CHAPTER 2

From Markov to Bellman

Before we build, we must understand the structure beneath

This and the following chapter are like learning the basics of driving a car: how to change gears, accelerate from a standstill, and brake. Only once these fundamentals are mastered, driving in traffic is safe and justified. Therefore, even if the analogy is not perfect, it is advisable to study Reinforcement Learning (RL) techniques first after the foundations are cemented. A reminder: it is advisable to check the nomenclature if an unknown term is experienced.

2.1 AN INTRODUCTORY PROBLEM

Bob is a diligent high school student. Every week, he faces one exam. Each day, he can decide how many hours to study, but he doesn't exactly know how study time affects his score. He's noticed there's a sweet spot: studying improves performance up to a point, but beyond that, additional hours don't help much and take away time from other enjoyable or important activities. Over many weeks, Bob wants to learn a study strategy that consistently gives him the best possible exam scores, without overstudying or burning out.

Table 2.1 illustrates how Bob studies over several weeks, with each row representing a weekly history that links the total number of study hours to the resulting exam score. Bob can study 0, 1, or 2 hours per day, a limitation shaped by his school schedule, sports

TABLE 2.1 Study Habits and Exam Score for the Student Bob

Week/Day	1	2	3	4	5	Score
1	0	1	0	0	0	1
2	0	1	1	0	1	3
3	1	1	1	1	1	5
4	2	2	2	1	1	5
5	0	1	0	0	0	2

Note: Each cell under Columns 1–5 represents the number of hours studied. *Score* denotes the exam result for that week.

DOI: 10.1201/9781003634409-2

activities, and the time he wants to spend with family and friends. This upper bound reflects a realistic trade-off between schoolwork and other aspects of daily life. The table shows how different study patterns affect Bob's exam performance, allowing him to learn from experience and gradually improve his study routine.

From intuition and personal experience, Bob (like most readers of this book) can make the following guesses: (1) if studying is postponed during the first days of the week, it quickly becomes too late to catch up before the exam; (2) it is generally wise to study at least a little each day, for example, one hour. However, such guesses only amount to a possible policy. What is needed instead is a fact-based policy. The more weeks of experience and data Bob collects, the stronger his conclusions about which study strategies work best will become.

This chapter introduces key concepts that help formulate Bob's study problem in a precise and structured way. With such a formulation, we can apply learning techniques to discover a strategy for how Bob should plan his studies. These techniques allow decisions to be guided by data and learned experience—not just intuition. During the chapter, Bob's study problem will be revisited to clarify introduced concepts.

2.2 MARKOV PROCESS AND STATE PRESENTATION

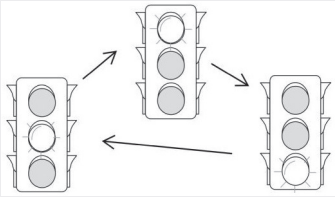
A **Markov process** is a common way to represent dynamic systems. It consists of a **state space** and **transition probabilities**. The state space is all possible states. For a traffic light, see Worked example 2.1, it is green, yellow, and red. The transition probabilities define the likelihood of moving from one state to another. A Markov process can be interpreted as a description of how a system evolves over time. Markov processes are widely used in RL, where modelling sequential dependencies is crucial.

A **state** is a fundamental concept represented by the term s. Worked examples 2.1–2.5 illustrate how a state expresses the status of a system. A state can consist of one or several variables, which may be integers, floating-point numbers, or Boolean values.

WORKED EXAMPLE 2.1

Present an example of a Markov process with deterministic transitions.

Deterministic means there is no randomness when changing states. A traffic light, see the figure below, can explain the concepts of state and Markov process. The state is the current color—green (right), yellow (left), or red (top). Transition probabilities determine, for example, that there is a 100% probability to shift from green to yellow.



Traffic light states. Red light (top), green (right), and yellow (left).

WORKED EXAMPLE 2.2

In Section 2.1, Bob the student was introduced. Propose a Markov state description for that problem.

Define:
hours = accumulated number of studied hours
daysleft= number of days left to exam
—Proposal 1—
(*day, hours*)
—Proposal 2—
(*daysleft, hours*)

Comment 1: Representing the state using only the total number of study hours is not Markovian, as the future state also depends on the current day.
Comment 2: Choosing between proposals 1 and 2 is largely a matter of preference. However, it can be argued that using "number of days left until the exam" is more intuitive, since a low value naturally indicates a "worse" or more urgent state.

