Chapter 1

Introduction

> *Destiny is not a matter of chance: it is a matter of choice; it is not a thing to be waited for, it is a thing to be achieved*[1].
>
> William Jennings Bryan[2], American orator and politician, 1860-1925.

The sentiment in William Jennings Bryan's quote captures the essence of Markov decision processes (MDPs) and reinforcement learning (RL). These frameworks provide methods to assist a decision maker or agent who seeks, through a series of choices, to guide a system to achieve desirable outcomes. Unlike forecasting, which focuses on predicting what will happen under fixed or external dynamics, Markov decision processes and reinforcement learning regard future outcomes as consequences of actions. The goal of these methods is not merely to anticipate the future but to learn policies that actively shape it. In this view, destiny is determined by decision-making, not chance–a principle that underlies the design and analysis of control strategies in dynamic environments.

Consequently, this book explores the mathematical and algorithmic foundations of sequential decision-making under uncertainty, with a focus on problems involving trade-offs, randomness, and temporal structure. By learning to formulate and solve dynamic decision models, readers will gain tools not only to anticipate future outcomes but also to influence and optimize them through informed action.

[1] Bryan [1899].

[2] Marty's wife, Dodie, and William Jennings Bryan were both born in Salem, Illinois. In a curious coincidence, Dodie later lived on Bryan Lane, the location of the Bryan family farm. Hopefully, this shared hometown and street name will not give rise to any conflicts of interest.

This book examines two primary approaches to sequential decision making in uncertain environments:

1. A *model-based* Markov decision process approach in which the agent or decision maker has full knowledge of the environment's structure—states, actions, transition probabilities, and reward functions — and uses algorithms such as value iteration, policy iteration, or linear programming to compute optimal policies.

2. A *model-free* reinforcement learning approach in which the agent or decision maker learns good policies through trial-and-error interactions with the environment or a simulation model, without requiring explicit knowledge of transition probabilities or reward functions. Algorithms include Q-learning, policy gradient, and actor-critic methods.

These categories represent conceptual extremes; the boundaries between them have become blurred over time. Modern approaches often blend features of both. In some cases, models may be available but too difficult to explicitly formulate or to solve using standard algorithms, making model-free methods more attractive. On the other hand, in model-free settings, the agent may learn approximate models of the environment and leverage them to to enhance learning and planning. Two applications distinguish these differences and their overlap.

Managing pre-boarding security

Consider the challenge of managing the pre-boarding screening area at a large airport. This is an ongoing challenge that is best analyzed offline using a model-based approach. The objective is to find policies that specify how and when to augment or decrement the workforce size and composition.

A model-based MDP approach specifies the system state to be the number of passengers of each type waiting to be cleared and the configuration of the workforce at each instant of time. Transition probabilities encode the probability of changes in system state between decision epochs given decisions regarding workforce levels. Viewing this as a cost minimization problem, costs are associated with workforce wages and penalties for passenger delay.

In theory, a model of this system can be developed and solved using Markov decision process algorithms. However, doing so will be challenging because of the large number of possible states and actions and the difficulty to explicitly represent the impacts of decisions in terms of transition probabilities. Such a model suffers from *the curse of dimensionality*[3]. Instead, it may be far easier to represent this system by a simulation and analyze it using reinforcement learning methods that were developed for model-free environments.

[3]This terminology was introduced by Bellman [1957] to describe the exponential growth of state spaces in multi-dimensional applications.

Robot navigation

Consider a robot with sensors dropped in a foreign landscape with no other instructions but to sense and reason autonomously in order to reach a certain target. This is an episodic problem that terminates if and when the target is reached. Developing a model would be challenging since the environment is unknown, although the actions the robot can choose are determined by its configuration and guidance system. There is no obvious intermittent reward; a reward corresponds to reaching the target.

In this new environment, the robot must learn online using trial-and-error. Model-free reinforcement learning algorithms can be used to guide the robot to its target. From this perspective, the methods are focused on learning, through experimentation, often from scratch.

Under this scenario, the robot does not have the benefit of running simulations to explore policy options, it must make decisions in real time. However, in training robots to perform tasks, it has become common practice to develop simulation models of the robot's capabilities and train the robot offline in simulations[4]. A major advantage of this approach is that using a simulator enables the robot to explore actions that cause costly damage in a real environment.

