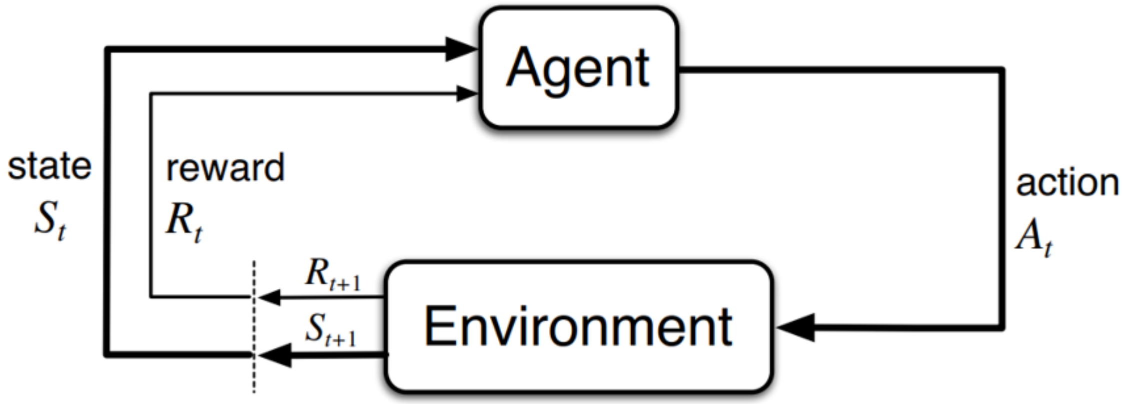
L1: RL Basics

> **Course**: Deep Reinforcement Learning
>
> **Info**: 2025 Spring, Tsinghua University

Contents ⬇️

What & RL v.s. Supervised Learning

1. No supervision, only reward signals.
2. Rewards are delayed.
3. Not i.i.d data, but sequential.



MDP

Basic Concepts

Markov Dicision Process (MDP) problems are non-derterminisic problems.

The **definition** of MDP is as below:

- A set of states $s \in S$
- A set of observations $o \in O$ (sometimes)
- A set of actions $a \in A$
- A transition function $T(s, a, s')$
- A reward function $R(s, a, s')$

Markov Property:

$$P(S_{t+1=s'}|S_{t=s_t}, A_t = a_t, \ldots, S_0 = s_0) = P(S_{t+1=s'}|S_{t=s_t}, A_t = a_t) \tag{1}$$

Fully v.s. Partially Observability: Whether state is the same as observation. If not, transformed into POMDP.

Policy: A policy gives an action for each state. an optimal policy ($\pi^* : S \to A$) is the one that maximizes expected utility (sum of rewards).

Discount Factor: Used in calculating the sum of rewards. Why we introduce Discount Factor:

- Avoid cyclic reward

- Infinite horizon convergence
- Worry about future uncertainties
- Time value & Human like

Utility: Sum of discounted rewards.

Optimal Quantities & Bellman Equation

The optimal policy: $\pi^*(s) =$ optimal action from a state s.

The optimal value (utility) of a state: $V^*(s) =$ expected utility starting in s and act optimally.

The optimal Q value: $Q^*(s,a) =$ expected utility taking action a from state s and acting optimally. (the action of this timestep is not necessarily the optimal)

$$V^*(s) = \max_a Q^*(s,a) \tag{2}$$

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \tag{3}$$

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')] \tag{4}$$

Value Iteration

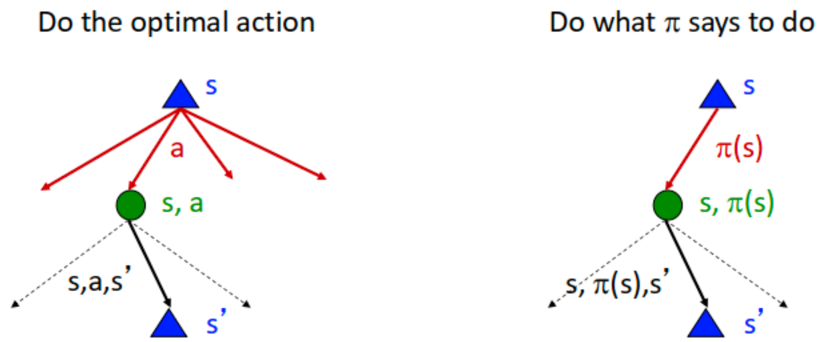
Define $V_k(s)$ to be the optimal value of s if the game ends in k more time steps, $V_0(s) = 0$, then we can compute $V_{k+1}(s)$:

$$V_k(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^k(s')] \tag{5}$$

Repeat this process until convergence.

> **Note**
>
> Some **problems** with value iteration:
>
> 1. Policy converges **earlier** than value.
> 2. The complexity of value iteration is $O(S^2A)$.

