Part III - Reinforcement Learning

With four parameters I can fit an elephant, and with five I can make him wiggle his trunk[1].

John von Neumann, Hungarian-American mathematician, physicist, computer scientist, game theorist and more, 1903-1957.

This part of the book contains three chapters:

- Chapter 9: Value function approximation
- Chapter 10: Simulation in tabular models
- Chapter 11: Simulation with function approximation

The material in Chapter 9 is often referred to as *approximate dynamic programming (ADP)* and that in Chapters 10 and 11 as *reinforcement learning*.

While von Neumann's quote was relevant for statistical models in the 1950s, things have changed. Many modern reinforcement learning methods, particularly those based on neural networks, use models with millions of parameters and still generalize well to new settings. This progress comes from advances in neural network techniques, access to massive datasets, and powerful optimization algorithms. As a result, the link between model complexity and performance is far more subtle than in von Neumann's time.

Reinforcement learning can be viewed from two perspectives:

- The *artificial intelligence (AI)* perspective is that reinforcement learning methods seek to develop an "agent which can interpret any environment, and learn a task to superhuman ability, all with minimal user interaction[2]."

[1] Dyson [2004].

[2] McKenzie and McDonnell [2022]

- The *operations research (OR)* perspective is that in a model-based environment in which rewards and transition probabilities are known, reinforcement learning provides a toolbox of methods that can be used to find effective policies.

Although these appear to be distinct, the boundaries are far from clear cut in practice. Table 1 sheds light on how these perspectives have evolved in practice.

Aspect	OR perspective	AI perspective
Horizon	Infinite or finite and fixed	Finite and random
Model	Specified	Not specified
Rewards	At each decision epoch	At termination
Possible actions	Pre-specified	Pre-specified
Transition probabilities	Known	Not known
Algorithms	Offline	Online

Table 1: Distinguishing features of the OR and AI perspectives on reinforcement learning. These are not intended to be exhaustive or mutually exclusive, the boundary between them is often blurred.

Overview

Chapters **??** - **??** provided computational algorithms (e.g., value iteration, policy iteration, linear programming) for finding optimal policies in a fully or partially observable Markov decision process that can be stored on a computer. Since these methods are computationally prohibitive in models with extremely large (or continuous) state spaces, this section of the book and most current research develops and investigates methods to do so.

Motivated by the need to reduce the dimension of the state space, Chapter **??** provides generalizations of value iteration, policy iteration and linear programming for analyzing Markov decision processes in which the underlying value function is approximated by a low-dimensional linear or nonlinear function of features of the state space or state-action space. Such algorithms are applied directly to the Markov decision process model. No simulation is involved.

The penultimate chapter, Chapter **??**, represents a major departure from previous chapters. In it, simulation (or real-time experimentation) replaces exact computation. It focuses on models that can be represented in *tabular* form. That is, models that are sufficiently small so that value functions and state-action value functions can be easily stored in memory. In such models, approximating value and state-action value functions is unnecessary. The reason for focusing on tabular models is to compare the performance of simulation to classical methods. Approaches include Monte Carlo methods, temporal differencing and Q-learning. This chapter also distinguishes *model-based* analyses, in which components of the underlying Markov decision process are used in algorithms, and *model-free* analyses, which are based only on the generated or

observed sequence of states, actions and rewards. The underlying mathematical tools are stochastic approximation and stochastic gradient descent.

The concluding chapter, Chapter **??**, combines the concepts from the previous two chapters, namely value function or state-action value function approximation and simulation and introduces policy-based approaches as well. It is motivated by applications in which the state space (and/or action space) is so large (or continuous) that tabular representations and exact computation are impractical. The two distinct approaches are:

1. **Value-based methods** use the reward to update the stored state-action value function or an approximation to it. When rewards corresponding to an action are greater than expected, the state-action value function for the specified state-action pair is increased, making the chosen action in that state more attractive. Conversely, when the reward is less than expected, the state-action value function is decreased making the action less attractive. Thus, the rewards lead these algorithms to learn a good approximations to the "true" underlying state-action value function from which actions can be selected greedily in subsequent implementations.

