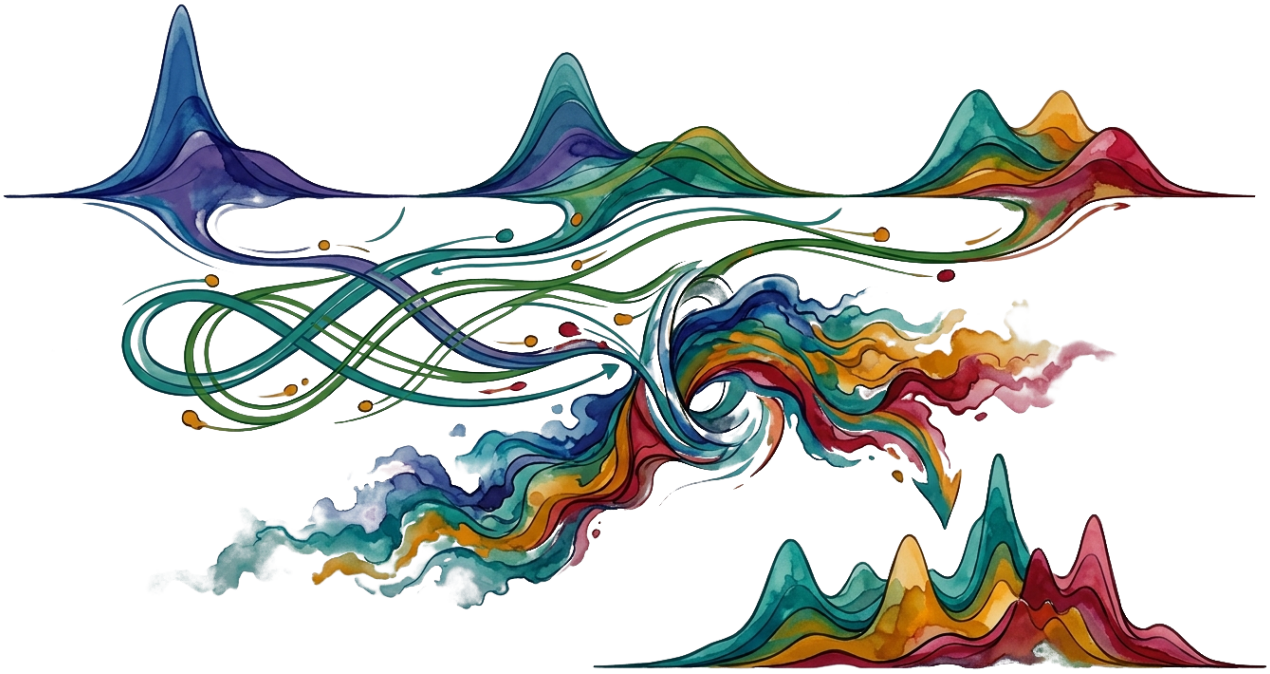


Risk-aware Reinforcement Learning

a Distributional Approach

Aurélien Garivier

ENS de Lyon

Companion to the Master 2 course *Reinforcement Learning* (CR15), ENS de Lyon — 2026

Typeset by the author with LaTeX. All figures were produced by the scripts of the `figures/scripts` folder.

The drawing on the title page was generated with an image-generation model from the author's description, and reworked by the author.

Preface

This book accompanies the Master 2 course *Reinforcement Learning* taught at ENS de Lyon. It grew out of lecture notes and was written with one conviction: the natural object of reinforcement learning is not a number but a probability distribution. An agent acting in a random environment collects a random cumulated reward—the *return*—and everything one may wish to know about a policy (its average performance, its variability, the risk of a disastrous outcome, the chance of an exceptional one) is a functional of the return's law. Classical reinforcement learning studies the expectation of the return; the expectation is linear, and linearity is what makes dynamic programming possible. The conviction is not this book's own: it is the programme of distributional reinforcement learning. We take it seriously and ask, throughout, three questions. Which statistics of the return can be propagated exactly by Bellman-type recursions? Which functionals can be optimized exactly? And what can be done, at what price, for the others? The answers to the first two are known; the third is the subject of most of the research the book reports on.

The answers organize the book. Part I sets up Markov decision processes, Monte Carlo estimation and finite-horizon dynamic programming, stating from the outset the Bellman equation as an identity between return *distributions*. Part II characterizes the statistics that are closed under this recursion—polynomial and exponential moments—shows that only generalized means can be optimized exactly, and builds from exponential moments a theory of risk: the entropic β-means and the entropic value at risk. Part III moves to the discounted, infinite-horizon setting, to learning from samples, to approximation and to policy gradient methods—by way of the average-reward criterion and the Poisson equation—always keeping the distributional recursion in view. Part IV opens onto current research: exploration, regularization, learning from demonstrations and from human feedback, adversarial environments and games, and the multi-agent and mean-field limits in which the population distribution becomes the state.

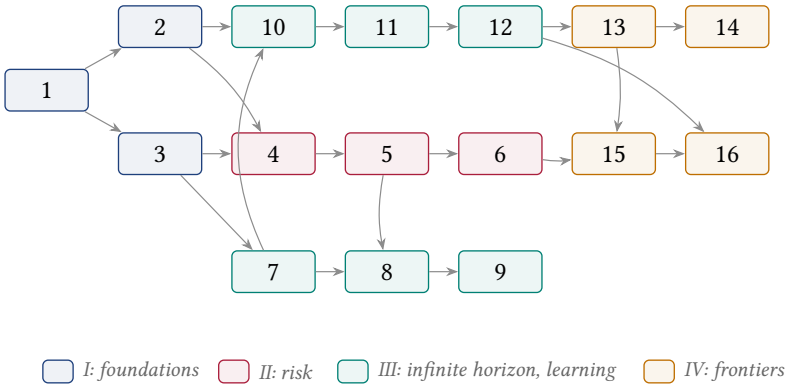


Figure 1. How the chapters depend on one another: an arrow means that the target uses results of the source. Chapters 1–3 are the common foundation, 4–6 develop the distributional and risk theory in finite horizon, 7–9 move to the infinite horizon, 10–12 are learning, and 13–16 are the research frontiers. A reader interested only in risk can follow 1, 3, 4, 5, 6; one interested only in algorithms, 1, 3, 7, 8, 10, 11, 12.

Each chapter corresponds to one lecture. It starts from a concrete example, states and proves the results that form the core of the theory, describes the corresponding algorithms in pseudo-code, and ends with exercises, pointers to the hands-on sessions of the course, and bibliographical notes.

Hints and solutions to the exercises are collected in Appendix F; the appendices before it gather the probabilistic (Appendix A), concentration (Appendix B), convex-analytic (Appendix C) and metric (Appendix D) facts used in the text, and Appendix E is a table of the notation. Proofs are complete for foundational results; for research-level results we state the theorems precisely, sketch the argument and refer to the original papers.

The prerequisites are a first course in probability (conditional expectation, laws of large numbers, Markov chains), linear algebra, and the rudiments of convex analysis; the appendices gather what is needed. Some familiarity with Python is useful for the hands-on sessions.

Acknowledgements

This book grew out of the Master 2 course *Reinforcement Learning* (CR15) at ENS de Lyon, and it owes most to the people who made that course and the research behind it.

My interest in reinforcement learning, and the way I understand it, come from many years of joint work with more collaborators than can be named here. Two of them matter particularly to this book. Juliet Bringas-Miranda, my doctoral student, is the one with whom my interest in the distributional point of view really developed. The doctoral work of Alexandre Marthe provides the backbone of Chapters 4 to 6: the characterization of the statistics that dynamic programming propagates exactly and of the criteria it optimizes exactly, the entropic optimality front with the algorithms FindBreaks and Dolfin, and the concentration of the plug-in β-mean (Chapter 2) are his, in joint work with Claire Vernade; his thesis (Marthe, 2026) gives the proofs in full and should be read alongside those chapters.

I owe a particular debt to my long-standing colleagues Emilie Kaufmann, Olivier Cappé and Claire Vernade.