Origins

The model-based and model-free approaches have evolved largely independently within distinct academic communities. The model-based, Markov decision process framework has its roots in operations research and control theory, with applications primarily in economics, engineering, and management. The model-free, reinforcement learning approach emerged from behavioral science, neuroscience and computer science and has been applied extensively in robotics, gaming, and generative artificial intelligence.

Despite significant conceptual differences in the foundations of these disciplines, they share a common mathematical base. Ultimately, both aim to determine optimal policies, albeit in different ways.

1.1 What is a Markov decision process?

This book's framework for describing decision-making will be a Markov decision process (MDP) model. Markov decision processes excel at representing the sequential nature of decision problems where current choices probabilistically impact future outcomes and options.

Relevant information for making a decision is captured by a *state* variable. In each state, the decision-maker chooses an *action* and receives an immediate *reward* with known functional form. This action choice must weigh the trade-off between the immediate reward and future rewards, both of which may be subject to uncertainty.

[4]See, for example, NVIDIA Isaac [2025].

Finally, the system is sequential: after choosing an action, the system *transitions* probabilistically according to a known transition probability function to a new state from which the decision-making process repeats. The objective in analyzing an MDP is to find a *policy*, that is a sequence of state-dependent actions that maximizes expected accumulated rewards over the future.

The expression *dynamic programming* is often used synonymously with *Markov decision process*. This book distinguishes these terms. It views dynamic programming as an inductive algorithm for solving sequential decision problems based on decomposing them into a sequence of interrelated single-period problems. Backwards induction and value iteration correspond to applying dynamic programming to a Markov decision process.

1.1.1 A historical perspective

The expression Markov decision process originates with Bellman [1957]. His comprehensive book formulates a wide range of problems as Markov decision processes and introduces the basic concepts of states, actions, transition probabilities, rewards and optimality equations.

A description of some antecedents follows. One of the earliest published examples of a sequential decision problem was that of the eminent English mathematician Arthur Cayley. His example is a precursor of the "secretary problem"[5], which is referred to as the online dating problem in Chapter **??**. He described it as follows [Cayley, 1875]:

> *A lottery is arranged as follows: There are n tickets representing rewards of $a, b, c, \ldots$ pounds respectively. A person draws once; looks at his ticket; and if he pleases draws again (out of the remaining $n-1$ tickets); looks at his ticket and if he pleases draws again (out of the remaining $n-2$ tickets) and so on, drawing in all not more than k times; and he receives the value of the last drawn ticket. Suppose he regulates his drawings in the manner most advantageous to him according to the theory of probabilities, what is the value of his expectations?*

The foundations of Markov decision process theory and application were laid in the mid-20th century. Two important (and independent) precursors were the work of Wald [1947] on statistical sequential analysis and the research of Massé [1946] on water resource management, which is described in Gessford and Karlin [1958]. However, the modern development of "dynamic programming" began with collaborations of Arrow, Bellman, Blackwell, Dvoretsky, Girschik, Isaacs, Kalman, Karlin, Kiefer, La Salle, Robbins, Shapley and Wolfowitz beginning in the late 1940s at RAND corporation. Excellent summaries of this pioneering work appear in Bellman [1957] and Arrow et al.

[5] A nice overview of the secretary problem and Cayley's original formulation appears in Ferguson [1989].

[1958]. The work of Shapley [1953] regarding value iteration in stochastic games is especially prescient.

The next important milestone was Howard's PhD dissertation research at MIT, which is colorfully described in Howard [1960]. In this work, Howard developed *policy iteration* for discounted and undiscounted Markov decision processes. A path-breaking insight was that the average reward model required distinct analyses when underlying Markov chains had single and multiple closed classes.

The 1960s saw the development of a rigorous mathematical theory of Markov decision processes. Inspired by Howard's book, the seminal paper by Blackwell [1962] developed a framework for rigorously analyzing infinite horizon, finite state and action, discounted and undiscounted Markov decision process. He formulated the model using matrix notation and established the optimality of stationary policies in a discounted model by showing that policy improvement converges. However, the most significant results in this paper concern the existence and computation of optimal policies as the discount factor approaches one. To do so, he uses Markov chain results from Kemeny and Snell [1960] concerning the fundamental matrix to relate the average and discounted cases through a partial Laurent expansion. This concept is now referred to as *Blackwell optimality*. This chain of arguments was subsequently refined in Veinott [1969], where he introduces the concepts of n-discount optimality.

