

Deep Reinforcement Learning

[Kaixiang Lin](#)

Email: kaixianglin.cs@gmail.com

Outline

Why RL? Motivation

What is RL? Key Concepts & Terminology

How to solve RL?

- Dive into RL algorithms, from basics to the SOTA.
- RL algorithm overview

Real World Application: RLHF in ChatGPT

Part 0: Why RL?

From Games to Real World Impact

Human Level Performance on Games

- In 2015, Deep Learning's impact propagate to Reinforcement Learning. We have human level performance on Atari games.

LETTER

doi:10.1038/nature14236

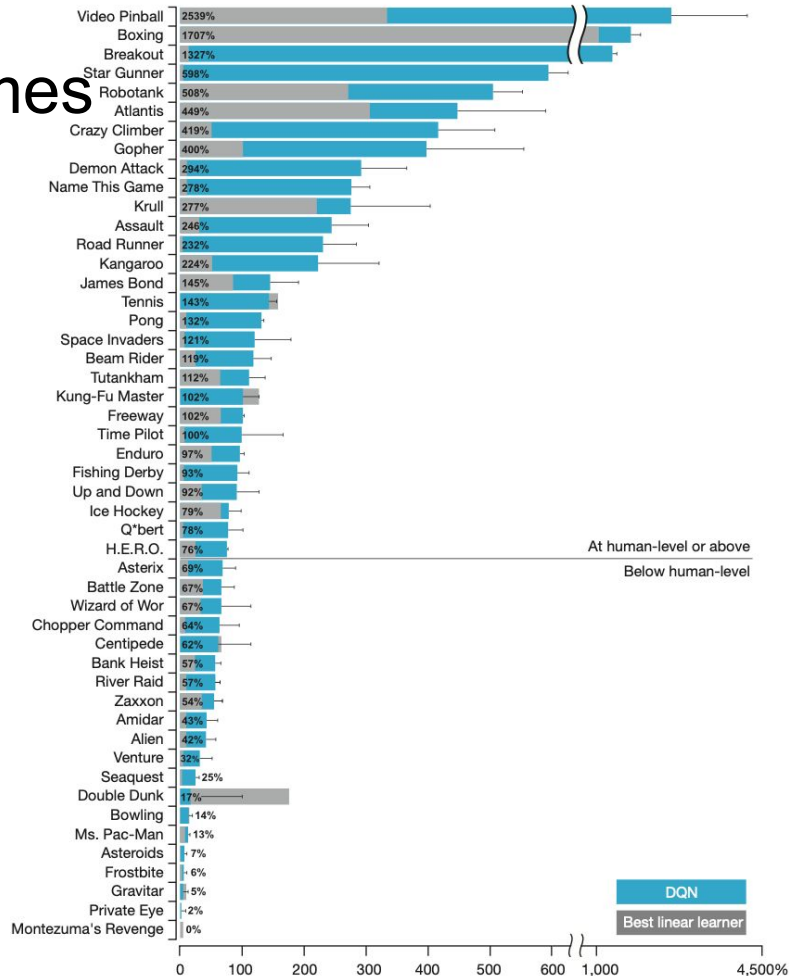
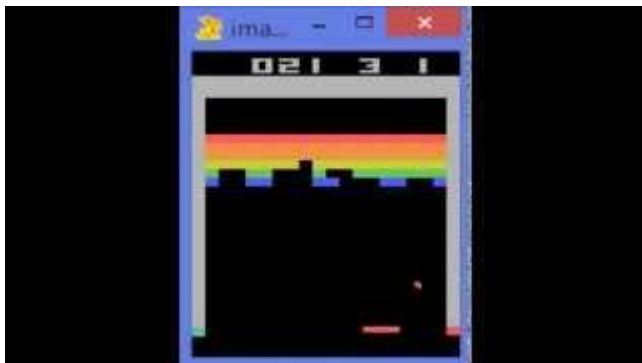
Human-level control through deep reinforcement learning

Volodymyr Mnih^{1*}, Koray Kavukcuoglu^{1*}, David Silver^{1*}, Andrei A. Rusu¹, Joel Veness¹, Marc G. Bellemare¹, Alex Graves¹, Martin Riedmiller¹, Andreas K. Fiedjeland¹, Georg Ostrovski¹, Stig Petersen¹, Charles Beattie¹, Amir Sadik¹, Ioannis Antonoglou¹, Helen King¹, Dharshan Kumaran¹, Daan Wierstra¹, Shane Legg¹ & Demis Hassabis¹



Human Level Performance on Games

- In 2015, Deep Learning's impact propagate to Reinforcement Learning. We have human level performance on Atari games.



Mastering the game of Go

- In 2016, AlphaGo beats world Champion Lee Sedol. 4-1 in a five-game Go match.

Mastering the game of Go with deep neural networks and tree search

David Silver^{1*}, Aja Huang^{1*}, Chris J. Maddison¹, Arthur Guez¹, Laurent Sifre¹, George van den Driessche¹, Julian Schrittwieser¹, Ioannis Antonoglou¹, Veda Panneershelvam¹, Marc Lanctot¹, Sander Dieleman¹, Dominik Grewe¹, John Nham², Nal Kalchbrenner¹, Ilya Sutskever², Timothy Lillicrap¹, Madeleine Leach¹, Koray Kavukcuoglu¹, Thore Graepel¹ & Demis Hassabis¹



Human-level ChatBot

- In 2022, ChatGPT reveals the new era of Generative AI.

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*

Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell[†]

Peter Welinder

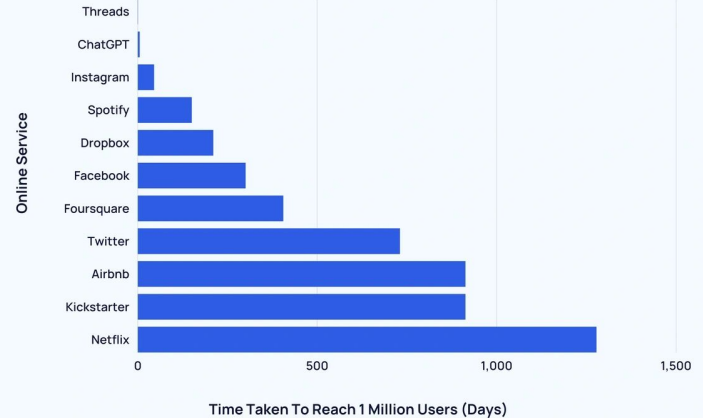
Paul Christiano*[†]

Jan Leike*

Ryan Lowe*

OpenAI

Time taken to reach 1 million users



Human-level ChatBot

- In 2022, ChatGPT reveals the new era of Generative AI.

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*

Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell†

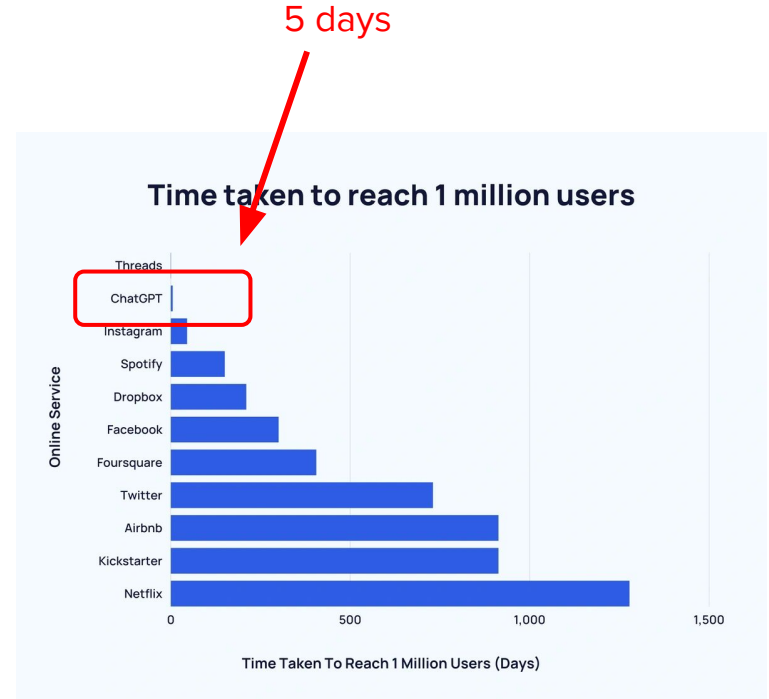
Peter Welinder

Paul Christiano*†

Jan Leike*

Ryan Lowe*

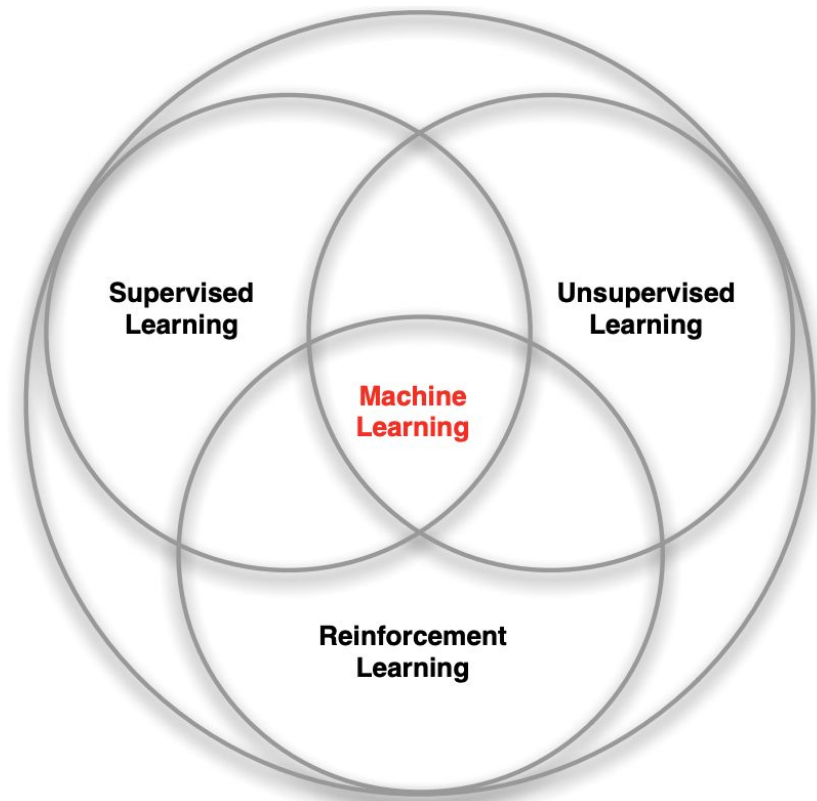
OpenAI



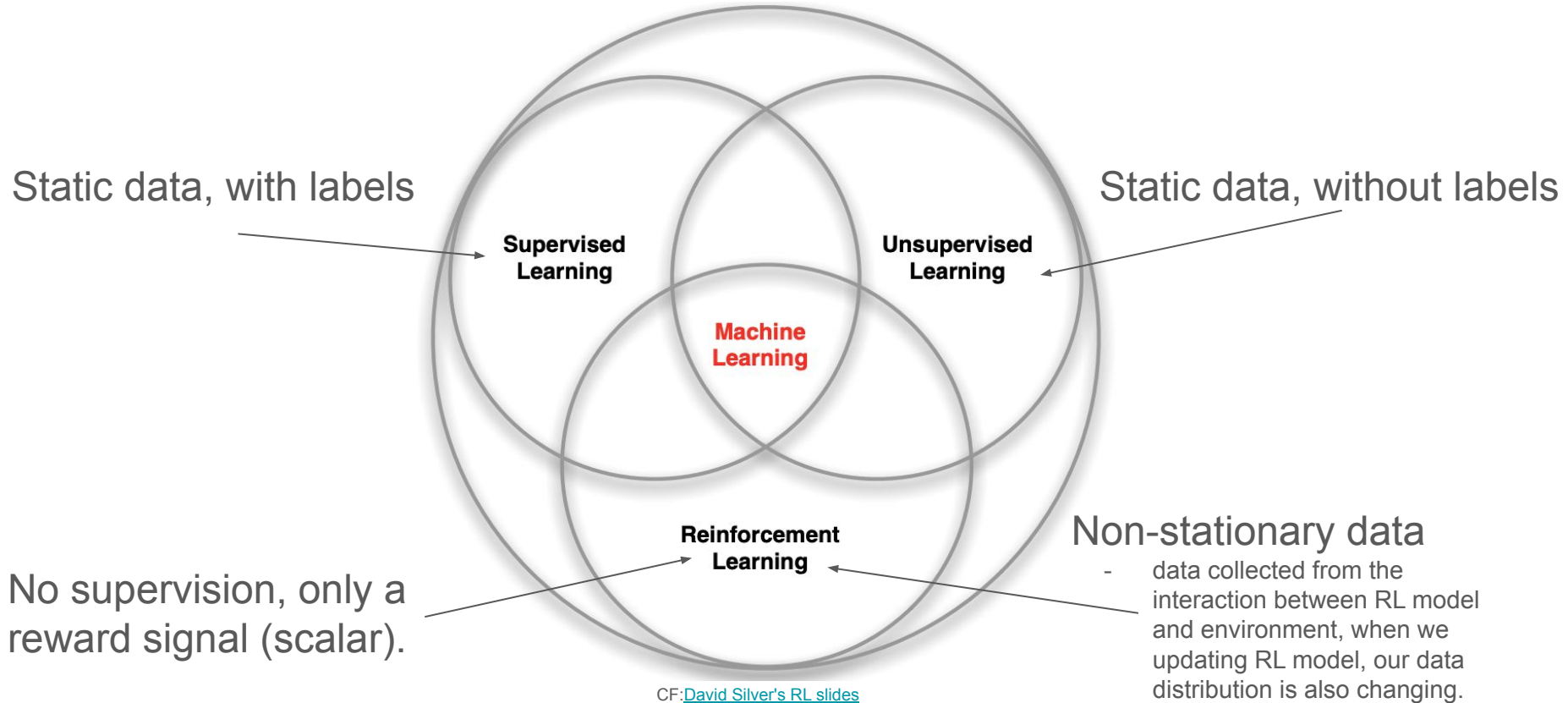
Part I: Key Concepts

What is RL? What RL is solving?

Where does RL stand in ML?