The students of CR15 at ENS de Lyon are the book's first readers and its most exacting ones. Several of the examination problems they were set have become material here: the functional-equation proof of Theorem 4.8, and the median and third-moment counterexamples of Chapter 5. Their questions and feedback are responsible for a good part of what the text bothers to explain.

The remaining errors are mine.

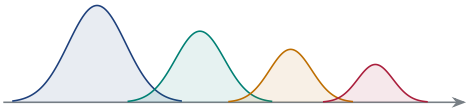
Contents

Part I

Foundations

1

Sequential Decisions and Markov Decision Processes



> *Reinforcement learning is about acting in an uncertain, evolving environment so as to collect rewards. This chapter sets the stage. It surveys what reinforcement learning is used for, from inventory control to games, robotics, medicine and language models; introduces Markov decision processes through a handful of examples that are miniatures of these applications; defines policies and the probability law they induce on trajectories; and singles out the object around which the whole book revolves: the* distribution *of the cumulated reward. It closes with the three problems studied afterwards—policy evaluation, planning, and learning.*

1.1 From prediction to decision

Machine learning is often presented through two archetypal problems. In *supervised learning*, one observes pairs $(X_1, Y_1), \dots, (X_n, Y_n)$ of inputs and outputs and looks for a decision rule $x \mapsto \hat{y}(x)$ that predicts the output of a new input: recognizing digits, translating sentences, forecasting a demand. In *unsupervised learning*, one observes inputs only and looks for structure in them: clusters, low-dimensional representations, densities. In both cases the learner is a spectator. It observes data that it did not influence, and its decisions have no effect on what it will observe next.

Reinforcement learning is about an agent that *acts*. At each time step it observes the state of an environment, chooses an action, receives a numerical reward, and the environment moves to a new state whose law depends on the state and on the action. The agent wants the cumulated reward to be large. Two features make this problem genuinely different from prediction. First, the feedback is *evaluative* rather than *instructive*: nobody tells the agent what the right action was, only how good the chosen one turned out to be. Second, actions have *delayed consequences*: a choice made now changes the states visited later and hence the rewards available later, so that credit for a good outcome must be distributed over a whole sequence of decisions.

The mathematical model for this interaction, sketched in Figure 1.1, is the *Markov decision process*. Before defining it, let us look at the kind of problems it is meant to capture.

1.2 What reinforcement learning is used for

Reinforcement learning has two lineages. One comes from the mathematics of sequential decisions: dynamic programming, developed by Bellman (1952) and Bellman (1957b) as a general method for multistage decision problems, and Markov decision processes, whose theory took shape in the 1950s

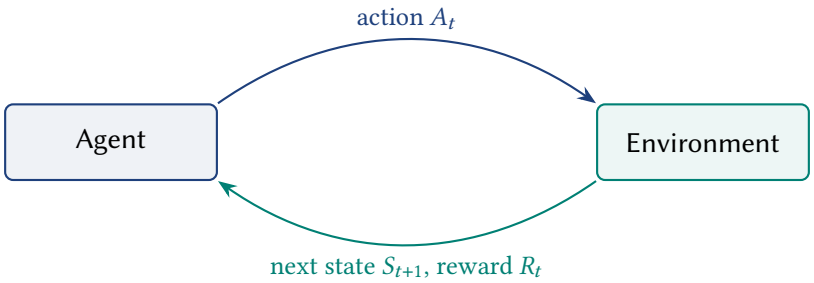


Figure 1.1. The interaction loop of reinforcement learning. At time t the agent observes S_t, chooses A_t, and the environment responds with a random next state and a reward.

(Shapley, 1953; Bellman, 1957b) and was popularized, with policy iteration, by Howard (1960). The other comes from psychology and the study of animal learning: the law of effect of Thorndike (1911), according to which actions followed by satisfaction are repeated, and the conditioning models of Rescorla and Wagner (1972), which describe how the value attached to a stimulus is corrected by the difference between what was expected and what was received. The modern framework emerged in the 1980s from the synthesis of these ideas with earlier computational precursors such as Samuel's checker player: the actor–critic architecture of Barto et al. (1983), the formalization of *temporal-difference learning* by Sutton (1988), and Q-learning, by which Watkins (1989) connected learning from experience to the optimality equations of dynamic programming. The applications reviewed below draw on both lineages: they are decision problems, and they are solved by learning from interaction.

1.2.1 Operations research and the control of physical systems

The oldest applications are the ones that motivated Bellman. *Inventory management* asks how much to order when demand is random and storage is costly (Arrow et al., 1951); *equipment maintenance* asks when to repair or replace a machine whose condition deteriorates randomly (Derman, 1963); *revenue management* asks how to price and allocate a perishable inventory of seats or rooms, unsold at departure and thus worthless (Talluri and van Ryzin, 2004); queueing, scheduling and routing problems have the same structure. In the classical versions of these problems the model is known or well estimated and the state space is moderate, so that dynamic programming solves them exactly, and the structure of the solution—order up to a target level, replace beyond a wear threshold, accept a booking above a price threshold—is often as valuable as its numerical value; large instances, in contrast, call for the approximation methods of Chapter 11. Randomness is not a nuisance here but the heart of the matter: an inventory policy is often judged by its service level—how rarely it runs out of stock—as much as by its average profit, and this is a first reason to look at the whole distribution of the outcome.

The control of physical systems is the engineering counterpart. Linear-quadratic control (Kalman, 1960) steers aircraft, satellites and chemical plants; its finite-horizon version is the rocket example of Chapter 3. Recent years have seen the same ideas, combined with learned models, applied to systems that resist classical modeling: the cooling of data centers by model-predictive control with a linear model identified from a few hours of safe exploration (Lazic et al., 2018), or the management of electricity demand and storage in smart grids by reinforcement learning (Vázquez-Canteli and Nagy, 2019). Heating a building (Exercise 3.6) is the domestic version of these problems.

1.2.2 Games

Games have been the showcase of reinforcement learning since Samuel (1959) taught a computer to play checkers by playing against itself and adjusting its evaluation function. The method that took the field by storm was TD-Gammon (Tesauro, 1995): a neural network evaluating backgammon

positions, trained by temporal-difference learning on millions of self-played games, reached the level of the best human players. Two decades later, Mnih et al. (2015) trained a single architecture—a deep Q-network fed with raw pixels—to play dozens of Atari video games at human level, using the Arcade Learning Environment of Bellemare et al. (2013) as a testbed. The game of Go, long considered out of reach, fell to AlphaGo (Silver et al., 2016), which combined Monte Carlo tree search with policy and value networks, pre-trained on human games and then improved by self-play; its successors AlphaGo Zero and AlphaZero learned Go, chess and shogi from the rules alone (Silver et al., 2017; Silver et al., 2018), and MuZero learned without being given the rules of the game, planning in a learned model (Schrittwieser et al., 2020). Games of imperfect information followed: DeepStack (Moravčík et al., 2017), Libratus and Pluribus beat professional poker players in two-player and then six-player no-limit Texas hold'em (Brown and Sandholm, 2018; Brown and Sandholm, 2019), building on the counterfactual regret minimization that we shall meet in Chapter 15, combined with real-time search; DeepNash reached expert human level at Stratego (Perolat et al., 2022), and AlphaStar reached grandmaster level in the real-time strategy game StarCraft II through a league of agents trained against one another (Vinyals et al., 2019). Closer to physics, Gran Turismo Sophy outraced champion drivers in a realistic racing simulator (Wurman et al., 2022)—and its learning algorithm, a quantile-regression actor–critic, was *distributional*: it estimated the whole distribution of future rewards rather than their mean.

Why games? Because they offer what learning needs most: a perfect simulator, hence unlimited experience at no cost and no risk; an unambiguous reward; and, in self-play, an opponent that improves with the agent. These conditions are rarely met outside games, and the transfer of these methods to the real world is precisely what makes the rest of this list difficult.

