1 Introduction

Artificial Intelligence is commonly defined as the study of computational methods to produce intelligent behavior. As a definition, it borders on circular: what is *intelligent behavior?* We might attempt to go a step further, and say that intelligent behavior is behavior which is appropriate to the given situation, but then we need to unpack what is *appropriate* (and for that matter, what is *behavior* or a *situation*, and so on). Another approach is to work from examples: humans are commonly supposed to be intelligent and general-purpose computers, for all of their computational ability, are generally taken to fall short of being intelligent; Artificial Intelligence is then the study of how to bridge this gap. Better still, is to work from concrete examples of a real-world tasks. We would like a robotic lawn mower that recognizes a garden hose, or children's toys left in the yard, or large rocks, or whatever foreign object has found its way on to our property, and avoids running over them, while attacking weeds with extreme prejudice and otherwise doing its best to maintain the grass. How can we capture the difference between these unanticipated discoveries, and how to respond to them? Artificial Intelligence is a very broad field, spanning from perception of the world, to evaluating possible courses of action, to acting. In this book, we focus on the computational study of one approach to the problems that arise in Artificial Intelligence.

1.1 Our approach: the PAC semantics for knowledge

The approach we study here is based on the *Probably Approximately Correct (PAC) semantics* for *knowledge*. *Knowledge* refers to descriptions of possible states of the world, and there are three ways in which we can come to possess it:

1. *innate*: we may be born or designed with it,
2. *declarative*: we may receive an explicit representation of it, or
3. *inductive*: we may infer it from experience.

The PAC semantics, introduced in Chapter 2, assigns meaning to such descriptions in a way that allows us to treat knowledge from all three of these sources on equal footing.

We are not claiming that intelligent behavior *must* be based on knowledge. We are also not claiming that knowledge *must* be based on the PAC semantics. But, we certainly see value in acquiring knowledge from declarative sources (such as this book).

Indeed, as we will see when we study inductive learning starting in Chapter 4, there are some severe limits to the inductive acquisition of knowledge. Thus, we desire an approach that can incorporate declaratively acquired knowledge. While there are other approaches to inductive learning, many of these do not allow us to interpret their results in terms of knowledge in the same sense as declarative statements, and it is not clear how to integrate declarative knowledge into these other approaches to inductive learning. By contrast, we develop approaches to integrate declarative knowledge into inductive learning in Chapters 6 and 8 for the PAC semantics.

At the same time, as we discuss in Chapter 6, declarative knowledge is perhaps surprisingly limiting as a means to achieve intelligent behavior. Much crucial knowledge is never represented declaratively, and we do not have a sufficiently rigorous grasp of it to design systems that possess it innately; for example, what should our robot consider to be a *"weed"*? This leaves only the inductive route to systems that possess such knowledge. Unfortunately, as we review in Chapter 2 (and develop further in Chapter 6), the classical conception of knowledge from formal logic is not fit for this purpose: it does not permit inductive generalization of knowledge from particular instances. By contrast, the PAC semantics gives a relatively simple model of inductive learning.

Having acquired such knowledge from our combined sources, our applications will require that we use it to answer questions, for example towards determining a course of action. There are a variety of such problems that we collectively refer to informally as *reasoning*. We review classical deductive reasoning based on entailment in Chapter 5, and develop a wider variety of reasoning problems, including *abductive* reasoning, in the PAC semantics in Chapters 6 and 8; in the final part of the book (Chapters 9, 10, and 11) we consider how to extend the PAC semantics to encompass more sophisticated kinds of reasoning.

Reasoning problems are frequently computationally intractable; approaches based on solving intractable problems are typically not viable for applications. Thus, our approach to the subject, beginning in Chapter 5, is to focus on when reasoning is tractable. Classically, work on computational reasoning focused on the syntactic features of the languages used to state knowledge and pose questions: if *any* such questions are intractable to answer, then the problem was considered to be intractable (in the worst case). Unfortunately, as we will see, even very simple logics pose intractable reasoning problems. Rather than restricting the features of the language, which only leaves a rather impoverished family of representations, we develop an approach based on *proof complexity*, which grades questions based on quantitative features of the reasoning needed to reach the corresponding answers. We will be able to develop algorithms for reasoning with running time that scales with these measures of proof complexity: we can prove that the algorithms are efficient when the question can be answered by sufficiently simple reasoning. We will see that this approach allows us to consider efficient algorithms for reasoning problems with more expressive languages. By comparing the families of proofs where given algorithms are efficient, we are also able to compare the relative power of different approaches.

