1

Introduction

To define *reinforcement learning* (RL), it is first necessary to define *automatic control*. Examples in your everyday life may include the cruise control in your car; the thermostat in your air conditioning, refrigerator, and water heater; and the decision-making rules in a modern clothes dryer. There are sensors that gather data, a computer to take the data to understand the state of the "world" (is the car traveling at the right speed? Are the towels still damp?), and based on these measurements an algorithm powered by the computer spits out commands to adjust whatever needs to be adjusted: throttle, fan speed, heating coil current, etc. More exciting examples include space rockets, artificial organs, and microscopic robots to perform surgery.

The dream of RL is automatic control that is truly automatic; without any knowledge of physics or biology or medicine, an RL algorithm tunes itself to become a super controller: the smoothest ride into space, and the most expert microsurgeon!

The dream is surely beyond reach in most applications, but recent success stories have inspired industry, scientists, and a new generation of students.

DeepMind's AlphaGo set the world on fire following the defeat of Go champion Fan Hui in 2015, and Lee Sedol the following year (a story told in the 2017 film [185]). Soon after was the astonishing sequel AlphaZero, which learns to play chess and Go by "self play" without any help from experts [150, 310, 322].[1]

1.1 What You Can Find in Here

Reinforcement learning today rests on two pillars of equal importance:

1. Optimal control: the two most famous RL algorithms, TD- and Q-learning, are all about approximating the *value function* that is at the heart of optimal control. Similarly, actor-critic methods are based on state feedback, which is motivated by optimal control theory.
2. Statistics and information theory, especially the topic of exploration, as in bandit theory. Consider the annoying ads on YouTube, which provide one example of Google's

[1] These success stories follow a longer history of demonstrations based on board games. An earlier breakthrough is Tesauro's implementation of TD-learning with neural network function approximation to obtain an RL algorithm for backgammon [350]. The 1997 victory of IBM's "Deep Blue" over chess champion Garry Kasparov was also front-page news [373]. While this algorithm resembles *model predictive control* rather than any RL algorithm described in this book, we will be liberal in our taxonomy of control and RL since we want to benefit from the best of both.

exploration: "will Diana click???" [171, 216, 307]. Exploration in RL is a rapidly evolving field – it will surely generate many new books in the years to come.

The big focus of the book is the first pillar, emphasizing the geometry of optimal control, and why it should not be difficult to create reliable algorithms for learning. We will not ignore the second pillar: motivation and successful heuristics will be explained without diving deeply into the theory. The reader will learn enough to begin experimenting with homemade computer code, and have a big library of options for algorithm design. Before completing half of the book, I hope that a student will have a solid understanding of why these algorithms are expected to be useful, and why they sometimes fail.

This is only possible through mastery of several fundamentals:

(i) The philosophical foundations of control design.
(ii) The theory of optimal control.
(iii) Ordinary differential equations (ODEs): stability and convergence. The ODE method, including translation to algorithm.
(iv) Basics of machine learning (ML): function approximation and optimization theory.

Any reader who knows the author will wonder why the list is so short! The following topics are seen as fundamental in RL theory, and are fundamental to much of my research:

(i) Stochastic processes and Markov chains.
(ii) Markov decision theory.
(iii) Stochastic approximation and convergence of algorithms.

Yes, we will get to all of this. However, I want to make it clear that there is no need for probability theory to understand many of the important concepts in RL.

The first half of the book contains RL theory and design techniques without any probability prerequisites. This means we pretend that the world is purely deterministic until the surname "Markov" appears in the second half of the book. Justification comes in part from the control foundations covered in Chapters 2 and 3. Do you think we modeled the probability of hitting a seagull when we designed a control system to take astronauts to the moon? The models used in traditional control design are often very simple, but good enough to get insight on how to control a rocket or a pancreas.

Beyond this, once you understand RL techniques in this simplified setting, it does not take much work to extend the ideas to the more complex probabilistic realm.

Among the highlights of Part I of the book are the following

► *ODE design.* The ODE method has been a workhorse for algorithm *analysis* since the introduction of the stochastic approximation technique of Robbins and Monro in the early 1950s [301]. In this book, we flip the narrative, and start off in the continuous time domain. There is tremendous motivation for this point of view:

(i) We will see that an ODE is much simpler to describe and analyze than the discrete-time noisy algorithm that is eventually implemented.
(ii) Remarkable discoveries in the optimization literature reinforce the value of this approach: first design an ODE with desirable properties, and then find a numerical analyst to implement this on a computer [318]. It is now known that the famous acceleration techniques of Polyak and Nesterov can be interpreted in this way.

