Preliminaries

Chapter 1

Introduction

This chapter presents an overview of the book. We provide a gentle introduction to decision-making under uncertainty and survey the structure and contents of each chapter. We also provide an overview of the notation used throughout the book.

1.1 A gentle start

Consider the environment in Fig. 1.1, depicting a household divided into 9 distinct areas. We want to deploy a household robot to assist the person living in this house in her daily chores. Our role is that of *robot programmer*: given a request by the human user—e.g., assisting her in some task in the kitchen—our role involves determining the most adequate sequence of actions required to complete the request. For ease of presentation, we restrict our attention to the navigation—in this case, reaching the kitchen.

The robot can move back and forth between any two communicating divisions except in the passages marked with a dashed line in the map. Dashed lines correspond to steps in the environment that only allow the robot to move in the indicated direction. There is some uncertainty associated with the motion of the robot and, eventually, also with its localization.

Let us then consider a request for assistance in the kitchen—which, from a navigation perspective, just means reaching the kitchen. As programmers of the robot, one approach would be to explicitly program a sequence of actions from its current location to the kitchen. Not knowing, beforehand, where the robot will be when the user issues the corresponding request, we must indicate the sequence of actions for every possible location.

Alternatively, we can compactly provide the robot with a *contingency plan* or *policy* that dictates, at each possible location, where the robot should move next:

- If in the Bedroom, move to HW1;
- If in the Dining room, move to the Kitchen;
- If in the Hall, move to the Living room;

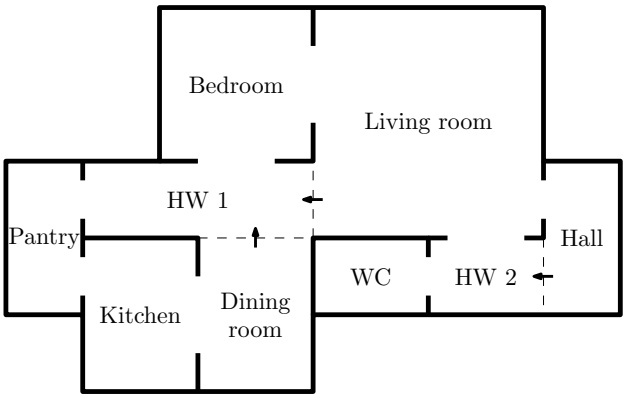


Figure 1.1 Household environment used as a running example throughout the book. Solid lines correspond to walls.

- If in Hallway 1, move to the Pantry;
- If in Hallway 2, move to the Living room;
- If in the Kitchen, do nothing;
- If in the Living room, move to Hallway 1;
- If in the Pantry, move to the Kitchen;
- If in the WC, move to HW2.

How good is the above policy given the request? At first sight, and if we assume that the actions of the robot always succeed, the policy appears to be the best possible, in the sense that it takes the least amount of steps from any location to the Kitchen.

However, if we consider that there is some degree of *uncertainty* in the motion of the robot, a better policy may exist. For example, if the robot often fails when crossing a step, it may be better to move from the Living room to the Bedroom, instead of going directly to Hallway 1.

◇

Throughout the book, we use the household robot scenario as a running example. This choice of running example has several appealing aspects:

- In spite of its simplicity, it is still remarkably close to real-world problems addressed in the decision-theoretic literature (Spaan and Vlassis, 2005). Therefore, it provides a flavor of some of the challenges faced in the application of the concepts discussed in the book to real-world problems.

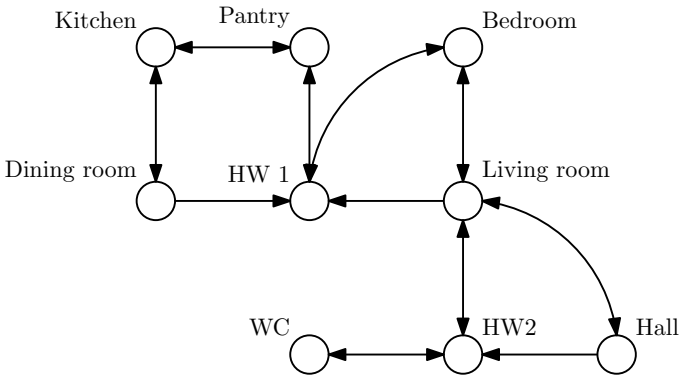


Figure 1.2 Graph representation of the household environment from Fig. 1.1.

