sim \rho$, $A_t \sim \pi_\theta(\cdot|X_t)$, and $R_t \sim \mathcal{R}(\cdot|X_t, A_t)$, we define

$$\hat{r}_n(x,a) = \frac{\sum_{i=1}^n R_i \mathbb{I}\{(X_i, A_i) = (x,a)\}}{\sum_{i=1}^n \mathbb{I}\{(X_i, A_i) = (x,a)\}}.$$

For (x,a) with $\rho(x) > 0$ and $\pi_\theta(a|x) > 0$, there is a non-zero probability of observing samples. In that case, as $n \to \infty$, $\hat{r}_n \to r$. We can then use

$$\frac{1}{n} \sum_{i=1}^n \left(\sum_{a \in \mathcal{A}} \hat{r}_n(X_i, a) \nabla_\theta \pi_\theta(a|X_i) \right), \tag{6.19}$$

as an estimate of the policy gradient.

This approach can be considered as a model-based approach, in which the model is estimated and used for the computation of some quantities, e.g., PG in this case.

Another approach, which turns out to be quite useful, is based on the observation that for a function $f : \mathbb{R} \to \mathbb{R}$, we have

$$\frac{\mathrm{d} \log f(x)}{\mathrm{d}x} = \frac{\frac{\mathrm{d}f}{\mathrm{d}x}(x)}{f(x)},$$

or more generally, for a function $f : \mathbb{R}^p \to \mathbb{R}$,

$$\nabla_x \log f(x) = \frac{\nabla_x f(x)}{f(x)}.$$

We use this identity to rewrite the term $\int r(x,a) \nabla_\theta \pi_\theta(a|x) \mathrm{d}a$ in (6.14) to

$$\begin{aligned} \int r(x,a) \nabla_\theta \pi_\theta(a|x) \mathrm{d}a &= \int r(x,a) \pi_\theta(a|x) \nabla_\theta \log \pi_\theta(a|x) \mathrm{d}a \\ &= \mathbb{E}_{A \sim \pi_\theta(\cdot|x)} \left[r(x,A) \nabla_\theta \log \pi_\theta(A|x) \right]. \end{aligned}$$

Here the desired quantity can be written as the expectation of $r(x, A)\nabla_\theta \log \pi_\theta(A|x)$ when $A \sim \pi_\theta(\cdot|x)$. Notice that the sampling distribution is the same as the one agent uses to choose its actions (i.e., π_θ), as opposed to the uniform distribution in the previous formulation (6.17), (6.18), in which the underlying distribution was uniform. Of course, the integrands differ. We are in the on-policy sampling scenario over the choice of actions.

Therefore, if $X \sim \rho$ and $A \sim \pi_\theta(\cdot|X)$, the random variable

$$r(X, A)\nabla_\theta \log \pi_\theta(A|X) \tag{6.20}$$

is an unbiased estimate of $\nabla_\theta J_\rho(\pi_\theta)$, i.e.,

$$\begin{aligned}\nabla_\theta J_\rho(\pi_\theta) &= \mathbb{E}\left[r(X, A)\nabla_\theta \log \pi_\theta(A|X)\right] \\ &= \mathbb{E}_{X\sim\rho}\left[\mathbb{E}_{A\sim\pi_\theta(\cdot|X)}\left[r(X, A)\nabla_\theta \log \pi_\theta(A|X) \mid X\right]\right].\end{aligned} \tag{6.21}$$

This unbiasedness means that we can estimate the gradient of the performance w.r.t. the parameters of the policy using data available through the interaction of the agent with its environment. We may use this estimate in (6.15) to update the policy parameters. The difference here, though, is that we are using an unbiased but noisy estimate of the gradient. This makes it a stochastic gradient ascent.[5]

For conciseness of our further discussions, let us denote

$$g(x; \theta) = \mathbb{E}_{A\sim\pi_\theta(\cdot|x)}\left[r(x, A)\nabla_\theta \log \pi_\theta(A|x)\right]. \tag{6.22}$$

The function $g : \mathcal{X} \times \Theta \to \mathbb{R}^p$ is the gradient of r^{π_θ} w.r.t. θ at state x, and is a p-dimensional vector.

Despite its unbiasedness, the p-dimensional r.v. (6.20), has variance due to two sources of randomness.

- The variance of estimating $g(x; \theta) = \mathbb{E}_{A\sim\pi_\theta(\cdot|X)}\left[r(X, A)\nabla_\theta \log \pi_\theta(A|X) \mid X = x\right]$ with a single sample $r(X, A)\nabla_\theta \log \pi_\theta(A|X)$.
- The variance of estimating $\mathbb{E}_{X\sim\rho}\left[g(X; \theta)\right]$ using a single sample.

One can show that the variance along the i-th dimension of this r.v. is

$$\mathrm{Var}\left[r(X, A)\frac{\partial \log \pi_\theta(A|X)}{\partial \theta_i}\right] = \mathbb{E}_{X\sim\rho}\left[\mathrm{Var}\left[r(X, A)\frac{\partial \log \pi_\theta(A|X)}{\partial \theta_i} \mid X\right]\right] + \mathrm{Var}_{X\sim\rho}\left[g_i(X; \theta)\right]. \tag{6.23}$$

[5] We may refer to this procedure by SGD, instead of SGA, even though we are moving in the ascent direction.

If we knew $r(x, a)$ and we could compute

$$g(x;\theta) = \mathbb{E}_{A\sim\pi_\theta(\cdot|x)}\left[r(X, A)\nabla_\theta \log \pi_\theta(A|X) \mid X = x\right] = \int r(x, a)\nabla_\theta\pi_\theta(a|x)\mathrm{d}a,$$

we wouldn't have the first source of variance, but we still would have the second one. In that case, the variance would be

$$\mathrm{Var}_{X\sim\rho}\left[g_i(X;\theta)\right]$$

These two sources of variance make our estimate of the gradient inaccurate. There are ways to reduce them, as we shall discuss in the rest of this section.

Exercise 6.2. *Prove the identity* (6.23).

6.3.2.1 Variance Reduction – Randomness of States

Suppose we can compute $g(x;\theta)$ exactly for any given $x \in \mathcal{X}$. The second source of variance can be reduced if instead of a single sample $g(X_1;\theta)$, we use multiple independent samples $X_1, \ldots, X_n$, all distributed according to ρ, to estimate the PG:

$$\begin{aligned}\nabla_\theta J_\rho(\pi_\theta) &\approx \frac{1}{n}\sum_{i=1}^{n} g(X_i;\theta)\\ &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{A\sim\pi_\theta(\cdot|X_i)}\left[r(X_i, A)\nabla_\theta \log \pi_\theta(A|X_i)\right].\end{aligned}$$

The variance of this estimator, along dimension i, is

$$\frac{1}{n}\mathrm{Var}_{X\sim\rho}\left[g_i(X;\theta)\right].$$

As $n \to \infty$, the variance goes to zero. This leads to more accurate estimate of the PG, hence more accurate update of the policy. Nonetheless, if we have a budget of total N interactions with the environment, and use n interactions to estimate the gradient, we can only perform $\frac{N}{n}$ policy updates using SGD (we use $t = 1, \ldots, n$ to compute an estimate of $\nabla_\theta J_\rho(\pi_{\theta_1})$; update the policy to get θ_2; we use $t = n + 1, \ldots, 2n$ to compute an estimate of $\nabla_\theta J_\rho(\pi_{\theta_2})$, and so on).