1.2.3 Robotics, engineering and the sciences

Robots learn in simulation and act in the real world. Legged robots have learned agile and robust locomotion over rough terrain from policies trained entirely in simulation and then transferred to the machine (Hwangbo et al., 2019; Lee et al., 2020); a quadrotor trained by reinforcement learning beat human champions at drone racing (Kaufmann et al., 2023); a robotic hand learned to manipulate a Rubik's cube (Akkaya et al., 2019); visuomotor policies map camera images directly to motor commands (Levine et al., 2016). A central bottleneck, along with safety and sample efficiency, is the *sim-to-real gap*: a policy that is optimal in the simulator may fail on the physical system, and the variability of its performance under perturbations matters as much as its average. The survey of Kober et al. (2013) discusses these challenges.

Two large-scale engineering successes deserve mention. The stratospheric balloons of Project Loon kept station over a target area by learning to exploit winds at different altitudes; the controller was trained by a distributional variant of Q-learning based on quantile regression, in an environment whose randomness—the wind field—is only partially known and forecast with errors (Bellemare et al., 2020). And the plasma of the TCV tokamak was shaped and stabilized by a learned controller acting directly on the magnetic control coils, trained through simulated interactions only, the simulator being calibrated on experimental data, and transferred to the machine without further tuning (Degrave et al., 2022).

In the sciences, reinforcement learning has been used as a search method for hard combinatorial problems cast as single-player games: AlphaTensor discovered faster algorithms for matrix multiplication (Fawzi et al., 2022) and AlphaDev found faster sorting routines that were incorporated in a standard library (Mankowitz et al., 2023), both by tree search guided by learned value functions, with a reward tied to the length and quality of the algorithm found—a cost of one per step, plus a terminal penalty for what remains undone. AlphaTensor is, again, a distributional success: its value network predicted quantiles of the return, and the search used a *risk-seeking* value, the mean of the upper quantiles, to hunt for rare exceptional algorithms rather than good average ones. Policy-gradient

methods placed the components of computer chips (Mirhoseini et al., 2021) and trained generative models that propose new drug candidates (Popova et al., 2018).

1.2.4 Decisions at scale: recommendation, advertising, logistics

Online services make billions of small decisions a day: which article to display, which advertisement to show, which driver to dispatch. The feedback is partial—typically only the outcome of the chosen action is observed—and the environment changes constantly. Many of these problems are *bandit* problems, a model introduced by Thompson (1933) for medical trials and formalized as sequential design of experiments by Robbins (1952): one decision at a time, with no lasting effect on the state; others, in which today's decision changes tomorrow's situation, are genuine Markov decision processes. Contextual bandits were deployed for news recommendation by Li et al. (2010); Chen et al. (2019) describe a recommender system trained by policy gradient from logged data, with the corrections needed to evaluate a new policy from the actions of an old one, and Qin et al. (2020) describe the dispatching of ride-hailing orders by reinforcement learning at DiDi. In computer systems, learned schedulers allocate resources in clusters (Mao et al., 2016). The specific difficulty of these applications is that experimentation is expensive and evaluation must largely be done *off-policy*, from data collected under another policy—the setting of *offline* reinforcement learning, which learns and evaluates policies from a fixed data set without interacting with the environment (Levine et al., 2020); we shall see in Chapter 2 how importance sampling makes this possible, and at what cost.

1.2.5 Health

A physician who must choose among treatments of unknown efficacy for a stream of patients faces a bandit problem—a contextual one when the choice depends on the patient's characteristics—and adaptive clinical trials are one of its historical motivations (Thompson, 1933; Villar et al., 2015). When treatments are adapted over time to the evolution of each patient, one speaks of *dynamic treatment regimes* (Murphy, 2003; Chakraborty and Murphy, 2014): sequences of decision rules, one per stage, that tailor the treatment to the patient's history; when a Markov state summarizing this history is available, they are the policies of a Markov decision process whose transitions are estimated from clinical data. Reinforcement learning has been proposed for the management of sepsis in intensive care (Komorowski et al., 2018), with a lively debate on how such policies can be validated when experimentation is impossible and the data come from physicians who did not follow them (Gottesman et al., 2019). Discovering a promising policy in observational records is one thing; showing that following it would *cause* better outcomes is a much harder problem, because the physicians' decisions depended on information that the records may not contain—off-policy evaluation (Chapter 2) assumes that this is not the case. In health more than anywhere, the average outcome is not the only thing that matters: a treatment that is best on average but occasionally catastrophic is not acceptable, and the distribution of outcomes must be looked at as a whole.

1.2.6 Finance and economics

Sequential decision under uncertainty is the daily bread of finance. Portfolio selection over time is a control problem (Merton, 1969); the execution of a large order over a trading day is a Markov decision process whose state is the remaining inventory and the market conditions (Nevmyvaka et al., 2006); pricing an American option is an optimal stopping problem, solved in practice by an approximate dynamic programming method based on regression (Longstaff and Schwartz, 2001)—closely related in spirit to the fitted value iteration of Chapter 11. Finance has also produced the vocabulary of risk that we adopt in Chapter 6: the value at risk, and the coherent measures such as the conditional value at risk that were designed to repair its defects, summarize the tail of a distribution of losses—the lower

tail of a distribution of gains, in the convention of this book.[1] In economics, the Bellman equation is a standard tool for modeling consumption, investment and growth (Stokey et al., 1989).

1.2.7 Language models

The most visible recent application is the training of large language models. A language model is a policy that chooses the next token given the text so far; after a first phase of imitation of human text, it is refined by *reinforcement learning from human feedback*: humans compare pairs of answers, a reward model is fitted to their preferences, and the policy is improved by policy-gradient methods under a penalty that keeps it close to the supervised reference model (Christiano et al., 2017; Ouyang et al., 2022). Variants bypass the reward model and optimize preferences directly (Rafailov et al., 2023), and reinforcement learning with automatically verifiable rewards has been used to improve the reasoning abilities of these models (Guo et al., 2025). Chapter 14 presents the theory behind these methods, in which the penalty toward a reference policy plays the central role.

1.2.8 Neuroscience and psychology

Reinforcement learning also offers models of how brains learn. The activity of dopamine neurons in the midbrain is well described by the prediction error of temporal-difference learning: they fire when a reward is better than expected, stay silent when it is as expected, and are depressed when it is worse (Schultz et al., 1997). More recently, Dabney et al. (2020) provided evidence that these neurons do not all encode the same prediction—some respond as optimists, some as pessimists—and that their population as a whole may encode the *distribution* of future rewards rather than its mean: exactly the object that this book puts at the center. The models of Chapter 10 are thus not only algorithms but also, to some extent, hypotheses about the brain.

1.2.9 What the successes share, and what remains hard

Looking back at this list, several ingredients recur in many of the successes. A *simulator* or a very large amount of data, since learning requires experience and experience in the real world is expensive and risky. A *well-defined reward*, which is easy in games and hard as soon as human preferences are involved. *Function approximation*, typically by neural networks, to generalize across states that are too numerous to be visited. And *planning* by search or dynamic programming whenever a model is available. None of them is indispensable, and the most spectacular results combine reinforcement learning with supervised learning, search and domain knowledge; but where they are present together, reinforcement learning has repeatedly turned hard sequential problems into solved ones.

What remains hard defines the frontier, and the later chapters of this book. Learning with few samples, and exploring an unknown environment safely and efficiently (Chapter 13). Specifying what one wants, when the objective is a human preference rather than a score (Chapter 14). Evaluating a policy from data collected by another one (Chapters 2 and 11). Acting when other agents learn and adapt at the same time—opponents in a game, but also vehicles at an intersection or bidders in a market (Chapters 15 and 16). And, transversally, controlling not only the average outcome but its variability and its worst cases: a robot, a patient, an investor or a power grid are not served by a policy that is excellent on average and disastrous once in a while. This last concern is the thread of the book. The examples of the next section are miniature versions of the applications above—inventory, maintenance, navigation under risk, exploration, treatment allocation, hiring—small enough to be solved by hand, rich enough to exhibit the phenomena that matter.

[1] Finance works with losses, to be minimized; we work with gains, to be maximized, and translate the definitions accordingly in Chapter 6.