- It is sufficiently simple to allow a thorough treatment. As such, we illustrate how to *model* it as a decision problem at different levels of detail and how to *solve* it using the algorithms presented in the text.
- The output of the algorithms—corresponding to the actions that the robot should select to reach its goal—can easily be visualized in terms of the environment, providing a natural intuition of how different components of the models and algorithms affect the resulting behavior of the robot.
- Finally, from a navigation perspective, the environment in Fig. 1.1 can be represented compactly as a graph, as depicted in Fig. 1.2. Such representation has been widely used in the robotics literature, and is referred as a *topological map*. The fundamental model used throughout the book to represent decision problems admits a similar representation, and intuition about one easily carries to the other.

The running example is complemented in each chapter with a number of additional examples and exercises.

1.2 Overview of the book

The book is divided in three parts, each containing three chapters. "Part 0" is a preliminary part used to introduce two important models that lie at the core of the decision-theoretic models and algorithms that we survey in the book. In **Chapter 2**, we introduce *Markov chains*. We describe several concepts of *stability* for Markov chains, that summarize the long-term term behavior of these processes. We also introduce Markov chain Monte Carlo, a method used extensively for both inference and optimization and which relies critically on the notions of stability surveyed.

Chapter 3 introduces the notion of *partial observability*, a central concept when, later on, we discuss decision problems where uncertainty affects both action and perception. A Markov chain with partial observability is known as a *hidden*

Markov model (HMM), and Chapter 3 is devoted to the study of this class of processes. We also present the main algorithms used for inference in HMMs.

The two models introduced in the preliminary chapters provide the background for the developments in Part I. Part I is devoted to formalizing and solving decision-making under uncertainty. We introduce principled ways to address the two main questions that arose in our running example, namely

(i) For a given task, how can we assess the quality of a policy in light of the uncertainty in the environment?

(ii) For a given task, what is the best policy in light of the uncertainty in the environment?

We start in **Chapter 4**, by introducing the simplest decision problem, where the decision-maker has a single decision to make. We introduce the theory of *expected utility* and discuss how this theory can be used computationally for making rational choices, by striking an adequate balance between uncertainty and cost.

Chapter 5 is the core chapter in Part I, addressing situations where the decision-maker has multiple inter-related decisions to make. To formalize this class of decision problems, we bring together the Markov chain model from Chapter 2 and the expected utility theory from Chapter 4, to introduce the *Markov decision problem* (MDP). Chapter 5 also goes over the main algorithmic approaches to solve MDPs; the ideas behind these methods permeate many of the developments in upcoming chapters.

Chapter 6 extends MDPs to include partial observability. The resulting model, known as *partially observable Markov decision problem* (POMDP) is significantly more powerful than MDPs as a modeling framework. Unfortunately, such representational power comes at a great cost in complexity. Chapter 6 goes over several classes of POMDP solution methods, while pointing out some of the main challenges involved in solving this class of problems.

MDPs and POMDPs are the two most important single-agent models for sequential decision-making. Both models are well studied and well understood, and have extensive applications in AI. Additionally, many other more complex models build on the properties of MDPs and POMDPs—namely in multiagent settings. We do not cover multiagent decision-making in this book, but many good references exist (see, for example, Shoham and Leyton-Brown, 2009).

The methods introduced in Part I of the book assume that the decision-maker has access to the model (MDP or POMDP) of the decision problem that it is meant to solve. Hence, one can think of the algorithms surveyed in Part I as *planning* algorithms: they are given a model of a decision problem and output a contingency plan/policy that solves such problem.

Part II is focused on the problem of *learning*—i.e., solving a decision problem when a model thereof is lacking. We cover two learning approaches:

- *Learning from examples*, where the decision-maker is provided with no information on the problem but an "expert" exemplifies the expected behavior.