6.3.2.2 Variance Reduction – Randomness of Actions

Let us consider the variance of estimating $g(x;\theta)$ (6.22), the contribution to PG by state x, using a single sample $r(x,A)\nabla_\theta \log \pi_\theta(A|x)$ with $A \sim \pi_\theta(\cdot|x)$. One way to reduce this variance is based on the perhaps surprising observation that adding a state-dependent function $b : \mathcal{X} \to \mathbb{R}$ to $r(x,a)$ does not change the expectation. For each dimension i, we have

$$\begin{aligned}
\mathbb{E}\left[\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} b(x) \mid x\right] &= \int \pi_\theta(a|x) \frac{\partial \log \pi_\theta(a|x)}{\partial \theta_i} b(x) \mathrm{d}a \\
&= \int \frac{\partial \pi_\theta(a|x)}{\partial \theta_i} b(x) \mathrm{d}a \\
&= b(x) \int \frac{\partial \pi_\theta(a|x)}{\partial \theta_i} \mathrm{d}a \\
&= b(x) \frac{\partial}{\partial \theta_i} \underbrace{\int \pi_\theta(a|x) \mathrm{d}a}_{=1} = 0,
\end{aligned}$$

where in the last equation we benefitted from the fact that the integral of a distribution is always constant (equal to one), so its derivative w.r.t. θ_i is zero. This shows that

$$\mathbb{E}\left[\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} r(x,A) \mid x\right] = \mathbb{E}\left[\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} \left(r(x,A) + b(x)\right) \mid x\right]. \tag{6.24}$$

For each dimension i of the PG, we can use a different state-dependent function. Therefore, we can write that for any state-dependent function $b : \mathcal{X} \to \mathbb{R}^p$, the PG (6.21) is

$$\nabla_\theta J_\rho(\pi_\theta) = \mathbb{E}\left[r(X,A)\nabla_\theta \log \pi_\theta(A|X)\right] = \mathbb{E}\left[\left(r(X,A)\mathbf{1} + b(X)\right) \odot \nabla_\theta \log \pi_\theta(A|X)\right],$$

where $\mathbf{1}$ is a p-dimensional vector with all components equal to 1, and $\odot$ is a pointwise (Hadamard) product of two vectors, i.e., for $u, v \in \mathbb{R}^p$, $[u \odot v]_i = u_i v_i$. Of course, if we simply choose a scalar function b, which is often the case in practice, we have

$$\nabla_\theta J_\rho(\pi_\theta) = \mathbb{E}\left[\left(r(X,A) + b(X)\right) \nabla_\theta \log \pi_\theta(A|X)\right].$$

The function b is called the *baseline*. It can be used in order to minimize the variation of p-dimensional random vector. We may use the variance for this purpose. As the variance is defined for a scalar r.v., a not a multivariate r.v., we use the

summation of the variance of each dimension as our criteria. That is, we would like to find a function $b : \mathcal{X} \to \mathbb{R}^p$ such that for all $x \in \mathcal{X}$,

$$\min_b \sum_{i=1}^{p} \operatorname{Var}\left[(r(x,A) + b_i(x)) \frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} \mid x\right] = \operatorname{Tr} \operatorname{Cov}\left((r(x,A)\mathbf{1} + b(x)) \odot \nabla_\theta \log \pi_\theta(A|x) \mid x\right)$$

We have

$$\sum_{i=1}^{p} \operatorname{Var}\left[(r(x,A) + b_i(x)) \frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} \mid x\right] = \sum_{i=1}^{p} \mathbb{E}\left[(r(x,A) + b_i(x))^2 \left(\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i}\right)^2 \mid x\right] - \mathbb{E}\left[(r(x,A) + b_i(x)) \frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} \mid x\right]^2.$$

We take derivative w.r.t. $b(x)$ and make it equal to zero. Note that the second term (expectation of PG squared) is independent of the choice of $b(x)$, as shown in (6.24), so its derivate w.r.t. $b(x)$ is zero. For the first term, we have

$$\sum_{i=1}^{p} \frac{\partial}{\partial b(x)} \mathbb{E}\left[\left(r^2(x,A) + b_i^2(x) + 2r(x,A)b_i(x)\right) \left(\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i}\right)^2 \mid x\right] = 2\sum_{i=1}^{p} \mathbb{E}\left[(b_i(x) + r(x,A)) \left(\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i}\right)^2 \mid x\right] = 0.$$

As $b_i(x)$ is not a function of a, we can take it outside the expectation. The result is that the optimal baseline, in terms of the variance criteria, is

$$b_i(x) = \frac{-\mathbb{E}\left[r(x,A) \left(\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i}\right)^2 \mid x\right]}{\mathbb{E}\left[\left(\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i}\right)^2 \mid x\right]}. \tag{6.25}$$

We could choose a single scalar function $b : \mathcal{X} \to \mathbb{R}$ instead. In that case, the solution would be

$$b(x) = \frac{-\mathbb{E}\left[r(x,A) \left\|\nabla_\theta \log \pi_\theta(A|x)\right\|_2^2 \mid x\right]}{\mathbb{E}\left[\left\|\nabla_\theta \log \pi_\theta(A|x)\right\|_2^2 \mid x\right]}.$$

Exercise 6.3. *What is the variance of*

$$\mathrm{Var}\left[(r(x,A)+b_i(x))\,\frac{\partial \log \pi_\theta(A|x)}{\partial \theta_i} \mid x\right]$$

with the optimal choice of $b_i(x)$ (6.25)*?*

6.3.3 Policy Gradient for Continuing Tasks

We derive the PG for continuing tasks. The difference with the immediate reward case is that the performance $J_\rho(\pi_\theta)$ depends on the dynamics $\mathcal{P}^{\pi_\theta}$ too. As we change θ, the dynamics $\mathcal{P}^{\pi_\theta}$ changes as well. This seems to complicate the gradient computation. It turns out that despite this challenge, the PG can be written in an elegant, and relatively easy to compute, form.