1.3 Motivating examples

The examples below are used throughout the book. Some are toy problems chosen for their pedagogical value; others are simplified versions of the operational questions of Section 1.2. In each one, the reader should identify what is observed (the state), what is decided (the action), what is gained (the reward), and where the randomness comes from.

Example 1.1 (The bakery). Every morning, a baker decides how many loaves $a \geq 0$ to bake (we treat a as a real number; when the demand is integer-valued, the optimal quantity found below is an integer). The demand D of the day is random, with cumulative distribution function F. Each loaf sold brings a margin $g > 0$; each unsold loaf costs $\ell > 0$ (ingredients, energy, disposal). The profit of the day is

$$R(a) \;=\; g\,(a \wedge D) \;-\; \ell\,(a - D)_+.$$

Loaves cannot be kept until the next day, so there is no carry-over: the decision of one day does not affect the next. This is a *one-step* decision problem—the classical *newsvendor* problem—and it can be solved in closed form.

Proposition 1.2 (Newsvendor formula). Let $D \geq 0$ be a random demand with cumulative distribution function F and let $\varphi(a) \coloneqq \mathbb{E}[R(a)]$. Then φ is concave and it is maximized at any τ-quantile of D with $\tau = g/(g+\ell)$, in particular at

$$a^\star \;=\; F^{-1}\!\left(\frac{g}{g+\ell}\right) \;\coloneqq\; \inf\left\{a \in \mathbb{R} : F(a) \geq \frac{g}{g+\ell}\right\}.$$

Proof. Since $a \wedge D = a - (a-D)_+$, we have $\varphi(a) = g\,a - (g+\ell)\,\mathbb{E}[(a-D)_+]$. For $a \geq 0$, Fubini's theorem gives $\mathbb{E}[(a-D)_+] = \mathbb{E}\big[\int_0^a \mathbb{1}\{D \leq u\}\,\mathrm{d}u\big] = \int_0^a F(u)\,\mathrm{d}u$, hence

$$\varphi(a) = g\,a - (g+\ell)\int_0^a F(u)\,\mathrm{d}u.$$

As F is nondecreasing, φ is concave. Its right derivative at a is $g - (g+\ell)F(a)$ and its left derivative is $g - (g+\ell)F(a^-)$, where $F(a^-) = \mathbb{P}(D < a)$. A concave function is maximized exactly at the points where the left derivative is nonnegative and the right derivative is nonpositive, that is, where $F(a^-) \leq g/(g+\ell) \leq F(a)$: these are the τ-quantiles of D with $\tau = g/(g+\ell)$. The generalized inverse $F^{-1}(\tau)$ is one of them. ∎

The optimal decision is a *quantile* of the demand, not its mean: the whole distribution of D matters, through the balance between the cost of a lost sale and the cost of an unsold loaf. This is a first, elementary manifestation of a theme that runs through the book. When the baker does not know F, she has to learn it from past sales—and past sales are *censored* by past decisions, since demand exceeding the stock is not observed. We return to this in Chapter 2.

Example 1.3 (The bike retail store). A shop sells bikes and has room for M of them. At the beginning of each week t the manager observes the stock $S_t \in \{0, \dots, M\}$, pays a storage cost λ per bike in stock, and orders $A_t \in \{0, \dots, M - S_t\}$ bikes, which are delivered immediately. During the week a random demand D_{t+1} arrives; the shop sells $(S_t + A_t) \wedge D_{t+1}$ bikes and the stock at the beginning of the next week is

$$S_{t+1} \;=\; \big(S_t + A_t - D_{t+1}\big)_+.$$

The manager pays a fixed plus proportional ordering cost $f(a) = C_0\mathbb{1}\{a > 0\} + C\,a$ and earns h

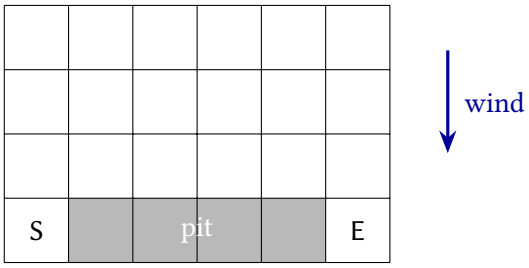


Figure 1.2. The windy hike. The hikers start at S and must reach the bar at E without stepping into the pit (gray). The wind pushes them south with probability p at every step. The grid, the start and goal and the safe-path/short-path contrast are those of the cliff-walking example of Sutton and Barto (2018, Example 6.6).

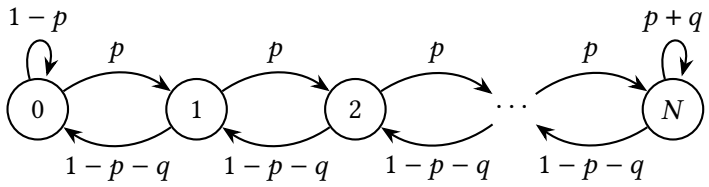


Figure 1.3. Riverswim: transitions under the action R (swim to the right); the self-loops of probability q at the interior states are omitted. The action L moves deterministically to the left. The reward is collected when arriving in state N.

per bike sold. The reward of week t is

$$R_t \;=\; -f(A_t) - \lambda S_t + h\big((S_t + A_t) \wedge D_{t+1}\big).$$

Contrary to the bakery, unsold bikes remain in stock: the decision of a week affects the situation of the next one. The order quantity must balance immediate costs against the value of the stock left for the future, and the problem has to be considered over a horizon of H weeks.

Example 1.4 (The windy hike, after Sutton and Barto, 2018, Example 6.6). Two hikers—a daring child and a prudent mother—walk on the grid of Figure 1.2, with m rows and n columns. They start in the bottom-left cell $\mathsf{S} = (1, 1)$ and want to reach the exit $\mathsf{E} = (1, n)$, where a bar awaits them; the cells $(1, 2), \dots, (1, n-1)$ of the bottom row are a deep pit. At each step a hiker chooses a direction $a \in \{\mathrm{N}, \mathrm{S}, \mathrm{E}, \mathrm{W}\}$. With probability $1 - p$ she moves one cell in that direction; with probability p the wind blows and she moves one cell south instead. A move against a wall leaves the position unchanged. Stepping into the pit sends the hiker to an absorbing *sink* state, from which no reward can ever be collected; the exit is absorbing as well. In the simplest version, reaching the exit yields a reward of 1: within a horizon of H steps the cumulated reward is $\mathbb{1}\{\text{the exit is reached}\}$, and maximizing its expectation means maximizing the probability of success. In the version of the hands-on session, each step spent at the bar before the horizon yields a reward of 1, so that the return is the time enjoyed there (Example 3.6). The short path along the pit is fast but dangerous; a path higher up is safer but takes longer. The child and the mother do not weigh these two aspects in the same way, and this example will follow us through the book.

Example 1.5 (Riverswim). A swimmer is in a river represented by the states $\{0, 1, \dots, N\}$, see Figure 1.3. The action L (swim with the current) moves her deterministically from s to $(s-1) \vee 0$. The action R (swim against the current) moves her from s to $(s+1) \wedge N$ with probability p, leaves her in s with probability q, and pushes her back to $(s-1) \vee 0$ with probability $1-p-q$. A reward of 1 is collected each time she arrives in state N, and nothing otherwise. Swimming against the current is slow (p is small), and the agent must persist for a long time before seeing any reward. This example is a standard test bed for *exploration*: an agent that only knows the rewards it has already observed has no incentive to swim to the right. In the original formulation, a small reward is also available in state 0 under the action L, which makes the temptation to stay left even stronger.

Example 1.6 (The photo booth). A photo booth takes H successive shots $X_1, \dots, X_H$, independent and identically distributed, with values in a finite set $\{v_1 < \cdots < v_K\}$ and probabilities $p_1, \dots, p_K$. After each shot the customer sees it and decides either to keep it, which ends the session, or to discard it and continue; if she reaches the last shot she must keep it. The cumulated reward is the value of the shot kept. The state after t shots is the value X_t of the current shot, together with the information that the session is still going on. This is a simple instance of an *optimal stopping* problem.