Indeed, the final distinguishing characteristic of our approach is our focus on *guarantees of computational efficiency* throughout the book. The PAC semantics is

a framework that, in addition to being a simple framework that encompasses both declarative and inductive knowledge, and allowing us to pose deductive, inductive, and abductive inference problems, also allows the design of computationally efficient algorithms for these problems. The focus on computational efficiency, beginning in Chapter 3, will allow us to clearly perceive one of the main tensions in Artificial Intelligence, between *expressive power* and *computational tractability*: on the one hand, we desire rich languages for representing knowledge and methods for solving a wide variety of reasoning problems that we may encounter in the world. But, richer languages and wider families of problems are inherently harder to solve, and in some cases no efficient method is possible. In general, there is a trade-off between the breadth of problems we can solve and the computational cost of solving them. We will see that sometimes by reducing the scope of our tasks in other ways, for example, by foregoing an explicit representation of learned knowledge in a tight integration of learning and reasoning, we can obtain computationally feasible methods in Chapters 6 and 8.

1.2 Overview of contents

We now give brief summaries of each chapter, highlighting the main conceptual points raised in each.

1.2.1 Part I: Truth and Complexity in Learning and Reasoning

The first part of the book introduces the PAC semantics, reviews the quantitative study of representations and computation, and develops learning and reasoning as separate topics.

1.2.1.1 Chapter 2: Probably Approximate Truth

We first recall the classical semantics of mathematical logic in Section 2.1 and its limitations as a semantics of learned knowledge in Section 2.2. We then introduce the PAC semantics and discusses some basic properties and methods for evaluating the *approximate* validity of statements in Section 2.4.

1.2.1.2 Chapter 3: Representational Tradeoffs

We next introduce knowledge representations, and develop the notion of relative expressive strength of different representations. We then introduce some basic computational tasks associated with knowledge representations and observes that expressive representations feature inherently intractable tasks. Indeed, Section 3.6 recalls the theory of *NP-completeness*, which directly connects expressivity and intractability.

1.2.1.3 Chapter 4: The Inductive Learning Task

We introduce the basic supervised learning task as a computational problem, and how Occam's Razor provides inductive generalization in Section 4.1. Important examples

of learning algorithms for standard representations are introduced in Sections 4.2 and 4.4, and learning via reductions is introduced in Section 4.3.

Sections 4.5-4.6 develop the crucial distinction between the breadth of representations that an algorithm can learn, and the representations that the algorithm uses as its output: indeed, we see that sometimes by using a more expressive output representation (an *"improper"* learning algorithm), we can obtain an efficient algorithm, where finding a representation in a more restrictive output language is intractable. Of course, the downside is that this representation in a more expressive language may result in us needing to solve a harder reasoning problem later.

In any event, we finally develop the adversarial connection between learning and cryptography in Section 4.7, leading to the conclusion that there are ultimate limits to what can be efficiently learned: in Section 4.8, we show learning is intractable for even seemingly simple representations (special cases of two-layer neural networks), regardless of what output representation is used.

1.2.1.4 Chapter 5: The Deductive Reasoning Task

We introduce the basic deductive reasoning task based on entailment in Section 5.1, which we quickly determine to be intractable in the worst case for almost all languages of interest. To be able to identify feasible reasoning tasks, we therefore subsequently introduce proof systems in Section 5.2, and observe that as with knowledge representations, we can (sometimes) directly compare the expressive power of proof systems.

We develop our metrics of proof complexity for the system of resolution in Section 5.4. We establish in Section 5.5 that the reasoning task associated with resolution is intractable in general, and therefore that many standard reasoning algorithms must have poor worst-case behavior, by a connection developed in Section 5.3. In contrast, we show in Section 5.7 that small modifications of these standard algorithms are efficient for problems with limited proof complexity.

Finally, paralleling Section 4.7, in Section 5.8 we show that cryptography also has an adversarial relationship with automated reasoning, and thus we can identify proof systems with inherently intractable reasoning problems, regardless of the method used.

1.2.2 Part II: A Unified Framework for Learning and Reasoning

The second part of the book is aimed at developing an integration of the learning and reasoning tasks in the PAC semantics, and the role of these tasks in Artificial Intelligence. This is the central part of the book.

