# Fundamentals of Reinforcement Learning

**Anonymous authors**
Paper under double-blind review

## Abstract

This paper provides an overview of key introductory concepts in Reinforcement Learning (RL), suitable for beginners. It defines the Markov Decision Process (MDP) and its components, elaborates on the distinctions between RL and Supervised Learning (including Imitation Learning), describes a real-world application using RL terminology, and mathematically formulates the RL objective function.

## 1 What is a Markov Decision Process (MDP)?

A **Markov Decision Process (MDP)** is a framework used to describe sequential decision-making problems, where a decision-maker (agent) interacts with an environment over time. At each time step $t$, the agent observes an observation of the state of the world $s_t$ (possibly partial $o_t$), takes an action $a_t$, and then receives a reward $r_t$ while making a transition to the new state $s_{t+1}$. This cycle continues, and the agent's ultimate goal is to find a **policy** (a strategy for choosing actions) that maximises the total expected reward.

The key idea and assumption of MDP is rooted in the **Markov property**, which states that the next state depends **only** on the current state and action, not on the history of previous states or actions. That doesn't mean that the future is perfectly predictable; there might still be randomness, but knowing the past doesn't help at all to resolve that randomness.

An MDP is defined by a **5-tuple**: $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$.

- **States ($\mathcal{S}$)**: The set of all possible states the environment of an agent can be in. Each state $s \in \mathcal{S}$ is a **complete** description of the environment, while an observation $o$ is a partial description of a state. A state $s$ must satisfy the **Markov Property**: If the current state $s_t$ is given, the future state $s_{t+1}$ and future rewards are independent of past states $(s_1, \ldots, s_{t-1})$ and actions $(a_1, \ldots, a_{t-1})$. Mathematically, $P(s_{t+1}|s_t, a_t, \ldots, s_1, a_1) = P(s_{t+1}|s_t, a_t)$.
- **Actions ($\mathcal{A}$)**: The set of all possible & valid actions (moves or decisions) that the agent can make in any state is often called the action space. Some environments have discrete action spaces, while others have continuous action spaces where actions are real-valued vectors.
- **Transition Probability Operator/Function($T$)**: It is a **tensor** which specifies the probability of transitioning from the current state $s$ to the next state $s'$ after taking action $a$. $T_{i,j,k} = p(s_{t+1} = i|s_t = j, a_t = k)$ also denoted as $P(s'|s, a)$. The environment's response to an action can be **deterministic** (always leads to the same next state) or **stochastic** (leads to different next states).
- **Reward Function ($R$)**: The reward function $R$ is very important in reinforcement learning, which is a mapping from $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. It depends on the current state of the world and the action just taken. It quantifies the desirability of an action or state, typically denoted as $R(s)$ or $R(s, a)$. The agent's ultimate goal in RL is to maximise ***not*** *the immediate* but the *cumulative* reward over time (over a trajectory).
- **Discount Factor ($\gamma$)**: For tasks that continue indefinitely, the cumulative sum of rewards could be infinite. We need ways to ensure the objective remains finite. Discount Factor is a value between 0 and 1 ($\gamma \in [0, 1)$ typically 0.9 to 0.99) which weights rewards received sooner more heavily than rewards received later to guarantee that the infinite sum converges to a finite value.

## 2 How Reinforcement Learning Differs from Supervised Learning (Imitation Learning)

Reinforcement Learning (RL) and Supervised Learning (SL), including Imitation Learning (IL), learn in fundamentally different ways and are distinguishable by their data sources, goals, inherent assumptions, and feedback mechanisms. The Supervised Learning works fine under the assumption that each data example is Independent and Identically Distributed. **Imitation Learning (Behavioral Cloning)** is a form of Supervised Learning applied to sequential decision-making. It trains a policy $\pi_\theta$ to mimic an expert's actions. However, IL inherits SL's limitations, particularly the distributional shift problem, meaning it cannot guarantee optimal performance outside the expert's observed trajectory space and cannot discover behaviors superior to the expert's. RL directly addresses these limitations by learning through interaction and optimizing for long-term outcomes.

- In supervised learning, including imitation learning, the agent learns from a fixed dataset (**Independent and Identically Distributed data**) of input-output pairs $\mathcal{D} = \{(X_i, Y_i)\}_{i=1}^{N}$. The model learns from these static examples and receives direct, immediate, and explicit "ground truth" labels (correct answers) for each input.

  In contrast, reinforcement learning does not assume access to an expert or labelled actions. Instead, the agent sequentially interacts with the environment directly. It observes the state, selects actions, and receives rewards based on those actions. The agent's goal is not to imitate an expert but to maximise the total cumulative reward it receives over time.
- **Learning Objective**: In supervised learning, the goal is to generalise patterns from given data and copy the behaviour or labels. i.e. learn mapping $f_\theta : \mathcal{X} \to \mathcal{Y}$ that accurately predicts outputs for unseen inputs and minimises prediction error.
  In Reinforcement Learning, the goal is to learn a **policy** $\pi_\theta : \mathcal{S} \to \mathcal{A}$ that maximises the **total future reward** over a sequence of interactions. It's about figuring out what the agent should do to reach a goal, potentially discovering solutions better than any human expert could show (like AlphaGo's surprising "Move 37"). It maximises $J(\theta) = \mathbb{E}_{\tau \sim P_\pi(\tau)} \left[ \sum_{t=1}^{T} \gamma^t R(s_t, a_t) \right]$.

## 3 Real-World Case: Autonomous Drone trained by RL

Let's describe a real-world case of an autonomous drone that navigate through a forest or different obstacles to reach a destination while avoiding any barriers. The AI-powered drone's goal is to learn a policy that maximises its total reward, meaning it learns to reach its destination safely while minimising crashes and being efficient.