WORKED EXAMPLE 2.3

Choose a state description from Worked example 2.2 and exemplify how the state can be updated during a week. Choose two weeks from Table 2.1.

$$\text{Choose: } state = (day, hours)$$

where hours is accumulated number of studied hours.

Week\Day	1	2	3	4	5
2	(1,0)	(2,0)	(3,1)	(4,2)	(5,2)
4	(1,0)	(2,2)	(3,4)	(4,6)	(5,7)

Comment: The state describes the accumulated number of studied hours when entering state (what has happened before).

WORKED EXAMPLE 2.4

Present an example of a Markov process with stochastic transitions.

Stochastic implies that there is randomness in the transition from one state to another. For example, weather is a system with stochastic transitions: if it is rainy today, it may or may not be sunny tomorrow. One could assume, for instance, a 50% probability that the current weather condition will persist to the next day.

WORKED EXAMPLE 2.5

Given the weather prediction model in Worked example 2.4, present two example sequences of how the weather can change from day to day.

Introduce: S=sunny, R=rainy										
Day	1	2	3	4	5	6	7	8	9	10
Weather	S	S	S	R	R	S	S	R	S	S
Day	1	2	3	4	5	6	7	8	9	10
Weather	R	S	S	S	R	S	S	R	S	S

With the concept of state established, a natural question arises: what makes for a good state representation? The idea of a **Markovian** state representation is central here. A system is Markovian if the current state contains all the information needed to predict the next state. In other words, the future depends only on the present, not on the sequence of past states. For example, consider a moving car. Knowing only the position does not provide a Markov state, because the future position also depends on the speed. To fully capture the system in a Markovian manner, both position and speed must be included in the state.

One aspect characteristic of a Markov process is whether transitions are **deterministic** or **stochastic**. Deterministic systems are those in which outcomes are fully determined by the current state, with no randomness involved. In contrast, stochastic systems involve randomness, meaning that outcomes are uncertain and best described by probability distributions rather than fixed rules. The difference is highlighted in Worked examples 2.1 and 2.4. Worked example 2.5 exemplifies how states can vary in time for a system with stochastic transitions.

2.3 REWARD

The **reward** is the immediate feedback received after taking an **action** in a state, expressed as a real number indicating how good or bad that choice was. The reward signals the desirability of the outcome. Higher rewards encourage actions that lead to better states, while lower or negative rewards discourage poor decisions. An action represents a decision the agent can take when in a particular state, chosen from the set of all available alternatives.

A stochastic reward means that the outcome of the reward is not exactly known when the agent takes a specific action in a specific state. In other words, the same action in the same state may lead to different rewards due to inherent randomness. A common real-life example is throwing a die: the action is to throw, but the reward, the number of dots shown, is highly stochastic. Despite identical actions, the result varies due to chance.

A common real-life example of a **delayed** reward is playing chess. The true reward for a sequence of actions is only revealed at the end of the game, when one player is checkmated. It is only then that the consequences of all preceding moves become clear.

The concepts of reward and action are illustrated in Worked examples 2.6 and 2.7, using the Student Bob problem introduced in Section 2.1.

WORKED EXAMPLE 2.6

In Section 2.1, Bob the student was introduced. What is the reward in that problem?

There is only one single reward: the score on the exam (received at the end of the week). All the transitions from day 1 up to day 5 yield a reward of zero.

Comment 1: This is an example of a delayed reward problem, where feedback is only received after a sequence of actions has been completed.

Comment 2: The reward is stochastic because days 1 and 5 yield different rewards despite ending in the same state.

WORKED EXAMPLE 2.7

In Section 2.1, Bob the student was introduced. What is the action in that problem?

The action corresponds to the number of hours studied on a specific day. According to Table 2.1, the action set is {0,1,2}.

2.4 MARKOV DECISION PROCESS

Compared to a Markov Process, which only includes states and transition probabilities, a **Markov Decision Process** (**MDP**) adds actions and rewards, enabling the modeling of goal-directed behavior.