1.2.2.1 Chapter 6: Learning to Reason with Full Examples

We first consider a full-information setting, directly extending the supervised learning model from Chapter 4; it first develops a basic entailment reasoning task in this setting in Section 6.1, which turns out to be quite easy, in contrast to the standard entailment task from Section 5.1. In Section 6.2, we next examine the task of learning imperfect representations, that feature "exceptions", which turns out to be much more challenging computationally. Unfortunately, we find in Section 6.3 that this kind of

exception-tolerance is essential for Artificial Intelligence, as it is one of the key features of the essential, non-declarative, "common sense" knowledge. Fortunately, a natural extension of the approach to the full-information entailment task based on reasoning in a given context, developed in Section 6.4, features such exception-tolerance, along with many of the other desiderata for reasoning with common-sense knowledge. We conclude the chapter by considering the problem of choosing a "best" context for the reasoning in Section 6.5, and the related problem of abductive inference in Section 6.6.

1.2.2.2 Chapter 7: Inductive Learning from Partial Examples

The main weakness of the approach in Chapter 6 is that it requires complete observations of the values of all attributes, which are not available in many situations. So, in Chapter 7, we extend the learning task from Chapter 4 to the task of supervised learning from *partial* examples, that may omit specifying the values of some attributes.

The basic partial information model is developed in Section 7.1, and a theory of learning from such examples is developed in Section 7.2, leading to a general method for extending learning algorithms for monotone representations to learn from partial examples, in Section 7.3. Indeed, moreover, we also present an algorithm for learning halfspaces from partial examples in Section 7.3, which are one of the most expressive representations with efficient learning algorithms. At the same time, in Section 7.4, we see that some representations that were learnable from complete examples are not efficiently learnable from partial examples.

One of the main limitations of the approach to learning from partial examples based on *bounded concealment* that we developed in Section 7.2 is that it does not allow us to use the kind of reductions considered in Section 4.6, that enrich the feature space. In Section 7.5 we develop an alternative approach that can allow learning using such expanded features (equiv., learning of composed representations) under narrower conditions, when the values of the functions can be determined by a more limited kind of *partial evaluation* on the examples.

1.2.2.3 Chapter 8: Learning to Reason with Partial Examples

Finally, Chapter 8 leverages the partial information model developed in Chapter 7 to extend the formulation of learning and reasoning from Chapter 6 to allow the use of partial examples. Crucially, these partial examples now may encompass situations where some of the attributes are latent and described only in our background knowledge, or observed in some situations and not others, such as when a robot can only observe its immediate surroundings. We can no longer use the simple solution for the full-information problem developed in Sections 6.1 and 6.4, as classical reasoning to cope with the unobserved attributes now appears as an inherent part of the task.

We first consider a simple approach in which we explicitly learn chaining rules to fill in missing attributes, in Section 8.1, and analyze an algorithm that applies several rounds of such chaining in Section 8.2: although the amount of data needed grows exponentially with the number of rounds, we find that multiple rounds may indeed enable us to accurately predict more of the missing attributes. We next develop a method of *implicit* learning in Section 8.3, that enables us to design efficient

algorithms for reasoning from learned knowledge with exception-tolerance, as needed for common-sense reasoning. Finally, in Section 8.4, we consider the challenges in extending the full-information abduction task of Section 6.6 to the partial information setting using this implicit learning technique, and develop an efficient algorithm for such a task.

1.2.3 Part III: Learning with Advanced Reasoning Semantics

In the third part of the book, we consider several important extensions to the reasoning problem.

1.2.3.1 Chapter 9: Learning and First-Order Relational Reasoning

The first extension, in Chapter 9, considers first-order relational logic, i.e., with statements featuring the quantifier symbols $\forall$ (*"for all"*) and $\exists$ (*"there exists"*). Indeed, the rest of the book considered only propositional Boolean logic on small domains, with the complexity of our algorithms scaling polynomially with the number of these Boolean attributes. We recall the language and classical semantics of first-order logic in Section 9.1.

In general, as we recall in Section 9.4, reasoning in first-order logic is impossible: correct algorithms do not exist. But, for some more restricted families of first-order formulas, we can reduce the reasoning problem to various propositional reasoning problems. We first recall the naïve *propositionalization* technique for small, explicit domains in Section 9.2, and the more sophisticated *"grounding trick"* that can capture unknown or infinite domains in Section 9.3. Thus, we can leverage some of the methods we developed in Chapter 5 for efficient reasoning with first-order expressions. We also consider algorithms that can help search for *"lifted"* resolution proofs, that directly use first-order expressions with function symbols, in Section 9.5.