Example 1.7 (The secretary problem). H candidates are interviewed one after the other in a uniformly random order. After each interview, the employer knows how the current candidate ranks among those seen so far, and must either hire her on the spot—ending the process—or dismiss her forever. The employer only cares about hiring the very best candidate: the reward is 1 if the hired candidate is the best of all H, and 0 otherwise. Equivalently, the candidates have i.i.d. values $U_1, \dots, U_H$ uniform on $[0, 1]$, the employer observes at time t whether $U_t = \max(U_1, \dots, U_t)$, and wants to maximize the probability of stopping at the overall maximum. Contrary to the photo booth, the reward is not observed when the decision is made: whether the hired candidate is the best depends on the candidates *not* interviewed. We shall see in Chapter 3 that the optimal strategy has a remarkably simple form.

Example 1.8 (Machine replacement). A machine wears out. Its level of wear at the beginning of year t is $S_t \in \mathbb{R}_+$; during a year the wear increases by a random amount $X_{t+1} \geq 0$ (exponentially distributed in the hands-on session), and the yearly maintenance cost $c(s)$ is a nondecreasing function of the wear. Each year the owner either *keeps* the machine, paying $c(S_t)$, or *replaces* it by a new one at price C, paying $C + c(0)$; a new machine wears out like any other, so that $S_{t+1} = S_t + X_{t+1}$ after keeping and $S_{t+1} = X_{t+1}$ after replacing. The rewards are the negatives of these costs, and the owner minimizes the expected total cost over H years. Intuition suggests replacing the machine when its wear exceeds a threshold; Chapter 3 confirms it, and Chapter 7 treats the case of an unbounded horizon. This is the simplest of the maintenance problems of Section 1.2.

Example 1.9 (Treatments). Patients with the same disease arrive one after the other at a hospital where K treatments are available. For each patient the physician chooses a treatment $A_t \in [K]$ and observes its outcome $R_t \in [0, 1]$, from 0 (death) to 1 (full recovery); the outcomes of a given treatment a are i.i.d. with an unknown mean θ_a. The physician wants to maximize the

Example	State	Action	Source of randomness	Nature of the problem
Bakery	—	quantity baked	daily demand	one-step decision
Bike store	stock level	quantity ordered	weekly demand	inventory control
Windy hike	position	direction	wind	navigation, risk
Riverswim	position	left / right	current	exploration
Photo booth	value of current shot	keep / continue	quality of shots	optimal stopping
Secretary	relative rank	hire / dismiss	order of candidates	stopping, hidden reward
Machine replacement	level of wear	keep / replace	deterioration	maintenance, threshold
Treatments	— (information)	treatment	patients' responses	learning, exploration

Table 1.1. The examples of Section 1.3 at a glance.

expected total outcome $\mathbb{E}[\sum_{t \le H} R_t]$ over H patients—the expected number of recoveries when outcomes are binary. If the θ_a were known the problem would be trivial: always prescribe the best treatment. Since they are not, each choice serves two purposes—curing the current patient and learning about the treatment for the next ones—and a treatment that looks poor after a few trials may deserve a second chance, at a cost. There is no dynamics here: the outcome of a patient depends only on the treatment, so that, as a Markov decision process, the problem has a single state. What evolves is the physician's *information*. This is the multi-armed bandit problem, the purest form of the exploration–exploitation dilemma, studied in Chapter 13.

These examples share a common structure, summarized in Table 1.1: a state that summarizes the situation, an action chosen by the agent, a random transition to a new state, and a reward attached to the transition. Other examples, with continuous states and actions (controlling the temperature of a building, steering a rocket), appear in Chapter 3.

1.4 Markov decision processes

Definition 1.10 (Finite-horizon Markov decision process). A *Markov decision process* (MDP) with finite horizon is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, (P_t)_{t\in[H]}, (r_t)_{t\in[H]}, H)$ where

- $\mathcal{S}$ is the *state space* and $\mathcal{A}$ the *action space*, both finite unless otherwise stated, with cardinalities $S = |\mathcal{S}|$ and $A = |\mathcal{A}|$;
- $H \in \mathbb{N}^*$ is the *horizon*, the number of decisions to be made;
- for each $t \in [H]$, P_t is a *transition kernel*: for every $(s,a) \in \mathcal{S}\times\mathcal{A}$, $P_t(\cdot \mid s,a) \in \mathcal{M}_1(\mathcal{S})$ is the law of the next state when action a is chosen in state s at time t;
- for each $t \in [H]$, $r_t : \mathcal{S}\times\mathcal{A}\times\mathcal{S} \to \mathbb{R}$ is a *reward function*: $r_t(s,a,s')$ is the reward received when the transition from s to s' under action a occurs at time t.

The MDP is *stationary* when $P_t = P$ and $r_t = r$ do not depend on t. We write $P_a \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}$ for the matrix $P_a(s,s') = P(s' \mid s,a)$, and

$$r(s,a) \coloneqq \sum_{s'\in\mathcal{S}} P(s' \mid s,a)\, r(s,a,s')$$

for the expected one-step reward, and similarly $r_t(s,a) \coloneqq \sum_{s'} P_t(s' \mid s,a)\, r_t(s,a,s')$ in the time-dependent case.

The interaction between the agent and the environment—the *protocol*—is as follows. An initial state

S_1 is drawn from an initial distribution $\mu \in \mathcal{M}_1(\mathcal{S})$. For $t = 1, \dots, H$: the agent observes S_t, chooses an action $A_t \in \mathcal{A}$, the environment draws the next state $S_{t+1} \sim P_t(\cdot \mid S_t, A_t)$, and the agent receives the reward $R_t = r_t(S_t, A_t, S_{t+1})$. The interaction stops after S_{H+1} is reached. The word *Markov* refers to the fact that the law of S_{t+1} depends on the past only through the current state and action.

Remark 1.11 (Variations on the definition).

(i) *Admissible actions.* In many problems only a subset $\mathcal{A}(s) \subset \mathcal{A}$ of actions is available in state s (the shop cannot order more bikes than it can store). This is handled by restricting the maximizations below to $\mathcal{A}(s)$, or by making the forbidden actions equivalent to an allowed one. We ignore this refinement in the notation.

(ii) *Random rewards.* One sometimes wants the reward to be random given (s, a, s'). Nothing is lost by our convention when the reward takes finitely many values: it suffices to enlarge the state to (s', x) where x is the realized reward, drawn jointly with s' (a continuous reward law leads to a continuous state space, one of the situations of item (v)). Making the reward a function of the transition keeps the notation light and makes the source of randomness explicit, which will matter when we study the whole distribution of the cumulated reward.

(iii) *Terminal reward.* A reward $r_{H+1}(s)$ collected in the final state is covered by adding it to $r_H(s, a, s')$, since $r_H(s, a, s') + r_{H+1}(s')$ is again a function of the last transition.

(iv) *Costs.* We adopt the *gain convention*: rewards are to be maximized. A cost is a negative reward.

(v) *Infinite spaces.* Many examples have continuous states or actions. The theory of this book is developed for finite spaces, where all sums are finite and no measurability issue arises; we indicate along the way what extends to general spaces and refer to Bertsekas and Shreve (1978) and Puterman (1994, Chap. 2) for the technicalities.

(vi) *What is the state?* The state must contain everything in the past that is relevant for the future: the Markov property is a modeling assumption, not a fact of nature. In the secretary problem, the full sequence of interviews is a valid state, but the relative rank of the current candidate turns out to be a *sufficient statistic* for the future (Exercise 1.7), and the problem becomes tractable once the state is reduced to it. When the agent does not observe the state but only a noisy function of it, one speaks of a *partially observable* MDP; the conditional law of the state given the observations (the *belief*) is then a sufficient statistic, at the price of a continuous state space. We do not treat partial observability in this book.

It is often more natural to describe a random transition as the output of a deterministic map fed with exogenous noise, as in Example 1.3 where S_{t+1} is a function of S_t, A_t and the demand D_{t+1}. The two descriptions are equivalent.

