Algorithmic and Theoretical Foundations of RL

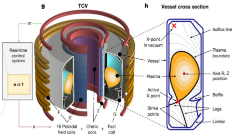
Introduction

Ke Wei
School of Data Science
Fudan University

Success of RL

(Nature, 2016)

(Nature, 2019)

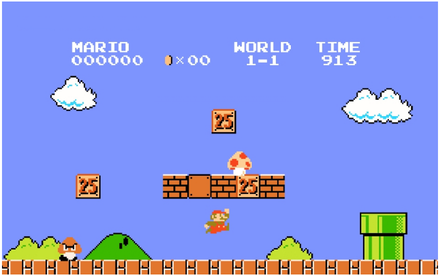


(Nature, 2022)

(ChatGPT, 2023)

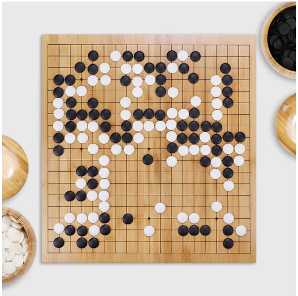
- RL has also been used to computationally difficult problems like traveling salesman problem and plays an important role in “AI for Science”.

Illustration: Super Mario



Super Mario makes a decision, then receives a reward and transfers to the next state; Goal: high long term cumulative reward by making right decisions.

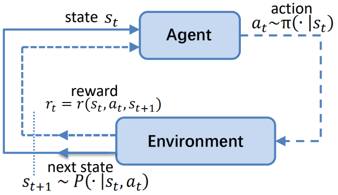
Challenges in RL



RL is a sequential decision problem and is essentially about efficient search (dynamic programming, control, game theory).

- High dimension (large state/action spaces)
- Highly nonconvex (distribution optimization, parameterization)
- Computational efficiency vs Reliability
- Plenty of scenarios $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$
- $\cdots\cdots$

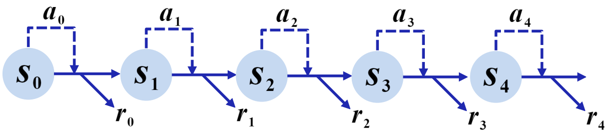
Formal Model: Markov Decision Process (MDP)



$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$$

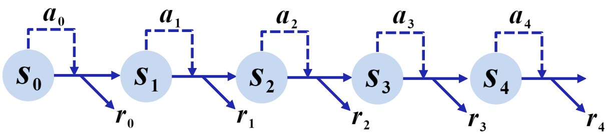
- $\mathcal{S}$: State space
- $\mathcal{A}$: Action space
- $P(\cdot|s, a)$: Transition probability
- $r(s, a, s')$: Immediate reward
- $\pi(\cdot|s)$: Policy (probability distribution on action space)

Agent selects action based on policy $a_t \sim \pi(\cdot|s_t)$ at state s_t, receives reward $r(s_t, a_t, s_{t+1})$, and transits to new state following $s_{t+1} \sim P(\cdot|s_t, a_t)$.



Trajectory: $\tau = (s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots)$

Formal Model: Markov Decision Process (MDP)



Trajectory: $\tau = (s_0, a_0, r_0, \cdots, s_t, a_t, r_t, \cdots)$

State value function at s and state-action value function at (s, a):

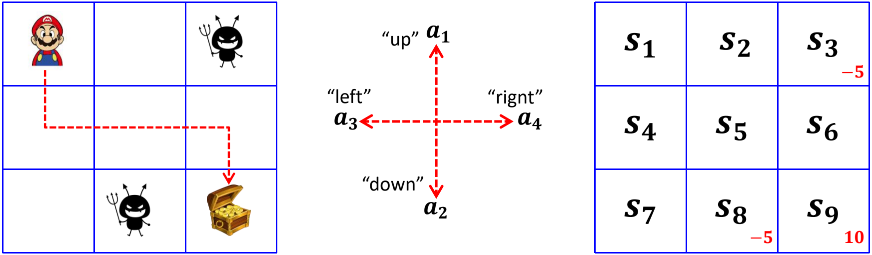
$$V^\pi(s) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s, \pi\right],$$

$$Q^\pi(s, a) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a\right].$$

Goal of RL is to a find a policy that maximizes weighted state values:

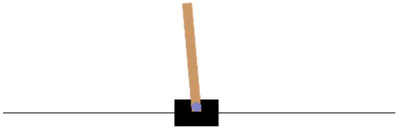
$$\max_\pi V^\pi(\mu), \quad \text{where } V^\pi(\mu) := \mathbb{E}_{s\sim\mu}\{V^\pi(s)\}.$$

Simple RL Example: GridWorld



- State space: $\mathcal{S} = \{s_i\}_{i=1}^{9}$
- Action space: $\mathcal{A} = \{a_i\}_{i=1}^{4}$
- Reward: $r = -5$ if hitting "obstacle" grid; $r = 10$ if arriving at "goal" grid
- Goal: Arriving "goal" grid while avoiding "obstacle" grid

Simple RL Example: CartPole



- State: $s = [x, y, \theta, \omega] \in \mathbb{R}^4$
  - $x \in [-4.8, 4.8]$: cart position
  - $y \in \mathbb{R}$: cart velocity
  - $\theta \in [-24^\circ, 24^\circ]$: pole angle
  - $\omega \in \mathbb{R}$: pole velocity at tip
- Action space: $\mathcal{A} = \{$left, right $\}$
- Reward: $r = 1$ if the pole remains upright, $r = 0$ otherwise
- Goal: prevent pole from falling over

More typical examples, such as Mountain Car and Cliff Walking, can be found in OpenAI Gym (https://github.com/openai/gym).

Basic RL Methods

- Valued-based methods: not directly optimize policy but seek optimal state or action values based on fixed point iteration or dynamic programming:

$$\begin{cases} \text{Value Iteration} \\ \text{Policy Iteration} \end{cases} \xrightarrow{\textit{sampling}} \begin{cases} \text{MC Learning} \\ \text{SARSA} \\ \text{Q-Learning} \end{cases} \xrightarrow[\textit{approximation}]{\textit{function}} \text{Deep Q-Learning}$$

Basic RL Methods

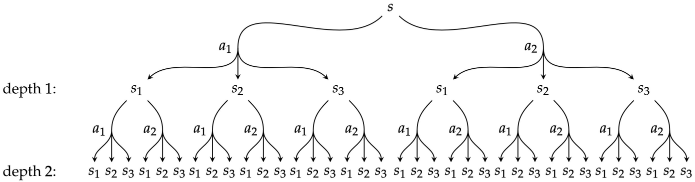
- Policy optimization: directly optimize policy via parameterization $\pi_\theta(\cdot|s)$:

$$V^{\pi_\theta}(\mu) = \mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})\right],$$

  where $P_\mu^{\pi_\theta}(\tau) = \mu(s_0)\prod_{t=0}^{\infty}\pi_\theta(a_t|s_t)p(s_{t+1}|s_t, a_t)$. Then maximize $V^{\pi_\theta}(\mu)$ is finite dimensional optimization problem about θ. Value exists in expression of policy gradient and policy optimization+value update= Actor-Critic.
- Online planning: MCTS (based on UCB for multi-armed bandit).

Policy optimization is also known as policy search, i.e., search over policy space directly. In contrast, value-based methods update state/action values and retrieve (optimal) policy from (optimal) state/action values.

Exploration & Exploitation



Since RL is about the search of “best” trajectory, a naive method is exhaustive search (suppose it is possible). However, computation cost will be prohibitively high, which requires smart search strategy.

- Exploitation: Use information of current experiences for efficient update;
- Exploration: Should allow more states and actions to be explored while using exploitation to reduce computation complexity.

Figure from “Algorithms for decision making” by Kochenderfer et al., 2022

Logistics

- **Prerequisites:** Probability and statistics, numerical optimization
- **Grading policy:** 60% Homework + 40% Final
- **Homework:**
  - Homework will be assigned via eLearning;
  - Coding language for this course is Python.
- **Course policies:**
  - Final exam is closed book.
  - Cheating in assignments and exams is not tolerated! Any sort of suspected cheating will result in zero grade of the corresponding assignments or exams, followed by penalty subject to university rules.

This course emphasizes basic methods and theory of RL, but hopefully there will be more practical projects.

Questions?