We next turn to the supervised learning problems associated with first-order representations in Section 9.6, and discover that this is, unfortunately, almost always intractable. In spite of this, in Section 9.7, we find that reasoning with *implicitly* learned first-order representations can be solved by efficient algorithms that leverage the grounding trick.

1.2.3.2 Chapter 10: Learning and Non-Monotonic Knowledge Representations

The second extension considers knowledge representations that, in contrast to classical logics, have *non-monotonic* semantics: the introduction of new knowledge can cause previously valid conclusions to be withdrawn. We note that not all important kinds of common sense knowledge can be learned in the partial information setting we developed in Chapters 7 and 8, particularly those that refer to unobservable attributes, and so we require alternative approaches.

The kind of representations we consider in Chapter 10 are based on *arguments*: in Section 10.1, we first consider the semantics of such arguments at an *abstract* level, i.e., ignoring the structure of the arguments themselves and only considering which arguments *"defeat"* which others. We introduce an efficient algorithm for deciding

which arguments to accept using such semantics in Section 10.2. Next, we consider a kind of argument system based on *"prioritized"* chaining rules, where the relative "priority" of the rules specifies which ones may override which others, and develop an efficient algorithm for determining what conclusions we may draw from these prioritized rules in Section 10.3.

Finally, in Section 10.4, we develop an efficient algorithm for acquiring a set of prioritized rules from a human teacher. This is a variant of learning that is distinct from the supervised learning model we considered previously.

1.2.3.3 Chapter 11: Learning and Causal/Temporal Reasoning

The third extension, in Chapter 11, concerns learning to reason about cause and effect in temporal settings, where the observations span a sequence of points in time. We previously considered how we could solve a basic version of the problem of reasoning about actions using classical reasoning in Section 5.6. In this simple version of the problem, we supposed that the rules governing the domain were fully known. In Section 11.1, by contrast, we consider a model for learning such domains when our observations of the domain are limited in various respects. We then develop an algorithm for solving a basic version of the problem in Section 11.2. We next consider the effect of the agent missing some of the intermediate steps in the environment's evolution in Section 11.3, and see that this leads to a considerably more challenging problem. Finally, we consider the issues that arise when the agent is making predictions that inform its choice of actions, in Section 11.4.

1.2.4 Appendix A: VC Dimension for Numeric Parameters and Sample Complexity

Finally, in Appendix A, we discuss the amount of data inherently required for learning. We postpone this discussion to an appendix because our focus in this book is primarily on computational aspects. Nevertheless, the data requirements of learning various representations is an important and well-understood topic in learning theory: there is a natural parameter of representations, called the *VC-dimension*, that precisely captures the amount of data that is necessary and sufficient to learn a given class of representations. We define the parameter in Section A.1, and show in Sections A.2 and A.3 that it respectively captures the amount of data that is necessary and sufficient for learning. More data is indeed required to learn more expressive representations. An important corollary of these bounds is that in order for learning to be possible at all, we must restrict the expressive power of the representations we are seeking to learn.

We give an application of the VC-dimension sample complexity bound to establish the efficient learnability of halfspaces (also known as linear threshold functions) in Section A.4. Indeed, we observe that the learning problem is essentially equivalent to linear optimization.

We then discuss the related but subtly different question of how the accuracy of a learned predictor scales as we collect more data from a distribution. Although this sounds like it is the same question, the difference is that the VC-dimension bounds are

a universal bound: they specify an amount of data that is sufficient *for all* distributions, whereas in Section A.5 we suppose that some distribution is fixed and then consider how the error rate shrinks as more data is collected from that distribution. The effect of this subtle switch is striking: whereas the VC-dimension bounds suggest that the error shrinks linearly with the amount of data, in many cases it shrinks *exponentially* with the amount of data for a fixed distribution.