Subsequent papers [Blackwell, 1965, 1967, Strauch, 1966] advanced theoretical foundations in abstract settings. In recognition of his influential contributions, in 2024, NVIDIA Corporation introduced its Blackwell architecture for AI computing.

Other noteworthy contributions to the development of MDP theory include Derman [1970], Schweitzer and Federgruen [1977] and Puterman and Brumelle [1979]. The books Bertsekas [1987], Ross [1983], Sennott [1999] and Puterman [1994] represented milestones in the formalization and consolidation of this body of theoretical development.

Ironically, while Markov decision process research had matured by the late 1980s, laying a rigorous mathematical and algorithmic foundation for sequential decision-making, (computational) reinforcement learning emerged as an active and complementary field of research and application. In the next decade, the relationship between these two distinct disciplines began to become apparent with the publication of Barto et al. [1989] and Bertsekas and Tsitsiklis [1996].

1.1.2 Markov decision process applications

Originally, Markov decision processes found widespread application in operations research problems including inventory management, queuing control, and equipment maintenance and replacement. In addition to providing algorithms for computing optimal policies, Markov decision process methods provided an approach for determining the structure of optimal policies. This is exemplified by Scarf [1960], who established

the optimality of an (s, S)-policy[6] in a single-product inventory model with a fixed cost of placing an order (see Figure 1.1). In addition to establishing an elegant result, it enabled development of specialized computational algorithms that restricted search for optimal policies within the class of (s, S)-policies.

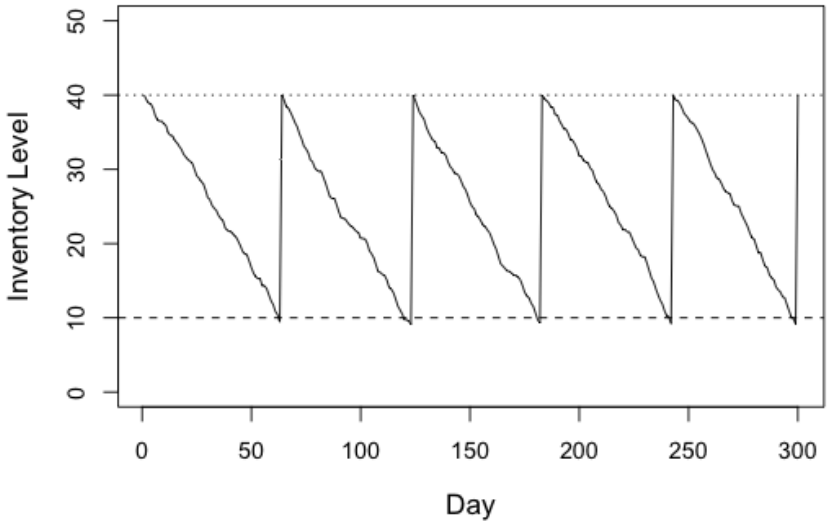

Figure 1.1: Graphical representation of the daily inventory level when controlled by an (s, S)-policy with $s = 10$ and $S = 40$. Jagged lines represent randomness in demand.

This avenue of research provided results in equipment replacement and queuing admission control models in which, under mild conditions, an optimal policy was shown to be of *control limit* form. That is, if the equipment condition deteriorated below a specified level, then it was optimal to replace it. Similarly, in a queuing admission control model, if the number of jobs in the system exceeded a specified level, then it was optimal to not admit further jobs.

Subsequently, the range of Markov decision process applications have broadened to include decision problems in:

> **Revenue Management:** Markov decision processes are used to dynamically adjust prices to maximize returns from time-limited or perishable resources. These techniques have been widely applied in the airline, hospitality, entertainment, and fashion retail industries.

[6] An (s, S) policy controls the inventory level as follows. Whenever the daily inventory level i falls below s, an (s, S)-policy places an order for $S - i$ units, raising the starting inventory level at the next day to S.

Healthcare: Applications include developing personalized treatment strategies—such as determining the optimal time to initiate therapy—and scheduling scarce resources like MRI/CT scanners and surgical staff, under uncertainty about future demand.

Finance: Markov decision process-based methods have been applied to dynamic portfolio optimization, risk management, and option pricing.

Communications Systems: Markov decision processes support the efficient allocation of network resources, including bandwidth distribution, job routing, and congestion control in dynamic environments.

Sports Analytics: Strategic decisions, such as when to pull a goalie in hockey or whether or not to punt on fourth down in American football, have been analyzed using Markov decision process models. They can also be used to optimize training regimens for individual athletes.

