Deep Reinforcement Learning

A Few Deep RL Highlights

- 2013 – Atari (DQN) [DeepMind]
- 2014 – 2D locomotion (TRPO) [Berkeley]
- 2015 – AlphaGo [DeepMind]
- 2016 – Real Robot Manipulation (GPS) [Berkeley, Google]
- 2017 – Dota2 (PPO) [OpenAI]
- 2018 – DeepMimic [Berkeley]
- 2019 – AlphaStar [DeepMind]
- 2019 – Rubik’s Cube (PPO+DR) [OpenAI]
- 2020 – AlphaFold2 [DeepMind]
- 2022 – RLHF [OpenAI]

One Goal for This Class

- Lay The Foundation for Deep Reinforcement Learning!

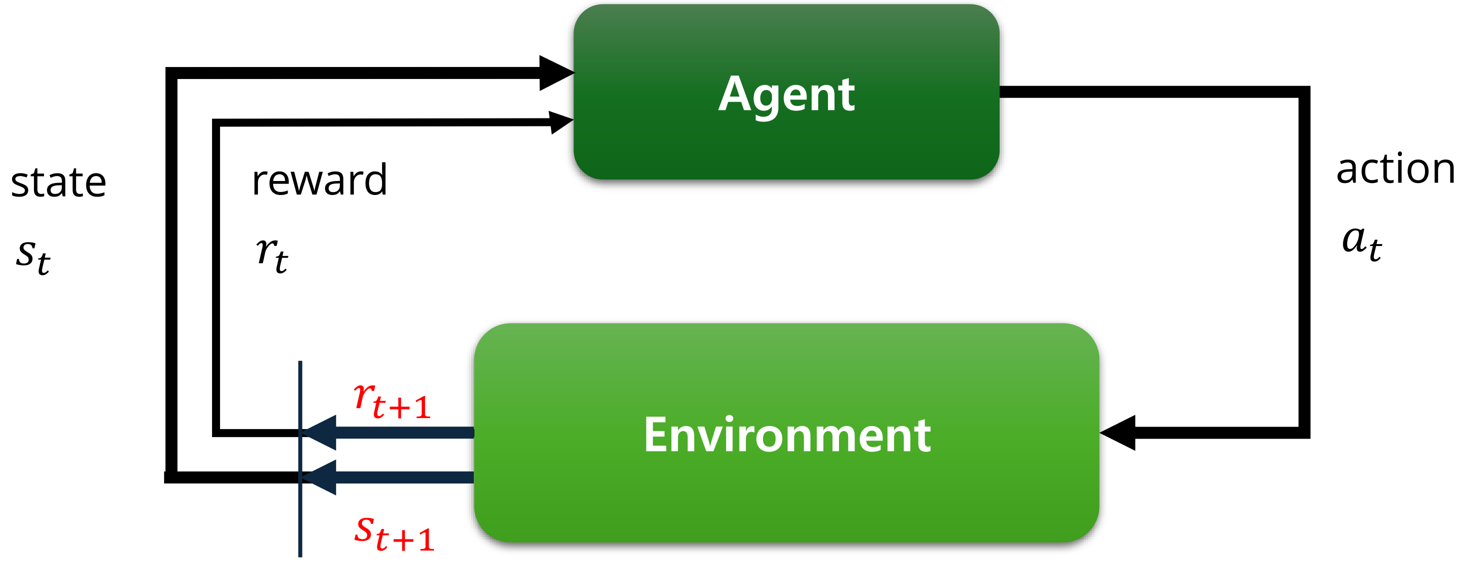
- How?
  - Understand basic theories.
  - Implement the theories and analyze the experimental results.

Fundamental Theories

1. Markov Decision Process
2. Deep Q-Learning
3. Policy Gradient
4. PPO(Proximal Policy Gradient)
5. DDPG(Deep Deterministic Policy Gradient)
6. Model Based Reinforcement Learning