Proposition 1.12 (Kernels and noisy dynamical systems). Let $\mathcal{S}$ be finite.

(i) For any kernel P from $\mathcal{S} \times \mathcal{A}$ to $\mathcal{S}$ there exists $f : \mathcal{S} \times \mathcal{A} \times [0,1] \to \mathcal{S}$ such that $f(s, a, U) \sim P(\cdot \mid s, a)$ for all (s, a) when U is uniform on $[0, 1]$.

(ii) Conversely, if $S_{t+1} = f_t(S_t, A_t, U_{t+1})$ where the U_t are i.i.d. with an arbitrary law, U_{t+1} being independent of $(S_1, A_1, \dots, S_t, A_t)$, then (S_t) evolves according to the kernels $P_t(s' \mid s, a) = \mathbb{P}\big(f_t(s, a, U_{t+1}) = s'\big)$.

Proof. For the first claim, enumerate $\mathcal{S} = \{s^{(1)}, \dots, s^{(S)}\}$ and, for each (s, a), cut $[0, 1)$ into S consecutive intervals $I_j(s, a)$ of respective lengths $P(s^{(j)} \mid s, a)$; set $f(s, a, u) = s^{(j)}$ for $u \in I_j(s, a)$. Then

$\mathbb{P}(f(s,a,U) = s^{(j)}) = |I_j(s,a)| = P(s^{(j)} \mid s,a)$. The second claim follows from the independence of U_{t+1} and $(S_1, A_1, \dots, S_t, A_t)$: $\mathbb{P}(S_{t+1} = s' \mid S_1, A_1, \dots, S_t, A_t) = \mathbb{P}(f_t(S_t, A_t, U_{t+1}) = s' \mid S_t, A_t)$. ∎

The map f is what a *simulator* of the environment implements: given the current state and the action, it draws the next state using fresh random numbers. Many algorithms of this book only require access to such a simulator (a *generative model*) rather than to the kernels themselves.

1.5 Policies and the law of a trajectory

The agent's behavior is described by a *policy*: a rule that specifies which action to take, possibly at random, as a function of everything observed so far.

Definition 1.13 (Histories and policies). A *history* up to time t is a sequence

$$h_t = (s_1, a_1, s_2, a_2, \dots, s_t) \in \mathcal{H}_t := (\mathcal{S} \times \mathcal{A})^{t-1} \times \mathcal{S}$$

of the states visited and actions taken before the t-th decision. A *policy* $\pi = (\pi_t)_{t \in [H]}$ is a sequence of maps $\pi_t : \mathcal{H}_t \to \mathcal{M}_1(\mathcal{A})$; $\pi_t(a \mid h_t)$ is the probability of choosing action a at time t after the history h_t. The policy is

- *Markov* if $\pi_t(\cdot \mid h_t)$ depends on h_t only through s_t; we then write $\pi_t(a \mid s)$;
- *deterministic* if each $\pi_t(\cdot \mid h_t)$ is a Dirac mass; for a deterministic Markov policy we write $\pi_t(s) \in \mathcal{A}$ for the chosen action;
- *stationary* if the MDP is stationary and $\pi_t = \pi_1$ for all t.

We denote by Π_{H} the set of all policies, by Π_{MR} the set of Markov (randomized) policies and by Π_{MD} the set of deterministic Markov policies, so that $\Pi_{\mathrm{MD}} \subset \Pi_{\mathrm{MR}} \subset \Pi_{\mathrm{H}}$.

The set Π_{MD} is finite, with A^{SH} elements. This number is astronomically large even for small problems—for the windy hike on a 4×6 grid with $H = 20$, it exceeds 10^{250}—so that no method based on the enumeration of policies is viable. The structure that makes MDPs tractable is the object of Chapter 3.

Once a policy is fixed, the protocol of Section 1.4 defines a random trajectory $(S_1, A_1, \dots, S_H, A_H, S_{H+1})$. Since all spaces are finite, its law is given by an explicit product formula.

Theorem 1.14 (Law of a trajectory). Let $\mathcal{M}$ be a finite-horizon MDP, $\mu \in \mathcal{M}_1(\mathcal{S})$ and $\pi \in \Pi_{\mathrm{H}}$. Let $\Omega_H := (\mathcal{S} \times \mathcal{A})^H \times \mathcal{S}$ and let $(S_1, A_1, \dots, S_H, A_H, S_{H+1})$ denote the coordinate maps on Ω_H; write $H_t = (S_1, A_1, \dots, S_t)$. There exists a unique probability measure $\mathbb{P}_\mu^\pi$ on Ω_H such that

(i) $\mathbb{P}_\mu^\pi(S_1 = s) = \mu(s)$ for all $s \in \mathcal{S}$;

(ii) for all $t \in [H]$, $h_t \in \mathcal{H}_t$ and $a \in \mathcal{A}$ with $\mathbb{P}_\mu^\pi(H_t = h_t) > 0$: $\mathbb{P}_\mu^\pi(A_t = a \mid H_t = h_t) = \pi_t(a \mid h_t)$;

(iii) for all $t \in [H]$, $h_t \in \mathcal{H}_t$, $a \in \mathcal{A}$ and $s' \in \mathcal{S}$ with $\mathbb{P}_\mu^\pi(H_t = h_t, A_t = a) > 0$: $\mathbb{P}_\mu^\pi(S_{t+1} = s' \mid H_t = h_t, A_t = a) = P_t(s' \mid s_t, a)$.

It is given, for every $\omega = (s_1, a_1, \dots, s_H, a_H, s_{H+1}) \in \Omega_H$, by

$$\mathbb{P}_\mu^\pi(\{\omega\}) \;=\; \mu(s_1) \prod_{t=1}^{H} \pi_t(a_t \mid h_t)\, P_t(s_{t+1} \mid s_t, a_t), \qquad h_t = (s_1, a_1, \dots, s_t). \tag{1.1}$$

Proof. Uniqueness. If a probability measure satisfies (i)–(iii), the chain rule for conditional probabilities yields (1.1) for every ω of positive probability, and also for the others (if some factor vanishes, both sides are zero: the first vanishing factor along ω forces the probability of the corresponding event, hence of $\{\omega\}$, to be zero).

Existence. Define $\mathbb{P}_\mu^\pi(\{\omega\})$ by (1.1). These numbers are nonnegative. Summing first over s_{H+1} gives a factor $\sum_{s'} P_H(s' \mid s_H, a_H) = 1$, then summing over a_H gives $\sum_a \pi_H(a \mid h_H) = 1$, and so on down to $\sum_{s_1} \mu(s_1) = 1$: the total mass is one. The same partial summations show that $\mathbb{P}_\mu^\pi(H_t = h_t) = \mu(s_1) \prod_{k<t} \pi_k(a_k \mid h_k) P_k(s_{k+1} \mid s_k, a_k)$, that $\mathbb{P}_\mu^\pi(H_t = h_t, A_t = a)$ is this quantity times $\pi_t(a \mid h_t)$, and that $\mathbb{P}_\mu^\pi(H_t = h_t, A_t = a, S_{t+1} = s')$ is the latter times $P_t(s' \mid s_t, a)$; taking ratios gives (i)–(iii). ∎

We write $\mathbb{E}_\mu^\pi$ for the expectation under $\mathbb{P}_\mu^\pi$, and $\mathbb{P}_s^\pi$, $\mathbb{E}_s^\pi$ when $\mu = \delta_s$. For a *Markov* policy $\pi \in \Pi_{\mathrm{MR}}$, $t \in [H]$ and $s \in \mathcal{S}$, we also denote by $\mathbb{P}_{t,s}^\pi$ the law of the partial trajectory $(S_t, A_t, \dots, S_H, A_H, S_{H+1})$ of the process *started at time t in state s*: it is defined by the product formula $\prod_{k=t}^{H} \pi_k(a_k \mid s_k) P_k(s_{k+1} \mid s_k, a_k)$ with $s_t = s$ (for a history-dependent policy the decision rules would require the missing past). When the first action is also imposed, we write $\mathbb{P}_{t,s,a}^\pi$. In the infinite-horizon setting of Chapter 7, the trajectory space is infinite and the existence of $\mathbb{P}_\mu^\pi$ rests on the Ionescu-Tulcea theorem, see Appendix A.