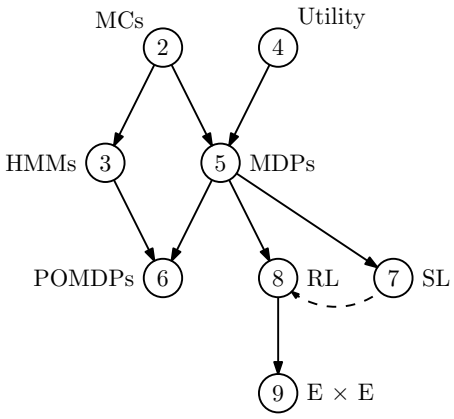


Figure 1.3 Structure of the chapters of the book.

This learning modality, known as *supervised learning*, is covered in **Chapter 7**. We discuss the relation between supervised learning and the sequential decision-making models from Part I, particularly with MDPs.

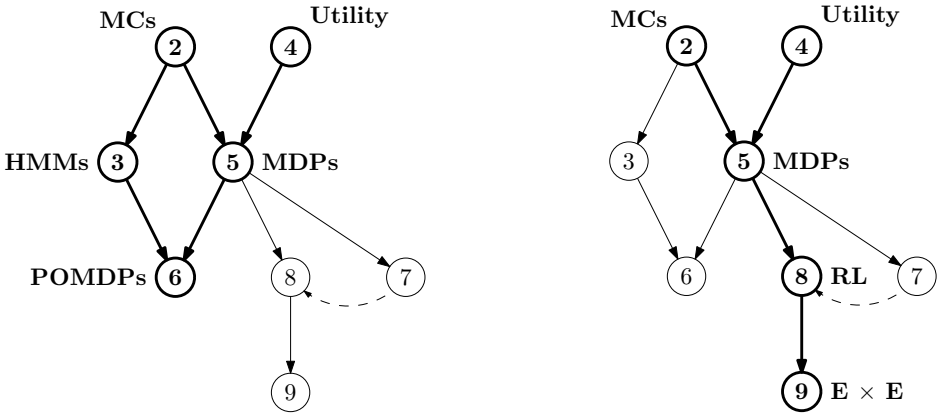
- *Learning by trial and error*, where the decision-maker must, on its own, determine what its decision problem is and how to solve it by experimenting. This learning modality, known as *reinforcement learning*, is covered in **Chapter 8** and builds extensively on the material from Chapter 5. We also introduce the *exploration vs exploitation* trade-off, central in reinforcement learning and reprised in Chapter 9.

Finally, **Chapter 9** explores in detail the trade-off between exploration and exploitation in learning. We distill this problem into its simplest form, the *multi-armed bandit* problem, and introduce the main algorithms designed to address this class of problems. We conclude by exploring deep ties between multi-armed bandits and the material in previous chapters, discussing ideas of *active learning*, *Monte Carlo planning* and surveying notable application scenarios that bring together important notions from reinforcement learning, planning in MDPs, and supervised learning.

1.2.1 Using this book

The book aims at being mostly self-contained. In practical terms, this means that we include proofs for the main results in each chapter, although most proofs are collected at the end of the corresponding chapter to avoid disrupting the text. Only key results are established in the main text. The proofs can be skipped without affecting the ability to understand the material, but the more interested reader is encouraged to go over them for a more in-depth study.

There is also some level of dependence between chapters, as summarized in Fig. 1.3. An arrow $A \longrightarrow B$ between Chapters A and B indicates that Chapter B should be read only after reading Chapter A. The dashed line between Chapters 7



(a) Sequential decision-making.

(b) Reinforcement learning.

Figure 1.4 Possible structure of specialized courses covering only part of the material in the book.

and 8 indicates that, although it may be helpful to read Chapter 7 before reading Chapter 8, the latter may still be fully understood even if Chapter 7 is skipped.