In order to present our results more compactly, we introduce some notations. Recall from Definition 1.3 in Section 1.2 that $\mathcal{P}^\pi(\cdot|x;k) = \mathcal{P}^\pi(\cdot|x)^k$ is the probability distribution of following policy π for $k \geq 0$ steps. We introduce *discounted future-state distribution* of starting from $x \in \mathcal{X}$ and following π as

$$\rho^\pi_\gamma(\cdot|x) = \rho_\gamma(\cdot|x;\mathcal{P}^\pi) \triangleq (1-\gamma)\sum_{k\geq 0}\gamma^k \mathcal{P}^\pi(\cdot|x;k). \tag{6.26}$$

It is easy to verify that $\rho^\pi_\gamma(\cdot|x)$ is a valid probability distribution, e.g., $\rho^\pi_\gamma(\mathcal{X}|x) = 1$. The relevant of this distribution becomes more clear if we note that

$$\begin{aligned} V^\pi(x) = \mathbb{E}\left[\sum_{t\geq 0}\gamma^t R_t | X_0 = x\right] &= \sum_{t\geq 0}\gamma^t \mathbb{E}\left[R_t|X_0=x\right] = \sum_{t\geq 0}\gamma^t \int \mathcal{P}^\pi(\mathrm{d}x'|x;t) r(x') \\ &= \frac{1}{1-\gamma}\int \rho^\pi_\gamma(\mathrm{d}x'|x) r(x') = \frac{1}{1-\gamma}\mathbb{E}_{X'\sim\rho^\pi_\gamma(\cdot|x)}\left[r(X')\right]. \end{aligned}$$

That is, the value function at a state x is the expected reward when X' is distributed according to $\rho^\pi_\gamma(\cdot|x)$.

One interpretation of this distribution is that the agent starts from state x and at each time step, it decides to follow π with probability γ or terminates the episode with probability $1-\gamma$.

Based on this distribution, we can define discounted future-state distribution of starting from ρ and following π as

$$\rho^\pi_\gamma(\cdot) = \rho_\gamma(\cdot|\mathcal{P}^\pi) \triangleq \int \rho_\gamma(\cdot|x;\mathcal{P}^\pi)\mathrm{d}\rho(x).$$

The performance measure $J(\pi_\theta)$ (6.3) is then

$$J(\pi_\theta) = \mathbb{E}_{X\sim\rho}\left[V^{\pi_\theta}(X)\right] = \frac{1}{1-\gamma}\mathbb{E}_{X\sim\rho^\pi_\gamma}\left[r(X)\right].$$

Theorem 6.4 (Policy Gradient Theorem – Sutton et al. 2000)**.** *Assume that π_θ is differentiable w.r.t. $\theta \in \Theta$. We have*

$$\nabla_\theta J_\rho(\pi_\theta) = \sum_{k\geq 0}\gamma^k \int \mathrm{d}\rho(x)\mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x;k)\int \nabla_\theta\pi_\theta(a'|x')Q^{\pi_\theta}(x',a')\mathrm{d}a' \tag{6.27}$$

$$= \frac{1}{1-\gamma}\int \rho^{\pi_\theta}_\gamma(\mathrm{d}x)\int \pi_\theta(a|x)\nabla_\theta\pi_\theta(a|x)Q^{\pi_\theta}(x,a)\mathrm{d}a \tag{6.28}$$

$$= \frac{1}{1-\gamma}\mathbb{E}\left[\nabla_\theta \log\pi_\theta(A|X)Q^{\pi_\theta}(X,A)\right], \qquad \text{with } X\sim\rho^{\pi_\theta}_\gamma, A\sim\pi_\theta(\cdot|X). \tag{6.29}$$

Proof. We write the value function at state $x \in \mathcal{X}$ as the expected value of the action-value function, i.e., $V^{\pi_\theta}(x) = \int \pi_\theta(a|x)Q^{\pi_\theta}(x,a)\mathrm{d}a$. We take its derivative w.r.t. θ and use the product rule to get

$$\nabla_\theta V^{\pi_\theta}(x) = \int \left[\nabla_\theta\pi_\theta(a|x)Q^{\pi_\theta}(x,a) + \pi_\theta(a|x)\nabla_\theta Q^{\pi_\theta}(x,a)\right]\mathrm{d}a. \tag{6.30}$$

As $Q^{\pi_\theta}(x,a) = r(x,a) + \gamma\int \mathcal{P}(\mathrm{d}x'|x,a)V^{\pi_\theta}(x')$,

$$\nabla_\theta Q^{\pi_\theta}(x,a) = \gamma\int \mathcal{P}(\mathrm{d}x'|x,a)\nabla_\theta V^{\pi_\theta}(x').$$

This alongside with (6.30) gives us the recursive Bellman-like equation for the gradient of $V^{\pi_\theta}(x)$:

$$\nabla_\theta V^{\pi_\theta}(x) = \int \nabla_\theta\pi_\theta(a|x)Q^{\pi_\theta}(x,a)\mathrm{d}a + \gamma\int \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x)\nabla_\theta V^{\pi_\theta}(x'). \tag{6.31}$$

Expanding $\nabla_\theta V^{\pi_\theta}(x')$ likewise, we get that

$$\nabla_\theta V^{\pi_\theta}(x) = \int \nabla_\theta\pi_\theta(a|x)Q^{\pi_\theta}(x,a)\mathrm{d}a + \gamma\int \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x)\left[\nabla_\theta\pi_\theta(a'|x')Q^{\pi_\theta}(x',a')\mathrm{d}a' + \gamma\int \mathcal{P}^{\pi_\theta}(\mathrm{d}x''|x')\nabla_\theta V^{\pi_\theta}(x'')\right].$$

Following this pattern recursively, we get that

$$\nabla_\theta V^{\pi_\theta}(x) = \sum_{k\geq 0} \gamma^k \int \mathcal{P}^{\pi_\theta}(\mathrm{d}x'|x;k) \int \nabla_\theta \pi_\theta(a'|x') Q^{\pi_\theta}(x',a')\mathrm{d}a'$$
$$= \frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x'|x) \int \nabla_\theta \pi_\theta(a'|x') Q^{\pi_\theta}(x',a')\mathrm{d}a'.$$

Also since $\nabla_\theta \pi_\theta(a'|x') = \pi_\theta(a'|x')\nabla_\theta \log \pi_\theta(a'|x')$, we can write the gradient as

$$\nabla_\theta V^{\pi_\theta}(x) = \frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x'|x) \int \pi_\theta(a'|x') \nabla_\theta \log \pi_\theta(a'|x') Q^{\pi_\theta}(x',a')\mathrm{d}a'$$
$$= \frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x'|x) \mathbb{E}_{A'\sim\pi_\theta(\cdot|X')} \left[\nabla_\theta \log \pi_\theta(A'|X') Q^{\pi_\theta}(X',A')\right].$$

As $J_\rho(\pi_\theta) = \int V^{\pi_\theta}(x)\mathrm{d}\rho(x)$, taking the average of x w.r.t. ρ, we get that

$$\nabla_\theta J_\rho(\pi_\theta) = \frac{1}{1-\gamma} \int \rho_\gamma^{\pi_\theta}(\mathrm{d}x) \int \pi_\theta(a|x) \nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a)\mathrm{d}a$$
$$= \frac{1}{1-\gamma} \mathbb{E}_{\substack{X\sim\rho_\gamma^{\pi_\theta} \\ A\sim\pi_\theta(\cdot|X)}} \left[\nabla_\theta \log \pi_\theta(A|X) Q^{\pi_\theta}(X,A)\right],$$

which is the desired result. □

This theorem provides an elegant formula for the PG. It relates the PG to the discounted future-state distribution $\rho_\gamma^{\pi_\theta}$, the action-value function $Q^{\pi_\theta}(x,a)$, and the gradient of π_θ (6.28).