2. **Policy-based methods** use the reward to directly update a randomized policy by gradient ascent. When an action chosen in a state produces a larger than expected reward, the probability of choosing that action in the state is increased. Conversely if the reward is less than expected, the probability of choosing that action is decreased. Repeating this process produces effective policies.

Objectives

This part of the book offers an accessible introduction to reinforcement learning concepts and algorithms, rather than a comprehensive (and soon outdated) overview. By mastering these fundamental methods, readers will establish a solid base from which to delve more deeply into rapidly evolving reinforcement learning research and applications.

So, how should one learn this material? Following the reinforcement learning philosophy, you must learn by doing. This means:

1. Carefully formulating your applications as Markov decision processes.

2. Developing your own code for the algorithms[3] in these chapters.

Often your codes will not replicate published results. Unlike the methods in Chapters **??** – **??**, reinforcement learning methods offer few practical guarantees on convergence and have many parameters to tune. Trial-and-error is a natural part of the process, and a necessity for developing the insights needed to create robust and effective algorithms.

[3]This is not recommended for linear programming and regression, where excellent commercial codes are available.

Figure III.1: Image a Brio Labyrinth owned by one of the authors of this book.

An example: The Brio labyrinth

To provide a flavor of the type of problem that the reinforcement learning community seeks to solve, consider the challenge of optimally guiding a ball through the Brio Labyrinth shown in Figure III.1. Bi and D'Andrea [2023] use reinforcement learning to find and implement a good strategy for playing this game. Details and methods are beyond the scope of this book but hopefully will be understandable after reading these chapters.

In this game, a player places a metal ball at a location designated as the start at the top center of the maze. Using the two knobs on the sides of the labyrinth, the player dynamically tilts the plane of the surface so as to guide the ball to a destination at the right center without falling through any of the 39 numbered holes[4].

To "solve" this problem using reinforcement learning requires complex engineering, considerable ingenuity and the use of state-of-the-art algorithms. The researchers developed a robotic system that used computer-controlled motors attached to the rotors to adjust the tilt of the surface. A camera mounted above the surface provided real-time information about the ball position, surface angles and the geometry around the ball.

An MDP model

The system was modeled by a discrete-time undiscounted infinite horizon (episodic) MDP. States represent information about the ball, the surface, and the local geometry (position of walls and holes) around the ball. The ball position is given by its (x, y)-

[4]Holes may be numbered twice when they can interact with the ball at two different segments of its path.

coordinates. The surface is summarized in terms of the angle of tilt of the surface in the x and y directions. Rates of change of these quantities are estimated using sequences of images. The local geometry around the ball is captured by 6 cm $\times$ 6 cm images centered at the ball, taken at discrete time steps and encoded in a 64×64 RGB image. Actions represent the angular velocity of the two motors attached to the knobs controlling the surface tilt.

Camera images also provided a basis for a reward structure that measures progress (in cms) along the directional path represented by the black line from the origin to the destination in Figure III.1. Note in episodic models such as this, it is customary to only receive a reward when achieving a goal or incur a penalty when terminating in an undesirable state. Such a reward structure presents challenges for reinforcement learning methods so the introduction of these intermediate rewards facilitates learning.

Algorithms and results

Good policies are found using a model-based actor-critic algorithm (see Section **??**) called DreamerV3 [Hafner et al., 2025]. Transition probabilities (obtained from "a world model"), policies, and values are approximated by neural networks. Underlying symmetries in the board structure improve algorithmic efficiency.

The system learned to "solve" the game in 5 hours (1 million time steps) using real-world experiences. Running the resulting policy over 50 replicates resulted in a success rate (reaching the target location) of 76% and a completion time of 15.73 ± 0.36 seconds. To put this in perspective, the human record is 15.95 seconds.

Bibliography

T. Bi and R. D'Andrea. Sample-efficient learning to solve a real-world labyrinth game using data-augmented model-based reinforcement learning, 2023. URL `https://arxiv.org/abs/2312.09906`.

F. Dyson. A Meeting with Enrico Fermi. *Nature*, 427(6972):297, 2004.

D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025.

M. C. McKenzie and M. D. McDonnell. Modern value based reinforcement learning: A chronological review. *IEEE Access*, 10:134704–134725, 2022.

Index