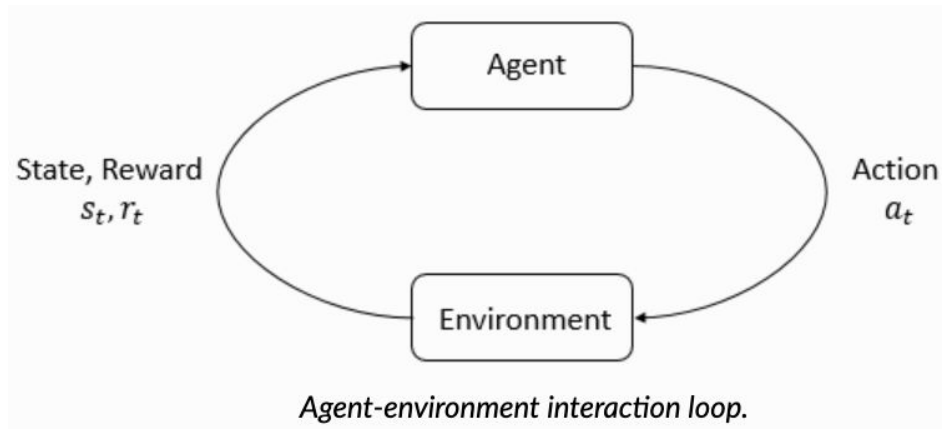


Where does RL stand in ML?



What is RL

- **Agent** interacting with environment:
- At every step t ,
- Agent seeing **state** of the world,
- Then agent deciding taking an **action**,
- The environment transits to next **state**,
- Agent might receive a **reward** from the world.
- The **goal** of agent is to maximize the **cumulative reward**.



CF: [OpenAI Spinning Up](#)

What is RL?

Let's look at an example!

- Goal: kill the opponent.
- State: board
- Action: move around or lay a bomb.
- Reward: -2 for each step,
100 for killing opponent.
- Horizon: finite.



What is RL?

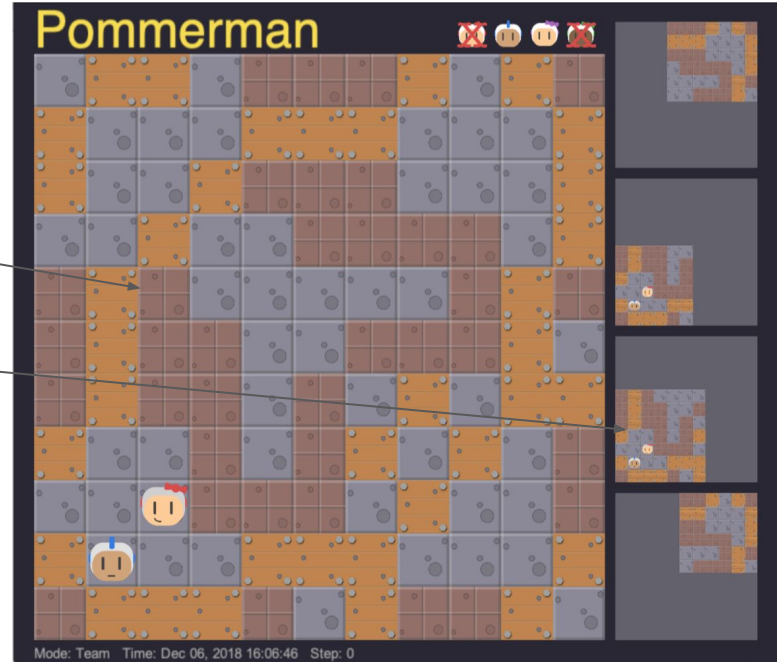
- **Agent and Environment**
- **Agent:** the man.
- **Environment:** the map.



What is RL?

States and Observations

- A **state** s is a complete description of the state of the world.
E.g., the entire status of map at a time step.
- An **observation** o is a partial description of a state which may omit information.
 - E.g., the part of map that man see.



What is RL?

States and Observations

- A **state** s is a complete description of the state of the world.
E.g., the entire status of map at a time step.
- An **observation** o is a partial description of a state which may omit information.
 - E.g., the part of map that man see.
- If observation = state, we say that the environment
 - is fully observed.
 - o.w. it is partially observed.

For simplicity, across this course, we use fully observed setting

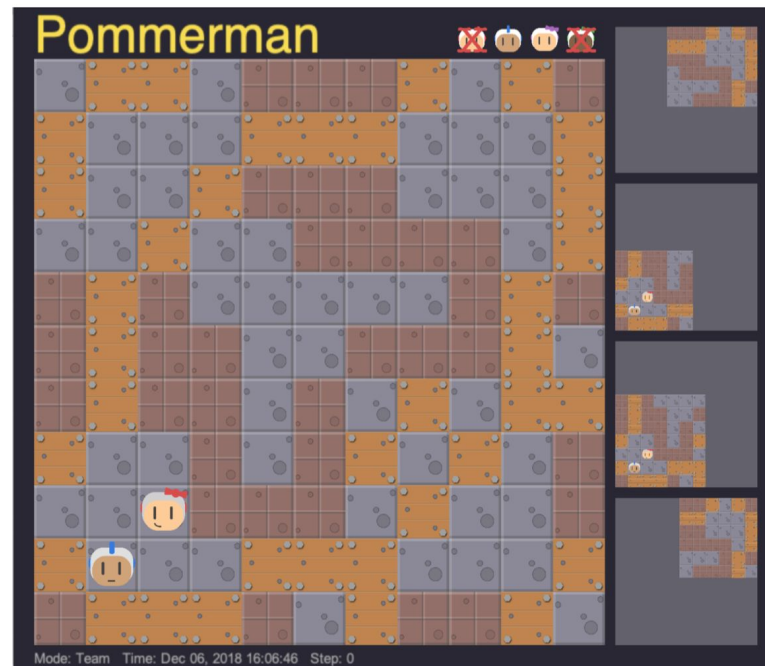


What is RL?

Action spaces and Policies

- **Action spaces:** The set of all validation actions.
 - E.g., move up, right, left, down, lay a bomb.
- **Policies:**
- E.g., Given what Pommerman see, red agent lay a bomb

$$a_t \sim \pi(\cdot | s_t)$$



What is RL?

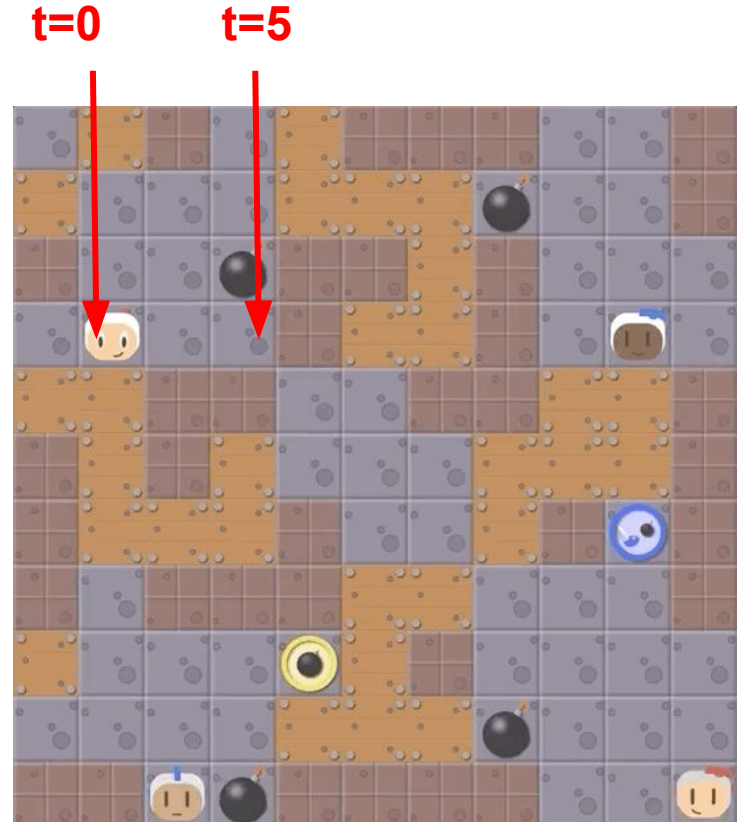
Trajectories(τ), Transition Probabilities

- A trajectory is a sequence of states and actions in the world.

$$\tau = (s_0, a_0, s_1, a_1, \dots)$$

- Transition Probabilities.
Given state s_t , if agent take action a_t , what's the probability of next states. (can be deterministic)

$$s_{t+1} \sim P(\cdot | s_t, a_t).$$



What is RL?

Reward, Return.

- The reward function R is a mapping from the current state of the world, the action just taken, and the next state of the world, to a scalar.

$$r_t = R(s_t, a_t, s_{t+1})$$

- It's usually simplified as $r_t = R(s_t, a_t)$
- The goal of the agent is to maximize some notion of cumulative reward over a trajectory.

$r(s_0, a_0)=0$

$r(s_{10}, a_{10})=100$



RL Formulation

Markov Decision Process (MDP)

- Set of states S
- Set of actions A
- Transition function $P(s'|s, a)$
 - Markov Property: future states only depends on current states and action.
- Reward function $R(s, a, s')$

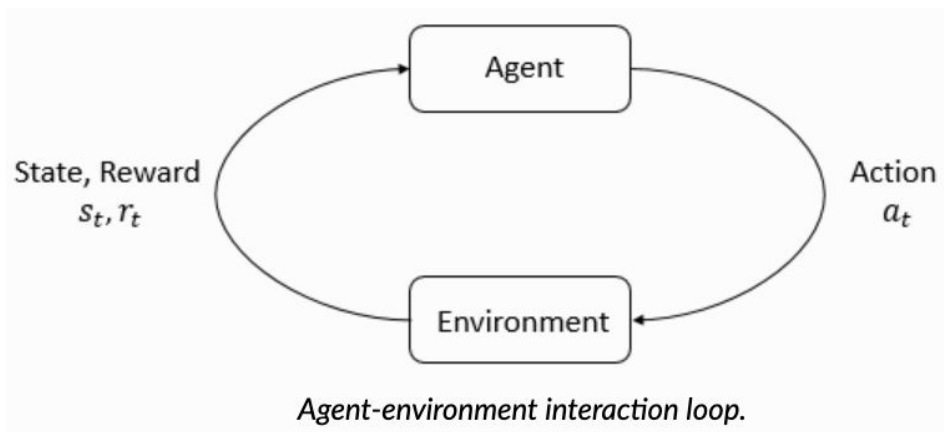
- Start state s_0

- Discount factor: γ

- Horizon: H

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

alternatively
$$J(\pi_{\theta}) = \int_{\tau} P(\tau | \theta) R(\tau)$$



CF: [OpenAI Spinning Up](#)

Finite horizon

$$R(\tau) = \sum_{t=0}^T \gamma^t R(s_t, a_t)$$

Infinite horizon

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t.$$

RL Formulation

$$J(\pi_\theta) = \int_{\tau} P(\tau|\theta)R(\tau)$$

Trajectory probability

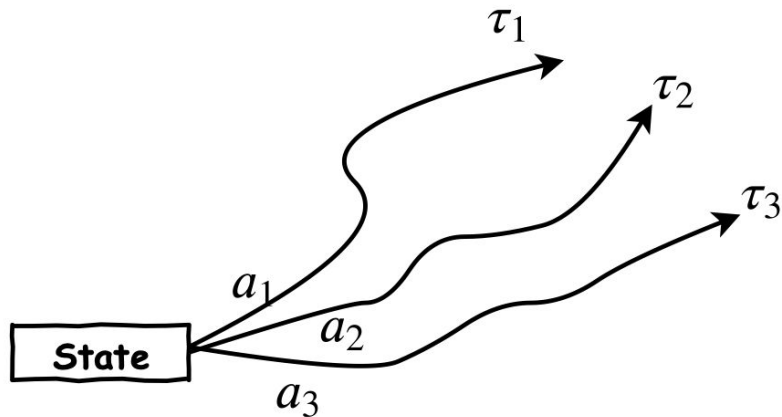
$$P(\tau|\theta) = \rho_0(s_0)\prod_{t=0}^T P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

Trajectory reward

$$R(\tau) = \sum_{t=0}^T R(s_t, a_t)$$

Goal of RL :

$$\pi^* = \arg \max_{\pi} J(\pi)$$

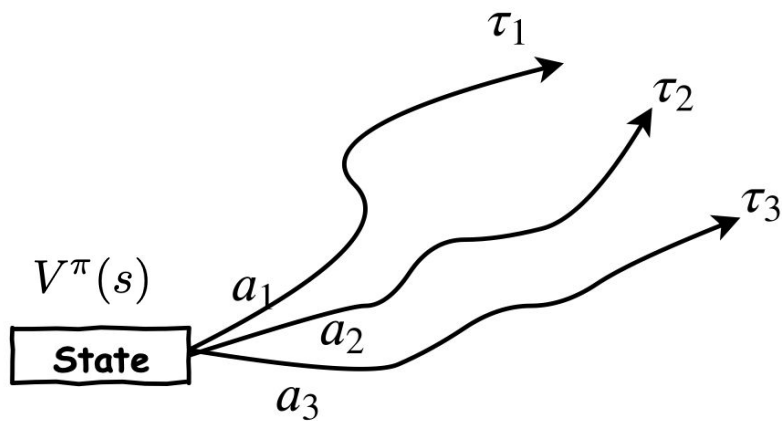


Value functions - On Policy

- the expected return if you start in that **state** or **state-action** pair, and then act according to a particular policy forever after

State-Value function

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$



Value functions - On Policy

- the expected return if you start in that **state** or **state-action** pair, and then act according to a particular policy forever after

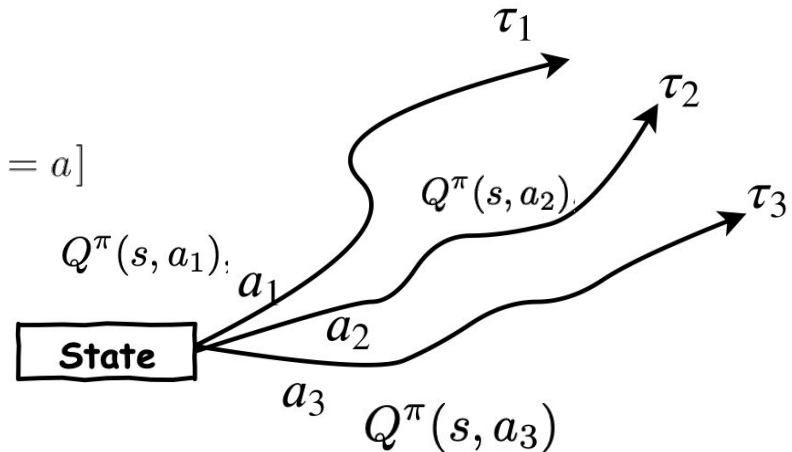
State-Value function

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$

Action-Value function

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a]$$

The on-policy value function evaluates the performance of current policy π .