To compute the PG in the RL setting, we have to estimate it using samples. The formula (6.29) shows that if we get

- a state X sampled from $\rho_\gamma^{\pi_\theta}$,
- an action A sampled from $\pi_\theta(\cdot|X)$, and
- know action-value $Q^{\pi_\theta}(X,A)$,

the random variables

$$\nabla_\theta \log \pi_\theta(A|X) Q^{\pi_\theta}(X,A)$$

is an unbiased estimate of the PG (cf. (6.20)). We can then use it in an SGD scheme to improve the policy.

Sampling from $\rho_\gamma^{\pi_\theta}$ is relatively straightforward in the on-policy sampling scenario when the agent follows π_θ. The formula (6.27) perhaps makes it more clear how it

should be generated: The agent starts an episode from $X_0 \sim \rho$ and follows π_θ. We get a sequence of states $X_0, X_1, \dots$. These would be samples from $\int \mathrm{d}\rho(x)\mathcal{P}^{\pi_\theta}(\cdot|x;k)$ for $k = 0, 1, \dots$. The distribution $\rho_\gamma^{\pi_\theta}$, however, has a γ^k factor for the k-th time step, see (6.26). Its effect is that the contribution to the gradient from X_k, which is

$$\mathbb{E}\left[\nabla_\theta \log \pi_\theta(A|X) Q^{\pi_\theta}(X,A)\right] = \int \pi_\theta(a|x)\nabla_\theta \pi_\theta(a|x) Q^{\pi_\theta}(x,a)\mathrm{d}a,$$

should be weighted by γ^k. Another way to directly sample from $\rho_\gamma^{\pi_\theta}$ is to follow π, but at each step terminate the episode with probability $1-\gamma$.[6]

An action A sampled from $\pi_\theta(\cdot|X)$ is automatically generated when the agent follows policy π_θ (on-policy).

The remaining issue is the computation of $Q^{\pi_\theta}(X,A)$ for $X \sim \rho_\gamma^{\pi_\theta}$ and $A \sim \pi_\theta(\cdot|X)$ using data. This is essentially a PE problem, and we may use various action-value function estimators that we have developed so far.

A very simple approach is based on the MC estimate $Q^{\pi_\theta}(X,A)$. This would lead to what is known as the REINFORCE algorithm by Williams [1992].[7] In the on-policy setting when the agent follows π_θ, it generates the sequence $X_0, A_0, R_0, X_1, A_1, R_1, \dots$ with $A_t \sim \pi_\theta(\cdot|X_t)$. The return (1.16)

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

provides an unbiased estimate of $Q^{\pi_\theta}(X_t, A_t)$. So we may replace the action-value function at that state-action with this return from time t onward.

The return, however, is a high variance estimate of the action-value function. One approach to reduce the variance of this MC estimate is to use a baseline. Another approach is to use an action-value function estimator instead. For instance, we may use the TD method (and its various variants), LSTD, and Fitted Value Iteration (for PE, and not for Control – so the underlying Bellman operator would be T^{π_θ} instead of T^*) to estimate $\hat{Q}^{\pi_\theta}$ and use it instead of Q^{π_θ}. Such a method is called an *actor-critic* method, where the actor refers to the policy (and often PG method to improve it) and the critic refers to the value function estimator used to criticize the policy (actor). The use of a critic, however, may induce a bias as $\mathbb{E}\left[\hat{Q}^{\pi_\theta}(X,A)|X,A\right]$ may be different from $Q^{\pi_\theta}(X,A)$, especially if we use a TD method (which introduces bias because of bootstrapping) or a function approximator (for large state-action spaces). Such a method would explicitly represent both policy and value function.

[6] This is usually ignored in practice.

[7] REINFORCE stands for REward Increment × Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility.

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 [2011b].

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:

$$\mathcal{F}^{|\mathcal{A}|} = \left\{ (Q_1, \ldots, Q_{|\mathcal{A}|}) \,:\, Q_i \in \mathcal{F},\ i = 1, \ldots, |\mathcal{A}| \right\}.$$

We use $Q_j(x) = Q(x, j)\,(j = 1, \ldots, |\mathcal{A}|)$ 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 $(z_1, \ldots, z_n)$. 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

$$\|f\|_{p,z_{1:n}}^p = \|f\|_{p,\mathcal{D}_n}^p \triangleq \|f\|_{p,\nu_n}^p = \frac{1}{n} \sum_{i=1}^{n} |f(z_i)|^p. \tag{A.4}$$

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 Functional Analysis: Spaces, Operators, and 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 are. We briefly review some basic concepts from functional analysis. 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 book. 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:*

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^{d} |x_i|^p}, \qquad 1 \leq p < \infty,$$
$$\|x\|_\infty = \max_{i=1,\ldots,d} |x_i|.$$

This can be generalized to infinite sequences too.

Example A.4. *For $p \geq 1$, the sequence space ℓ_p is the set of all sequences $(x_i)_{i\geq 1}$ such that $\sum_{i\geq 1} |x_i|^p < \infty$. The norm is defined as*

$$\|x\|_p = \begin{cases} \sqrt[p]{\sum_{i=1}^{\infty} |x_i|^p}, & 1 \leq p < \infty, \\ \max_{i\geq 1} |x_i|. & p = \infty \end{cases}$$

Example A.5. *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, which is an infinite dimensional object, as opposed to for a finite dimensional vector.*

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

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

Using this norm, we can define a supremum-norm-based distance between two value functions V_1 and V_2 as $d_\infty(V_1, V_2) = \|V_1 - V_2\|_\infty$ (and similarly for the action-value functions).

We can also define

A.3.1 Operators

Operators, or mappings, are transformations from one linear space $\mathcal{Z}$ to another linear space $\mathcal{W}$. An operator $L : \mathcal{Z} \to \mathcal{W}$ is a *linear* operator when

$$L(c_1 z_1 + c_2 z_2) = c_1 L z_1 + c_2 L z_2,$$

for all $c_1, c_2 \in \mathbb{R}$ and $z_1, z_2 \in \mathcal{Z}$.

A simple example is the operator $L : \mathbb{R} \to \mathbb{R}$ defined as $Lz = az$ with $a \in \mathbb{R}$ and $z \in \mathbb{R}$. It is a mapping from the space of real numbers to the space of real numbers. Here both $\mathcal{Z}$ and $\mathcal{W}$ are $\mathbb{R}$.

A generalization of this is $L : \mathbb{R}^m \to \mathbb{R}^n$ (with m, n being integer numbers) defined as the mapping that takes $z \in \mathcal{Z} = \mathbb{R}^m$ and maps it to $w \in \mathcal{W} = \mathbb{R}^n$ with the i-th component of w being

$$w_i = \sum_{j=1}^{m} l_{i,j} z_j,$$

for $l_{i,j} \in \mathbb{R}$ for $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, m\}$. Of course, this is the same as matrix-vector multiplication that we can succinctly write as $w = Lz$ with L being a matrix with components $l_{i,j}$. These are examples of linear operators.