1.2 What is reinforcement learning?

Computational reinforcement learning (RL) refers to a collection of artificial intelligence methods that trains agents (or computer programs) to make effective decisions in dynamic environments by using feedback from actions to influence future decisions. *Behavioral reinforcement learning*, on the other hand, refers to a body of psychological research originating in the early 20th century that developed a theory of how organisms "learn through reinforcement". The expression "reinforcement learning" appears to have originated in the work of Barto and Sutton in 1980s. Over time, these two disciplines have grown to share a common vocabulary; behavioral concepts are used to describe computational models, while computational concepts have become increasingly adopted in behavioral science discourse. With the exception of this introductory chapter, this book will focus on computational reinforcement learning.

1.2.1 Trial-and-error learning

Both behavioral and computational reinforcement learning are grounded in trial-and-error experimentation. Consider, for example, the experience of a child learning to walk. Falls may lead to tears while progress may brings smiles. The child's innate reward mechanisms reinforce the movements that produce smiles and diminish those that result in tears. Over time, this reinforcement pattern shapes neural pathways that enable the child to eventually walk.

Reinforcement learning concepts have roots in behavioral science. Building on these roots, artificial intelligence research has sought to replicate trial-and-error learning on

a computer. A friend of ours[7] asked "*How can a computer be rewarded and learn?*" This insightful question strikes at the core of computational reinforcement learning.

In the language of this book, the environment is represented on the computer, explicitly or implicitly, by a Markov decision process. When code selects an action in a state the computer model generates both a reward and a new state. The computer "learns" by using this information to increase the likelihood of choosing actions with positive rewards and decrease the likelihood of obtaining negative rewards. Over repeated interactions, this process leads to the development of effective operating policies. Chapters **??** and **??** make this high-level description precise.

Behavioral reinforcement learning focuses on investigating the psychological and neurological mechanisms underlying learning. In contrast, computational reinforcement learning is concerned with designing reward systems and developing algorithms that enable agents to learn effective behavior through interaction with their environment.

1.2.2 A historical perspective

Reinforcement learning has it roots in behavioral science, neuroscience and artificial intelligence.

Behavioral antecedents

Early research in behavioral science underlies the conceptual framework of reinforcement learning.

> **Pavlov**, in the 1890s, developed a theory of "conditioned learning" or *classical conditioning* to explain his observations that a dog's salivation in response to a reward (food) could instead be triggered by an unrelated stimulus (a metronome or bell) through conditioning.
>
> **Thorndike**, in the early 1900s, developed the "law of effect" to explain the learning that cats exhibited when placed in puzzle boxes where they needed to press a lever to escape and obtain a reward (food). He observed that over time, the cats were able to escape and obtain their reward more quickly, suggesting that behaviors learned by trial-and-error influence future behavior.
>
> **Skinner**, beginning in the 1930s, developed a theory of "operant conditioning" to explain how behavior could be modified through different reinforcement regimes and schedules. He tested his theory in a controlled environment now referred to as a "Skinner box" where he was able to control the timing and frequency of rewards and punishment.

[7] Dr. Robin Friedlander, Psychologist, BC Children's Hospital

Overall, this historical work on classical and operant conditioning provided a foundation for understanding how organisms learn through reinforcement. It highlighted the role of rewards, punishments, and their frequency and timing on shaping and modifying behavior.

This line of research has increasingly focused on determining the underlying neurological mechanisms that reinforce behavior. While the exact neural level processes are complex, growing evidence suggests that dopamine release following a reward plays a critical role in reinforcing reward-producing actions by strengthening synaptic connections in the brain.

Seminal work by Schultz et al. [1997] showed that specific midbrain neurons encode *reward prediction errors* - the difference between actual and expected reward. Positive prediction errors are associated with increased release of dopamine and reinforcement of preceding actions. Conversely, negative prediction errors lead to decreased dopamine activity and suppression of the likelihood of repeating behavior. This concept aligns with the notion of *temporal differencing* in computational reinforcement learning, which is the subject of Chapters **??** and **??**.

Credit assignment in behavioral reinforcement learning

It is not clear how organisms determine which actions to reinforce when a long sequence of actions is required to attain a reward. *Credit assignment* refers to the process of determining which actions or stimuli are responsible for an outcome.