Value functions - On Policy

- the expected return if you start in that **state** or **state-action** pair, and then act according to a particular policy forever after

State-Value function

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$

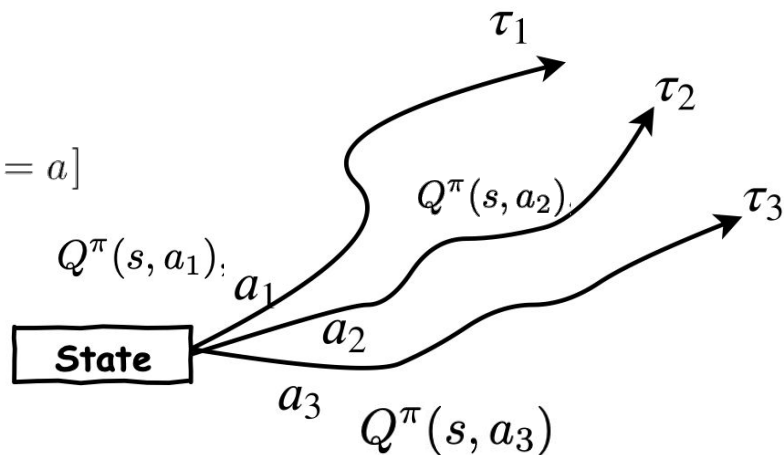
Action-Value function

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a]$$

Connection of two value functions:

$$V^\pi(s) = \mathbb{E}_{a \sim \pi} [Q^\pi(s, a)]$$

State value function is **expectation** of
Action value function over all possible actions.



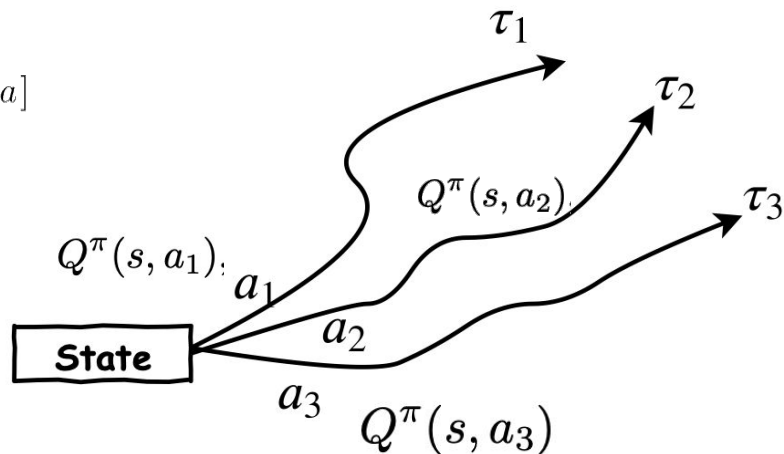
Value functions - Optimal Policy

- the expected return if you start in that **state** or **state-action** pair, and then act according to the best policy after that.

State-Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Action-Value function $Q^*(s, a) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a]$

The performance of best policy.



Value functions - Optimal Policy

- the expected return if you start in that **state** or **state-action** pair, and then act according to the best policy after that.

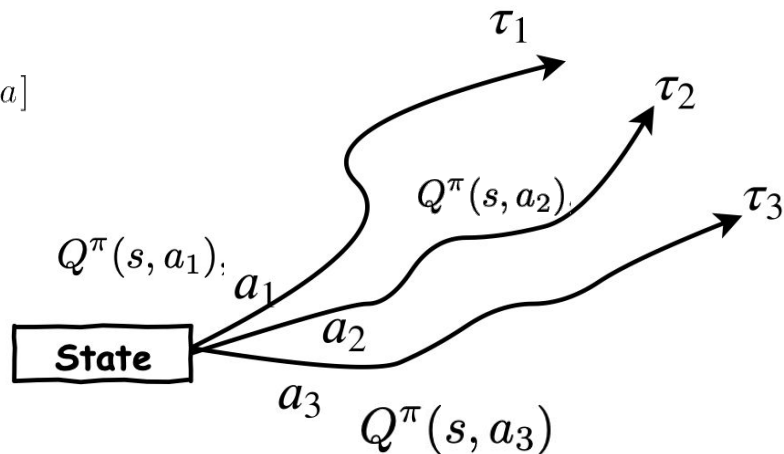
State-Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Action-Value function $Q^*(s, a) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a]$

Connection of two value functions:

$$V^*(s) = \max_a Q^*(s, a)$$

State value function is **max** of
Action value function over all possible actions.



Bellman Equations - On policy

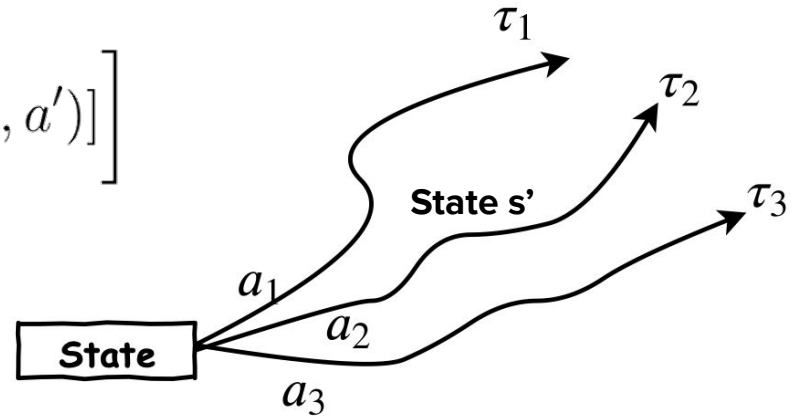
- The value of your starting point is the reward you expect to get from being there, plus the value of wherever you land next.

$$V^\pi(s) = \mathbb{E}_{\substack{a \sim \pi \\ s' \sim P}} [r(s, a) + \gamma V^\pi(s')],$$

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P} \left[r(s, a) + \gamma \mathbb{E}_{a' \sim \pi} [Q^\pi(s', a')] \right]$$

Current time step t

Next time step t+1



Bellman Equations - Optimal policy

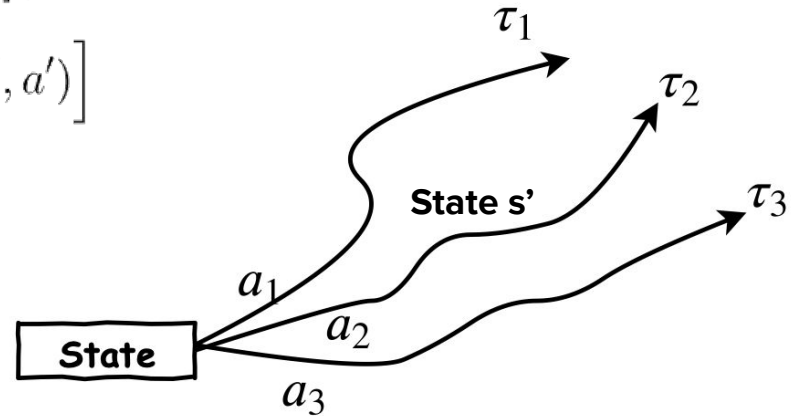
- The value of your starting point is the reward you expect to get from being there, plus the value of wherever you land next, following the optimal policy.

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$

$$Q^*(s, a) = \mathbb{E}_{s' \sim P} [r(s, a) + \gamma \max_{a'} Q^*(s', a')]$$

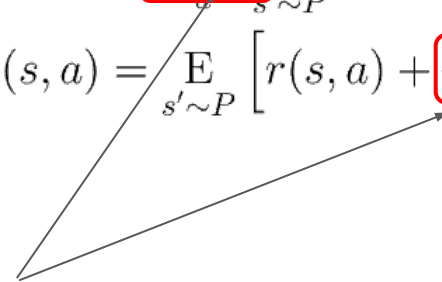
Current time step t

Next time step t+1

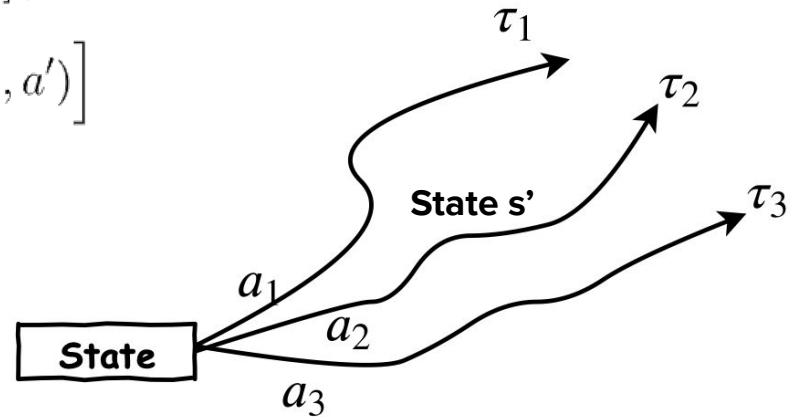


Bellman Equations - Optimal policy

- The value of your starting point is the reward you expect to get from being there, plus the value of wherever you land next, following the optimal policy.

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$
$$Q^*(s, a) = \mathbb{E}_{s' \sim P} [r(s, a) + \gamma \max_{a'} Q^*(s', a')]$$


Whether we have max over actions differentiates the on-policy vs optimal value function Bellman Equation.



Key Concepts

- RL is learning by trial and error.
- Bellman Equation: Relation between current state and future state.
- Exploration & Exploitation trade off
- On-policy vs Off-policy
- Online vs offline vs Hybrid
- Bias and Variance trade off

Part II: How to solve RL?

RL Algorithms (What we will cover?)

- Exact methods
 - **Value Iteration & Policy iteration**
- Deep RL algorithms [Model-free, Online approach]
 - **Policy gradient.**
 - **Actor acritic.**
 - Trust Region Policy Optimization.
 - **Proximal Policy Optimization.**
 - DQN
 - DDPG
 - SAC

RL Algorithms (What we will cover?)

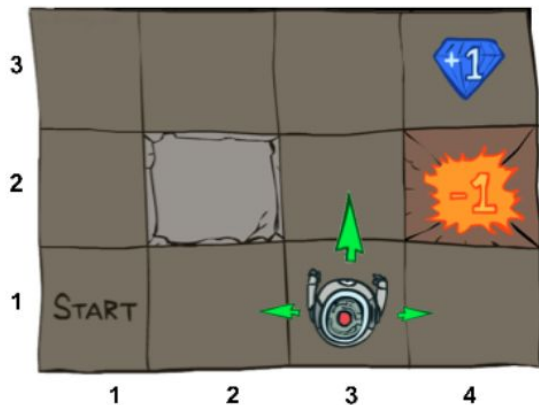
- Exact methods
 - Value Iteration & Policy iteration
- Deep RL algorithms [Model-free, Online approach]
 - **Policy gradient.**
 - **Actor acritic.**
 - Trust Region Policy Optimization.
 - **Proximal Policy Optimization.**
 - DQN
 - DDPG
 - SAC

Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')];$$



Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$

$$V^*(4,3) =$$

$$V^*(3,3) =$$

$$V^*(2,3) =$$

$$V^*(1,1) =$$

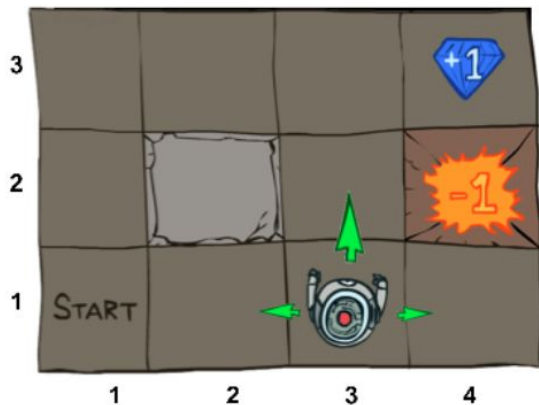
$$V^*(4,2) =$$

Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$



Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$

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

$$V^*(3,3) =$$

$$V^*(2,3) =$$

$$V^*(1,1) =$$

$$V^*(4,2) =$$

Computing Optimal Value Function

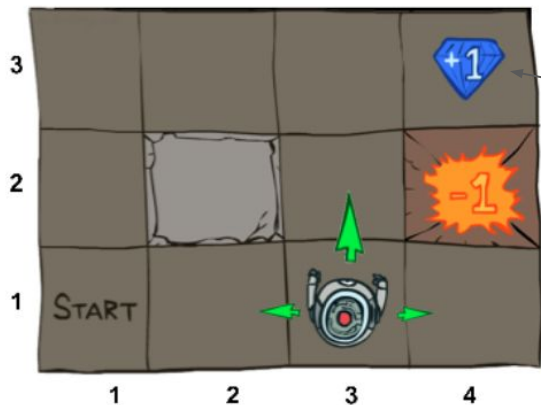
Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$

Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$



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

$$V^*(3,3) =$$

$$V^*(2,3) =$$

$$V^*(1,1) =$$

$$V^*(4,2) =$$

Computing Optimal Value Function

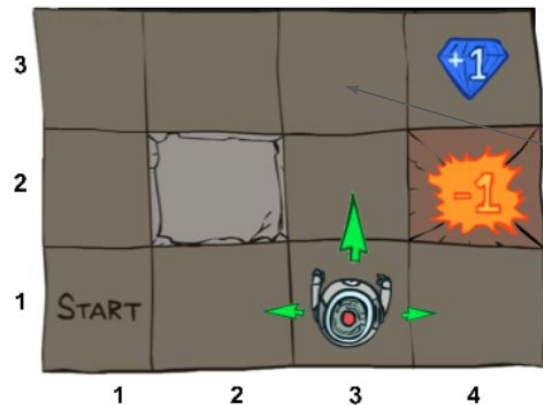
Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$

Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$



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

$$V^*(3,3) = 1$$

$$V^*(2,3) =$$

$$V^*(1,1) =$$

$$V^*(4,2) =$$

Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$

Let's assume:

actions deterministically successful, $\gamma = 1, H = 100$

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

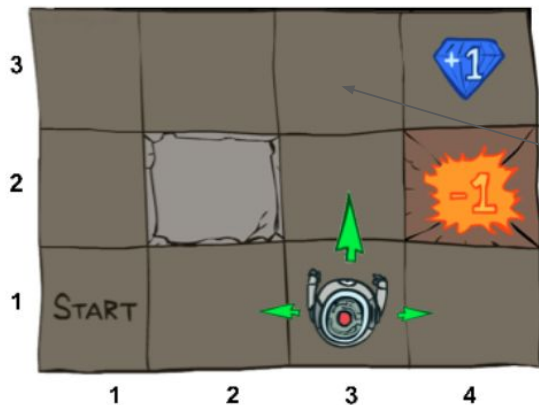
$$V^*(3,3) =$$

$$V^*(2,3) =$$

$$V^*(1,1) =$$

$$V^*(4,2) =$$

Starting at (3,3), acting optimally (i.e., moving right), reach (4,3) deterministically, then get +1 reward.



Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$

Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$

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

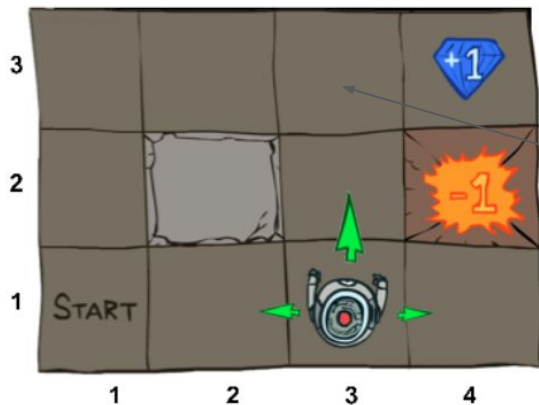
$$V^*(3,3) = 1$$

$$V^*(2,3) =$$

$$V^*(1,1) =$$

$$V^*(4,2) =$$

Starting at (3,3), acting optimally (i.e., moving right), reach (4,3) deterministically, then get +1 reward.

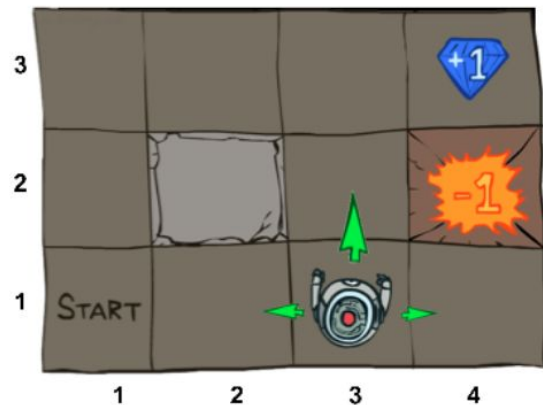


Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')];$$



Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$

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

$$V^*(3,3) = 1$$

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

$$V^*(1,1) = 1$$

$$V^*(4,2) =$$

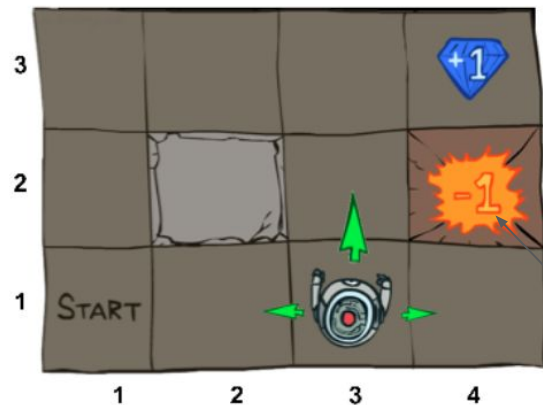
Similarly for (2,3) and (1, 1), the optimal policy is to move to (4,3) directly and collect reward 1.

Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$



Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$

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

$$V^*(3,3) = 1$$

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

$$V^*(1,1) = 1$$

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

Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')];$$

Let's assume:

actions deterministically successful, $\gamma = 1, H = 100$

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

$$V^*(3,3) = 1$$

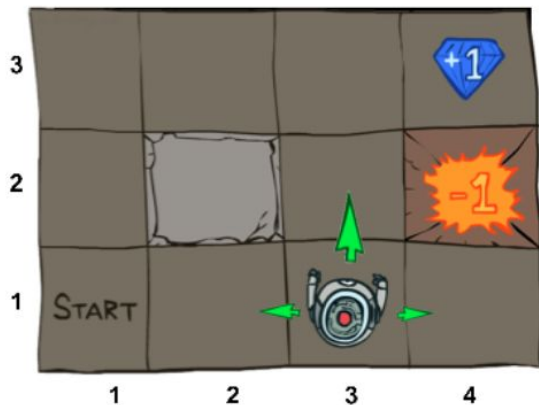
$$V^*(2,3) = 1$$

$$V^*(1,1) = 1$$

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

We are using the summation of trajectory reward to compute value function.

What if we use Bellman equation?



Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')];$$

Let's assume:

actions deterministically successful, $\gamma = 1, H = 100$

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

$$V^*(3,3) = 1$$

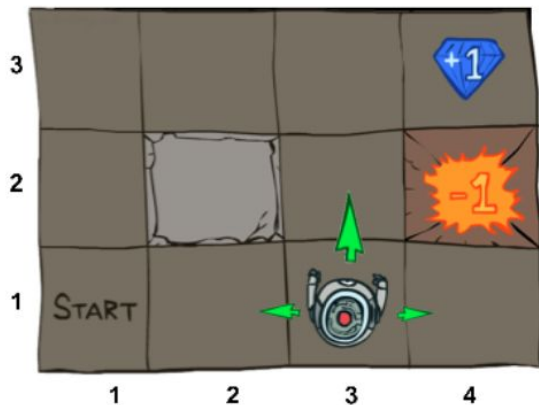
$$V^*(2,3) = 1$$

$$V^*(1,1) = 1$$

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

We are using the summation of trajectory reward to compute value function.

What if we use Bellman equation?

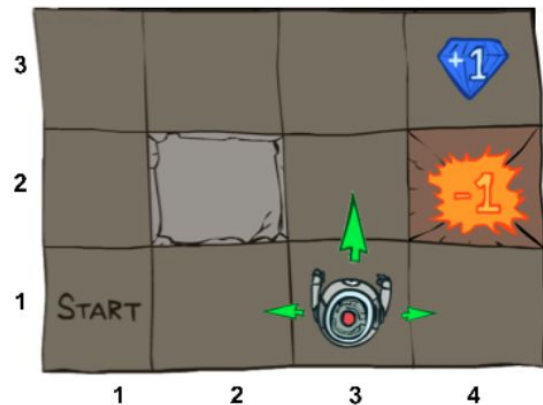


Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$



Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$

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

$$V^*(3,3) = 1 \quad V^*(2,3) = r(s = (2,3), a = \text{moving up}) + \gamma * V^*(3,3) :$$

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

$$V^*(1,1) = 1$$

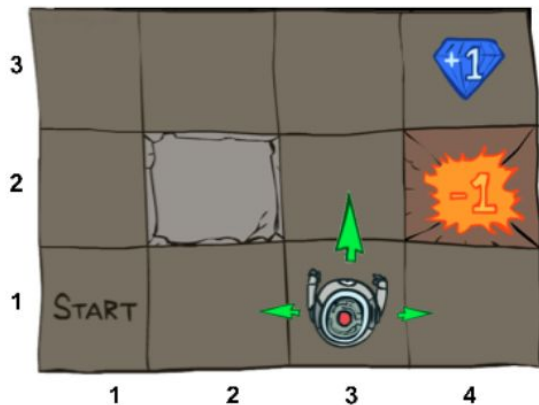
$$V^*(4,2) = -1$$

Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$



Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$

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

$$V^*(3,3) = 1 \quad V^*(2,3) = r(s = (2,3), a = \text{moving up}) + \gamma * V^*(3,3) :$$

$$V^*(2,3) = 1 \quad \quad \quad 0 \quad \quad \quad 1 * 1$$

$$V^*(1,1) = 1$$

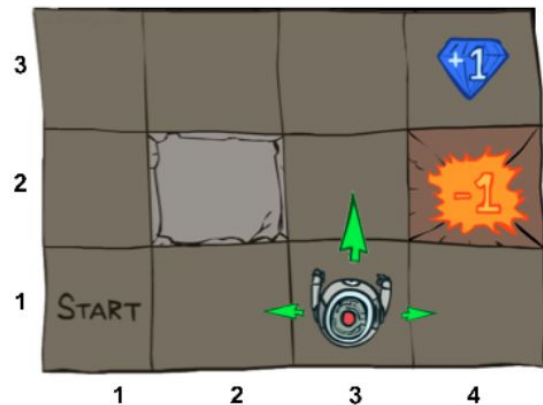
$$V^*(4,2) = -1$$

Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$



Let's assume:

actions deterministically successful, $\gamma = 1$, $H = 100$

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

$$V^*(3,3) = 1$$

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

$$\begin{aligned} V^*(2,3) &= r(s = (2,3), a = \text{moving up}) + \gamma * V^*(3,3) \\ &= 0 + 1 * 1 = 1 \end{aligned}$$

$$V^*(1,1) = 1$$

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

Computing Optimal Value Function

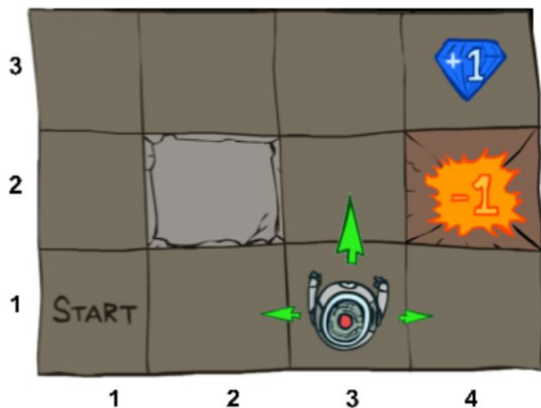
Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')];$$

Let's assume:

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



$$V^*(4,3) =$$

$$V^*(3,3) =$$

$$V^*(2,3) =$$

$$V^*(1,1) =$$

$$V^*(4,2) =$$

Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')],$$

Let's assume:

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

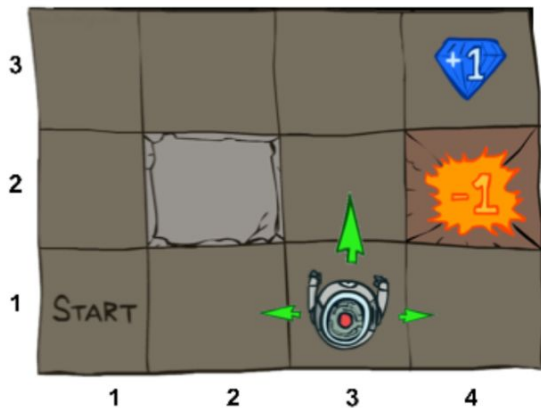
$$V^*(4,3) = 1$$

$$V^*(3,3) = 0.8 * 0.9 * V^*(4,3) + 0.1 * 0.9 * V^*(2,3) + 0.1 * 0.9 * V^*(3,2)$$

$$V^*(2,3) = \text{Taking best action from (3, 3): moving right.}$$

$$V^*(1,1) =$$

$$V^*(4,2) =$$



Computing Optimal Value Function

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')];$$

Let's assume:

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

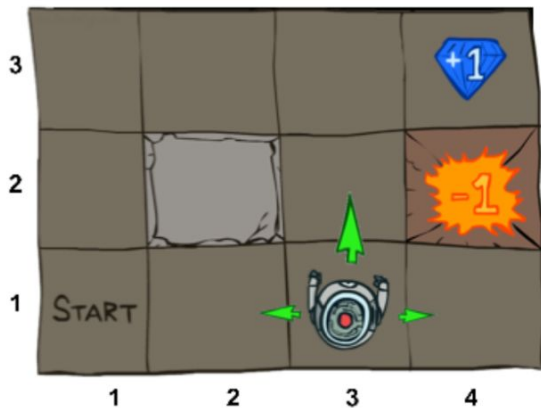
$$V^*(4,3) = 1$$

$$V^*(3,3) = 0.8 * 0.9 * V^*(4,3) + 0.1 * 0.9 * V^*(2,3) + 0.1 * 0.9 * V^*(3,2)$$

$$V^*(2,3) = \text{Recursive equation to the neighboring states}$$

$$V^*(1,1) =$$

$$V^*(4,2) =$$



Value Iteration

Indexing by how many steps left in the future, if 0, means no more steps left for us in the future. We initialize it by 0.

- $V_0^*(s)$ = 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 $H=2$
 - $V_2^*(s) = \max_a \sum_{s'} P(s'|s, a)(R(s, a, s') + \gamma V_1^*(s'))$
- $V_k^*(s)$ = optimal value for state s when $H = k$
 - $V_k^*(s) = \max_a \sum_{s'} P(s'|s, a)(R(s, a, s') + \gamma V_{k-1}^*(s'))$

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 $H=2$
 - $V_2^*(s) = \max_a \sum_{s'} P(s'|s, a)(R(s, a, s') + \gamma V_1^*(s'))$
- $V_k^*(s)$ = optimal value for state s when $H = k$
 - $V_k^*(s) = \max_a \sum_{s'} P(s'|s, a)(R(s, a, s') + \gamma V_{k-1}^*(s'))$

Decomposing to immediate reward, and one time step less time step left value function.

And then averaged across all possible neighboring states.

Value iteration in grid world

Recall the definition of Optimal Value function $V^*(s) = \max_{\pi} \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$

Bellman equation:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')];$$

Let's assume:

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

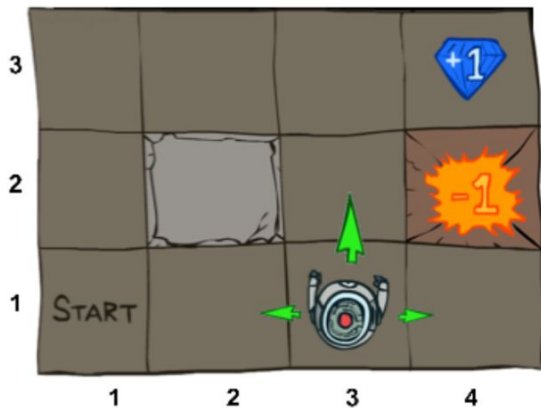
$$V^*(4,3) = 1$$

$$V^*(3,3) = 0.8 * 0.9 * V^*(4,3) + 0.1 * 0.9 * V^*(3,3) + 0.1 * 0.9 * V^*(3,2)$$

$$V^*(2,3) = \text{Revisit our previous example}$$

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

$$V^*(4,2) =$$



Value Iteration

Algorithm:

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

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

For all states 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**

Value Iteration

Algorithm:

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

For $k = 1, \dots, H$: Converge at infinite horizon: $H = \infty$

For all states 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**

Q-Value Iteration

Bellman Equation:

$$Q^*(s, a) = \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q^*(s', a'))$$

Q-Value Iteration:

$$Q_{k+1}^*(s, a) \leftarrow \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q_k^*(s', a'))$$

Q-Value Iteration

Bellman Equation:

$$Q^*(s, a) = \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q^*(s', a'))$$

Q-Value Iteration:

$$Q_{k+1}^*(s, a) \leftarrow \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma \max_{a'} Q_k^*(s', a'))$$

In Tabular cases, Bellman Backup -> Value iteration.

In large-scale cases, Bellman Backup -> Regression target for Q-learning

Similar for learning state value function, we will see soon.