The book has been used for teaching a one-semester MSc-level course on learning and decision making. In this course, I cover all 9 chapters of the book, without going into the more demanding sections (marked with a star, $\star$). However, I believe that the book can also be used for specialized graduate-level courses on:

- Sequential decision-making, covering only Chapters 2 through 6 (Fig. 1.4(a));
- Reinforcement learning, covering Chapters 2, 4, 5, 8 and possibly 9 (Fig. 1.4(b)).

Each chapter contains a number of exercises, both for going over the concepts covered in that chapter, getting a more "practical feeling" about the material, as well as extending some of the results in the text.

1.2.2 Final remarks

We conclude this chapter with some observations concerning presentation choices that were made in an effort to make the nomenclature and notation consistent and uniform across the whole book.

For convenience of presentation, most of the book focuses on *finite* problems. Covering the material in full generality would require significantly heavier mathematical machinery and the benefits of such generality would be scarce. We choose to maintain the presentation as accessible as possible; we discuss generalizations to infinite/continuous domains when deemed convenient, mainly when such generalizations imply critical precautions or difficulties.

In our overview of supervised learning we adopt a decision-theoretic perspective. This involves minor changes in the notation and nomenclature when compared

with the literature. Also, our coverage of supervised learning has no pretension at being exhaustive, as there are many other more competent alternative references on the subject (see, for example, Bishop, 2006; Murphy, 2012). Instead, our goal is to highlight the role decision-making in this type of learning, highlighting the similarities with other learning paradigms. For this same reason, we focus essentially in *classification*, the "discrete branch" of supervised learning.

A note on mathematical notation

Throughout the text, we write $\stackrel{\text{def}}{=}$ with the meaning "defined as". We abbreviate "random variable" as "r.v.", "with respect to" as "w.r.t." and "with probability one" as w.p.1.

Boldface letters are used to designate vector/matrix quantities. Upright letters are used for random quantities. For example, we write x, $\mathbf{x}$ and $\mathbf{X}$ to denote a random scalar, a random vector, and a random matrix, respectively. In contrast, we use *slanted* letters for non-random quantities. For example, we write x, $\boldsymbol{x}$ and $\boldsymbol{X}$ to denote a non-random scalar, a vector and a matrix, respectively. We write $\mathbf{x} = \boldsymbol{x}$ to denote the event where the random vector $\mathbf{x}$ takes as value the vector $\boldsymbol{x}$. Except where explicitly noted, vectors are taken to be *column vectors*, with two exceptions: one is the representation of discrete probability distributions, and the other is the vector representation of sequences (see ahead).

We write x_j and x_j to denote the jth component of the vector $\boldsymbol{x}$ and the random vector $\mathbf{x}$, respectively. Similarly, we write $X_{i,j}$ and $\mathrm{X}_{i,j}$ to denote the ijth element of the matrix $\boldsymbol{X}$ and the random matrix $\mathbf{X}$, respectively. Sometimes, to emphasize the nature of $\boldsymbol{X}$ and $\mathbf{X}$ as matrices, we write $[\boldsymbol{X}]_{i,j}$ and $[\mathbf{X}]_{i,j}$ with the same meaning. Similarly, to emphasize the vectorial nature of $\boldsymbol{x}$ and $\mathbf{x}$, we sometimes write $[\boldsymbol{x}]_i$ and $[\mathbf{x}]_j$ to denote x_i and x_j. We write $\boldsymbol{X}_{i,:}$ and $\mathbf{X}_{i,:}$ to denote the ith row of the matrix $\boldsymbol{X}$ and the random matrix $\mathbf{X}$, respectively. Similarly, we write $\boldsymbol{X}_{:,j}$ and $\mathbf{X}_{:,j}$ to denote the jth column of $\boldsymbol{X}$ and $\mathbf{X}$, respectively. We write $\boldsymbol{X}_{:}$ to denote the *stacked form* of $\boldsymbol{X}$, i.e., the vector obtained from $\boldsymbol{X}$ by stacking all its columns,

$$\boldsymbol{X}_{:} = \begin{bmatrix} \boldsymbol{X}_{:,1} \\ \boldsymbol{X}_{:,2} \\ \vdots \\ \boldsymbol{X}_{:,n} \end{bmatrix}.$$

We write $\boldsymbol{e}^i$ to denote the ith coordinate vector, given by

$$\boldsymbol{e}^i = [0, \ldots, 0, 1, 0, \ldots, 0]^\top,$$

where the 1 appears in position i. We write $\boldsymbol{I}$ to denote the identity matrix, i.e., the matrix with column i given by $\boldsymbol{e}^i$. In both cases, the dimensionality is implicit from the context.

