Lecture 1

Reinforcement Learning: Setup and Concepts

Goals of this lecture

1. Introduce *reinforcement learning* as a framework for decision-making through interaction.
2. Define the components of a *Markov Decision Process (MDP)*: states, actions, and rewards.
3. Formalize the *agent–environment interaction* and the role of the transition kernel.
4. Explain the *Markov property* and key modeling assumptions.
5. Present the concept of *policies* as mappings from observations to actions.

1.1 Motivation and Historical Context

Why study Reinforcement Learning? Reinforcement learning provides a principled framework for learning to act under uncertainty from *trial and error*. Breakthrough applications—ranging from TD–Gammon and autonomous helicopter flight to recent game–playing systems such as AlphaZero—demonstrate that approximate solutions to high–dimensional MDPs can match or surpass human performance. Beyond games, RL techniques underpin modern robotics, industrial control, recommendation systems, and adaptive experimentation in healthcare.

Milestones

- **1960s–1990s:** Dynamic Programming theory (Bellman), temporal–difference learning (Sutton & Barto), and the first empirical successes such as *TD–Gammon*.
- **2000s:** Convergence guarantees for Q–learning, PAC bounds for exploration (e.g., *E3*, *R–Max*).
- **2010s:** Deep RL era—function approximation with neural networks (DQN), actor–critic methods (A3C, PPO), and self–play systems (AlphaGo/AlphaZero).

These milestones illustrate how algorithmic progress follows advances in both theory (contraction mappings, stochastic approximation) and computation (GPU acceleration). A guiding theme of this course is understanding *why* and *when* these algorithms work.

1.2 Markov Decision Processes

We study sequential decision-making problems using the mathematical framework of *Markov decision processes (MDPs)*. An MDP is formally defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P)$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, and P is the transition probability kernel describing the environment. At each discrete time step $t = 0, 1, 2, \ldots$, the agent observes the current state $s_t \in \mathcal{S}$, selects an action $a_t \in \mathcal{A}$, and subsequently, the environment transitions to a new state s_{t+1} and returns a scalar reward r_{t+1}. This sequential interaction between the agent and the environment is clearly illustrated in Figure 1.1.

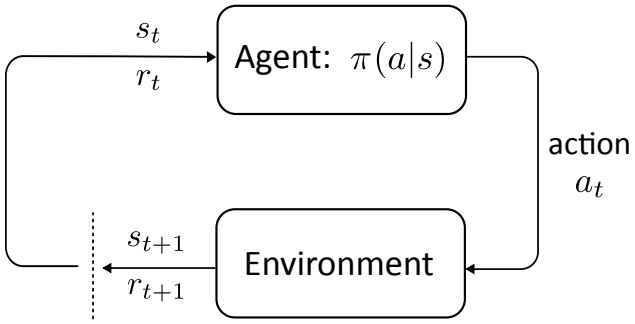


Figure 1.1: Agent–environment interaction loop under the Markov assumption. At each discrete time step t, the agent observes the current state s_t and reward r_t, selects an action a_t, and subsequently, the environment transitions to a new state s_{t+1} and provides a new reward r_{t+1}.

States. A state $s_t \in \mathcal{S}$ is a concise representation of the environment at time t, containing all necessary information for the agent to select its next action. The set of states $\mathcal{S}$ can be finite, countable, or continuous, depending on the problem setting. Selecting a suitable state representation is essential for effectively modeling and solving a sequential decision-making problem.

Actions. An action $a_t \in \mathcal{A}$ is a decision the agent makes at time t after observing the current state s_t. The action influences how the environment evolves, thereby affecting future states and rewards. Depending on the context, actions may be discrete (e.g., move left, right, up, or down) or continuous (e.g., setting the velocity of a robot arm). The choice of action set $\mathcal{A}$ significantly impacts the complexity and the performance of reinforcement learning methods.

Rewards. After executing an action, the agent receives a scalar reward $r_{t+1} \in \mathcal{R} \subseteq \mathbb{R}$, which provides feedback about the immediate utility or quality of the transition. The reward serves as the learning signal in reinforcement learning: the agent's objective is to select actions that maximize cumulative reward over time. The specific value of the reward depends on the environment dynamics and the current state-action context.