Value iteration max over over all actions to compute the optimal values. If we consider some fixed policy $\pi(s)$ (generally non-optimal), the tree would be much simpler.



Define the utility of a state s, under a fixed policy: $V_\pi(s)=$ expected total discounted rewards starting in s and following π. And the recursive relation (one-step look-ahead) is as below.

$$V_\pi(s)=\sum_{s'}T(s,\pi,s')[R(s,\pi,s')+\gamma V_\pi(s')] \tag{6}$$

Policy Iteration

Policy Evaluation

<u>Under a fixed policy</u>, how to calculate the V function?

Idea 1: Turn recursive Bellman equations into updates (like value iteration). $O(S^2)$ complexity.

$$V_0^\pi(s)=0 \tag{7}$$
$$V_{k+1}^\pi(s)=\sum_{s'}T(s,\pi(s),s')[R(s,\pi(s),s')+\gamma V_k^\pi(s')] \tag{8}$$

Idea 2: Without the maxes, the Bellman equations are just a linear system. Solve it. The complexity of matrix inversion operation (Gauss–Jordan elimination) is $O(S^3)$.

$$V=P(r+\gamma V) \tag{9}$$
$$V=(I-\gamma P)^{-1}R \tag{10}$$

Policy Improvement

Imagine we have the values.

Withdraw policy from values through one-step look-up (e.g. optimal condition V*):

$$\pi^*(s)=\arg\max_a\sum_{s'}T(s,a,s')[R(s,a,s')+\gamma V^*(s')] \tag{11}$$

One-step policy extraction:

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^{\pi_i}(s')] \tag{12}$$

Policy Iteration

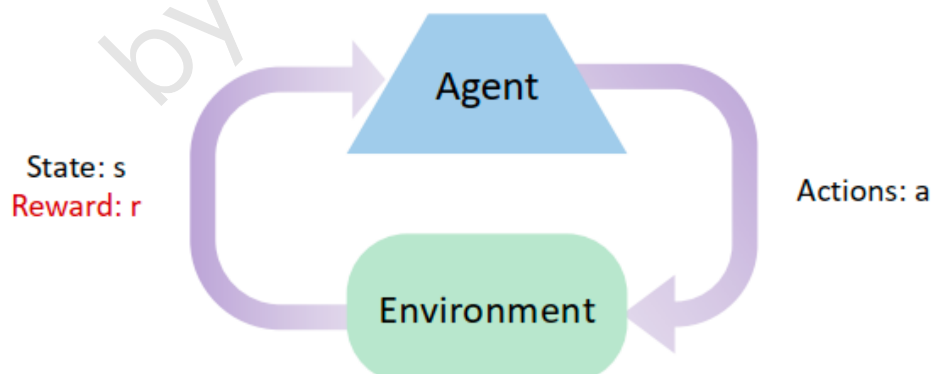
Require: Initialize π_0 arbitrarily
repeat
  Policy evaluation: compute $V^{\pi_k}(s)$ for all $s \in \mathcal{S}$
  Policy improvement: $\pi_{k+1}(s) = \arg\max_a Q^{\pi_k}(s,a)$ for all $s \in \mathcal{S}$
  $k \longleftarrow k+1$
until $\|\pi_k - \pi_{k-1}\|_1 = 0$
return Optimal policy π^*

We have converged $V^{\pi_k}(s)$ through policy evaluation, then extract the next policy π_{k+1} through "argmax_a" operation traversing the entire action space. While repeating, this "a" is different from $\pi_k(s)$.

> ⓘ **Note**
>
> - Policy iteration is still optimal.
> - It can converge much faster under some conditions.

From MDP to RL

Usually we do not know the transition, thus we introduced reinforcement learning method.



Basic ideas:

1. Receive feedback in the form of rewards.

2. Utility is defined by the **reward** function.
3. Must act to maximize **expected rewards**.
4. All learning is based on observed **samples of outcomes**.

A comprehensive comparison btw MDP and RL ⬇️

Aspect	MDP	RL
Definition	A mathematical model for sequential decision problems.	A learning method to solve sequential decision problems.
Scope	Theoretical foundation of RL; RL problems are typically modeled as MDPs.	Practical application of MDP; learns to solve MDP through interaction.
Known Information	<u>Assumes transition probabilities and reward functions are known.</u>	Typically assumes transition probabilities and reward functions are unknown.
Goal	Provides a framework to describe the problem.	Learns the optimal policy through interaction.
Methods	Based on dynamic programming (e.g., value iteration, policy iteration).	Based on trial and error (e.g., Q-learning, policy gradients).
Application	Used for theoretical analysis and problem modeling.	Used for practical learning and decision-making.