Policy Iteration

- Policy evaluation for current policy π_k :

- Iterate until convergence

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s)) [R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s')]$$

- Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg \max_a \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma V^{\pi_k}(s')]$$

Policy Iteration

Difference value iteration between policy evaluation.

■ Policy evaluation for current policy π_k :

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

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} P(s'|s, \pi_k(s)) [R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s')]$$

■ Policy improvement: find the best action according to one-step look-ahead

$$\pi_{k+1}(s) \leftarrow \arg \max_a \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma V^{\pi_k}(s')]$$

RL Algorithms (What we will cover?)

- Exact methods
 - Value Iteration & Policy iteration
- Deep RL algorithms [Model-free, Online approach]
 - Policy gradient.
 - Actor acritic.
 - Trust Region Policy Optimization.
 - **Proximal Policy Optimization.**
 - DQN
 - DDPG
 - SAC

Policy Gradient

Recall Goal of RL

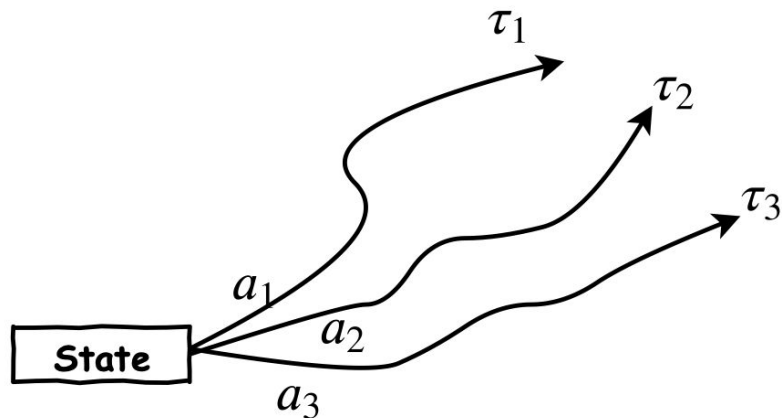
$$J(\pi_\theta) = \int_{\tau} P(\tau|\theta)R(\tau)$$

Trajectory probability

$$P(\tau|\theta) = \rho_0(s_0)\prod_{t=0}^T P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

Trajectory reward

$$R(\tau) = \sum_{t=0}^T R(s_t, a_t)$$



Policy Gradient

Recall Goal of RL

$$J(\pi_\theta) = \int_{\tau} P(\tau|\theta)R(\tau)$$

← Expected return

Trajectory probability

$$P(\tau|\theta) = \rho_0(s_0)\prod_{t=0}^T P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

Trajectory reward

$$R(\tau) = \sum_{t=0}^T R(s_t, a_t)$$

$$\max_{\pi} J(\pi_\theta) \quad \leftarrow \text{Maximizing return}$$

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} J(\pi_\theta)|_{\theta_k} \quad \leftarrow \text{Gradient ascent}$$

Policy Gradient

Recall Goal of RL

$$J(\pi_\theta) = \int_{\tau} P(\tau|\theta)R(\tau)$$

← Expected long-term reward

Trajectory probability

$$P(\tau|\theta) = \rho_0(s_0)\prod_{t=0}^T P(s_{t+1}|s_t, a_t)\pi_\theta(a_t|s_t)$$

Trajectory reward

$$R(\tau) = \sum_{t=0}^T R(s_t, a_t)$$

$$\max_{\pi} J(\pi_\theta)$$

← Maximizing long-term reward

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} J(\pi_\theta)|_{\theta_k}$$

← How do we compute this gradient?

Policy Gradient

Deriving policy gradient: $\nabla_{\theta} J(\pi_{\theta})$

$$\nabla_{\theta} J(\pi_{\theta}) = \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)] \quad \leftarrow$$

Expectation over all trajectories sampled from current policy.

$$= \nabla_{\theta} \int_{\tau} P(\tau|\theta) R(\tau) \quad \leftarrow$$

Exchange gradient and integration.

$$= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(\tau) \quad \leftarrow$$

Gradient of trajectory probability.

$$= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau)$$

$$= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)]$$

$$\therefore \nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]$$

Policy Gradient

How do we compute gradient of traj probability?

$$\begin{aligned}\nabla_{\theta} J(\pi_{\theta}) &= \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)] \\ &= \nabla_{\theta} \int_{\tau} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau) \\ &= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)] \\ \therefore \nabla_{\theta} J(\pi_{\theta}) &= \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]\end{aligned}$$

Step 1. Log probability of trajectory

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^T \left(\log P(s_{t+1}|s_t, a_t) + \log \pi_{\theta}(a_t|s_t) \right).$$

Policy Gradient

How do we compute gradient of traj probability?

$$\begin{aligned}\nabla_{\theta} J(\pi_{\theta}) &= \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)] \\ &= \nabla_{\theta} \int_{\tau} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau) \\ &= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)] \\ \therefore \nabla_{\theta} J(\pi_{\theta}) &= \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]\end{aligned}$$

Step 1. Log probability of trajectory

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^T \left(\log P(s_{t+1}|s_t, a_t) + \log \pi_{\theta}(a_t|s_t) \right).$$

Step 2. Grad of log prob of traj

$$\begin{aligned}\nabla_{\theta} \log P(\tau|\theta) &= \nabla_{\theta} \log \rho_0(s_0) + \sum_{t=0}^T \left(\nabla_{\theta} \log P(s_{t+1}|s_t, a_t) + \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \right) \\ &= \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t).\end{aligned}$$

Transition probabilities does not affect gradient.

Policy Gradient

How do we compute gradient of traj probability?

$$\begin{aligned}\nabla_{\theta} J(\pi_{\theta}) &= \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)] \\ &= \nabla_{\theta} \int_{\tau} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau) \\ &= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)] \\ \therefore \nabla_{\theta} J(\pi_{\theta}) &= \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]\end{aligned}$$

Step 1. Log probability of trajectory

$$\log P(\tau|\theta) = \log \rho_0(s_0) + \sum_{t=0}^T \left(\log P(s_{t+1}|s_t, a_t) + \log \pi_{\theta}(a_t|s_t) \right).$$

Step 2. Grad of log prob of traj

$$\begin{aligned}\nabla_{\theta} \log P(\tau|\theta) &= \nabla_{\theta} \log \rho_0(s_0) + \sum_{t=0}^T \left(\nabla_{\theta} \log P(s_{t+1}|s_t, a_t) + \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \right) \\ &= \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t).\end{aligned}$$

Step 3. The Log-Derivative Trick

Using $\nabla_x \log f(x) = 1/f(x) \nabla_x f(x)$ we have:

$$\nabla_{\theta} P(\tau|\theta) = P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta).$$

Policy Gradient

How do we compute gradient of traj probability?

$$\nabla_{\theta} J(\pi_{\theta}) = \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)]$$

$$= \nabla_{\theta} \int_{\tau} P(\tau|\theta) R(\tau)$$

Plug in this gradient

$$= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(\tau) \leftarrow$$

$$\nabla_{\theta} P(\tau|\theta) = P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta).$$

$$= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau)$$

$$= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)]$$

$$\therefore \nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]$$

Policy Gradient

How do we compute gradient of traj log probability?

$$\nabla_{\theta} J(\pi_{\theta}) = \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)]$$

$$= \nabla_{\theta} \int_{\tau} P(\tau|\theta) R(\tau)$$

$$= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(\tau)$$

$$= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau)$$

$$= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)]$$

Recall gradient of log prob of traj is

$$\nabla_{\theta} \log P(\tau|\theta) = \cancel{\nabla_{\theta} \log p_0(s_0)} + \sum_{t=0}^T \left(\cancel{\nabla_{\theta} \log P(s_{t+1}|s_t, a_t)} + \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) \right)$$

$$= \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t).$$

$$\therefore \nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]$$

Policy Gradient

Let's think a bit what did we do here:

$$\begin{aligned}\nabla_{\theta} J(\pi_{\theta}) &= \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)] \\ &= \nabla_{\theta} \int_{\tau} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau) \\ &= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)] \\ \therefore \nabla_{\theta} J(\pi_{\theta}) &= \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]\end{aligned}$$

We change it from trajectory probability to action probability.

$$P(\tau|\theta) = \rho_0(s_0) \prod_{t=0}^T P(s_{t+1}|s_t, a_t) \pi_{\theta}(a_t|s_t)$$

Notice that this is hard to compute since we don't have access to the transition probability.

Policy Gradient

Let's think a bit what did we do here:

$$\begin{aligned}\nabla_{\theta} J(\pi_{\theta}) &= \nabla_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta}} [R(\tau)] \\ &= \nabla_{\theta} \int_{\tau} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} \nabla_{\theta} P(\tau|\theta) R(\tau) \\ &= \int_{\tau} P(\tau|\theta) \nabla_{\theta} \log P(\tau|\theta) R(\tau) \\ &= \mathbb{E}_{\tau \sim \pi_{\theta}} [\nabla_{\theta} \log P(\tau|\theta) R(\tau)]\end{aligned}$$

$$\therefore \nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau) \right]$$

We take expectation over trajectories sampled from current policy. This means we can estimate it using sample mean:

$$\hat{g} = \frac{1}{|\mathcal{D}|} \sum_{\tau \in \mathcal{D}} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t|s_t) R(\tau),$$

$$\mathcal{D} = \{\tau_i\}_{i=1, \dots, N}$$

\mathcal{D} is the set of trajectories sampled during training, from π_{θ} .

Policy Gradient

Intuition:

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} J(\pi_{\theta})|_{\theta_k}$$

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right]$$

- Maximum likelihood, weighted by the trajectory reward.
- Increasing the action probability, if this action leads to high trajectory reward.
- Decreasing the action probability, if this action leads to low trajectory reward.
- Reinforce the good behavior

Policy Gradient

Comparing to Supervised learning:

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} J(\pi_{\theta})|_{\theta_k}$$

Data is dynamic

PG:

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right]$$

Weighted by Trajectory reward.

SL:

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{a_t, s_t \sim (Y, X)} [\nabla_{\theta} \log \pi_{\theta}(a_t | s_t)]$$

Data is static or sampled from static distribution.

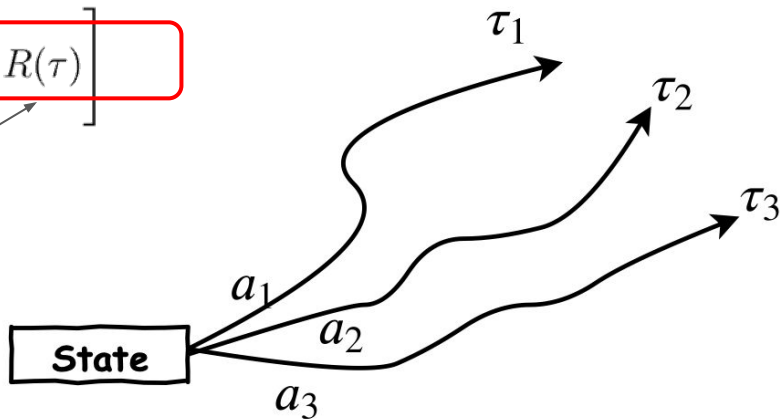
Policy Gradient

The problem of PG:

$$\theta_{k+1} = \theta_k + \alpha \nabla_{\theta} J(\pi_{\theta})|_{\theta_k}$$

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right]$$

- Gradient has high variance.



$$R(\tau_1) = 10 \quad R(\tau_2) = -1 \quad R(\tau_3) = 100$$

Policy gradients variants, and Actor Critic

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right]$$

- Different version of weights to replace $R(\tau)$ leads to different policy gradients methods:

Option 1. $R(\tau) = \sum_{t=0}^T R(s_t, a_t)$

Total reward of trajectory.

Option 2. $R_{t:T} = \sum_{t'=t}^T R(s_{t'}, a_{t'})$

Reward starting from time t.

Option 3. $R_{t:T} = \sum_{t'=t}^T R(s_{t'}, a_{t'}) - b(s_{t'})$

Baseline version of previous version to further reduce variance.

Policy gradients variants, and Actor Critic

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right]$$

- Different version of weights to replace $R(\tau)$ leads to different policy gradients methods:

Option 4. $Q^{\pi_{\theta}}(s_t, a_t)$ State-action value function

Option 5. $A^{\pi_{\theta}}(s_t, a_t) = Q^{\pi_{\theta}}(s_t, a_t) - V^{\pi_{\theta}}(s_t)$ Advantage function

Option 6. $R(s_t, a_t) + \gamma V^{\pi_{\theta}}(s_{t+1}) - V^{\pi_{\theta}}(s_t)$ Temporal Difference Residue

Policy gradients variants, and Actor Critic

$$\nabla_{\theta} J(\pi_{\theta}) = \mathbb{E}_{\tau \sim \pi_{\theta}} \left[\sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) R(\tau) \right]$$

- Different version of weights to replace $R(\tau)$ leads to different policy gradients methods:

Option 4. $Q^{\pi_{\theta}}(s_t, a_t)$

Option 5. $A^{\pi_{\theta}}(s_t, a_t) = Q^{\pi_{\theta}}(s_t, a_t) - V^{\pi_{\theta}}(s_t)$ ← **Actor critic algorithms**

Option 6. $R(s_t, a_t) + \gamma V^{\pi_{\theta}}(s_{t+1}) - V^{\pi_{\theta}}(s_t)$

Policy Gradient & Actor Critic Algorithm implementation

Algorithm 1 Vanilla Policy Gradient Algorithm

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go \hat{R}_t .
- 5: Compute advantage estimates, \hat{A}_t (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) |_{\theta_k} \hat{A}_t.$$

- 7: Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.

- 8: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

- 9: **end for**
-

Policy Gradient & Actor Critic Algorithm implementation

Algorithm 1 Vanilla Policy Gradient Algorithm

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go \hat{R}_t .
- 5: Compute advantage estimates, \hat{A}_t (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Estimate policy gradient as

Policy improvement

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) |_{\theta_k} \hat{A}_t.$$

- 7: Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.

- 8: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T (V_{\phi}(s_t) - \hat{R}_t)^2,$$

typically via some gradient descent algorithm.

- 9: **end for**
-

Option 1-6

Policy Gradient & Actor Critic Algorithm implementation

Algorithm 1 Vanilla Policy Gradient Algorithm

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go \hat{R}_t .
- 5: Compute advantage estimates, \hat{A}_t (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Estimate policy gradient as

Policy improvement

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) |_{\theta_k} \hat{A}_t.$$

Option 1-6

- 7: Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.