Environment Process. We model the environment as a stochastic process described formally by random variables. Specifically, we denote by S_t and R_t the state and reward random variables at each time t, and by s_t and r_t their realizations. Given the history of past interactions up to time t:

$$h_t = (s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{t-1}, a_{t-1}, r_t, s_t),$$

and the current action a_t, the environment's dynamics are fully described by the transition kernel P. For concreteness we will assume the sets $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$ to be *finite*. At each step t, the transition kernel outputs the next reward R_{t+1} and next state S_{t+1} according to the conditional distribution:

$$(R_{t+1},\, S_{t+1}) \;\sim\; P(\,\cdot,\,\cdot \mid H_t{=}h_t,\; A_t{=}a_t).$$

In other words, the probability

$$P(S_{t+1} = s',\, R_{t+1} = r \mid H_t = h,\, A_t = a)$$

represents the likelihood that the environment transitions to state s' and emits reward r, given that the interaction history equals h and the agent selects action a at time t.

This interaction forms a stochastic process:

$$S_0, A_0, R_1, S_1, A_1, R_2, \ldots$$

Markov Property. We assume the environment satisfies the *Markov property*, meaning that the distribution over next states and rewards depends only on the current state-action pair (s_t, a_t), and not on the entire past history h_t. Formally, this is expressed as:

$$P(S_{t+1}{=}s_{t+1},\, R_{t+1}{=}r_{t+1} \mid H_t{=}h_t,\, A_t{=}a_t) = P(S_{t+1}{=}s_{t+1},\, R_{t+1}{=}r_{t+1} \mid S_t{=}s_t,\, A_t{=}a_t), \quad \forall h_t.$$

To simplify notation, we write:

$$p(s', r \mid s, a) \;:=\; P(S_{t+1}{=}s',\, R_{t+1}{=}r \mid S_t{=}s,\, A_t{=}a).$$

This compact form will be used throughout to describe the environment's transition dynamics under the Markov assumption.

Model Assumptions. As mentioned before, to simplify the theory, we assume that the state space $\mathcal{S}$ and action space $\mathcal{A}$ are finite or countable. We also assume that $\mathcal{R}$ is finite; this is done without loss of generality. The environment is *stationary*, meaning that the transition distribution $P(S_{t+1}, R_{t+1} \mid S_t, A_t)$ does not change over time. Finally, we assume full observability: the agent observes the true environment state s_t at each time step. We will later relax several of these assumptions.

1.3 Modeling Transitions and Rewards

Having introduced the Markov property and the environment's transition kernel, we now examine how the joint distribution over next states and rewards can be represented, simplified, or factored.

Transition Probabilities. The Markov assumption allows us to describe the environment's dynamics in terms of the conditional probability $p(s', r \mid s, a)$ over next states and rewards. Often, it is convenient to separately represent state transitions by marginalizing over the rewards:

$$p(s' \mid s, a) = \sum_{r \in \mathcal{R}} p(s', r \mid s, a).$$

This state-transition probability describes how likely it is to transition into state s' after taking action a in state s, regardless of the reward received.

Reward Distributions and Expected Rewards. Given the transition probabilities, the conditional distribution of rewards given a state-action-next-state tuple can be recovered using Bayes' rule:

$$p(r \mid s, a, s') = \frac{p(s', r \mid s, a)}{p(s' \mid s, a)} \quad \text{for all } s' \text{ such that } p(s' \mid s, a) > 0.$$

This explicitly separates the reward dynamics from state transitions, allowing more detailed modeling of the environment.

In practice, it is often sufficient or computationally simpler to represent the reward signal using only its expectation. The *expected reward function* $r(s, a)$ summarizes the reward distribution into a single scalar quantity:

$$r(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a).$$

While the full MDP dynamics are described by the joint distribution $p(s', r \mid s, a)$, many reinforcement learning algorithms and analyses simplify modeling assumptions by using only the marginal transition probabilities $p(s' \mid s, a)$ and the expected reward function $r(s, a)$.