Worked example 2.8 defines state and action in the context of the student Bob example, see Table 2.1. In this case, an action corresponds to the number of hours studied on a given day. Worked example 2.9 exemplifies an MDP with deterministic transitions. In the

WORKED EXAMPLE 2.8

Choose a state description from Worked example 2.2 and exemplify how the state can be updated from one day to another. Choose two days from Table 2.1.

$$\text{Choose: } state = (day, hours)$$

where the variable *hours* is the accumulated number of hours studied. The term action in the table below refers to the number of study hours chosen for the day represented by the start state.

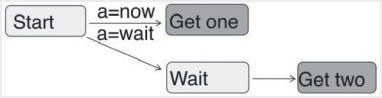
Start State	Action	New State
(1,0)	0	(2,0)
(1,0)	2	(2,2)
(5,5)	0	(6,5)

Comment: The new state (6,5) is a terminal state—the exam has taken place, and the result is now known. Based on Table 2.1, one can conclude that studying more hours likely wouldn't have improved Bob's score. Therefore, the chosen action (zero study hours on day 5) was probably a sensible decision.

WORKED EXAMPLE 2.9

Present an example of a Markov decision process with deterministic transitions.

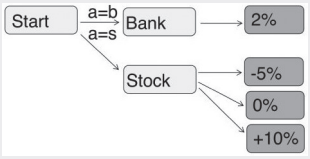
In this MDP, there are two available actions: "take one candy now" or "wait and receive two candies later". The process is illustrated below. Choosing the immediate reward (one candy) yields an instant reward and ends the process. Choosing to wait involves a transition to a wait state, followed by a subsequent state with a higher reward (two candies).



WORKED EXAMPLE 2.10

Present an example of a Markov decision process with stochastic transitions.

This example MDP models an investment decision. From the start state, one can choose between placing money in a bank or investing in stocks. The process is illustrated below. Choosing the bank results in a deterministic outcome of a 2% return. In contrast, choosing the stock leads to a stochastic outcome, with possible returns of −5%, 0%, or +10%. The bank action represents a low-risk, low-reward path, while the stock action reflects higher risk with the potential for both losses and higher gains.



example, the reward is fully known after an action is applied. In contrast, Worked example 2.10 is a system where one of the actions gives a random next state.

2.5 PARAMETER FITTING AND LEARNING RATE

This section explains the basic mechanism for updating parameters. Updating parameters is a core mechanism in RL, as it allows functions to be tuned to data. This section begins with the simplest possible memory: a single parameter. The general relation for updating a single parameter w from a reference value w_{ref} is

$$w \leftarrow w + \alpha \cdot \left(w_{ref} - w \right) \tag{2.1}$$

where α is the learning rate. The difference $\left(w_{ref} - w \right)$, the error, is computed to measure how far the current value w is from the reference value w_{ref}. This difference serves as the guiding signal for the update. The arrow $\leftarrow$ means "assigned to". The parameter w is updated by adding the current value with the learning rate times the error.

TABLE 2.2 Parameter Fitting Algorithm

1	`initialize w`	Set initial parameter value
2	**`while`** `fitCriterion`	Fitting loop
3	`e ← wRef-w`	Compute error
4	`w ← w+α·e`	Update parameter
5	**`end while`**	End fitting loop

Note: The `fitCriterion` is false when fitting is considered finishes. Can, for example, be when the error falls below a specific threshold.

It is now time for the first piece of pseudocode in this book; it is given in Table 2.2. Eq. (2.1) is applied on lines 3 and 4. The updating continues as long as the fit criterion is met. There is no single universal criterion; one common approach is to stop when the error falls below a specified threshold. In the analysis of executing the code in Table 2.2, presented later, the loop terminates after 50 iterations.

An example of parameter updating now follows. The parameter w starts with an initial value of zero, and the target value w_{ref} is one. The update process is carried out for four different learning rates, ranging from 0.1 to 0.8.

The results are presented in Figure 2.1. One observation is that the target value is reached more quickly with larger learning rates. Another observation is that the value of w changes most rapidly in the beginning, when the error, the difference from w_{ref}, is largest. More about learning rate setting is given in Worked example 2.11.

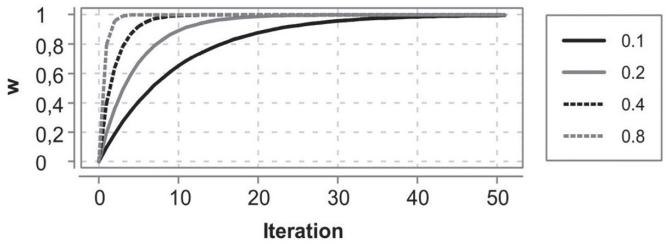


FIGURE 2.1 Updating. Learning rate is increasing from 0.1 to 0.8. A higher learning rate results in faster convergence of the parameter toward its reference value.