Not all operators are linear though. For example, the operator $L : \mathbb{R} \to \mathbb{R}$ defined as $Lz = az + b$ (with $a, b \in \mathbb{R}$) or $Lz = z^2$ or $Lz = \sin(z)$ or $Lz = \max\{0, z\}$ are not linear operators. We call them nonlinear operators. The first one is called *affine*.

The operators are not necessarily limited to finite dimensional linear spaces, but they can also be defined over the space of functions. An example is the linear integral operators. Consider a continuous function $k : [0, 1] \times [0, 1] \to \mathbb{R}$, and define $K : C([0, 1]) \to C([0, 1])$ (cf. Example A.5) as the operator that takes a continuous function $f \in C([0, 1])$ and returns another continuous function $Kf : [0, 1] \to \mathbb{R}$, whose value at $x \in [0, 1]$, that is $(Kf)(x)$, is

$$(Kf)(x) = \int_0^1 L(x, y) f(y) \mathrm{d}y.$$

Note the similarity of Kf with $\mathcal{P}f$ (Definition 1.4) and the Bellman operators applied to a value function, such as $T^\pi V$ or $T^* V$ (Definition 2.1). The Bellman operators, however, are not linear operators, but they are affine.

If the spaces $\mathcal{Z}$ and $\mathcal{W}$ are also equipped with norms, that is, we have $(\mathcal{Z}, \|\cdot\|_{\mathcal{Z}})$ and $(\mathcal{W}, \|\cdot\|_{\mathcal{W}})$, we can define a notion of operator norm based on the norm of these two spaces. First, we say that the operator L is bounded if there exists a finite $B < \infty$ such that for any $z \in \mathcal{Z}$, the norm of the output of the operator L, that is $w = Lz \in \mathcal{W}$, satisfies

$$\|Lz\|_{\mathcal{W}} \leq B \|z\|_{\mathcal{Z}} .$$

Note that as $w = Lz$ belongs to the space $\mathcal{W}$, its size should be measured with $\|\cdot\|_{\mathcal{W}}$.

We define the *operator norm* of L as

$$\|L\| = \inf \{ B \, : \, \|Lz\|_{\mathcal{W}} \leq B \|z\|_{\mathcal{Z}} \} . \tag{A.5}$$

This is equivalent to

$$\|L\| = \sup_{z \neq 0} \frac{\|Lz\|_{\mathcal{W}}}{\|z\|_{\mathcal{Z}}} = \sup_{\|z\|_{\mathcal{Z}}=1} \|Lz\|_{\mathcal{W}} . \tag{A.6}$$

The operator norm measures the maximum any input z can be expanded after going through the operator L and creating $z = Lw$.

An immediate consequence of this definition is that for any $z \in \mathcal{Z}$, we have

$$\|Lz\|_{\mathcal{W}} \leq \|L\| \|z\|_{\mathcal{Z}} . \tag{A.7}$$

Another useful property is that for two operators $L_1 : \mathcal{Z} \to \mathcal{W}$ and $L_2 : \mathcal{W} \to \mathcal{Y}$, whose joint operation $L_2 L_1 : \mathcal{W} \to \mathcal{Y}$, their norm is sub-multiplicative:

$$\|L_2 L_1\|_{\mathcal{W} \to \mathcal{Y}} \leq \|L_1\|_{\mathcal{Z} \to \mathcal{W}} \|L_2\|_{\mathcal{W} \to \mathcal{Y}} . \tag{A.8}$$

Here we used subscripts $\mathcal{Z} \to \mathcal{W}$ and $\mathcal{W} \to \mathcal{Y}$ to emphasize that these two operator norms are defined over different spaces. A further discussion of this XXX

A.3.2 Contraction Mapping

We are ready to define the contraction mapping/operator 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]

$$d(L(z_1), L(z_2)) \leq a d(z_1, z_2).$$

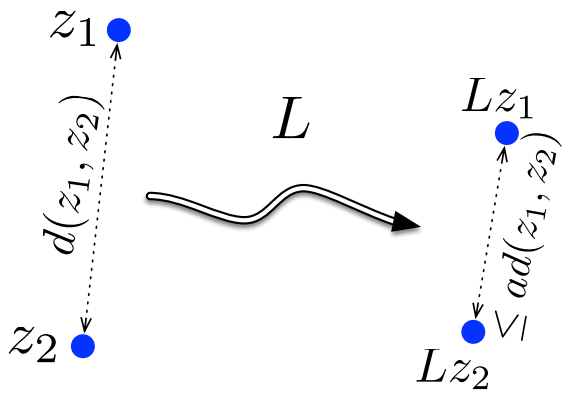

Figure A.1: Visualization of an a-contraction mapping L.

This is visualized in Figure A.1.

Example A.6. *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 $\|\cdot\|_p$ ($1 \le p \le \infty$, 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.

[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.

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

$$z_{k+1} = Lz_k, \qquad k = 0, 1, \ldots .$$

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.6). 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*.*

[3]There are various notions of stability in control theory. What we consider as stable is the same as globally exponentially stable.

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 $f(z) = 0$.

Example A.7. *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 of its Properties

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].

[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 vector induced p-norm of a matrix $A \in \mathbb{R}^{d\times d}$ is defined as

$$\|A\|_p = \sup_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p} = \sup_{\|x\|_p=1} \|Ax\|_p . \tag{A.9}$$

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. These are all examples of operator norm (A.6) for linear operator A between the normed spaces $(\mathcal{Z} = \mathbb{R}^d, \|\cdot\|_p)$ and $(\mathcal{W} = \mathbb{R}^d, \|\cdot\|_p)$, as we discussed in Appendix A.3.1.

We have the following identities for the matrix norms:

- $\|A\|_1 = \max_{1\leq j\leq d} \sum_{i=1}^{d} |a_{i,j}|$ (maximum of the sum over rows)
- $\|A\|_\infty = \max_{1\leq i\leq d} \sum_{j=1}^{d} |a_{i,j}|$ (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

$$\|\mathcal{P}^\pi\|_\infty = 1. \tag{A.10}$$

A useful property of any vector-induced p-norm is that for any $x \in \mathbb{R}^d$,

$$\|Ax\|_p \leq \|A\|_p \|x\|_p . \tag{A.11}$$

It is worth paying attention that these norms are semantically different: the norms $\|Ax\|_p$ and $\|x\|_p$ are the p-norms on the vector space $\mathbb{R}^d$, while $\|A\|_p$ is a matrix norm on the space of $\mathbb{R}^{d\times d}$ matrices. This result is essentially the same as (A.7).

Another useful property of the vector induced p-norms is that they are sub-multiplicative: For two matrices A and B, we have

$$\|AB\|_p \leq \|A\|_p \|B\|_p . \tag{A.12}$$

This is the analogous result to (A.8), specialized to matrices.

