Frontmatter

The very beginning of the book presents, for example, the author and target groups. It also defines terms frequently used in the book.

From Rules to Learning

This chapter sets Reinforcement Learning (RL) in a context. It is also a history lesson.

From Markov to Bellman

Essential mathematical foundations are presented, covering Markov processes, parameter fitting, linear function approximation, and value functions. The principle of Bellman optimality is also defined and explained.

Reinforcement Learning Concepts

This chapter presents the core triad of RL: the trainer, the agent, and the environment. Their interactions and responsibilities are explained, along with fundamental RL concepts such as experience, episode, and action value. The chapter also outlines the general code structure common to nearly all RL algorithms.

Temporal Difference Learning

RL techniques covered in this chapter are foundational: one- step Q-learning and one-step SARSA. Their algorithms are explained and illustrated through grid-world examples to enhance understanding. The examples highlight key concepts such as variance, the need for hyperparameter tuning, and exploration. Unsuccessful results are also discussed to provide deeper insight, and practical tips are included to support hands-on implementation.

Monte Carlo Learning

Monte Carlo (MC) methods utilize the principle of sampling returns, where the model generates values based on randomly simulated outcomes. Consequently, value estimates can only be performed upon the completion of an episode. This chapter presents MC-based methods used for value function $V(s)$ estimation for a given policy. In addition, the chapter describes MC-based estimation of the state-action value function $Q(s, a)$, which in turn defines a policy. Example problems, such as a Dice Game, are also presented in the chapter.

n-Step Learning

Temporal-Difference (TD) methods refine value estimates incrementally after each interaction, whereas Monte Carlo (MC) methods update values only after an episode has finished. This chapter begins by outlining the key differences between TD and MC methods, noting that MC methods typically suffer from higher variance. The concept of n-step updates

for both state-value and state–action value functions is then introduced. These updates can be viewed as a hybrid between TD and MC methods and often lead to significantly faster learning. The chapter concludes with practical demonstrations on two grid-world problems.

Safe-Action Reinforcement Learning

In this chapter, the original trainer–agent–environment setup is extended with a safety layer. The safety layer prevents unsafe actions by performing a safety check: if an action is deemed unsafe, it is blocked, and a safe alternative is selected. In this chapter, this concept is integrated into a single-step Q-learning framework. The Gold Treasure problem, characterized by numerous failure states, is used for demonstration. The demonstration shows that safe-action RL can substantially improve learning speed.

Non-Episodic Learning

In some problems, episodes do not naturally terminate. Such problems are referred to as non- episodic, as the task continues indefinitely. This chapter introduces key concepts needed to address non-episodic Reinforcement Learning (RL) problems, with particular emphasis on the estimation of average reward, a critical component in this setting. Appropriate RL algorithms for solving these problems are presented. To demonstrate the approach, a parking area access problem is introduced, in which a queue of vehicles arrives at a parking area. An artificial agent is trained to decide whether to accept or reject each request, with the goal of maximizing long- term performance.

Next-Level Concepts

In real-world problems, inputs and actions are often continuous rather than discrete. Moreover, the dimensionality of such problems is typically challenging, involving many input variables and a vast number of possible states. To manage this complexity, this chapter introduces key concepts that form the foundation for more advanced reinforcement learning methods. A central focus is gradient descent learning, a fundamental technique in machine learning for fitting function approximators. The chapter also presents two powerful function approximators: radial basis function networks and neural networks. Through simple, illustrative problems, readers will learn how to apply these techniques effectively.

Policy Gradient Methods

This chapter presents an alternative way of action selection: a policy expressed as a function that directly selects actions, without relying on a value function. Such a policy typically gives the probability of choosing action a in state s. The parameters of this function determine how s is mapped to action probabilities. The policy parameters are updated by calculating a gradient. The gradient points in the direction in the parameter space that will make the policy more likely to choose high-value actions. A classic gradient algorithm, REINFORCE, is applied to two example problems. One example problem involves discrete actions: selecting an arm when playing a Two-Armed bandit. Another example problem utilizes continuous actions for setting the angle elevation when firing a cannon.

Actor-Critic Methods

Actor–critic methods can be viewed as an extension of policy gradient methods. One major limitation of the most basic policy gradient method, REINFORCE, is that it requires complete episodes for policy updating. In contrast, actor-critic methods can also learn during episodes. This chapter describes the actor-critic architecture: the actor and the critic operate in tandem within a feedback loop. The actor selects actions, and the critic provides the actor with a learning signal, guiding it between good and bad actions. The chapter also includes a description of the Lunar Lander problem and demonstrates how it can be solved using an actor– critic algorithm. The Lunar Lander problem involves safely landing a spacecraft on the surface of the Moon by controlling the rocket engine's thrust.

Deep Reinforcement Learning

This chapter deals with Deep Q-learning Network, often abbreviated as DQN. As in tabular Q- learning, action values are trained. One advantage over the tabular setting is that the inputs can be continuous, allowing the method to handle more complex and realistic environments. Algorithms for DQN learning are presented and discussed, including techniques such as experience replay and target networks, which help stabilize the training process. Understanding is supported by two demonstration problems. One of the problems is a classical control problem: the Inverted Pendulum.

Monte Carlo Tree Search

This chapter introduces search as an alternative to learning. Unlike learning, which aims for a global policy, search focuses on the current state, with forward simulation of possible outcomes as a central component. Monte Carlo Tree Search (MCTS), which incorporates several reinforcement learning concepts, is the technique discussed. The four phases of an MCTS iteration—traversal, expansion, simulation, and backpropagation—are explained in detail. MCTS is then applied to two example problems: Jumper, inspired by side-scrolling platformer video games, and Lane Change, illustrating a driving scenario. To deepen understanding, visualizations such as the growth of the search tree with iterations in the Jumper problem are included.

Combining Learning and Search

This chapter explores the combination of short- and long-term memory. While AlphaZero is a highly sophisticated algorithm, this book is intended for readers who are new to reinforcement learning. Therefore, a simplified variant of AlphaZero is presented here. The distinctive feature of this setup is the bidirectional flow of information between learning and search: search supports learning by guiding action selection, while learning supports search through value function estimation. Cooking a dish illustrates this idea: you plan the ingredients and steps, learn from tasting the outcome, and use that feedback to improve your plan the next time you cook. The simplified AlphaZero variant is demonstrated on the Pong game, where the agent controls a paddle to catch a bouncing ball.

Multi-Agent Reinforcement Learning

This chapter introduces Multi-Agent Reinforcement Learning (MARL), an extension of reinforcement learning to environments where multiple agents interact. Unlike the single agent setting, where the environment is stationary and governed by a single agent, MARL involves multiple agents whose actions influence one another. This interdependence introduces challenges such as non-stationarity, credit assignment, and an exponential increase in the size of the action space.
The chapter reviews fundamental concepts including joint action spaces, decentralized and centralized training. Key algorithmic frameworks, such as independent learning and centralized critic methods, are presented. An example environment, Dual Pong, is introduced to illustrate concepts in practice.

Outlook

This final chapter highlights limitations of current Reinforcement Learning (RL), including sample inefficiency, limited generalization, and high computational demands. It explores emerging research directions and advanced methods aimed at overcoming these challenges, such as improved exploration strategies, model-based approaches, and the use of transformers, the same architecture used in generative models like ChatGPT. By examining what might come next, the chapter offers insights into the future of RL and outlines promising paths toward more scalable, adaptable, and efficient learning systems.

Appendix

Complementary material such as a comparison of earlier presented RL methods is presented in appendix.