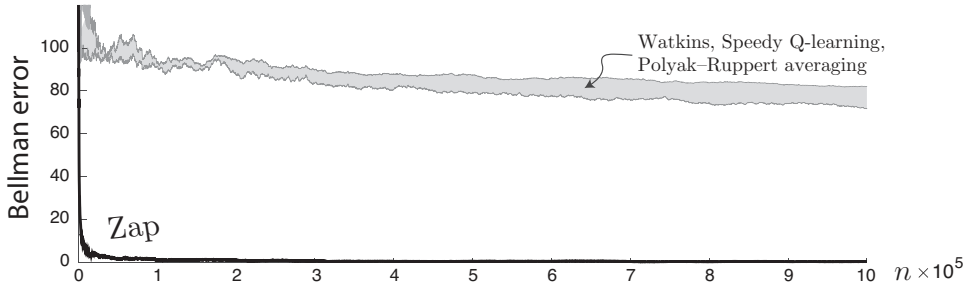

Figure 1.1 Maximum Bellman error $\{\overline{\mathcal{B}}_n : n \geq 0\}$ for various Q-learning algorithms [107].

(iii) Zap Q-learning will be one of many algorithms described in the book. It is a particular ODE design based on the Newton–Raphson flow introduced in the economics literature, and first analyzed by Smale in the 1970s [325]. The Zap ODE is universally stable and consistent, and hence when translated with care, provides new techniques for RL design [89, 107, 110, 112]. The power of Zap design is illustrated in Figure 1.1 – see Section 9.8.2 for details.

► *Quasistochastic approximation.* The theory of "stochastic approximation" amounts to justifying a discrete time algorithm based on an ODE approximation. An understanding of the general theory requires substantial mathematical background.

The reader will be introduced to stochastic approximation without any need to know the meaning of "stochastic." This is made possible by substituting mixtures of sinusoids for randomness, which is justified by an emerging science [11, 51, 52, 87, 93, 228]. Not only is this more accessible, but the performance in application to *policy gradient* methods in RL is remarkable.

The plots shown in Figure 1.2 are based on experiments described in Chapter 4, comparing exploration using sinusoids versus traditional random "independent and identically distributed (i.i.d.)" exploration. The histograms were generated through 1,000 independent experiments. The traditional approach labeled "1SPSA" requires additional training of many orders of magnitude when compared to quasistochatic approximation (QSA).

► *Batch RL methods and convex Q-learning.* One of the founders of AlphaGo admits that extension of these techniques is not trivial: "This approach won't work in more ill-structured problems like natural-language understanding or robotics, where the state space is more complex and there isn't a clear objective function" [150].

In applications to controlling building systems, the energy grid, robotic surgery, or autonomous vehicles, we need to think carefully about more structured learning and control architectures, designed so that we have a reliable outcome (hopefully with some guarantees on performance as well as the probability of disaster). The basic RL engine of AlphaZero is deep Q-learning (DQN) [259–261]: a batch Q-learning method that is easily explained, and offers great flexibility to allow the inclusion of "insights from experts." Completely new in this book are convex-analytic approaches to RL that have a far stronger supporting theory and can be designed to impose bounds on performance.

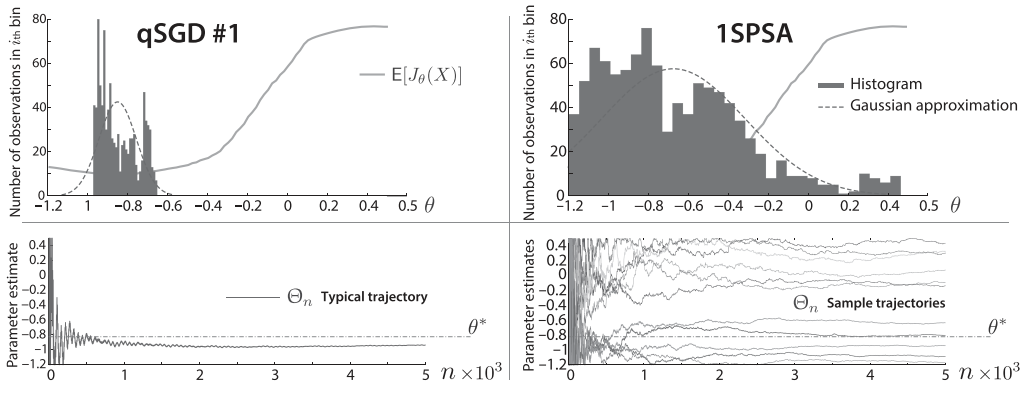

Figure 1.2 Error analysis for two policy gradient algorithms for Mountain Car, using QSA and traditional randomized exploration.