Markov Decision Process

Markov Decision Process

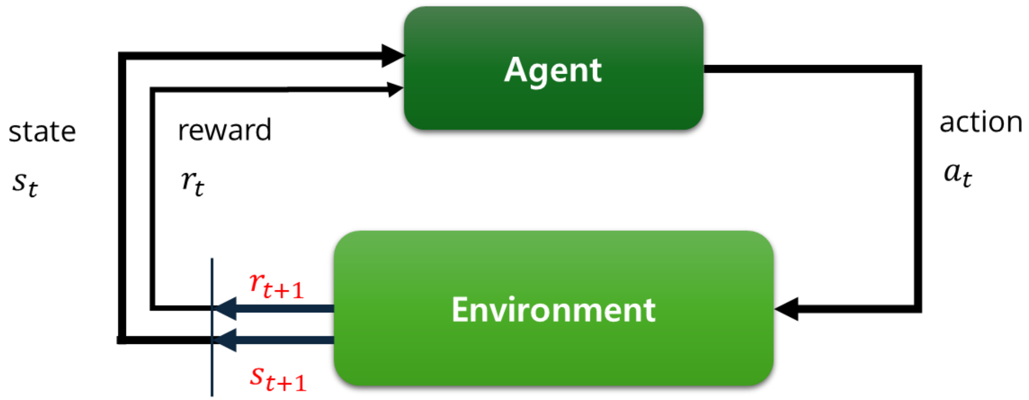


Assume that agent gets to observe the state.

MDP (Markov Decision Process)

An MDP is defined by

- Set of states S
- Set of actions A
- Transition function $P(s'|s, a)$
- Reward function $R(s, a, s')$
- Start state s_0
- Discount factor γ
- Horizon H



Example

MDP $S, A, P, R, s_0, \gamma, H$

Goal: $\max_{\pi} E\left[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi\right]$

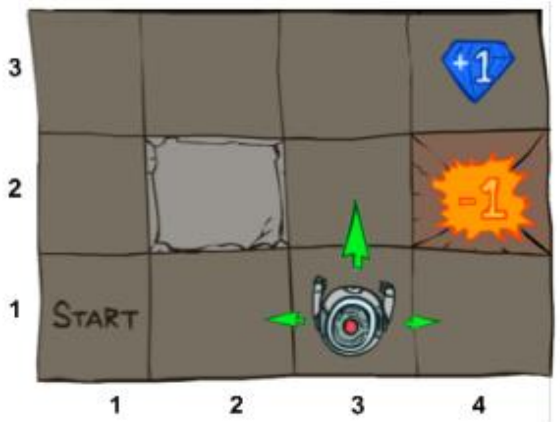
- Cleaning robot
- Walking robot
- Pole balancing
- Games: tetris, backgamon

- Server management
- Shortest path problems
- Models for animal, people

Example MDP: Grid World

An MDP is defined by

- Set of states S
- Set of actions A
- Transition function $P(s'|s,a)$
- Reward function $R(s,a,s')$
- Start state s_0
- Discount factor γ
- Horizon H

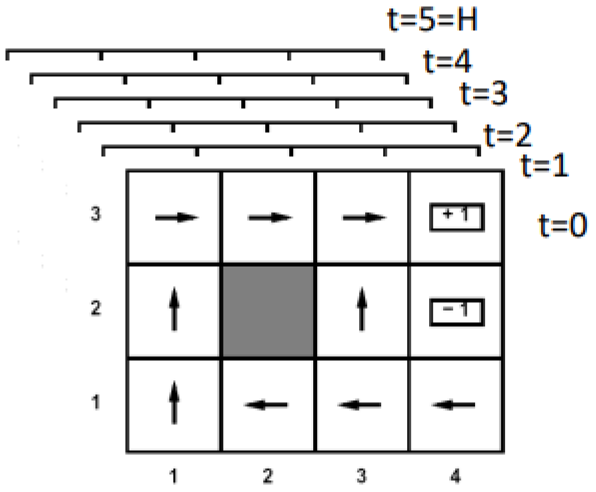


Goal: $$max_{\pi}\mathrm{E}[\sum_{t=0}^{H}\gamma^{t}R(S_t,A_t,S_{t+1})|\pi]$$

π:

Solving MDPs

- In an MDP, we want to find an optimal policy $\pi^*: S \times O: H \rightarrow A$
  - A policy π gives an action for each state for each time



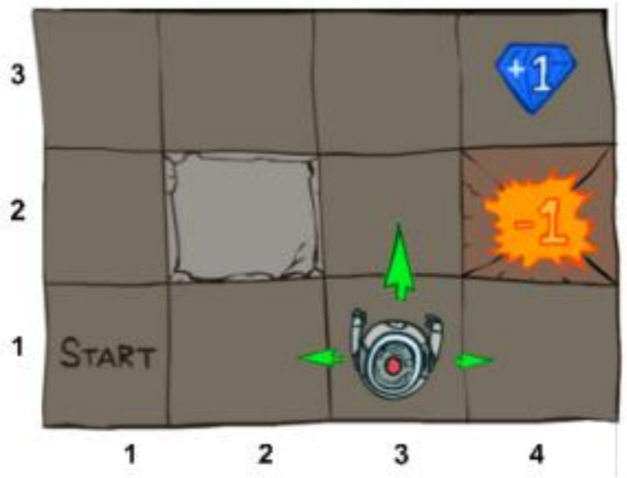
  - An optimal policy maximizes expected sum of rewards