Researchers in psychology and neuroscience study the mechanisms animals and humans use to solve the credit assignment problem. Their studies investigated mechanisms including action similarity, temporal discounting and dopamine signaling in the brain's reward system. Experiments, like those reported in Tang et al. [2024], explore how organisms, in this case mice, learn from controlled delayed and temporally distributed feedback. It found that when action pairs were separated by long time intervals, credit was assigned in a stepwise fashion: initially, behaviors closest in time to the reward were reinforced, followed by gradual refinement of earlier, more distant actions. As the authors note, *"Thus, a retrospective reinforcement mechanism promotes not only reinforcement, but also gradual refinement of the entire behavioral repertoire to assign credit to specific actions and action sequences that lead to dopamine release."*

This concept aligns with the notion of backward induction in Markov decision processes and computational reinforcement learning where rewards are propagated backwards from terminal nodes to earlier actions in a sequence.

Towards machine learning: Samuel's checkers

The transition from studies of learning in animals to machine learning was exemplified by the seminal work of Arthur Samuel in the 1950s. Samuel, a computer scientist at IBM, used the game of checkers [Samuel, 1959] to study how computers might learn through interacting with an environment. He chose checkers rather than chess because

"the simplicity of its rules permits greater emphasis on learning techniques".

Features of checkers that made it appealing to study were:

- Checkers is a *sequential decision process*. The current decision affects subsequent outcomes and decisions.
- Checkers is *Markovian*. The current board position summarizes the current state. No past information is necessary to determine the future.
- Checkers is *episodic*. Episodes end with a win, loss or stalemate.
- The *objective*, winning, is well-defined.
- Opponents' strategies introduce *randomness*. In most cases, the player does not have a model for the opponent's strategy.
- There is no algorithm that guarantees a win.
- The rules of checkers are well-defined and widely known.
- Checkers, including the ability to accept an opponent's moves and self-play, can be easily programmed on a computer.
- Computer-generated results can be tested in real-time against human players.

His research introduced many fundamental reinforcement learning concepts including:

Self-improvement: The goal of this research was to show that it was possible to develop a computer program that improved its outcomes over time without external interaction by repetitive play.

High dimensionality: The number of possible board configurations was extremely large.

Value function approximation: Weighted combinations of feature[8] values were used to assign a "score" to each board configuration. Features with the largest weights were regarded to be most important.

Roll out: Actions (moves) were evaluated on the basis of multi-step roll out from the current board position using the value function approximation to assess the quality of the move. Using a value function approximation avoided the need to play the game to its conclusion to evaluate each alternative move.

Value function backup: The largest value from each node in the roll out was backed up and assigned to the current board position.

[8]Features included piece advantage, the number of kings and indicators of board control.

Parameter updating: Feature weights were updated on the basis of outcomes. The described updating method had some similarity with temporal differencing. Feature sets and weights were initialized on the basis of available games.

Replay: Games can start from any configuration and more than one game can be played at the same time against different opponents.

Self-play: To speed up learning and fulfill the objective of autonomy, games were played in which the player and opponent switched roles at each turn. The same value function was used for each player but the features were adjusted to accommodate the board configuration facing each player.

Empirical validation: Results were validated through play with human opponents.

A fundamental dichotomy: Distinction between general purpose methods and special purpose methods such as that developed to play checkers was discussed.

Although not noted therein, it should be obvious to the reader that many ingredients of a Markov decision process formulation and analysis, especially backwards induction, were present in Samuel's work.

Credit assignment in computational reinforcement learning

Checkers provides a classic example of the credit assignment problem. In checkers, a long series of actions (moves) leads eventually to winning, losing or ending in a draw. However, it is difficult to identify which specific moves most greatly impacted the outcome.

This challenge is fundamental to learning in computational reinforcement learning. For an agent to improve performance, it must accurately identify which actions are beneficial and which are not. If the agent fails to assign credit accurately it may reinforce ineffective actions or fail to reinforce effective ones, making the learning process counterproductive.

With the recognition of the formal relationship between computational reinforcement learning and Markov decision processes, it became apparent that *Markov decision process value functions solve the credit assignment problem.* In online methods, value functions (or policies) are updated incrementally during task execution. As a result, actions that are consequential for generating rewards increase value functions, effectively assigning more credit to beneficial actions and less credit to those with negligible impact.