We finally give one more important application of the VC-dimension generalization bound, to *boosting* in Section A.6. In this final topic, we show that in order to learn highly accurate predictors, it suffices to give efficient algorithms that can find predictors with accuracy only slightly greater than a random guess. This highly accurate predictor is obtained by combining a sequence of such "weak" predictors with a weighted vote. A corollary of boosting is that if learning of some family of representations is intractable, then it is impossible to predict the labels of the representations better than chance—i.e., they are indistinguishable from random. This gives a converse to the connection between learning and cryptography: not only do cryptographic problems give hard learning problems, but hard learning problems can be used as cryptographic problems. And, indeed as we alluded to in Section 4.8, cryptographic problems based on hard learning problems are important candidates for post-quantum cryptography.

1.3 Prerequisites and target audience

This book is aimed primarily at the advanced undergraduate or beginning graduate level. The book assumes familiarity with basic data structures and the analysis of algorithms: students should be familiar with the library of basic data structures and their capabilities. They should also be comfortable reading pseudocode, analyzing its asymptotic worst-case running time, and reasoning about the function it computes using inductive arguments. In service of such analysis, we will also rely on some basic counting, including binomial coefficients.

More substantially, it assumes a solid understanding of discrete probability, at the level used to analyze randomized algorithms in an upper-level undergraduate algorithms course, e.g., based on Cormen et al. (2022) or Kleinberg and Tardos (2006). This includes the basic concepts of probability distributions, random variables, their expected value, and conditional expectation. Students should be comfortable reasoning about independence and ideally be familiar with standard tools for establishing probability bounds, such as Markov's Inequality and the (weak) law of large numbers. Students who have completed such an algorithms course should have all of the necessary background, but the book does not rely on any of the specific paradigms of the design and analysis of algorithms covered in such courses. Thus, a sufficiently solid background in these more basic areas should also be adequate, for example as covered by Harchol-Balter (2023).

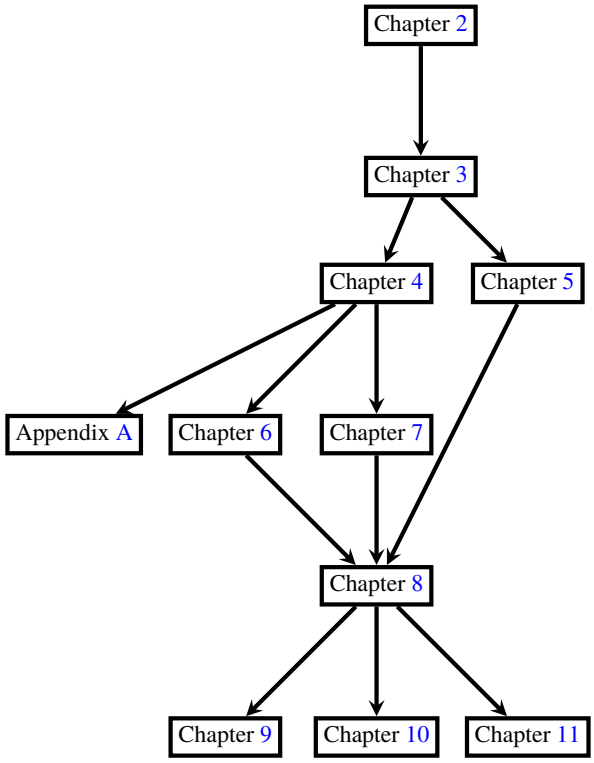

Figure 1.1 Dependencies among the chapters

1.4 Sample course outline(s) and notes to instructors

This book is primarily designed to facilitate a standard semester course (12 or 13 weeks long). A typical outline for such a course is as follows:

Chapter 2	Introduce PAC semantics	0.5 weeks
Chapter 3	Knowledge representations & complexity	1 week
Chapters 4 + 5	Introduce learning & reasoning	5 weeks
Chapter 6	Full-information integration	1 week
Chapter 7	Learning from partial information	1.5 weeks
Chapter 8	Partial-information learning and reasoning	1.5 weeks
	Advanced topic(s)	1.5–2.5 weeks

Table 1.1 Outline of a standard semester-long course on learning and reasoning

For the advanced chapters, Chapters 10 and 11 each can reasonably be covered in 1.5 weeks, whereas Chapter 9 and Appendix A each contain 2.5 weeks' worth of material. Of course, an instructor can choose to cover a portion of each of these.

In the outline above, we stress that we *strongly recommend* covering Chapter 4 and Chapter 5 in parallel, for example spending one lecture per week on a topic from Chapter 4 and one on Chapter 5. The main reason, put bluntly, is that this ensures that

students retain familiarity with both topics when the time comes to integrate them in Chapters 6 and 8. Otherwise, whichever topic is presented first has often been "evicted" from memory by the time the second has concluded.