- 8: Fit value function by regression on mean-squared error:

Policy evaluation

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T (V_{\phi}(s_t) - \hat{R}_t)^2,$$

On policy value function, this requires using data sampled only from policy π_k

This means samples are used for training once then throw away.

typically via some gradient descent algorithm.

- 9: **end for**

Policy Gradient & Actor Critic Algorithm implementation

Algorithm 1 Vanilla Policy Gradient Algorithm

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go \hat{R}_t .
- 5: Compute advantage estimates, \hat{A}_t (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) |_{\theta_k} \hat{A}_t.$$

- 7: Compute policy update, either using standard gradient ascent,

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.

- 8: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T (V_{\phi}(s_t) - \hat{R}_t)^2,$$

typically via some gradient descent algorithm.

- 9: **end for**
-

In fact, there is a minor issue in this implementation. Could you figure out where it is?

Option 1-6

On policy value function, this requires using data sampled only from policy π_k

This means samples are used for training once then throw away.

Policy Gradient & Actor Critic Algorithm implementation

Algorithm 1 Vanilla Policy Gradient Algorithm

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go \hat{R}_t .
- 5: Compute advantage estimates, \hat{A}_t (using any method of advantage estimation) based on the current value function V_{ϕ_k} .
- 6: Estimate policy gradient as

$$\hat{g}_k = \frac{1}{|\mathcal{D}_k|} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) |_{\theta_k} \hat{A}_t.$$

We should do

1. policy evaluation

(step 4, 8, 5): Compute policy update, either using standard gradient ascent,

2. policy improvement.

(step 6, 7)

$$\theta_{k+1} = \theta_k + \alpha_k \hat{g}_k,$$

or via another gradient ascent algorithm like Adam.

- 8: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k| T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T (V_{\phi}(s_t) - \hat{R}_t)^2,$$

typically via some gradient descent algorithm.

- 9: **end for**
-

Option 1-6

On policy value function, this requires using data sampled only from policy π_k

This means samples are used for training once then throw away.

RL Algorithms (What we will cover?)

- Exact methods
 - **Value Iteration & Policy iteration**
- Deep RL algorithms [Model-free, Online approach]
 - **Policy gradient.**
 - **Actor acritic.**
 - Trust Region Policy Optimization.
 - **Proximal Policy Optimization**
 - DQN
 - DDPG
 - SAC

From Policy Gradient to PPO

Recall policy gradient

- Estimate advantage: $A^{\pi_\theta}(s_t, a_t) = Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)$
- Gradient update:

$$\nabla_\theta J(\pi_\theta) = \mathbf{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^T \nabla_\theta \log \pi_\theta(a_t | s_t) A^{\pi_\theta}(s_t, a_t) \right]$$

- All the samples are from current policy π_θ
- This means, once we do one step gradient ascent, we cannot use our previously collected data.
- What if we want to use old data more efficiently?

From Policy Gradient to PPO

Let's look at our RL formulation again

$$J(\pi_{\theta}) = \int_{\tau} P(\tau|\theta)R(\tau)$$

How do we leverage the data collected by previous policies to improve current policy?

- Can we establish relation between different policies?

$$J(\theta) - J(\theta_{\text{old}}) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

From Policy Gradient to PPO

$$J(\theta) - J(\theta_{\text{old}}) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

- Maximizing RL objectives is equal to maximizing
 - Advantage of old policy, under the expectation of new policy π_{θ}

$$E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] = \sum_t E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)} \left[\gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right]$$

From Policy Gradient to PPO

$$J(\theta) - J(\theta_{\text{old}}) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

- Maximizing RL objectives is equal to maximizing
 - Advantage of old policy, under the expectation of new policy π_{θ}

$$\begin{aligned} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] &= \sum_t E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)} \left[\gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \\ &= \sum_t E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \end{aligned}$$

Importance sampling

From Policy Gradient to PPO

$$J(\theta) - J(\theta_{\text{old}}) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

- Maximizing RL objectives is equal to maximizing
 - Advantage of old policy, under the expectation of new policy π_{θ}

$$\begin{aligned} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] &= \sum_t E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)} \left[\gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \\ &= \sum_t E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \end{aligned}$$

If the state also sampled from old policy, then we can optimize this by sampling data from old policies, directly optimize new policy π_{θ}

From Policy Gradient to PPO

$$J(\theta) - J(\theta_{\text{old}}) = E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right]$$

- Maximizing RL objectives is equal to maximizing
 - Advantage of old policy, under the expectation of new policy π_{θ}

$$\begin{aligned} E_{\tau \sim p_{\theta}(\tau)} \left[\sum_t \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] &= \sum_t E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)} \left[\gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \\ &= \sum_t E_{\mathbf{s}_t \sim p_{\theta}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \\ &\approx \sum_t E_{\mathbf{s}_t \sim p_{\theta_{\text{old}}}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \gamma^t A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}_t, \mathbf{a}_t) \right] \right] \end{aligned}$$

When can we approximate π_{θ} 's state distribution with state distribution from old policy?

From Policy Gradient to PPO

- Surrogate loss

$$\sum_t E_{\mathbf{s}_t \sim p_{\theta_{\text{old}}}(\mathbf{s}_t)} \left[E_{\mathbf{a}_t \sim \pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \left[\frac{\pi_{\theta}(\mathbf{a}_t | \mathbf{s}_t)}{\pi_{\theta_{\text{old}}}(\mathbf{a}_t | \mathbf{s}_t)} \gamma^t A^{\pi_{\theta_{\text{old}}}(\mathbf{s}_t, \mathbf{a}_t)} \right] \right]$$

- Subject to two policies are close to each other.

$$KL(\pi_{\theta_{\text{old}}} || \pi_{\theta}) \leq \epsilon$$

- How do we enforce two policies are close during the updates?

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- We want to optimize π_{θ} to maximize above value, but at the same time, not push too far away from old policy.

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- We want to optimize π_{θ} to maximize above value, but at the same time, not push too far away from old policy.
- Let's divide this into different cases based on Advantage

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- We want to optimize π_{θ} to maximize above value, but at the same time, not push too far away from old policy.
- Let's divide this into different cases based on Advantage
- If $A > 0$, we want increase the probability of π_{θ} for this state action pair, but not too large

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- If $A > 0$, we want increase the probability of π_{θ} for this state action pair, but not too large.
- Let's denote ratio of two policies as $r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$
- If r is too large
- we bound the maximal update by $1 + \epsilon$
- O.w. we improve it based on r
- Let's denote this new loss by L^{CLIP}

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- If $A > 0$, we want increase the probability of π_{θ} for this state action pair, but not too large.
- Let's denote ratio of two policies as $r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$
- More concretely

$$L^{CLIP} = A \left\{ \begin{array}{ll} 1 + \epsilon & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ r & r \leq 1 - \epsilon \end{array} \right\}$$

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{old}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{old}}}(\mathbf{s}, \mathbf{a})$$

- If $A > 0$, we want increase the probability of π_{θ} for this state action pair, but not too large.
- Let's denote ratio of two policies as $r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{old}}(a | s)}$
- More concretely

$$L^{CLIP} = A \left\{ \begin{array}{ll} 1 + \epsilon & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ r & r \leq 1 - \epsilon \end{array} \right\}$$

We want to increase π_{θ} , right now it has low probability on (s, a) . $\pi_{\theta}(a | s) < (1 + \epsilon)\pi_{\theta_{old}}(a | s)$

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- If $A > 0$, we want increase the probability of π_{θ} for this state action pair, but not too large.
- Let's denote ratio of two policies as $r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$
- More concretely

$$L^{CLIP} = A \left\{ \begin{array}{ll} 1 + \epsilon & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ r & r \leq 1 - \epsilon \end{array} \right\}$$

We want to increase π_{θ} , right now it has low probability on (s, a) .

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- If $A > 0$, we want increase the probability of π_{θ} for this state action pair, but not too large.
- Let's denote ratio of two policies as $r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$
- More concretely

$$L^{CLIP} = A \left\{ \begin{array}{ll} 1 + \epsilon & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ r & r \leq 1 - \epsilon \end{array} \right\}$$

We can increase π_{θ} by r^*A

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- If $A > 0$, we want increase the probability of π_{θ} for this state action pair, but not too large.
- Let's denote ratio of two policies as $r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$
- More concretely

$$L^{CLIP} = A \left\{ \begin{array}{ll} 1 + \epsilon & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ r & r \leq 1 - \epsilon \end{array} \right\}$$

We want to increase π_{θ} , it already has high probability on (s, a) . No additional benefit to make it larger.

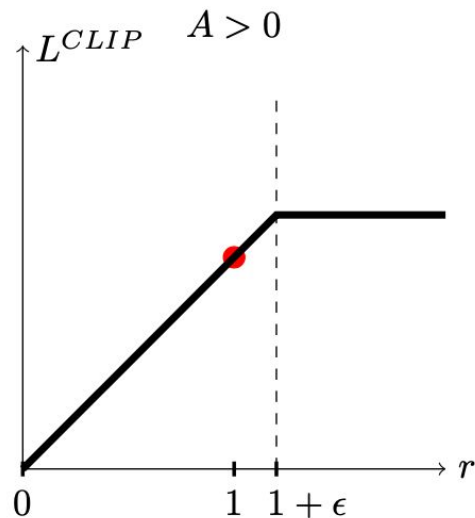
From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- If $A > 0$, we want increase the probability of π_{θ} for this state action pair, but not too large.
- Let's denote ratio of two policies as $r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$
- More concretely

$$L^{CLIP} = A \left\{ \begin{array}{ll} 1 + \epsilon & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ r & r \leq 1 - \epsilon \end{array} \right\}$$



From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})}$$

- If $A < 0$, we want decrease the probability of π_{θ} for this state action pair, but not to further decrease, when it's already small.

-

$$L^{CLIP} = A \left\{ \begin{array}{ll} r & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ 1 - \epsilon & r \leq 1 - \epsilon \end{array} \right\}$$

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}}(\mathbf{s}, \mathbf{a}) \quad r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$$

- If $A < 0$, we want decrease the probability of π_{θ} for this state action pair, but not too large, when it's already small.

$$L^{CLIP} = A \left\{ \begin{array}{ll} r & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ 1 - \epsilon & r \leq 1 - \epsilon \end{array} \right\}$$

$$\pi_{\theta}(a | s) \geq (1 - \epsilon)\pi_{\theta_{\text{old}}}(a | s)$$

In this case, π_{θ} has higher probability on (s, a) , than $\pi_{\theta_{\text{old}}}$.
We can then take big update to make it smaller.

From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{old}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{old}}}(\mathbf{s}, \mathbf{a}) \quad r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{old}}(a | s)}$$

- If $A < 0$, we want decrease the probability of π_{θ} for this state action pair, but not too large, when it's already small.

$$L^{CLIP} = A \left\{ \begin{array}{ll} r & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ 1 - \epsilon & r \leq 1 - \epsilon \end{array} \right\}$$

$$\pi_{\theta}(a | s) \leq (1 - \epsilon) \pi_{\theta_{old}}(a | s)$$

In this case, π_{θ} already has a smaller probability than $\pi_{\theta_{old}}$
No additional benefit to update π_{θ}

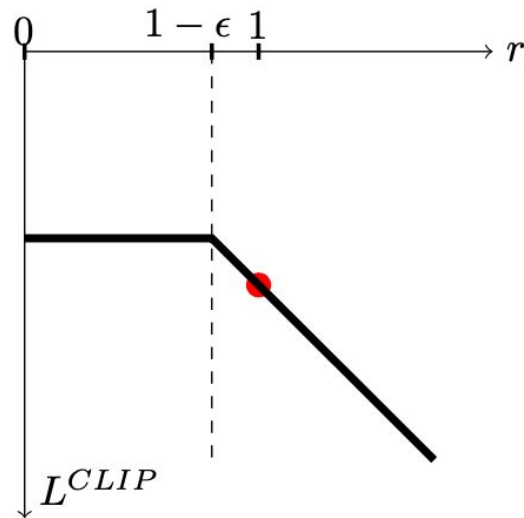
From Policy Gradient to PPO

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})} \quad r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$$

- If $A < 0$, we want decrease the probability of π_{θ} for this state action pair, but not too large, when it's already small.

$$L^{CLIP} = A \left\{ \begin{array}{ll} r & r \geq 1 + \epsilon \\ r & r \in (1 - \epsilon, 1 + \epsilon) \\ 1 - \epsilon & r \leq 1 - \epsilon \end{array} \right\}$$



Proximal Policy Optimization

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})} \quad r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$$

- Let's combine the two cases: $L^{CLIP} = \min(rA, g(\epsilon, A))$

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0. \end{cases}$$

- Or equivalently $L^{CLIP} = \min(rA, \text{clip}(r, 1 - \epsilon, 1 + \epsilon)A)$

Proximal Policy Optimization

- Let's look at one state action pair first,

$$\frac{\pi_{\theta}(\mathbf{a} | \mathbf{s})}{\pi_{\theta_{\text{old}}}(\mathbf{a} | \mathbf{s})} A^{\pi_{\theta_{\text{old}}}(\mathbf{s}, \mathbf{a})} \quad r = \frac{\pi_{\theta}(a | s)}{\pi_{\theta_{\text{old}}}(a | s)}$$

- Let's combine the two cases: $L^{CLIP} = \min(rA, g(\epsilon, A))$

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0. \end{cases}$$

- Recap intuition:

if a given state-action pair has negative advantage A , the optimization wants to make $\pi_{\theta}(a | s)$ smaller, but no additional benefit to the objective function is gained by making $\pi_{\theta_{\text{old}}}(a | s)$ smaller than $(1 - \epsilon)\pi_{\theta_{\text{old}}}(a | s)$

if a given state-action pair has positive advantage A , the optimization wants to make $\pi_{\theta}(a | s)$ larger, but no additional benefit to the objective function is gained by making $\pi_{\theta}(a | s)$ larger than $(1 + \epsilon)\pi_{\theta_{\text{old}}}(a | s)$

Proximal Policy Optimization

Algorithm 1 PPO-Clip

- 1: Input: initial policy parameters θ_0 , initial value function parameters ϕ_0
- 2: **for** $k = 0, 1, 2, \dots$ **do**
- 3: Collect set of trajectories $\mathcal{D}_k = \{\tau_i\}$ by running policy $\pi_k = \pi(\theta_k)$ in the environment.
- 4: Compute rewards-to-go \hat{R}_t .
- 5: Compute advantage estimates, \hat{A}_t (~~using any method of advantage estimation~~) based on the current value function V_{ϕ_k} . In PPO, we use GAE [1]
- 6: Update the policy by maximizing the PPO-Clip objective:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left(\frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right),$$

typically via stochastic gradient ascent with Adam.

- 7: Fit value function by regression on mean-squared error:

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left(V_{\phi}(s_t) - \hat{R}_t \right)^2,$$

typically via some gradient descent algorithm.