- Contrast: If environment were deterministic, then we would just need an optimal plan, or sequence of actions, from start to a goal

Optimal Value Function V^*

$$V^*(s) = \max_{\pi} E\left[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi, s = s_o\right]$$

= sum of discounted rewards when starting from state s and **acting optimally**



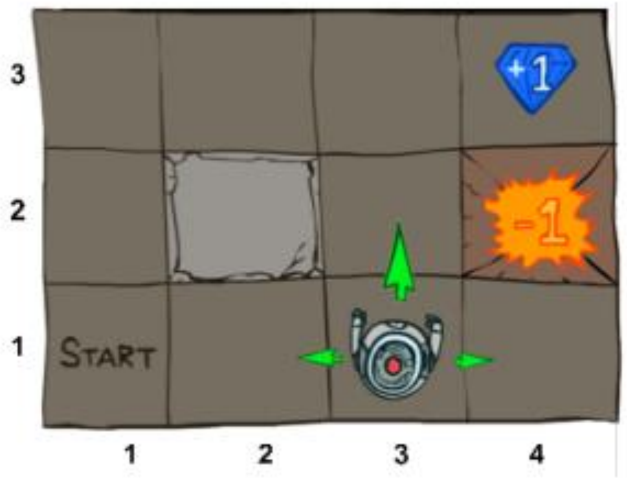
Let's assume:
actions deterministically successful, $\gamma = 1, H = 100$

- $V^*(4,3) =$ 1
- $V^*(3,3) =$ 1
- $V^*(2,3) =$ 1
- $V^*(1,1) =$ 1
- $V^*(4,2) =$ -1

Optimal Value Function V^*

$$V^*(s) = \max_{\pi} E\left[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi, s = s_o\right]$$

= sum of discounted rewards when starting from state s and **acting optimally**



Let's assume:

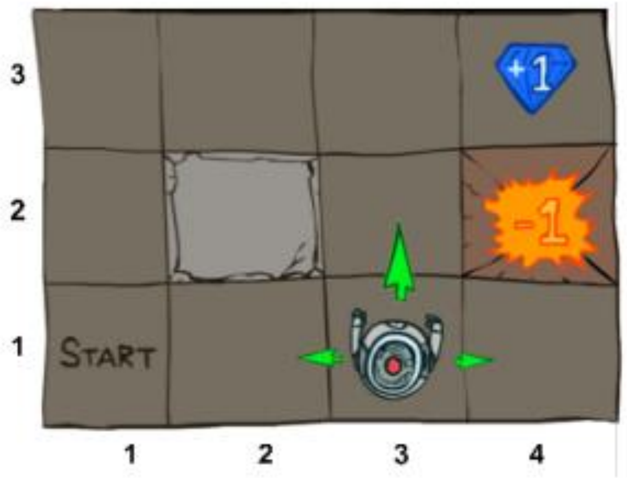
actions deterministically successful, $\gamma = 0.9, H = 100$

- $V^*(4,3) =$ 1
- $V^*(3,3) =$ 0.9 = 0.9
- $V^*(2,3) =$ 0.9 * 0.9 = 0.81
- $V^*(1,1) =$ 0.9 * 0.9 * 0.9 * 0.9 * 0.9 = 0.59
- $V^*(4,2) =$ -1

Optimal Value Function V^*

$$V^*(s) = \max_{\pi} E\left[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi, s = s_o\right]$$

= sum of discounted rewards when starting from state s and **acting optimally**



Let's assume:

actions successful w/p 0.8, $\gamma = 0.9, H = 100$

- $V^*(4, 3) =$ 1
- $V^*(3, 3) =$ $0.8 * 0.9 * V^*(4,3)$

  right(0.8), up(0.1), down(0.1) $+ 0.1 * 0.9 * V(3,3)$ $+ 0.1 * 0.9 * V(3,2)$

Value Iteration

- $V_0^*(s)$ = optimal value for state s when $H = 0$
  - $V_0^*(s) = 0 \quad \forall s$
- $V_1^*(s)$ = optimal value for state s when $H = 1$
  - $V_1^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_0^*(s'))$
- $V_2^*(s)$ = optimal value for state s when $H = 1$
  - $V_2^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_1^*(s'))$
- $V_k^*(s)$ = optimal value for state s when $H = 1$
  - $V_k^*(s) = \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$

Value Iteration

Algorithm

Start with $V_0^*(s) = 0$ for all s

For $k = 1, \dots, H$:

For all state s in S:

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$$

$$\pi_k^*(s) \leftarrow \arg\max_a \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma V_{k-1}^*(s'))$$

This is called a value update or Bellman update/back-up