Alternative Environment Description. Instead of specifying the joint distribution $p(s', r \mid s, a)$ directly, an equivalent and often convenient representation explicitly separates state transitions from reward distributions. This alternative specification involves first modeling the transition probabilities $p(s' \mid s, a)$ and then the conditional reward distributions $p(r \mid s, a, s')$. Formally, the joint distribution can be recovered using the chain rule:

$$p(s', r \mid s, a) = p(s' \mid s, a)\, p(r \mid s, a, s').$$

This factorization neatly decouples randomness in transitions from randomness in rewards, simplifying analysis, simulations, and algorithmic implementations.

An important simplification occurs when rewards depend only on the current state-action pair (s, a) and not on the next state s'. In this special (but common) case, the reward distribution simplifies to:

$$p(r \mid s, a, s') = p(r \mid s, a) \quad \text{for all } s'.$$

If the reward distribution is further specialized to a deterministic function $r(s, a)$, we obtain:

$$p(r \mid s, a) = \begin{cases} 1 & \text{if } r = r(s, a), \\ 0 & \text{otherwise.} \end{cases}$$

These assumptions simplify the representation significantly, making analysis and computation more convenient, without losing the essential structure needed by most reinforcement learning algorithms.

1.4 Policies and Agent Behavior

Policies. A policy defines the strategy by which an agent selects actions based on the information it observes. In the most general setting, a policy maps the entire interaction history h_t to a probability distribution over actions. Formally, a general (history-dependent) policy $\pi \in \Pi$ selects actions according to:

$$A_t \sim \pi(\cdot \mid h_t), \quad \text{or equivalently,} \quad \pi(a_t \mid h_t) = P_\pi(A_t = a_t \mid H_t = h_t).$$

When the environment satisfies the Markov property, it is natural and convenient to restrict attention to policies that depend only on the current state s_t. Such a *Markovian policy* $\pi \in \Pi_M$ specifies the probability of selecting an action given the current state:

$$A_t \sim \pi(\cdot \mid s_t).$$

A policy can also be *deterministic*, $\pi \in \Pi_D$ assigning a single action to each state:

$$a_t = \pi(s_t).$$

These policy classes form nested subsets:

$$\Pi_D \subseteq \Pi_M \subseteq \Pi.$$

Notes and Implications. Given a policy π and an initial state $S_0 = s$, or distribution $S_0 \sim \rho$ the interaction between the policy and the environment induces a probability distribution P_π over the sequences of states, actions, and rewards, formally described as:

$$H_t \sim P_\pi(\,\cdot\,|S_0 = s) \quad \text{or} \quad H_t \sim P_\pi(\,\cdot\,|S_0 \sim \rho)$$

We write the associated expectation as $\mathbb{E}_\pi[\cdot|S_0 = s]$, explicitly denoting dependence on the policy π. If the initial state S_0 is drawn from an initial distribution ρ, we denote this as $\mathbb{E}_{\pi,\rho}[\cdot]$.

Several key theoretical and practical implications arise:

- For most MDPs, optimal policies exist within the class of Markovian policies Π_D, significantly simplifying theoretical analysis and the design of algorithms.
- The definitions and results implicitly assume full observability of the state s_t. Extensions to partially observed scenarios lead naturally to the generalization known as *partially observable Markov decision processes (POMDPs)*.

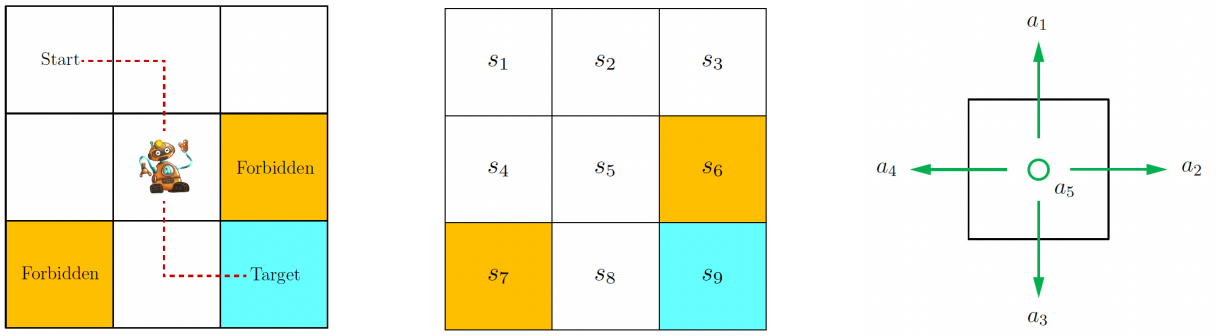
Throughout the course, we will frequently encounter these ideas, examining the theoretical advantages of Markovian policies and their practical applicability, as well as discussing generalizations beyond full state observability.