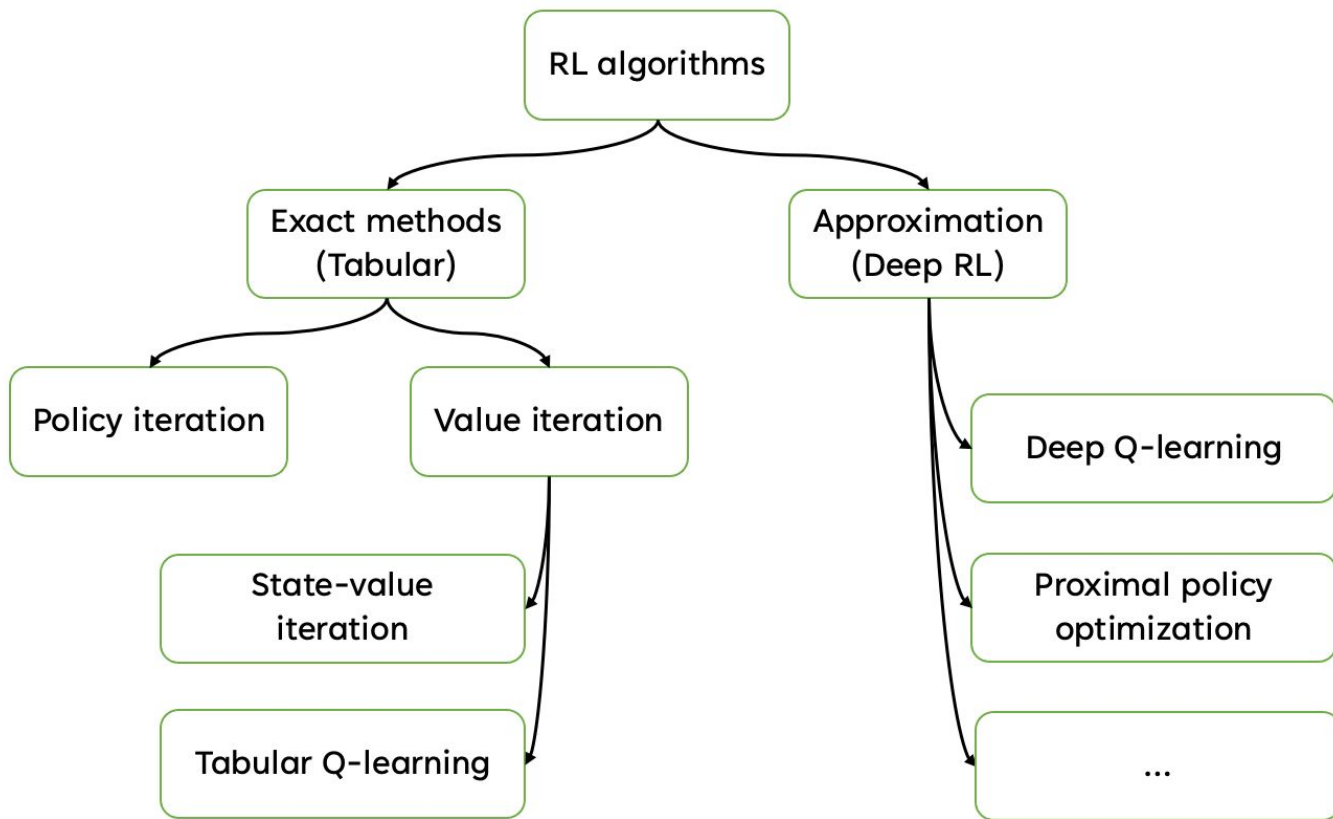
- 8: **end for**

[1] Schulman, John, et al. "High-dimensional continuous control using generalized advantage estimation." *arXiv preprint arXiv:1506.02438* (2015).

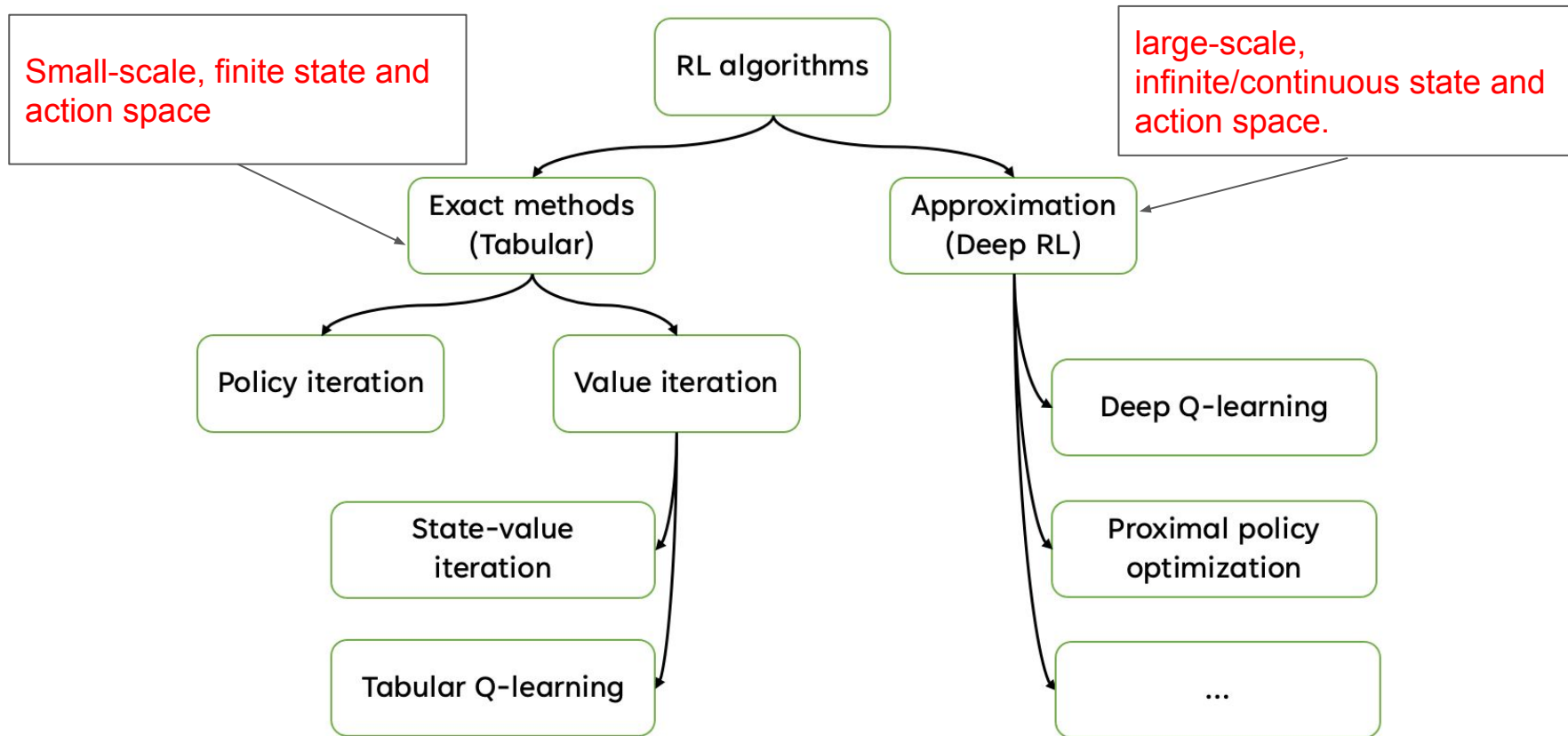
Part II: How to solve RL?

RL Algorithms Landscape

RL Algorithm Landscape (from the scale of problem)



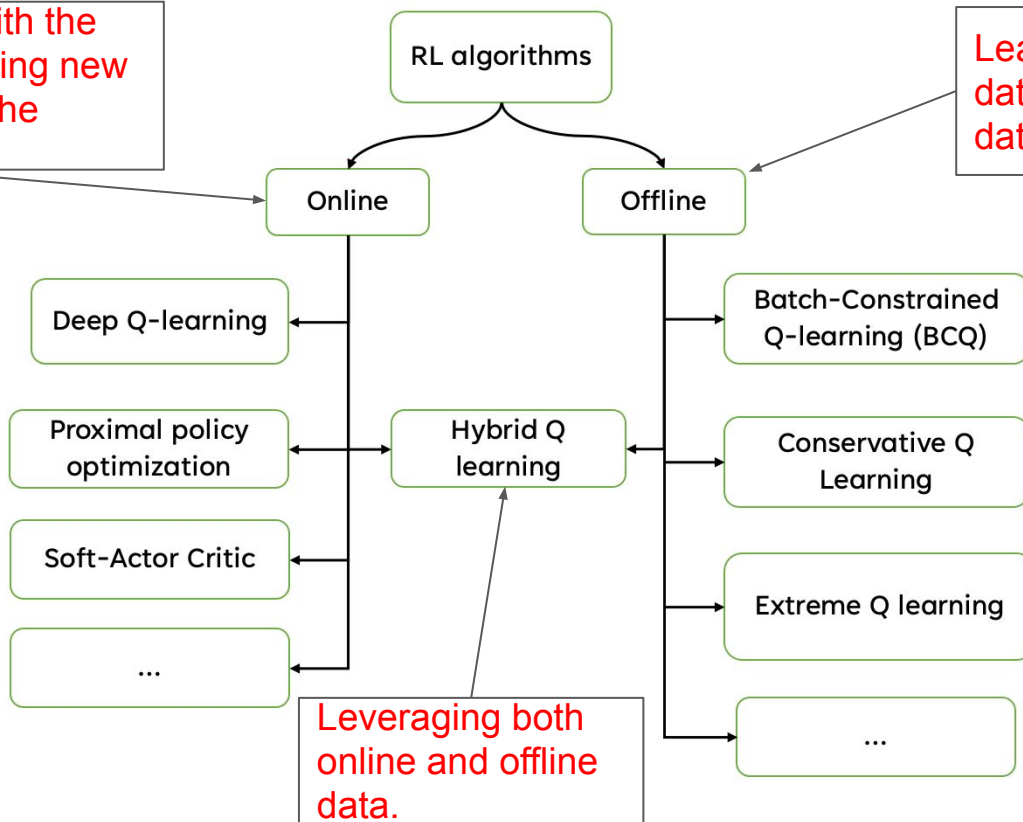
RL Algorithm Landscape (from the scale of problem)



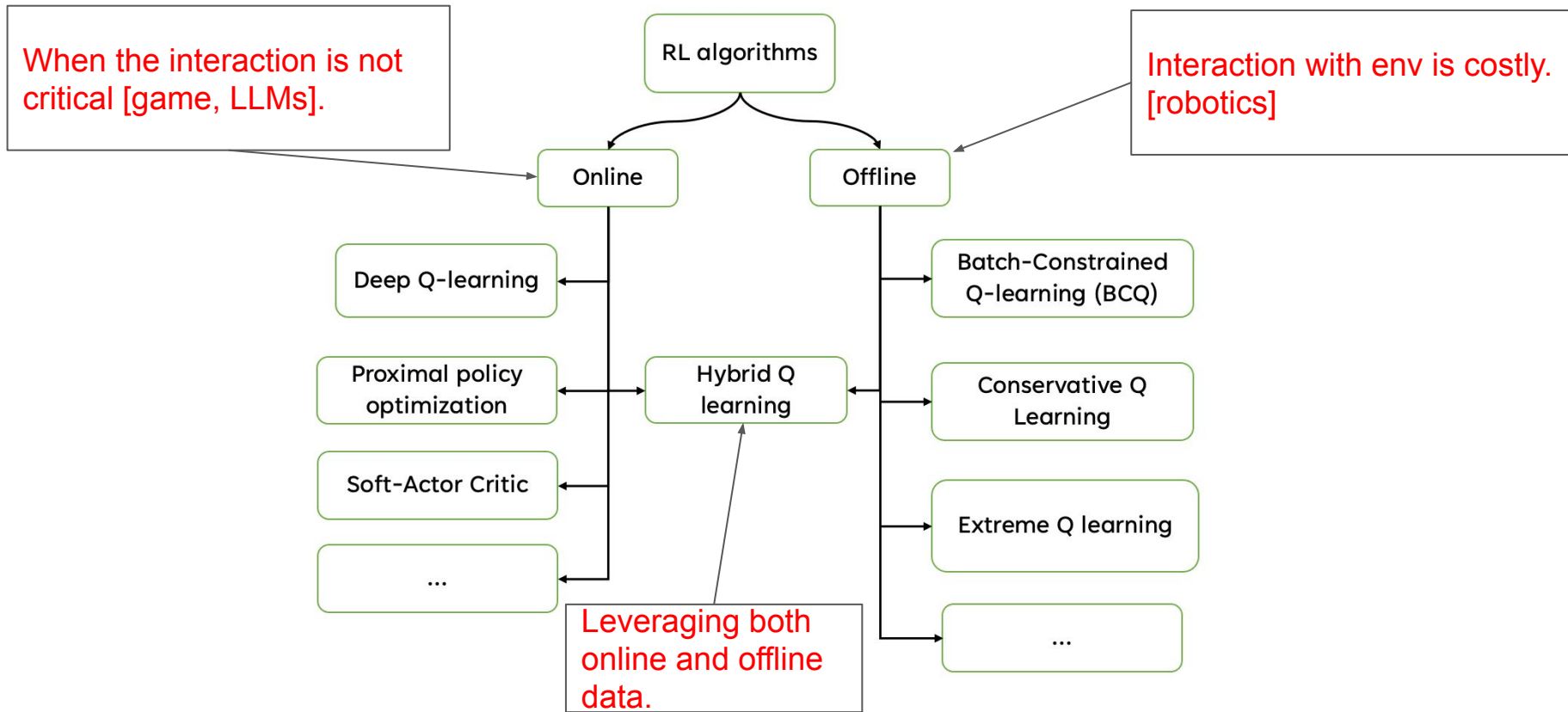
RL Algorithm Landscape (whether interacting with env)

Online interaction with the environment, collecting new data sampled from the training policy.