Furthermore, as noted in the overview, there are strong parallels between learning and automated reasoning: in both cases, there is a distinction between the complexity of the target representation or proof, and the complexity of the representations considered by the algorithm to solve the task. Solving tasks with more complex targets requires greater computational resources, whereas using more complex representations often reduces the resources required. For the sake of developing tractable algorithms, we need to focus on relatively simple target representations. And, moreover, if we drop consideration of the complexity of representations being used, then the inherent limitations on solving both kinds of tasks are uncovered by considering their relationship to cryptography. Presenting the topics side-by-side provides an opportunity to highlight these parallels.

We note that in each instance of this correspondence between the two subjects, the constructions and arguments that arise in learning are always simpler, and should be presented first. For example, compare Section 4.5 on the hardness of proper learning and Section 5.5 on the hardness of automating resolution: the former is a straightforward NP-completeness reduction, while the latter relies on the machinery of proof complexity developed in Section 5.4, to support a much more sophisticated reduction. Or, compare the efficient learning algorithm for decision lists in Section 4.4 to the efficient reasoning algorithm for space-bounded treelike resolution in Section 5.7. (We remark that Algorithm 8 can be adapted to properly learn bounded-rank decision trees, generalizing Algorithm 2.) In any case, this is natural to expect, as the knowledge representations that are the targets of learning comprise individual lines of the proofs used in reasoning. Proofs are strictly more complex.

Depending on the context of the institution and students' backgrounds, the book could also support a more advanced course. For example, supposing students had a prior course in machine learning theory, for example following Shalev-Shwartz and Ben-David (2014), Mohri et al. (2018), or Kearns and Vazirani (1994), that covered the contents of Chapters 3-4 and Appendix A, one could follow the following outline for a course aimed at introducing reasoning and then integrating learning into it:

Chapter 5	Introduce propositional reasoning	2.5 weeks
Chapter 6	Full-information integration with learning	1 week
Chapter 7	Learning from partial information	1.5 weeks
Chapter 8	Partial-information learning and reasoning	1.5 weeks
Chapter 9	Relational reasoning and learning	2.5 weeks
Chapter 10	Non-monotonic reasoning and learning	1.5 weeks
Chapter 11	Temporal/causal learning and reasoning	1.5 weeks

Table 1.2 Outline of an advanced semester-long course on learning and reasoning

More generally, the dependencies among the chapters are depicted in Figure 1.1, and other selections are possible. Naturally, it is also possible to teach an analogous course that introduces learning and then integrates it into reasoning, assuming a prior course

on knowledge representation and reasoning or on proof complexity. In the former case, following a less-quantitative course on knowledge representation, the instructor may still need or wish to cover the sections of Chapter 5 introducing proof complexity, but presumably will already have covered portions of Chapters 9 and 10 that are quite standard in such courses.

1.4.1 On the exercises

We have included exercises for each chapter, serving one or more of the three (non-exclusive) following purposes:

1. Practicing concepts
2. Making a conceptual point
3. Developing an important technique

In particular, some of the proofs are deferred to exercises, and we strongly recommend assigning these as homework problems, e.g., Exercise 5.6 on the restriction-closure property of resolution. Some of these are technical exercises that build on one another: for example, Exercise 4.3 introduces the *rank* parameter of trees, that is related to the *clause space* complexity of treelike resolution in Exercise 5.5. On the one hand, this concept underpins the state-of-the-art in learning decision trees in Exercise 4.4, and on the other hand characterizes reasoning with the unit propagation inference rule in Exercise 5.7. Our underlying philosophy is that these kinds of techniques are most effectively learned by direct engagement. Their use and significance is much harder to grasp from passively reading about their applications. Experience in attempting to solve problems is essential to mastering this material.

We have attempted to provide hints for the exercises and broken them into steps to keep their level of difficulty reasonable. We do not provide sample solutions to the exercises. It is true that in many cases, the exercises are either well-known problems or derived from research articles, and it is not hard to find their solution by searching the Internet or, more recently, by asking a chatbot. Still, we chose not to make finding the solutions even more "frictionless" by collecting them in the text, in an effort to encourage students to spend some time and effort thinking about them. As explained above, simply reading the solutions undermines their pedagogical value. Thus, furthermore, in cases where the solutions are *not* readily available, we chose to leave them that way.