As an example of how these are relevant to the topic of this book, consider two policies π_1 and π_2, their policy induced transition kernels (matrices) $\mathcal{P}^{\pi_1}, \mathcal{P}^{\pi_2} \in$

$\mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$, and a value function $V \in \mathbb{R}^{|\mathcal{X}|}$, all for a finite state space $\mathcal{X} = \{x_1, \ldots, x_d\}$. The expression

$$\mathcal{P}^{\pi_1}\mathcal{P}^{\pi_2}V$$

is an $|\mathcal{X}|$-dimensional vector whose i-th component is the expected value of V for the agent starting from x_i and follows π_1 for the first step and π_2 for the second step. From (A.11) and (A.12), we have

$$\left\|\mathcal{P}^{\pi_1}\mathcal{P}^{\pi_2}V\right\|_p \leq \left\|\mathcal{P}^{\pi_1}\mathcal{P}^{\pi_2}\right\|_p \|V\|_p \leq \left\|\mathcal{P}^{\pi_1}\right\|_p \left\|\mathcal{P}^{\pi_2}\right\|_p \|V\|_p .$$

If $p = \infty$, by (A.10) we have $\left\|\mathcal{P}^{\pi_1}\right\|_\infty = \left\|\mathcal{P}^{\pi_2}\right\|_\infty = 1$, so overall, we get

$$\left\|\mathcal{P}^{\pi_1}\mathcal{P}^{\pi_2}V\right\|_\infty \leq \|V\|_\infty .$$

This is intuitive: the maximum value of the expectation of the value function V that the agent gets following any policies is not going to be larger than the maximum of V itself. If the norm was $p < 1$, this may not hold.

We also note that this argument is not limited to finite state problems, as we could use the properties of operator norms to get the same results for more general state spaces.

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

$$\left\|(\mathbf{I} - A)^{-1}\right\|_p \leq \frac{1}{1 - \|A\|_p}.$$

The consequence of this result for us is that we can write

$$(\mathbf{I} - \gamma\mathcal{P}^{\pi})^{-1} = \sum_{k\geq 0}(\gamma\mathcal{P}^{\pi})^k,$$

and conclude that

$$\left\|(\mathbf{I} - \gamma\mathcal{P}^{\pi})^{-1}\right\|_\infty \leq \frac{1}{1 - \gamma}.$$

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.

A.6 Concentration Inequalities

Consider $X_1, \ldots, X_n$ be independent real-valued random variables. Their average

$$S_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

is a random variable itself, and it tends to be *concentrated* arounds its expectation $\mathbb{E}[S_n]$. To see what this means, we provide a series of results that quantifies a notion of concentration.

First, for the simplicity of exposition, assume that all of X_i have the same mean μ and variance σ^2. By the linearity of the expectation, we have

$$\mathbb{E}[S_n] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[X_i] = \frac{1}{n}\mathbb{E}[\mu] = \mu.$$

By benefitting from the independence of X_i and X_j, we get that the variance of S_n is

$$\begin{aligned}
\mathrm{Var}\left[S_n\right] &= \mathbb{E}\left[(S_n - \mathbb{E}\left[S_n\right])^2\right] = \mathbb{E}\left[\left(\frac{1}{n}\sum_{i=1}^{n}(X_i-\mu)\right)^2\right] \\
&= \frac{1}{n^2}\mathbb{E}\left[\sum_{i,j=1}^{n}(X_i-\mu)(X_j-\mu)\right] \\
&= \frac{1}{n^2}\mathbb{E}\left[\sum_{i=1}^{n}(X_i-\mu)(X_i-\mu) + \sum_{i,j=1;i\neq j}^{n}(X_i-\mu)(X_j-\mu)\right] \\
&= \frac{1}{n^2}\left[\sum_{i=1}^{n}\sigma^2 + \sum_{i,j=1;i\neq j}^{n}\mathbb{E}\left[(X_i-\mu)(X_j-\mu)\right]\right] \\
&= \frac{\sigma^2}{n} + \frac{1}{n^2}\sum_{i,j=1;i\neq j}^{n}\mathbb{E}\left[(X_i-\mu)\right]\mathbb{E}\left[(X_j-\mu)\right] = \frac{\sigma^2}{n}.
\end{aligned}$$

This shows that as n increases, the variance of S_n decreases with a rate of $\frac{1}{n}$. Variance is a notion of dispersion of a random variable arounds its mean, so this result shows that S_n is increasingly more concentrated around μ.

We can use these results on the mean and variance of S_n to derive a high probability notion of concentration.

Recall the Markov's inequality, which states that for a non-negative random variables Z, for any $\varepsilon > 0$, we have

$$\mathbb{P}\left\{Z > \varepsilon\right\} \leq \frac{\mathbb{E}\left[Z\right]}{\varepsilon}. \tag{A.13}$$

This means that the probability that a non-negative r.v. Z is much larger than its expectation is decreasing. For instance, $\mathbb{P}\left\{Z > k\mathbb{E}\left[Z\right]\right\} \leq \frac{1}{k}$.

A direct consequence of the Markov's inequality, applied to the non-negative r.v. $Z = |S_n - \mu|^2$ is that

$$\mathbb{P}\left\{|S_n-\mu| > \varepsilon\right\} = \mathbb{P}\left\{|S_n-\mu|^2 > \varepsilon^2\right\} \leq \frac{\mathbb{E}\left[|S_n-\mu|^2\right]}{\varepsilon^2} = \frac{\mathrm{Var}\left[S_n\right]}{\varepsilon^2} = \frac{\sigma^2}{n\varepsilon^2}. \tag{A.14}$$

This shows that for any $\varepsilon > 0$, as $n \to \infty$,

$$\lim_{n\to\infty}\mathbb{P}\left\{|S_n-\mu| > \varepsilon\right\} \to 0.$$

This means that asymptotically, the probability that S_n is more than ε different from μ is zero, no matter how small ε is. This is the *convergence in probability* of S_n to μ. This result is known as the *weak Law of Large Number (LLN)*.

We also have the *strong* LLN, which states that

$$S_n \to \mu \quad \text{almost surely}$$

under mild assumptions, such as $\mathbb{E}[|Z_i|] < \infty$ for all i.

Both versions of LLN are about the convergence of S_n to μ, but they do not specify how different S_n from μ is. The statement (A.14) provides a rather loose upper bound on the deviation of S_n from μ. There are way to provide a tighter statements.