1.5 Running Example: Gridworld (3×3)

We illustrate the fundamental concepts of reinforcement learning using a small, intuitive 3×3 gridworld environment illustrated in Figure 1.2a.[1] In this example, an agent moves between adjacent cells, aiming to reach a designated goal state while avoiding forbidden cells. At each step, the agent chooses from five possible actions: move up, down, left, right, or stay still .

Environment. The environment consists of nine states, corresponding to the nine grid cells, indexed as $s_1, s_2, \ldots, s_9$. Each state corresponds uniquely to the agent's position on the grid (see Figure 1.2b).

[1] This running example has been taken from [1]. I found it to be a good grounding example of several concepts.



(a) Gridworld layout (b) State numbering (c) Available actions

Figure 1.2: The 3×3 Gridworld environment: (a) grid layout with goal and forbidden states, (b) state indices, and (c) possible actions from a given state.

Actions and Transition Dynamics. From each state, the agent can select one of five possible actions: up, down, left, right, or stay still. If an action would move the agent outside the grid boundary, the agent remains in the current state, effectively "bumping" against the boundary. Similarly, attempting to enter a forbidden cell may result in the agent remaining in place or incurring other specified penalties. Figure 1.2c shows the possible actions visually.

Formally, the environment's dynamics are represented by transition probabilities $p(s' \mid s, a)$, indicating the probability of moving from state s to state s' after taking action a. In our Gridworld example, the transitions are deterministic—meaning that each action from a given state always leads to exactly one next state, with probability 1. Consequently, for each pair (s, a), there is a single state s' such that:

$$p(s' \mid s, a) = 1, \quad \text{and} \quad p(\tilde{s} \mid s, a) = 0 \text{ for all } \tilde{s} \neq s'.$$

Table 1.1 explicitly lists these deterministic state transitions for each action-state pair. For instance, from state s_1, taking the upward action a_1 results deterministically in remaining at s_1, since this action would attempt to move the agent outside the grid boundary. Similarly, taking the rightward action a_2 from s_1 deterministically transitions the agent to state s_2. Such clear specification helps in understanding the agent's possible moves, simplifying both theoretical analysis and practical implementation.

Rewards. The agent receives numerical feedback from the environment in the form of rewards, according to the following structure:

- $r_{\text{target}} = +1$: when transitioning into the goal cell.
- $r_{\text{boundary}} = -1$: when the agent attempts an action leading outside the grid boundary.
- $r_{\text{forbidden}} = -1$: upon entering a forbidden cell.
- $r_{\text{other}} = 0$: for all other transitions.

These rewards incentivize the agent to efficiently find the optimal path to the goal state while avoiding penalties.

Given these rules, we define the reward function at the granularity of individual transitions $r(s, a, s')$. For example, from state s_1, taking the upward action a_1 leads to a boundary collision, resulting in the agent remaining in the same state with a penalty:

$$r(s_1, a_1, s_1) = -1\,.$$

	a_1 (upward)	a_2 (rightward)	a_3 (downward)	a_4 (leftward)	a_5 (still)
s_1	s_1	s_2	s_4	s_1	s_1
s_2	s_2	s_3	s_5	s_1	s_2
s_3	s_3	s_3	s_6	s_2	s_3
s_4	s_1	s_5	s_7	s_4	s_4
s_5	s_2	s_6	s_8	s_4	s_5
s_6	s_3	s_6	s_9	s_5	s_6
s_7	s_4	s_8	s_7	s_7	s_7
s_8	s_5	s_9	s_8	s_7	s_8
s_9	s_6	s_9	s_9	s_8	s_9

Table 1.1: A tabular representation of the deterministic state transitions. Each cell shows the next state given the current state and chosen action.