For example, in a soccer game, online methods would incrementally update value estimates in real time after each pass, gradually improving understanding of how sequences of passes contribute to scoring a goal. In contrast offline or batch methods require waiting until the end of the game to adjust estimates. This online approach underlies real-time updating of odds in real-time sports gambling.

Foundations of reinforcement learning

Barto, Sutton and colleagues have been significant contributors to the development of computational reinforcement learning. Their text, Sutton and Barto [2018], provides a modern introduction to reinforcement learning from the computer science perspective. Motivated by this early work, Bertsekas and Tsitsiklis developed rigorous foundations for reinforcement learning methods from a control theory perspective. Much of their early work appears in Bertsekas and Tsitsiklis [1996].

A brief description of some fundamental references in the development of reinforcement learning follows. Barto et al. [1983] introduces many of the concepts and models underlying reinforcement learning including temporal difference (TD) learning, neural network approximations and an actor-critic architecture. It applied the actor-critic algorithm to solve the "cart-pole" problem and explored the relationship of this approach to behavioral theories.

Sutton [1988] provides an in-depth analysis of model-free temporal difference learning. The paper includes illustrative examples, a convergence proof, a discussion of the relationship to gradient descent and explores its relationship to Samuel's work. It distinguishes temporal difference learning as a form of incremental learning distinct from supervised learning, in which learning does not occur until a goal is achieved. Tsitsiklis and Roy [1997] develop and apply a rigorous foundation for analyzing temporal difference methods.

Sutton et al. [1999] introduced an approach to reinforcement learning that focuses on optimizing policies directly instead of state or state-action value functions. The approach is based on using a parametric model of a randomized decision rule. A key result, referred to as *The Policy Gradient Theorem*, provides a representation for the gradient of the value functions with respect to policy parameters that can be evaluated through simulation.

Watkins and Dayan [1992] proposed Q-learning as a method for model-free learning by shifting the object of analysis from a value function to a state-action value function. Updating the state-action value function through experience avoided the need for an underlying model. This paper described a Q-learning recursion and provided a convergence proof. Tsitsiklis [1994] explores the relationship between Q-learning and stochastic approximation and provides a proof of the convergence of Q-learning in greater generality.

Reinforcement learning applications

Reinforcement learning has demonstrated remarkable success in a variety of domains, including game playing, robotics, and natural language processing.

Perhaps motivated by Samuel's seminal study of checkers, many advances in reinforcement learning have come about through game playing including backgammon (Tesauro [1992]), Atari games (Mnih et al. [2013]), chess and Go (Silver et al. [2016]) and the Brio Labyrinth (Bi and D'Andrea [2023]). Bertsekas [2022] provides a control

theory perspective on the foundations of DeepMind's work on Go.

Reinforcement learning methods have been fundamental in driving advancements in robotics and autonomous vehicle guidance. By treating robotic systems as agents that learn from interactions with their environment, reinforcement learning has enabled the development of robots capable of performing complex tasks such as manipulation, navigation, and decision-making in dynamic settings. In the realm of autonomous vehicles, reinforcement learning has been instrumental in addressing challenges like path planning, obstacle avoidance, and adaptive control, leading to significant improvements in vehicle safety and efficiency.

Recently, reinforcement learning methods have played a fundamental role in the broad impact and remarkable progress of the generative AI applications ChatGPT, Gemini and Copilot. Reinforcement learning methods provide a framework for reward function design, policy optimization, trading off exploration with exploitation, and incorporating user feedback. Note we have occasionally called on such applications while writing this book, most notably to improve wording, generate and debug code, generate TikZ-based figures, and identify historical references.

1.3 Book structure

Figure 1.2 provides a graphical representation of the interrelationship between topics covered in this book. Note that Markov decision process and reinforcement learning methods lie in distinct boxes, ignoring the fact reinforcement learning methods are frequently used to solve large, intractable Markov decision process.

The book is divided into three parts:

- Part I establishes foundations by providing a rigorous formulation of the Markov decision process model and illustrating it through applications. It corresponds to the content itemized in the boxes **Foundations** and **Derived Quantities**, and provides an introduction to **Optimality** concepts.

- Part II covers classic Markov decision process theory and algorithms. This part of the book provides an in-depth treatment of the items in the **Optimality** and **MDP methods** boxes. For technical reasons, these methods are discussed separately for discounted, total, and average reward criteria. This part concludes with a description of partially observable Markov decision process (POMDP) models and results.