For Markov policies, the past influences the future only through the current state. This is the property on which every recursion of the book relies.

Proposition 1.15 (Markov and restart properties). Let $\pi \in \Pi_{\mathrm{MR}}$ and $\mu \in \mathcal{M}_1(\mathcal{S})$. For every $t \in [H]$ and every $h_t \in \mathcal{H}_t$ with $\mathbb{P}_\mu^\pi(H_t = h_t) > 0$, the conditional law of $(S_t, A_t, \dots, S_H, A_H, S_{H+1})$ given $H_t = h_t$ under $\mathbb{P}_\mu^\pi$ is $\mathbb{P}_{t,s_t}^\pi$. In particular:

(i) $(S_t)_{t \in [H+1]}$ is a Markov chain with transition matrices $P_t^{\pi_t}(s, s') \coloneqq \sum_{a \in \mathcal{A}} \pi_t(a \mid s) P_t(s' \mid s, a)$;

(ii) for every bounded function g of the partial trajectory,

$$\mathbb{E}_\mu^\pi\big[g(S_t, A_t, \dots, S_{H+1}) \,\big|\, H_t\big] = \mathbb{E}_{t,S_t}^\pi\big[g(S_t, A_t, \dots, S_{H+1})\big].$$

Proof. Fix h_t with positive probability and a continuation $(a_t, s_{t+1}, \dots, s_{H+1})$. By (1.1) and the formula for $\mathbb{P}_\mu^\pi(H_t = h_t)$ obtained in the proof of Theorem 1.14, the conditional probability of the continuation given $H_t = h_t$ is the ratio of the two products, in which all factors indexed by $k < t$ cancel:

$$\mathbb{P}_\mu^\pi\big(A_t = a_t, S_{t+1} = s_{t+1}, \dots, S_{H+1} = s_{H+1} \,\big|\, H_t = h_t\big) = \prod_{k=t}^{H} \pi_k(a_k \mid s_k)\, P_k(s_{k+1} \mid s_k, a_k),$$

where we used that $\pi_k(a_k \mid h_k) = \pi_k(a_k \mid s_k)$ for a Markov policy. The right-hand side is $\mathbb{P}_{t,s_t}^\pi$ evaluated at the continuation, and depends on h_t only through s_t. The two consequences follow by summation. ∎

For a history-dependent policy the conclusion fails in general: the conditional law of the future given h_t still exists, but it is the law of the process started at (t, s_t) under the *shifted* policy obtained by prefixing every history with h_t, which depends on the whole of h_t.

1.6 The return and its distribution

Definition 1.16 (Return). The *return* of a trajectory is the cumulated reward

$$W_H \coloneqq \sum_{t=1}^{H} R_t, \qquad R_t = r_t(S_t, A_t, S_{t+1}).$$

For $t \in [H+1]$, the *reward-to-go* from time t is $W_{t:H} \coloneqq \sum_{k=t}^{H} R_k$, with the convention $W_{H+1:H} = 0$.

The return is a random variable, whose law depends on the initial state and on the policy. This law is the central object of the book.

Definition 1.17 (Return distribution and value). Let $\pi \in \Pi_{\mathrm{H}}$. The *return distribution* of π from state s is the law of the return under $\mathbb{P}_s^\pi$:

$$v^\pi(s) \coloneqq \mathcal{L}_{\mathbb{P}_s^\pi}(W_H) \in \mathcal{M}_1(\mathbb{R}),$$

and more generally $v^\pi(\mu) \coloneqq \mathcal{L}_{\mathbb{P}_\mu^\pi}(W_H) = \sum_s \mu(s) v^\pi(s)$. The *value* of π from s is its expectation,

$$V^\pi(s) \coloneqq \mathbb{E}_s^\pi[W_H] = \int_{\mathbb{R}} x \, v^\pi(s)(\mathrm{d}x),$$

and $V^\pi(\mu) \coloneqq \mathbb{E}_\mu^\pi[W_H] = \sum_s \mu(s) V^\pi(s)$.

For a probability measure $\nu \in \mathcal{M}_1(\mathbb{R})$ with a finite first moment, we use the shorthand $\mathbb{E}[\nu] \coloneqq \int x \, \nu(\mathrm{d}x)$ for its mean, and similarly $\mathbb{V}\mathrm{ar}[\nu]$ for its variance; thus $V^\pi(s) = \mathbb{E}[v^\pi(s)]$.

Remark 1.18 (Finite support). Since Ω_H is finite, $v^\pi(s)$ is a finitely supported measure: it charges at most $(SA)^H$ points (the number of trajectories from a fixed initial state; see also Remark 3.9), and in practice far fewer, since many trajectories share the same return. In the simplest version of the windy hike, $v^\pi(s)$ is a Bernoulli law. In the bike store with integer demands and costs, it is supported by a set of integers. In general, however, the support may be as large as the set of trajectories, and *representing* $v^\pi(s)$ exactly is already a computational question; we come back to it in Chapters 3 and 4.

Which features of $v^\pi(s)$ should an agent care about? The classical answer is the mean $V^\pi(s)$, and there are good reasons for it. If the same decision problem is faced many times independently, the law of large numbers makes the average cumulated reward converge to the value, so that a policy with a larger value eventually wins. More importantly for computation, the expectation is a *linear* functional of the return distribution, and we shall see in Chapter 3 that linearity is exactly what makes the value of a policy computable by a recursion over time.

But the mean is not always the right criterion, and it is not the only feature of the return that can be computed.

Example 1.19 (Safe or risky). Consider a single decision ($H = 1$) between two actions. Action a yields a reward of 1 for sure. Action b yields 2 with probability $1/2$ and 0 otherwise. The two return distributions are $v^{\mathrm{a}} = \delta_1$ and $v^{\mathrm{b}} = \frac{1}{2}(\delta_0 + \delta_2)$; they have the same mean 1 but are very different: a has zero variance, the probability of a zero return is $1/2$ under b, the lower median of

ν^{b} is 0. A cautious agent, or one who needs at least one unit of reward, prefers a; an agent who needs two units has no choice but b. Which action is "better" depends on which functional of the return distribution one maximizes.

The functionals of the return distribution most often considered are:

- the mean $\mathbb{E}[\nu]$, and more generally the moments $\int x^k \nu(\mathrm{d}x)$ and the variance;
- the probability $\nu([u,\infty))$ of exceeding a threshold u, and the quantiles $F_\nu^{-1}(\tau)$, $\tau \in (0,1)$;
- the *utilities* $\psi_f(\nu) \coloneqq \int f \,\mathrm{d}\nu$ for a nondecreasing function $f : \mathbb{R} \to \mathbb{R}$, which weigh outcomes according to the agent's preferences—concave f for a risk-averse agent, convex f for a risk-seeking one;
- *risk measures* such as the value at risk or the conditional value at risk, which summarize the lower tail of ν (Chapter 6).

Given a functional $\psi : \mathcal{M}_1(\mathbb{R}) \to \mathbb{R}$, the agent's problem is to make $\psi(\nu^\pi(\mu))$ as large as possible. A large part of this book is devoted to understanding for which ψ this problem is tractable, and what to do when it is not.

1.7 Evaluation, planning, learning

Three problems, of increasing difficulty, organize the field.

Policy evaluation. The MDP and a policy π are given; compute the return distribution ν^π, or a functional of it such as the value V^π. This is a question about a Markov chain (Proposition 1.15) and can be addressed either by simulation—running the policy many times and averaging (Chapter 2)—or, when the kernels are known and the state space is small, by exact recursions (Chapter 3).

Planning. The MDP is known; find a policy that maximizes the chosen criterion, classically the expected return $V^\pi(\mu)$. Bellman's principle of optimality (Chapter 3) reduces this search over the huge set Π_{H} to H successive optimizations over $\mathcal{A}$, one for each state and time. Which other criteria admit such a reduction is the subject of Chapters 4 and 5.