The first is the *Central Limit Theorem*, which states that as $n \to \infty$, the distribution of S_n converges to the Gaussian (normal) distribution with mean μ and variance $\frac{\sigma^2}{n}$, that is

$$S_n \xrightarrow{d} N\left(\mu, \frac{\sigma^2}{n}\right). \tag{A.15}$$

Here $\xrightarrow{d}$ denotes the convergence in distribution, which means that for any $t \in \mathbb{R}$

$$\lim_{n\to\infty} \mathbb{P}\left\{\frac{S_n - \mu}{\sigma\sqrt{n}} \le t\right\} \to \Phi(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-\frac{x^2}{2}} \mathrm{d}x.$$

Two remarks are worth mentioning. The first is that the CLT is an asymptotic result and holds exactly only when $n \to \infty$. When n is finite, say 100 or 10,000, the distribution of S_n is only approximately Gaussian. The second is that (A.15) shows that the tail behaviour of S_n is (approximately) like a Gaussian distribution, which means that

$$\mathbb{P}\{|S_n - \mu| > \varepsilon\} \approx 2\exp(-$$

The following result is a simplified form of Lemma 6.3 of Györfi et al. 2002, which itself is Theorem 2 of Hoeffding [1963].

Lemma A.3. *(Hoeffding's Inequality) Let $X_1, \ldots, X_n$ be independent real-valued random variables bounded by B almost surely, i.e., $|X_i| < B$. For any $\varepsilon > 0$, we have*

$$\mathbb{P}\left\{\left|\frac{1}{n}\sum_{i=1}^{n}(X_i - \mathbb{E}[X_i])\right| > \varepsilon\right\} \le 2\exp\left(-\frac{n\varepsilon^2}{2B^2}\right).$$

A.7 Information Theory

We provide a very brief overview of some definition from information theory, which are occasionally used in the book. For a more detailed treatment, including intuition about these concepts, refer to MacKay [2003]; Cover and Thomas [2006].

Given a discrete random variable X with probability distribution p, its entropy is defined as

$$\mathbb{H}[X] \triangleq \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{p(x)}. \tag{A.16}$$

The joint entropy of (X, Y) is defined similarly:

$$\mathbb{H}[X, Y] \triangleq \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} p(x, y) \log \frac{1}{p(x, y)}. \tag{A.17}$$

The conditional entropy of X given Y is defined as

$$\mathbb{H}[X|Y] \triangleq \sum_{(x,y) \in \mathcal{X} \times \mathcal{Y}} p(x, y) \log \frac{1}{p(x|y)}. \tag{A.18}$$

The KL divergence between distribution p and distribution q is defined as

$$\mathsf{KL}(p||q) = \int p(\mathrm{d}x) \log \frac{p(x)}{q(x)}. \tag{A.19}$$

The KL divergence is always non-negative. Whenever it is zero, it means that two distributions are the same, almost surely.

The mutual information between X and Y is

$$\mathbb{I}[X; Y] = \mathbb{H}[X|Y] - \mathbb{H}[X]. \tag{A.20}$$

The mutual information can also be written as

$$\mathbb{I}[X; Y] = \mathsf{KL}(p(x, y)||p(x)p(y)).$$

It can be shown than the mutual information is symmetric ($\mathbb{I}[X; Y] = \mathbb{I}[Y; X]$). It is also non-negative $\mathbb{I}[X; Y] \geq 0$, as a direct consequence of non-negativity of the KL divergence. When $\mathbb{I}[X; Y] = 0$, the X and Y are independent.

A.8 Algebraic Inequalities

In the theoretical analysis of ML and Statistics algorithms, in general, and in RL algorithms, in particular, we often go through some steps of upper bounding (or occasionally lower bounding) certain quantities. To prove those upper bounds, we use techniques that can be categorized, at the high level, as probabilistic arguments or algebraic ones. In many cases, to complete a proof, we need to use both types of arguments.

An example of the probabilistic argument is to benefit from the observation that a random variable, such as $G^\pi(x)$, is concentrated around its mean, which is $V^\pi(x) = \mathbb{E}\left[G^\pi(x)|x\right]$, and the probability of G^π being very different from the mean V^π is small, and becomes even smaller when we average multiple independent samples from that random variable. We discuss this in Appendix A.6.

An example of the algebraic one is to show that if r is $R_{\max}$-bounded and Q is $Q_{\max} = \frac{R_{\max}}{1-\gamma}$-bounded, we also have $T^\pi Q = r + \gamma \mathcal{P}^\pi Q$ is $(1 + \frac{\gamma}{1-\gamma}) = 1$-bounded, hence the Bellman error $\|Q - T^\pi\|_2^2$ is also $(2Q_{\max})^2$-bounded. Many of the algebraic proofs use inequalities, which we briefly summarize here.

We review Cauchy-Schwarz, Hölder, and Jensen inequalities. They all appear in different forms, such as an inequality for a finite dimensional vector, an infinite sequence, functions, or random variables. They can all be unified with the appropriate selection of the function space (and measure), but we present them separately. Considering that all of them are

In the following, let $u, v \in \mathbb{R}^d$ with $u = (u_1, \dots, u_d)$ and $v = (v_1, \dots, v_d)$; let $a = (a_1, a_2, \dots)$ and $b = (b_1, b_2, \dots)$ be infinite sequences, and $f, g : \mathcal{X} \to \mathbb{R}$ be functions. XXX

Lemma A.4.

- *For finite dimensional vectors $u, v \in \mathbb{R}^d$, we have*

$$\sum_{i=1}^{d} u_i v_i \le \sqrt{\sum_{i=1}^{d} |u_i|^2} \sqrt{\sum_{i=1}^{d} |v_i|^2},$$

  or more compactly, $\langle u, v \rangle \le \|u\|_2 \|v\|_2$, with the norms defined as in Example A.3.

- *The same holds for sequences $u = (u_1, u_2, \dots)$ and $v = (v_1, v_2, \dots)$ that satisfy $\|u\|_2, \|v\|_2 < \infty$: $\langle u, v \rangle \le \|u\|_2 \|v\|_2$ (cf. Example A.4).*

- *For functions $f, g : \mathcal{X} \to \mathbb{R}$ with $\int f^2(x)\mathrm{d}x$ and $\int g^2(x)\mathrm{d}x$ both being finite, we have*

$$\int f(x)g(x)\mathrm{d}x \leq \sqrt{\int |f(x)|^2\mathrm{d}x}\sqrt{\int |g(x)|^2\mathrm{d}x}.$$

- *If the random variables X and Y satisfy $\mathbb{E}\left[|X|^2\right], \mathbb{E}\left[|Y|^2\right] < \infty$, we have*

$$\mathbb{E}\left[XY\right] \leq \mathbb{E}\left[|XY|\right] \leq \mathbb{E}\left[|X|^2\right]^{\frac{1}{2}} \mathbb{E}\left[|Y|^2\right]^{\frac{1}{2}}.$$

  This can be written as $\langle X, Y \rangle \leq \|XY\|_1 \leq \|X\|_2 \|Y\|_2$ (cf. (A.1)).

- *Given a random variables $X \sim \nu$, and two functions $V, U : \mathcal{X} \to \mathbb{R}$ that have finite second order moments $\mathbb{E}\left[|V(X)|^2\right], \mathbb{E}\left[|U(X)|^2\right] < \infty$, we have*

$$\langle U, V \rangle_\nu \leq \|V\|_{2,\nu} \|U\|_{2,\nu},$$

  with inner product and norm defined as in (A.1) and (A.2).