- Part III focuses on function approximation and simulation-based methods developed within the reinforcement learning literature. It corresponds to the topics in the **RL methods** box. Reinforcement learning provides a model-free perspective by developing methods that learn good policies without requiring prior knowledge of transition probabilities or reward functions.

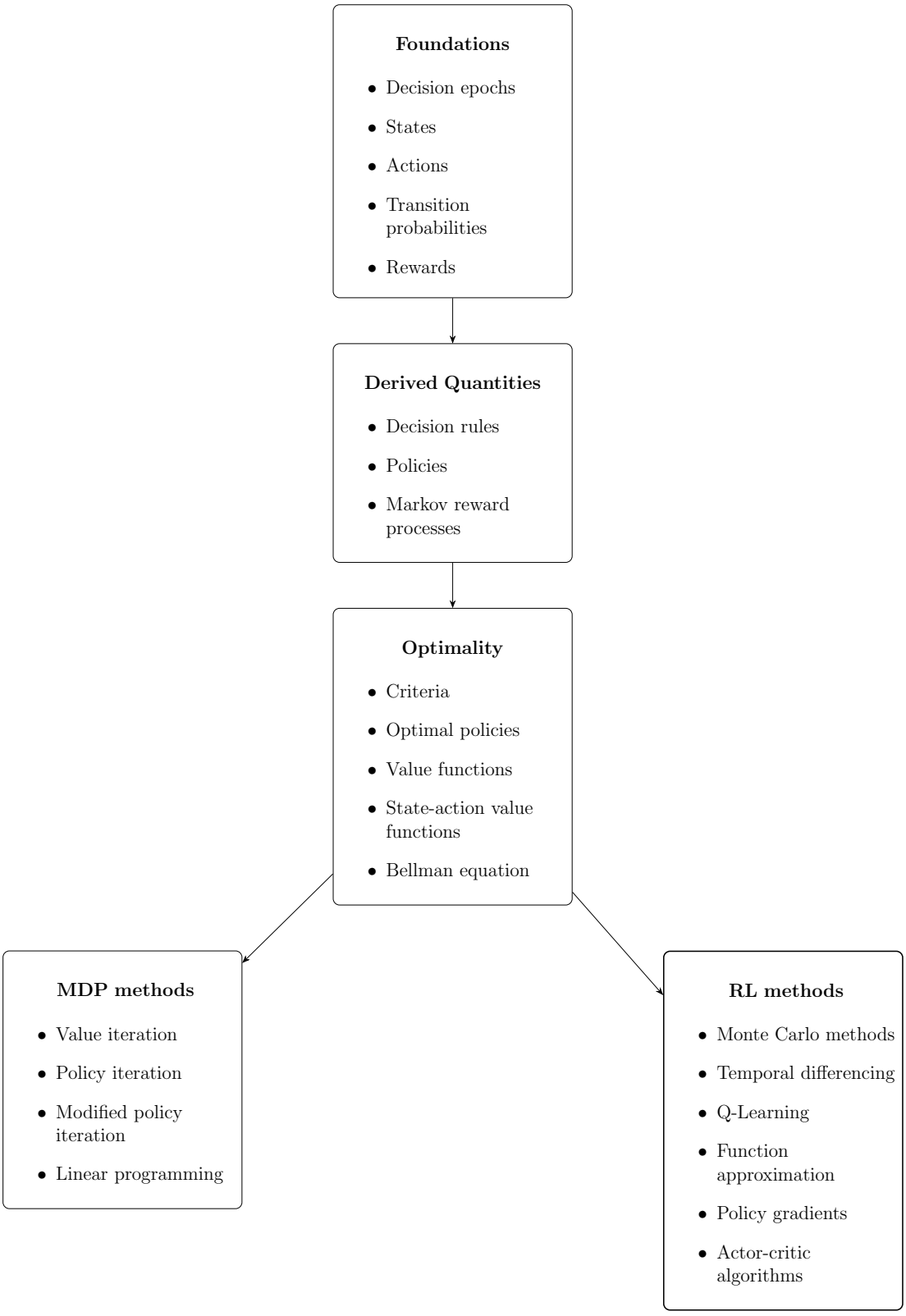

Figure 1.2: Relationship between fundamental topics in this book.

Bibliography

K. J. Arrow, S. Karlin, and H. E. Scarf. *Studies in the Mathematical Theory of Inventory and Production*. Stanford University Press, Stanford, CA, 1958.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man. Cybern.*, SMC-13 (5):834–846, 1983.