Learning. The kernels P_t (and possibly the rewards) are unknown; the agent can only interact with the environment, or with a simulator of it, and must find a good policy from the data it collects. Two regimes are distinguished. In the *online* regime the agent learns while acting, and must trade off the exploitation of what it already knows against the exploration of what it does not (Chapter 13); the riverswim of Example 1.5 is the paradigmatic example. In the *offline* regime a fixed data set of transitions is available. Learning methods are *model-based* when they estimate the kernels and then plan in the estimated model, and *model-free* when they directly estimate value functions or policies from samples (Chapters 10 and 11). When the state space is large, estimates must moreover be *approximated* within a class of functions, which is where statistical learning and neural networks enter the picture (Chapter 11).

Distributional viewpoint

The return W_H is a random variable and its law $\nu^\pi(s)$ carries everything that can be said about the policy π started from s. The classical theory of reinforcement learning studies one functional of this law, the mean $V^\pi(s) = \mathbb{E}[\nu^\pi(s)]$. This choice is legitimate—and, as we shall see, computationally privileged—but it is a choice. The newsvendor of Example 1.1 maximizes a mean

and finds that the answer is a quantile; the two actions of Example 1.19 cannot be distinguished by their means although they lead to very different situations. Three questions guide the following chapters.

(i) Which statistics of ν^π can be computed exactly by a recursion over time?

(ii) Which functionals $\psi(\nu^\pi)$ can be maximized exactly over policies?

(iii) What can be done, at what price, for the others?

Hands-on

The notebook `bakery.ipynb` (session H01) poses the bakery problem of Example 1.1 in two versions. When the law of the demand is known, the agent should implement the newsvendor formula of Proposition 1.2. When it is unknown, the agent must learn from past sales, which is the subject of Chapter 2. The notebook `retailStore.ipynb` (session H04) implements the bike store of Example 1.3 as a simulator in the sense of Proposition 1.12; it is used from Chapter 3 on for planning and learning.

Exercises

Exercise 1.1 (Newsvendor with salvage value). In Example 1.1, suppose that each loaf costs c to produce and is sold at price $p > c$, and that unsold loaves are sold the next day at a salvage price $v < c$. Write the profit of the day, show that the optimal production is again a quantile of D, and identify its level. Compute it when D is a Poisson random variable with parameter $\lambda = 20$, $p = 3$, $c = 1$ and $v = 0.5$.

Exercise 1.2 (The windy hike as an MDP). Write explicitly the state space, the action space and the kernel $P(s' \mid s, a)$ of Example 1.4, including the sink and the absorbing exit. In the simplest version, check that the return W_H takes values in $\{0, 1\}$ and that $V^\pi(\mathrm{S})$ is the probability of reaching the exit within H steps. How many deterministic Markov policies are there for a 4×6 grid and $H = 20$?

Exercise 1.3 (Randomization in the state). Let $\pi \in \Pi_{\mathrm{MR}}$. Using Proposition 1.12, build an MDP on the state space $\mathcal{S} \times [0, 1]$ (or $\mathcal{S} \times \{1, \dots, A\}$ with a suitable auxiliary kernel) and a *deterministic* Markov policy on it such that the law of the sequence $(S_t, A_t)_{t \in [H]}$ is the same as under π in the original MDP. Conclude that randomization can always be delegated to the environment.

Exercise 1.4 (Marginal laws). Let $\pi \in \Pi_{\mathrm{MR}}$. Using (1.1), show that the law of S_t under $\mathbb{P}_\mu^\pi$ is the row vector $\mu P_1^{\pi_1} P_2^{\pi_2} \cdots P_{t-1}^{\pi_{t-1}}$, where $P_k^{\pi_k}$ is the matrix of Proposition 1.15. Deduce that $V^\pi(\mu) = \sum_{t=1}^{H} \sum_{s,a} \mathbb{P}_\mu^\pi(S_t = s, A_t = a)\, r_t(s, a)$: the value is a linear function of the *occupation measures* $d_t^\pi(s, a) \coloneqq \mathbb{P}_\mu^\pi(S_t = s, A_t = a)$.

Exercise 1.5 (Safe or risky, twice). Consider two successive decisions ($H = 2$) in the setting of Example 1.19, each between a and b, with independent outcomes. Compute the return distribution of the four deterministic Markov policies and of the history-dependent policy "play b, then a if the first reward was 2 and b otherwise". Which policies maximize $\mathbb{P}(W_2 \geq 1)$? $\mathbb{P}(W_2 \geq 2)$? $\mathbb{P}(W_2 \geq 3)$? Observe that the ranking depends on the threshold, that for each single threshold a Markov policy does as well as the history-dependent one, but that the latter is the only policy achieving $\mathbb{P}(W_2 \geq 2) = \frac{3}{4}$ and $\mathbb{P}(W_2 \geq 3) = \frac{1}{2}$ simultaneously. Chapter 5 exhibits criteria for which memory is indispensable.

Exercise 1.6 (Treatments as an MDP on the information state). In Example 1.9, suppose the outcomes are Bernoulli, $R_t \in \{0, 1\}$, and that the physician models her uncertainty about θ_a by independent uniform priors on $[0, 1]$. Show that, conditionally on the past, the probability of a recovery under treatment a is $(u_a+1)/(u_a+v_a+2)$, where u_a and v_a are the numbers of recoveries and failures observed so far with treatment a (*hint:* Bayes' formula with a Beta posterior). Deduce that the problem is a finite-horizon MDP whose state is the vector $(u_a, v_a)_{a\in[K]}$, write its kernel and rewards, and count its states when H patients are treated. What does this suggest about the cost of computing an optimal policy by dynamic programming?

Exercise 1.7 (Relative ranks). Let $U_1, \dots, U_H$ be i.i.d. with a continuous distribution and let $I_t = \mathbb{1}\{U_t = \max(U_1, \dots, U_t)\}$ indicate that the t-th candidate is the best seen so far. Show that $I_1, \dots, I_H$ are independent with $\mathbb{P}(I_t = 1) = 1/t$. *Hint:* the vector of relative ranks $(\mathrm{rk}_1, \dots, \mathrm{rk}_H)$, where rk_t is the rank of U_t among $U_1, \dots, U_t$ (rank 1 for the largest), is in bijection with the permutation ordering the U's. Deduce that the secretary problem of Example 1.7 can be modeled as an MDP with state space $\{0, 1\} \cup \{\text{stop}\}$ and time-dependent kernels. (This is Lemma 3.18, proved in Chapter 3; try it before reading the proof.)

Bibliographical notes

Markov decision processes were formalized by Bellman (1957a) and Bellman (1957b), who coined the expression "dynamic programming", and by Howard (1960). The book of Puterman (1994) remains the standard mathematical reference; Bertsekas (2017) emphasizes control and continuous spaces, and Bertsekas and Shreve (1978) treat the measure-theoretic foundations. The reinforcement learning perspective—learning from interaction—is developed in Sutton and Barto (2018) and, with a theoretical emphasis, in Szepesvári (2010) and Agarwal et al. (2026).

The newsvendor problem goes back to Arrow et al. (1951). The windy hike is a variant of the cliff-walking example of Sutton and Barto (2018, Example 6.6): the grid geometry, the start and the goal, and the contrast between a short dangerous path and a long safe one are theirs, and are used again in Chapters 3 and 4; the stochastic wind, the return distributions and the two hikers are additions of this book. riverswim was introduced by Strehl and Littman (2008). The secretary problem has a colorful history, told by Ferguson (1989); its first published solution is due to Lindley (1961), and Gilbert and Mosteller (1966) study many variants.

The idea of looking beyond the mean of the return is old in operations research: moments were studied by Jaquette (1973), the variance by Sobel (1982), and the whole distribution by Chung and Sobel (1987). It was revived in machine learning by Morimura et al. (2010) and popularized by Bellemare et al. (2017); the monograph of Bellemare et al. (2023) is the reference on distributional reinforcement learning. The existence of the trajectory law on infinite products is the theorem of Ionescu Tulcea (1949), see also Kallenberg (2002, Chap. 6).