Part II might be considered a more traditional treatment of RL, since it begins with Markovian models and surveys the original TD methods developed in the 1990s. What is unique in this book is the attention to algorithm design, including ways to obtain insight in selecting "metaparameters" such as step-size gains. There is also new material, such as variance reduction methods (also known as *acceleration* techniques) for both Q-learning and actor-critic methods.

1.2 What's Missing?

The focus of the book is on control fundamentals that are most relevant to reinforcement learning, and a large collection of tools for RL algorithm design based on these fundamentals.

Some important topics will receive less attention:

(i) *Exploration.* This is a hot topic for research right now [171, 216, 244, 281, 307]. The theory is mature mainly within the subdiscipline of bandit theory. This book contains only a mention of bandit basics to explain the exploration/exploitation trade-off in RL.

(ii) *Data science.* Here I refer to the empirical side of statistics and computer engineering. Much success in RL may have been initially inspired by hardcore mathematical analysis, but success in applications requires clever computer engineers to create efficient code, and patience to test algorithms and hopefully improve them based on insight gained in particular examples.

There are no large-scale empirical studies described in this book. Also, there is no attempt to catalog the growing list of RL algorithms.

(iii) *Machine learning topics.* The book will explain the meaning of neural networks and kernels, but ask the reader to go elsewhere for details.

Sample complexity theory is covered only briefly. There is no question that sample complexity theory is the bedrock of statistical learning theory, and the theory of bandits. However, I personally believe that its value in RL is limited: Bounds are typically loose, and

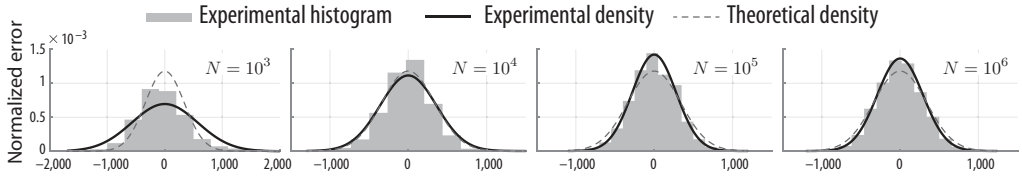

Figure 1.3 Comparison of Q-learning and relative Q-learning algorithms for the stochastic shortest path problem of [112]. The relative Q-learning algorithm is unaffected by large discounting.

to date they offer little insight for algorithm design. For example, I don't see how today's finite-n bounds will help DeepMind create better algorithms for Go or chess.

The value of asymptotic statistics is underappreciated in the RL literature. The best way to make this point is through images. Figure 1.3 is taken from [114] (many similar plots can be found in the thesis of A. Devraj [107]). Technical details regarding these plots may be found in Section 9.7.

The histograms show estimation error for a single parameter (one of many) using a particular implementation of *tabular Q-learning*. The integer N refers to the run length of the algorithm, and the histograms were obtained from 1,000 independent experiments. The "theoretical density" is what you can obtain from asymptotic statistics theory for stochastic approximation. This density is easily estimated based on limited data. In particular, in this experiment, it is clear after $N = 10^4$ samples that we have a very good variance estimate. This can then be used to obtain approximate confidence bounds after we run to $N = 10^7$ (we know how far we have to run once we have a variance estimate).

Variance theory for SA is used to decide run lengths. It is also used for algorithm *design*: for example, to adjust variables in an algorithm so that the asymptotic variance is minimized. Zap stochastic approximation and Polyak–Ruppert averaging are two examples of this approach. Outside of the bandits literature, *I am not aware of work in the sample complexity literature that can offer similar value for algorithm design in reinforcement learning*.

1.3 Resources

Many of the examples introduced in Chapters 2 and 3 are adapted from *Lecture Notes on Control System Theory and Design* [29]. The early chapters provide useful linear algebra background, but far more control theory than is needed to follow this book.

There are many great textbooks on deterministic control systems. See [15] and Murray's two-part online survey [268] for basics of modeling and design, and much more on linear control systems in [7, 76, 205]. Bertsekas [45, 46] provides a great treatment of optimal control for nonlinear systems, and an introduction to RL with a perspective that is similar to this book.

Luenberger [230] is my favorite introduction to optimization, and Boyd and Vandenberghe [73] is considered the bible, with far more depth in the domain of numerical methods. See also the new monograph of Bach [19].

The book of Sutton and Barto [338] is an encyclopedic introduction to RL. The books [44, 47, 347] are more foundational, and [289] is complementary to the treatment of RL here. The monographs [221, 360] contain essays by researchers at the intersection of RL and control. Finally, don't forget the resources at the Simons Institute website, `simons.berkeley.edu`. Videos and slides from the 2020 fall program on RL will be available whenever you have time to view a tutorial.