- *We have*

$$\sum_{i=1}^{d} u_i v_i \leq \sqrt{\sum_{i=1}^{d} |u_i|^2}\sqrt{\sum_{i=1}^{d} |v_i|^2} \quad \textit{or more compactly} \langle u, v \rangle \leq \|u\|_2 \|v\|_2,$$

$$\int f(x)g(x)\mathrm{d}x \leq \sqrt{\int |f(x)|^2\mathrm{d}x}\sqrt{\int |g(x)|^2\mathrm{d}x}$$

$$\mathbb{E}\left[XY\right] \leq \mathbb{E}\left[|X|^2\right]^{\frac{1}{2}} \mathbb{E}\left[|Y|^2\right]^{\frac{1}{2}}$$

Bibliography

Pierre-Luc Bacon. *Temporal Representation Learning*. PhD thesis, McGill University, 2018. 96

Pierre-Luc Bacon and Doina Precup. A matrix splitting perspective on planning with options. In *NIPS Continual Learning and Deep Networks Workshop*. 2016. 96

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957. 95

Richard Bellman. *Eye Of The Hurricane*. World Scientific, 1984. 95

Vasile Berinde. *Iterative approximation of fixed points*, volume 1912. Springer, 2007. 194, 196

Dimitri P. Bertsekas. *Abstract dynamic programming*. Athena Scientific Belmont, 2nd edition, 2018. ii, 7, 58, 65, 70, 84, 90, 95

Dimitri P. Bertsekas and Steven E. Shreve. *Stochastic Optimal Control: The Discrete-Time Case*. Academic Press, 1978. 7

Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996. ii, 7, 18, 94, 120

B J Casey, Leah H Somerville, Ian H Gotlib, Ozlem Ayduk, Nicholas T Franklin, Mary K Askren, John Jonides, Marc G Berman, Nicole L Wilson, Theresa Teslovich, Gary Glover, Vivian Zayas, Walter Mischel, and Yuichi Shoda. Behavioral and neural correlates of delay of gratification 40 years later. *Proceedings of the National Academy of Sciences of the United States of America*, 108(36): 14998–15003, 2011. 30

Paul F. Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 30

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2nd edition, 2006. 202

Amir-massoud Farahmand. Action-gap phenomenon in reinforcement learning. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS - 24)*, pages 172–180. Curran Associates, Inc., 2011a. 70

Amir-massoud Farahmand. *Regularization in Reinforcement Learning*. PhD thesis, University of Alberta, 2011b. 187

Amir-massoud Farahmand, Saleh Nabi, Piyush Grover, and Daniel N. Nikovski. Learning to control partial differential equations: Regularized fitted Q-iteration approach. In *IEEE Conference on Decision and Control (CDC)*, pages 4578–4585, December 2016. 29

Amir-massoud Farahmand, Saleh Nabi, and Daniel N. Nikovski. Deep reinforcement learning for partial differential equation control. In *American Control Conference (ACC)*, 2017. 29

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Remi Munos. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2024. 30

Peter Godfrey-Smith. *Metazoa: Animal life and the Birth of the Mind*. Farrar, Straus and Giroux, 2020. 29

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The John Hopkins University Press, 4th edition, 2013. 79, 80, 196, 198, 199

László Györfi, Michael Kohler, Adam Krzyżak, and Harro Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer Verlag, New York, 2002. 148, 201

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. 96

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. 201

Ronald A. Howard. *Dynamic programming and markov processes*. John Wiley, 1960. 96

De-An Huang, Amir-massoud Farahmand, Kris M Kitani, and J. Andrew Bagnell. Approximate MaxEnt inverse optimal control and its application for mental simulation of human interactions. In *AAAI Conference on Artificial Intelligence*, January 2015. 30, 31

John K. Hunter and Bruno Nachtergaele. *Applied analysis*. World Scientific Publishing Company, 2001. 189, 190, 193, 196

Kris N. Kirby. One-year temporal stability of delay-discount rates. *Psychonomic Bulletin & Review*, 16(3):457–462, 2009. 30

Harold J. Kushner and Allan J. Kleinman. Accelerated procedures for the solution of discrete Markov control problems. *IEEE Transactions on Automatic Control*, 16(2):147–152, April 1971. 96

David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. 202

Sean Meyn. *Control Systems and Reinforcement Learning*. Cambridge University Press, 2022. ii

Sean Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability*. Cambridge University Press, New York, NY, USA, 2009. 153

Walter Mischel, Ebbe B. Ebbesen, and Antonette Raskoff Zeiss. Cognitive and attentional mechanisms in delay of gratification. *Journal of personality and social psychology*, 21(2):204, 1972. 30

James R. Munkres. *Topology*. Pearson Modern Classic, 2nd edition, 2018. 58

Andrew Y. Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2000. 30, 31

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 30

Yangchen Pan, Amir-massoud Farahmand, Martha White, Saleh Nabi, Piyush Grover, and Daniel Nikovski. Reinforcement learning with function-valued action

spaces for partial differential equation control. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80, pages 3986–3995, Jul 2018. 29

Evan L. Porteus. Bounds and transformations for discounted finite markov decision chains. *Operations Research*, 23(4):761–784, 1975. 96

Amin Rakhsha, Andrew Wang, Mohammad Ghavamzadeh, and Amir-massoud Farahmand. Operator splitting value iteration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 96

Deepak Ramachandran and Eyal Amir. Bayesian inverse reinforcement learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2586–2591, 2007. 30

Jeffrey S. Rosenthal. *A Fist Look at Rigorous Probability Theory*. World Scientific Publishing, 2nd edition, 2006. 187

Stuart Russell. Learning agents for uncertain environments. In *Annual conference on Computational Learning Theory (COLT)*, pages 101–103, 1998. 30, 31

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics (SIAM), 2nd edition, 2003. 79

Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. *Mathematics of Operations Research*, 41(3):758–774, 2016. 96

Ingo Steinwart and Andreas Christmann. *Support Vector Machines*. Springer, 2008. 148

Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1988. 23

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. ii, 7, 77, 95, 159

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NIPS - 12)*, 2000. 184

Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan Claypool Publishers, 2010. ii, 7, 95

John N. Tsitsiklis. On the convergence of optimistic policy iteration. *Journal of Machine Learning Research (JMLR)*, 3(1):59–72, 2002. 114

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. 158, 159

Sara A. van de Geer. *Empirical Processes in M-Estimation*. Cambridge University Press, 2000. 148

Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, University of Cambride, 1989. 23

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992. 186

Christian Wirth, Riad Akrour, Gerhard Neumann, and Johannes Fürnkranz. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research (JMLR)*, 18(136):1–46, 2017. 30

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. 96

Brian D. Ziebart, J. Andrew Bagnell, and Anind K. Dey. The principle of maximum causal entropy for estimating interacting processes. *Information Theory, IEEE Transactions on*, 59(4):1966–1980, April 2013. ISSN 0018-9448. 30, 31