WORKED EXAMPLE 2.11

In general, what are possible learning rate values?

The learning rate must be greater than zero and less than or equal to one. If the learning rate is negative, the parameter updates move in the opposite direction of the gradient, leading to divergence. If the learning rate is greater than one, the updates can become too large, causing the parameters to diverge or oscillate.

2.6 OBSERVATION

The state concept was introduced in Section 2.2. A state represents a complete description of the environment at a given time, containing all relevant information. An **observation** is what the agent actually perceives from the environment, which in many real-world problems is only a partial or noisy representation of the true state. Mathematically, an observation is a subset of a state. This relationship is illustrated in Figure 2.2. The term "features" is explained in the next section.

A **partial observation** occurs when the agent can only perceive part of the environment's true state. In this case, the agent's decisions are based on incomplete or noisy information, which makes it harder to infer the full situation. In contrast, in the case of a **full observation**, the agent is provided with all information from the environment. Worked example 2.12 demonstrates the concept of full and partial observation.

2.7 THE FEATURE CONCEPT

The concept of **feature** is essential when fitting functions. Therefore, it will be explained before introducing function fitting. A feature is a property or characteristic used as input to a function. In Machine Learning, features are typically extracted from raw data (e.g., numerical values or categorical labels) and serve as the basis for predictions in learning tasks. As Figure 2.2 shows, the features may represent the entire observation or only a subset of it.

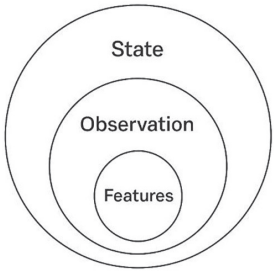


FIGURE 2.2 State-observation-features. Observation is a subset of state, and features is a subset of observation.

WORKED EXAMPLE 2.12

Use a self-driving car to describe the concept of partial observation.

A self-driving car illustrates the concept of partial observation well. The car does not have access to the full state of the environment, such as the exact positions or intentions of all nearby vehicles and pedestrians. Instead, it perceives a limited and sometimes noisy view through its sensors and cameras. This means that parts of the environment, like a pedestrian hidden behind a parked car or a vehicle just out of sensor range, remain unknown, and the car must make decisions based on incomplete information.

In RL, a feature can be seen as a reformulation of the environment's true state. A feature in RL is a variable used to describe the state s; in other words, it's what the agent "sees" when making decisions. The agent makes choices based on the combination of all features that define the current state. Good feature representation is essential for effective learning in RL. In Chapter 14, different feature candidates will be proposed and discussed for a specific problem.

Worked examples 2.13–2.16 are aimed to increase the intuition for the feature concept.

WORKED EXAMPLE 2.13

In Section 2.1, Bob the student was introduced. Use that problem to exemplify how a state represented by real numbers can be transformed into a state with discrete features.

In this case, the state is defined by the day (an integer) and the number of study hours, which may initially be a real number. To align with the discrete representation used in Section 2.1, a rounding operation is needed. For example, the state (2, 2.23) would be transformed into (2, 2), rounding the number of hours to the nearest integer or predefined discrete level.

Comment: This transformation makes it possible to apply tabular methods or finite-state models that require discrete state representations.

WORKED EXAMPLE 2.14

In Section 2.1, Bob the student was introduced. Use that problem to exemplify how a single feature can represent the outlook for achieving the maximum score on the exam.

According to Table 2.1, the maximum score is 5. Let *hours* denote the accumulated number of hours studied so far during the week. If the state variable *day* is 4 or 5, and the discrete state variable *hours* is 0, it is already too late to catch up—given the constraint that Bob can study at most two hours per day. Even if he studies the maximum amount for the remaining days, the total number of study hours will fall short of what is required for a high score. Mathematically, this can be formulated as:

$$isMaxPossible = \begin{cases} 0 & (day \in [4,5], hours = 0) \\ 1 & (else) \end{cases}$$

The table below exemplifies how different combinations of *day* and *hours* map to *isMaxPossible*.

day	*hours*	*isMaxPossible*
1	0	1
4	0	0
5	0	0