- **Agent**: The drone's control system (the AI brain that makes decisions about flying).
- **The state** $s_t$ at time $t$ includes all relevant information about the environment, such as the drone's position, speed, orientation, battery level, and a camera image (the pixels showing trees, clear paths, etc.) of its surroundings.
- **Actions ($\mathcal{A}$)**: These are the commands the drone can give to its motors to change its flight. The action $a_t$ could be continuous control commands like pitch and roll or **discrete** commands like "turn left," "turn right," or "go straight."
- **Reward Function ($R$)**: How the drone gets feedback on its performance to learn and improve:
  - **Positive Reward**: For moving closer to the destination, or for each second it stays flying without crashing.
  - **Negative Reward (Penalty)**:
    - A large negative reward for colliding with a tree or any other obstacle.
    - A small negative reward for using a lot of battery power.

- **Transition Probability Function ($P(s_{t+1}|s_t, a_t)$)**: This describes how the drone moves through the forest based on the current state when an action is applied, including sensor noise and wind disturbances. This is typically **stochastic** due to external factors like wind.
- **Policy ($\pi_\theta$)**: This is the learned strategy or "brain" of the drone. It tells the drone what action to take in the current state (e.g., "When it sees a particular state/image of obstacles ahead, tilt hard right!").
- **Trajectory ($\tau$)**: A sequence of states and actions along the way. $\tau = (s_1, a_1, s_2, a_2 \ldots)$.
- **Objective**: The objective is to learn policy parameters $\theta^*$ that maximises the **expected cumulative reward** over extended periods and to train the drone to learn this policy so that it can reach the destination safely and efficiently.

# 4 The Objective of Reinforcement Learning in Math

The fundamental objective in Reinforcement Learning is to find an **optimal policy ($\pi^*$)** that is defined by parameters ($\theta^*$) and maximises the **expected cumulative reward** (also known as "return") over time. This expectation accounts for the stochasticity of both the environment and the agent's policy.

## 4.1 Finite Horizon (Total Reward)

For tasks that have a fixed, finite number of time steps $T$ the objective is to maximise the expected sum of rewards the agent collects until the end of completing the task.

The **return** for a single trajectory $\tau = (s_1, a_1, \ldots, s_T, a_T, s_{T+1})$ is:

$$G_0 = \sum_{t=1}^{T} R(s_t, a_t)$$

The **objective function**, which we aim to maximise by optimising the policy's parameters $\theta$, is the **expected return**:

$$J(\theta) = \mathbb{E}_{\tau \sim P_\pi(\tau)} \left[ \sum_{t=1}^{T} R(s_t, a_t) \right]$$

where:

- $P_\pi(\tau)$: The probability of a specific trajectory $\tau$ occurring under policy $\pi_\theta$:

$$P_\pi(\tau) = P(s_1) \prod_{t=1}^{T} \pi_\theta(a_t|s_t) P(s_{t+1}|s_t, a_t)$$

- $R(s_t, a_t)$: The immediate reward received at time $t$ in state $s_t$ taking an action $a_t$.

An equivalent formulation for finding the optimal policy parameters $\theta^*$ in the finite horizon case is to maximise the sum of expected rewards at each time step. Here, $p_\theta(s_t, a_t)$ represents the probability of visiting state $s_t$ and taking action $a_t$ at time $t$ under policy $\pi_\theta$.

$$\theta^* = \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{(s_t, a_t) \sim p_\theta(s_t, a_t)} [R(s_t, a_t)]$$

## 4.2 Infinite Horizon (Average Reward)

For tasks where the agent operates continuously without a natural ending point and without a discount factor, the objective is to maximise the long-term **average** reward per time step.

The **average reward objective** is defined as:

$$J_{avg}(\theta) = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_{\tau \sim P_\pi(\tau)} \left[ \sum_{t=1}^{T} R(s_t, a_t) \right]$$

When the system converges to a stationary distribution $p_\theta(s, a)$, this objective simplifies to the expected immediate reward under that distribution:

$$J_{avg}(\theta) = \mathbb{E}_{(s,a)\sim p_\theta(s,a)}[R(s, a)]$$

The optimal policy parameters $\theta^*$ for the average reward case are therefore found by maximising this expected immediate reward under the stationary distribution

$$\theta^* = \arg\max_\theta \mathbb{E}_{(s,a)\sim p_\theta(s,a)}[r(\mathbf{s}, \mathbf{a})]$$

### 4.3 Infinite Horizon (Discounted Total Reward)

For continuous tasks without a natural ending point, we also use a **discounted sum of rewards** to ensure the total return remains finite and converges.

The **discounted return** for a single trajectory $\tau = (s_1, a_1, s_2, a_2, \ldots)$ is:

$$G_0 = \sum_{t=1}^{\infty} \gamma^t R(s_t, a_t)$$

The **objective function** to maximise is the **expected discounted return**:

$$J(\theta) = \mathbb{E}_{\tau\sim P_\pi(\tau)}\left[\sum_{t=1}^{\infty} \gamma^t R(s_t, a_t)\right]$$

where:

- $\gamma$: The **discount factor**, a value in $[0, 1)$ that exponentially weighs future rewards less than immediate ones.

The ultimate goal in Reinforcement Learning is to find the policy parameters $\theta^*$ that maximises this objective function:

$$\theta^* = \arg\max_\theta J(\theta)$$

This means we are searching for the best possible strategy that, on average, leads to the highest accumulation of rewards over the long run, considering both the agent's actions and the environment's stochastic (random) nature.