We use the letter t to represent the time index. We write either $\{x_0, \ldots, x_T\}$ or $\{x_t, t = 0, \ldots, T\}$ to denote a $T+1$-step sequence, and refer to T as *horizon* or *length* of the sequence. Similarly, we write either $\{x_0, \ldots, x_t, \ldots\}$ or $\{x_t, t \in \mathbb{N}\}$ to denote an infinite sequence. Sometimes, it is convenient to write a finite sequence

$\{x_t, t = 1, \ldots, T\}$ as a vector. When that is the case, we use the notation $\boldsymbol{x}_{0:T}$, where the vector $\boldsymbol{x}_{0:T}$ is understood as a *row vector*. In case of a vector-valued sequence $\{\boldsymbol{x}_t, t = 0, \ldots, T\}$, we write $\boldsymbol{X}_{0:T}$ to denote the matrix whose tth column corresponds to $\boldsymbol{x}_t$, and write $x_{k,t}$ to denote the kth element of $\boldsymbol{x}_t$. A similar notation is used for stochastic processes: given a stochastic process $\{\mathrm{x}_t, t = 0, \ldots, T\}$, we write it in vector form as $\mathbf{x}_{0:T}$.

Sets are usually represented using calligraphic letters such as $\mathcal{X}$. However, to avoid cumbersome notation, when it is clear from the context that we are referring to a set, we sometimes use uppercase letters to designate sets, as in A. We write $|\mathcal{X}|$ to represent the cardinality of set $\mathcal{X}$. We usually represent functions as *slanted*, uppercase letters, such as F. In contrast we use sans-seriffed letters to represent operators, such as M. Given a real-valued function $F : \mathcal{X} \to \mathbb{R}$, and an operator M, we write $\mathsf{M}F$ to denote the real-valued function obtained by applying M to F, and $(\mathsf{M}F)(x)$ to represent the value of $\mathsf{M}F$ at point $x \in \mathcal{X}$.

We write $\mathbb{P}[A]$ to denote the probability of event A. For example, we write $\mathbb{P}[\mathrm{x} = x]$ to denote the probability that the random variable x takes the value x. Given a r.v. x taking values in some set $\mathcal{X}$ and real-valued function $F : \mathcal{X} \to \mathbb{R}$, we write $\mathbb{E}[F(\mathrm{x})]$ to denote the expectation of F, defined as

$$\mathbb{E}[F(\mathrm{x})] \stackrel{\text{def}}{=} \sum_x F(x)\ \mathbb{P}[\mathrm{x} = x].$$

Given a distribution p over a set $\mathcal{X}$, we write $\mathrm{x} \sim p$ to indicate that the r.v. x is distributed according to p. We write $\mathbb{E}_{\mathrm{x}\sim p}[F(\mathrm{x})]$ or $\mathbb{E}_p[F(\mathrm{x})]$ to denote the expected value of F under the p, defined as

$$\mathbb{E}_{\mathrm{x}\sim p}[F(\mathrm{x})] = \mathbb{E}_p[F(\mathrm{x})] \stackrel{\text{def}}{=} \sum_x F(x)p(x).$$

When $\mathcal{X}$ is a discrete set, it is sometimes convenient to represent the distribution p as a vector. When that is the case, we write $\boldsymbol{p}$ to denote the *row vector* with xth component given by $p(x)$. Finally, given an event A, we write $\mathbb{I}[A]$ to represent the *indicator of event* A, defined as:

$$\mathbb{I}[A] \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if event } A \text{ is true/occurs;} \\ 0 & \text{otherwise.} \end{cases}$$

Thus we can write, for example,

$$[\boldsymbol{I}]_{i,j} = \mathbb{I}[i = j].$$