A. G. Barto, R. S. Sutton, and C. J. C. H. Watkins. Sequential decision problems and neural networks. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 2, pages 686–693. 1989.

R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice Hall, 1987.

D. P. Bertsekas. *Lessons from AlphaZero for Optimal, Model Predictive and Adaptive Control*. Athena Scientific, 2022.

D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

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`.

D. Blackwell. Discrete dynamic programming. *Ann. Math. Stat.*, 33:719–726, 1962.

D. Blackwell. Discounted dynamic programming. *Ann. Math. Stat.*, 36:226–235, 1965.

D. Blackwell. Positive dynamic programming. In *Proceedings of the 5th Berkeley Symposium on Probability and Statistics, Volume 1*, pages 415–418, 1967.

W. J. Bryan. America's Mission. Speech, Virginia Democratic Association Banquet, Washington, D.C., 1899. Delivered on February 22, 1899.

A. Cayley. Mathematical questions with their solutions, no. 4528. *Educational Times*, 23:18, 1875.

C. Derman. *Finite State Markovian Decision Processes*. Academic Press, New York, NY, 1970.

T. S. Ferguson. Who solved the secretary problem? *Stat. Sci.*, 4(3):282–296, 1989.

J. Gessford and S. Karlin. Optimal policies for hydroelectric operations. In K. Arrow, S. Karlin, and H. Scarf, editors, *Studies in the Mathematical Theory of Inventory and Production*, pages 179–200. Stanford University Press, Stanford, CA, 1958.

R. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Van Nostrand-Reinhold, New York, 1960.

P. Massé. *Les reserves et la regulation se l'avenir dans la vie economique, Vol I and II*. Hermann, 1946.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. In *Proceedings of the 27th International Conference on Machine Learning*, pages 330–337, 2013.

NVIDIA Isaac. AI robot development platform. `https://developer.nvidia.com/isaac/`, 2025. Retrieved Feb. 20, 2025.

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons., 1994.

M. L. Puterman and S. Brumelle. On the convergence of policy iteration in stationary dynamic programming. *Math. Oper. Res.*, 4:60–69, 1979.

S. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, New York, 1983.

A. Samuel. Some studies in machine learning using the game of checkers. *IBM J. Res. and Dev.*, 3:210–229, 1959.

H. Scarf. The optimality of (s, S)-policies in the dynamic inventory problem. In S. Karlin K. Arrow and P. Suppes, editors, *Studies in the Mathematical Theory of Inventory and Production*, pages 196–202. Stanford University Press, Stanford, CA, 1960.

W. Schultz, P. Dayan, and P. R. Montague. A neural substrate of prediction and reward. *Science*, 275:1593–1599, 1997.

P. J. Schweitzer and A. Federgruen. Asymptotic behavior of value iteration in Markov decision problems. *Math. Oper. Res.*, 2:360–381, 1977.

L. Sennott. *Stochastic Dynamic Programming and the Control of Queueing Systems.* John Wiley and Sons, 1999.

L. S. Shapley. Stochastic games. *Proc. Natl. Acad. Sci. USA*, 39:1095–1100, 1953.

D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

R. Strauch. Negative dynamic programming. *Ann. Math. Stat.*, 37:871–890, 1966.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Mach. Learn.*, 3:9–44, 1988.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An introduction, second edition.* MIT Press, Cambridge, MA, 2018.

R. S. Sutton, D. MacAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1063, 1999.

J. C. Y. Tang, V. Paixao, F. Carvalho, A. Silva, A. Klaus, J. Alves da Silva, and R. M. Costa. Dynamic behaviour restructuring mediates dopamine-dependent credit assignment. *Nature*, 626:583–592, 2024.

G. Tesauro. Practical issues in temporal difference learning. *Mach. Learn.*, 8:257–277, 1992.

J. N. Tsitsiklis. Asynchronous stochastic approximation and Q-learning. *Mach. Learn.*, 16:185–202, 1994.

J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximations. *IEEE Trans. Autom. Control*, 42:674–690, 1997.

A. F. Veinott, Jr. Discrete dynamic programming programming with sensitive discount optimality criteria. *Ann. Math. Stat.*, 40:1635–1660, 1969.

A. Wald. *Sequential Analysis.* John Wiley & Sons, New York, 1947.

C. Watkins and P. Dayan. Q-learning. *Mach. Learn.*, 8:279–292, 1992.

Index