At state s_8, taking the rightward action a_2 leads to the goal state s_9, producing a positive reward:

$$r(s_8, a_2, s_9) = +1\,.$$

For most other transitions—where the move is valid and does not lead to termination or penalties—the reward is neutral:

$$r(s, a, s') = 0 \quad \text{for all other } (s, a, s')\,.$$

This transition-level reward structure supports flexible environment modeling, allowing the agent to distinguish not just which actions to take, but also how the resulting state affects the received reward.

Policy. A *policy* in this gridworld scenario is a mapping from states to action probabilities. A deterministic policy always selects the same action at each state, while a stochastic policy assigns probabilities to multiple actions. Following a policy, the agent moves through the grid, generating a trajectory composed of states, actions, and rewards.

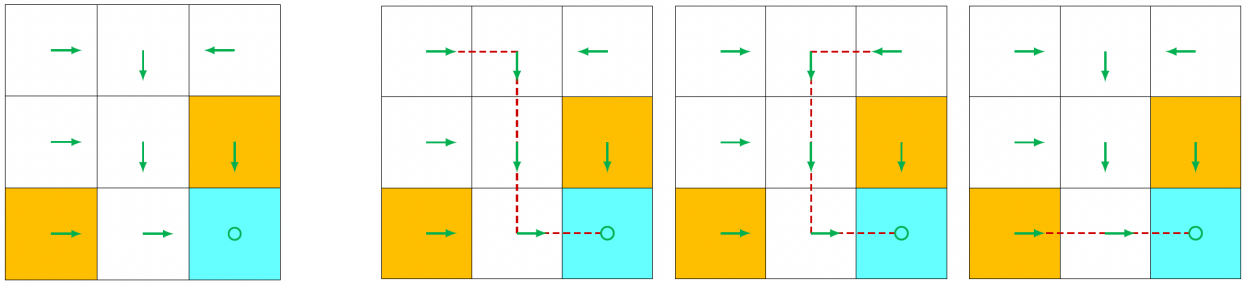
Figure 1.3 illustrates a deterministic policy in our 3×3 Gridworld example. Under this policy (Figure 1.3a), the agent selects a unique action in each state, leading to consistent, predictable trajectories toward the goal state (Figure 1.3b). In contrast, Figure 1.4 shows an example trajectory under a stochastic policy, where at certain states, actions are chosen probabilistically. This randomness leads to variability in the paths the agent takes toward the goal, highlighting how stochastic policies can help facilitate exploration.

Suggested Reading Before Lecture 2

- Sutton & Barto (2nd ed.), *Chapter 1 and Section 2.1–2.2*. Excellent narrative introduction.
- Szepesvári, *Algorithms for RL*, Section 2 for a more rigorous measure–theoretic treatment of MDPs.

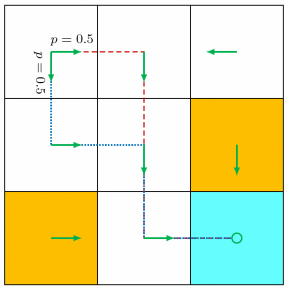
References for Lecture 1

[1] Shiyu Zhao. *Mathematical Foundations of Reinforcement Learning*. Springer Verlag, Singapore, 2025.



(a) Deterministic policy (b) Trajectories under deterministic policy

Figure 1.3: Illustration of policies in the 3×3 Gridworld: (a) Deterministic policy, and (b) sample trajectories under the deterministic policy.



$$\pi(a_1|s_1) = 0,$$
$$\pi(a_2|s_1) = 0.5,$$
$$\pi(a_3|s_1) = 0.5,$$
$$\pi(a_4|s_1) = 0,$$
$$\pi(a_5|s_1) = 0.$$

Figure 1.4: Sample trajectory under a stochastic policy in the 3×3 Gridworld. At certain states, the agent selects actions probabilistically, resulting in multiple possible paths toward the goal.