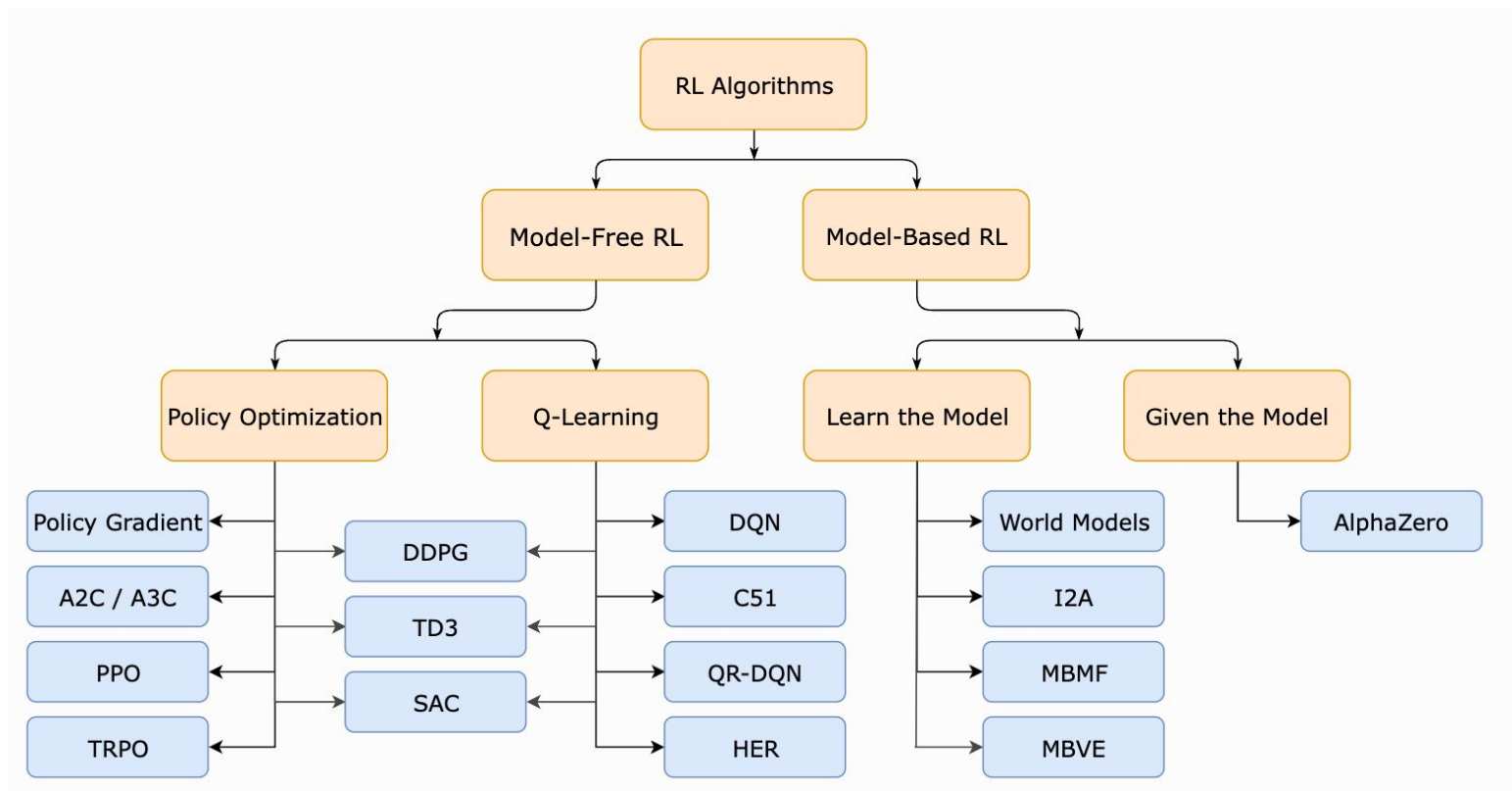
Learning on static offline dataset. Don't collect new data.



RL Algorithm Landscape (whether interacting with env)



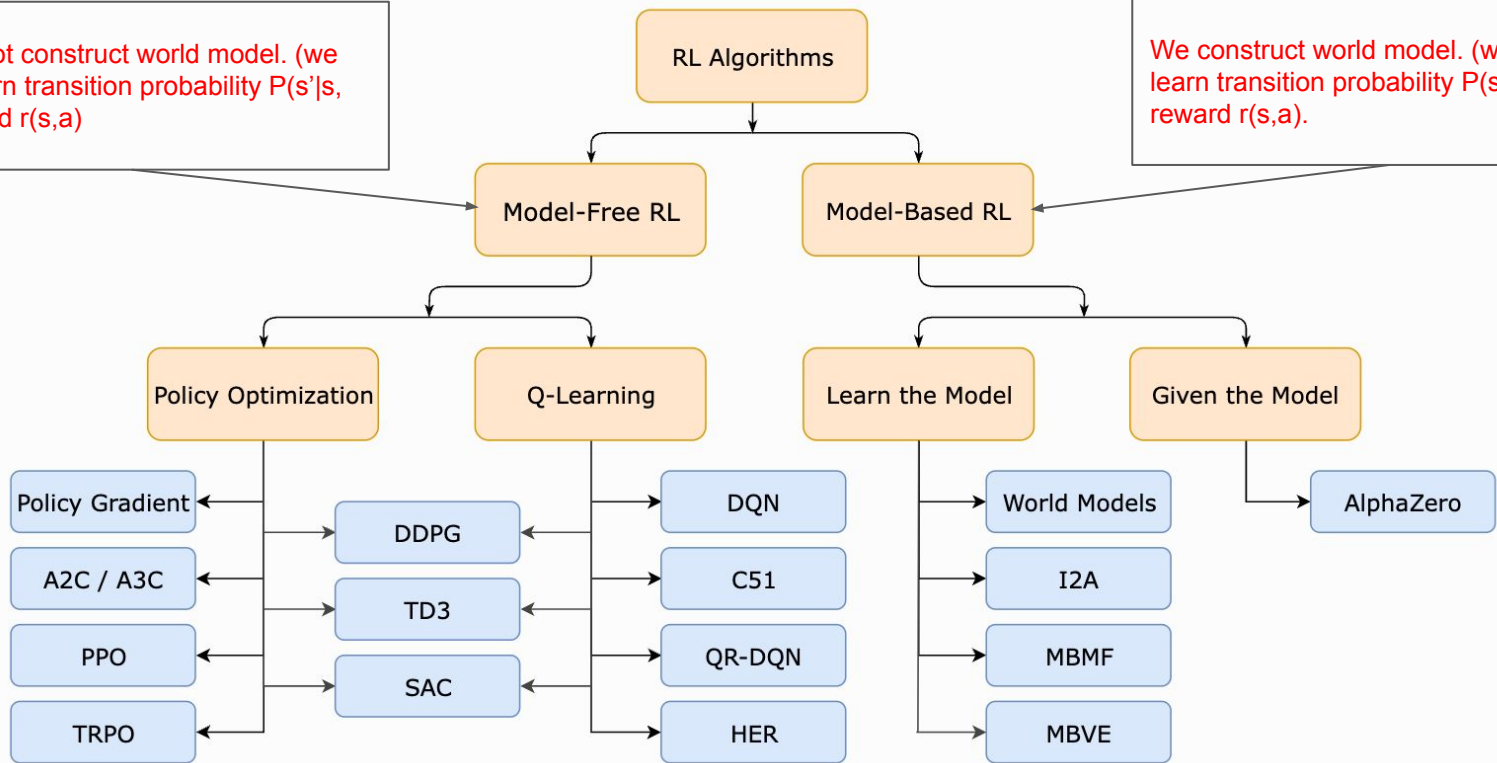
RL Algorithm Landscape (Do we have world model?)



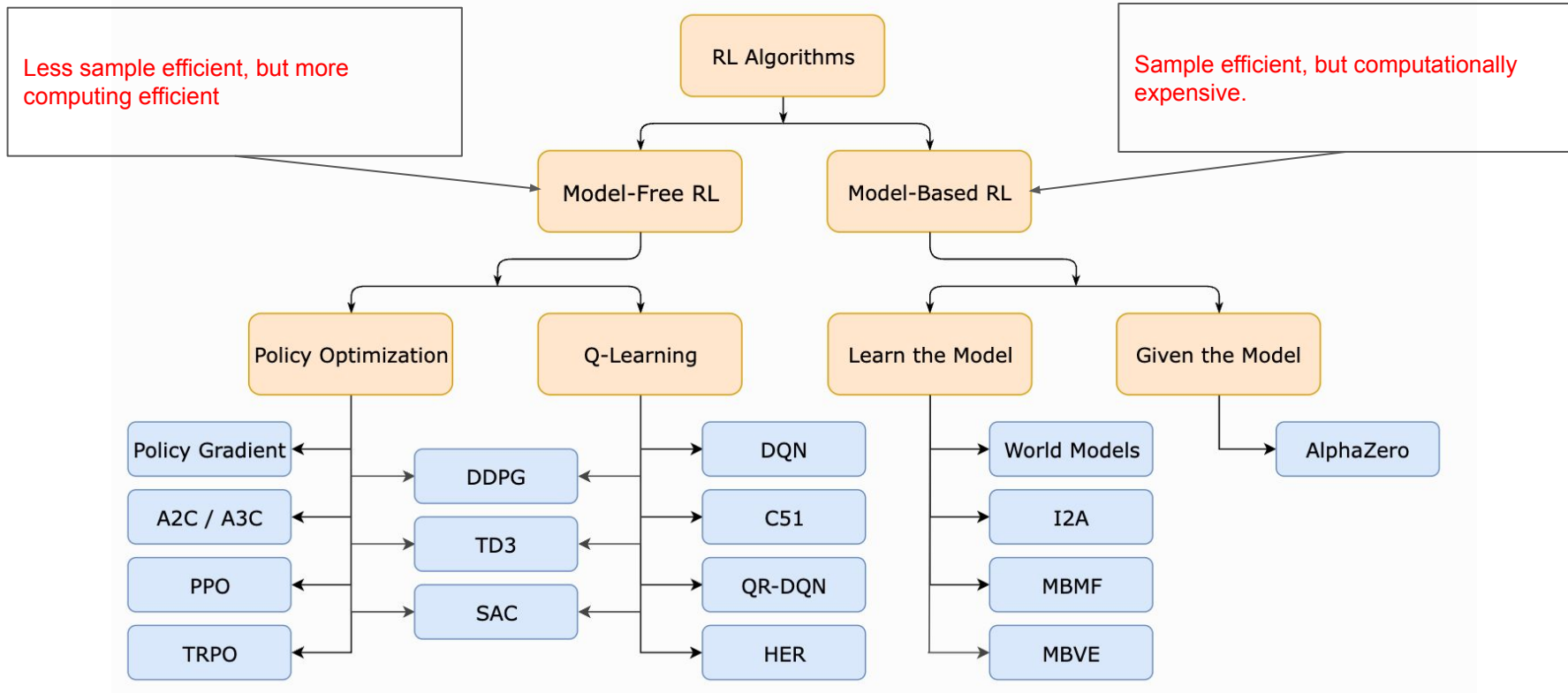
RL Algorithm Landscape (Do we have world model?)

We do not construct world model. (we don't learn transition probability $P(s'|s, a)$, reward $r(s,a)$)

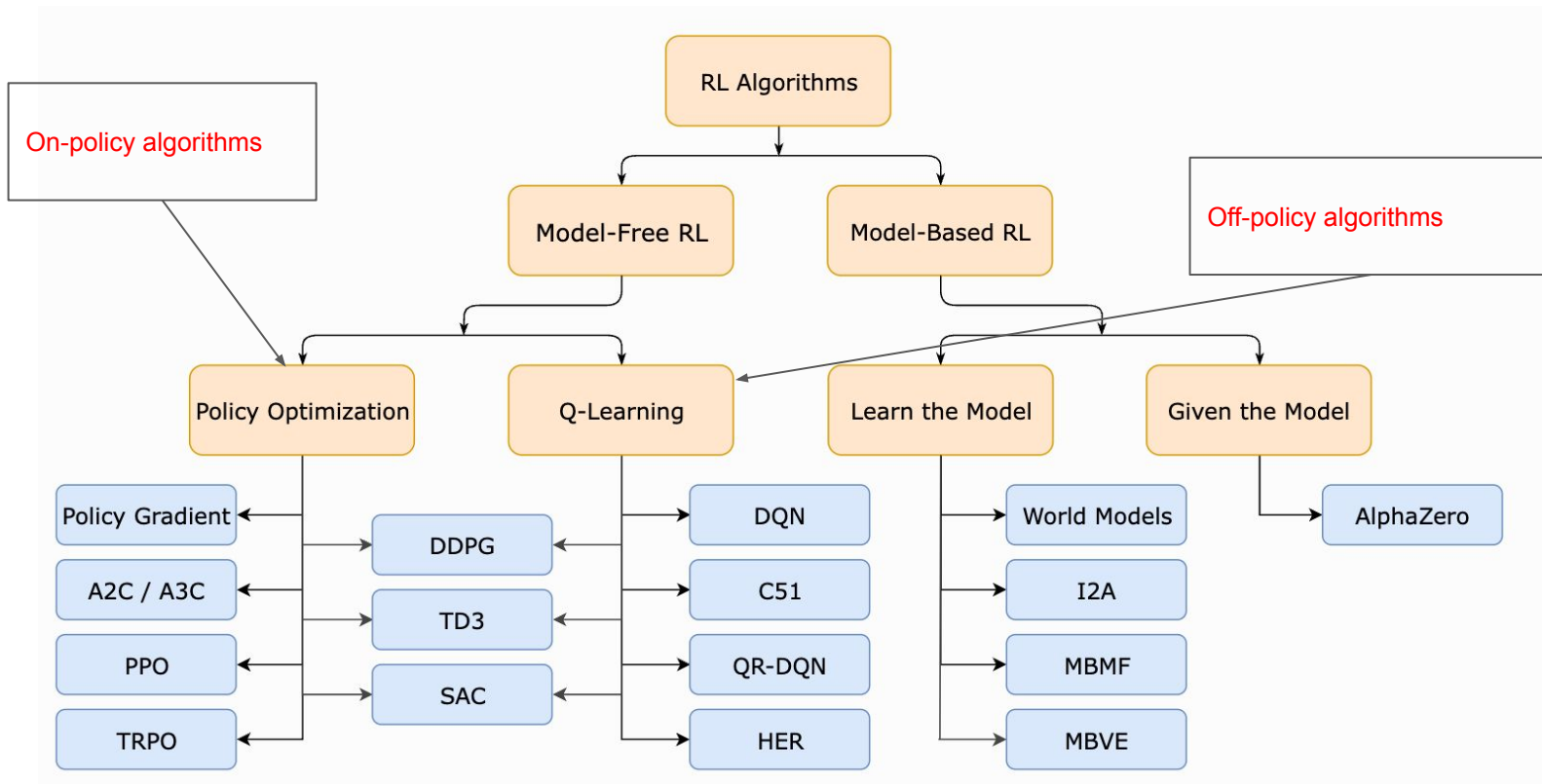
We construct world model. (we model learn transition probability $P(s'|s, a)$, reward $r(s,a)$.)



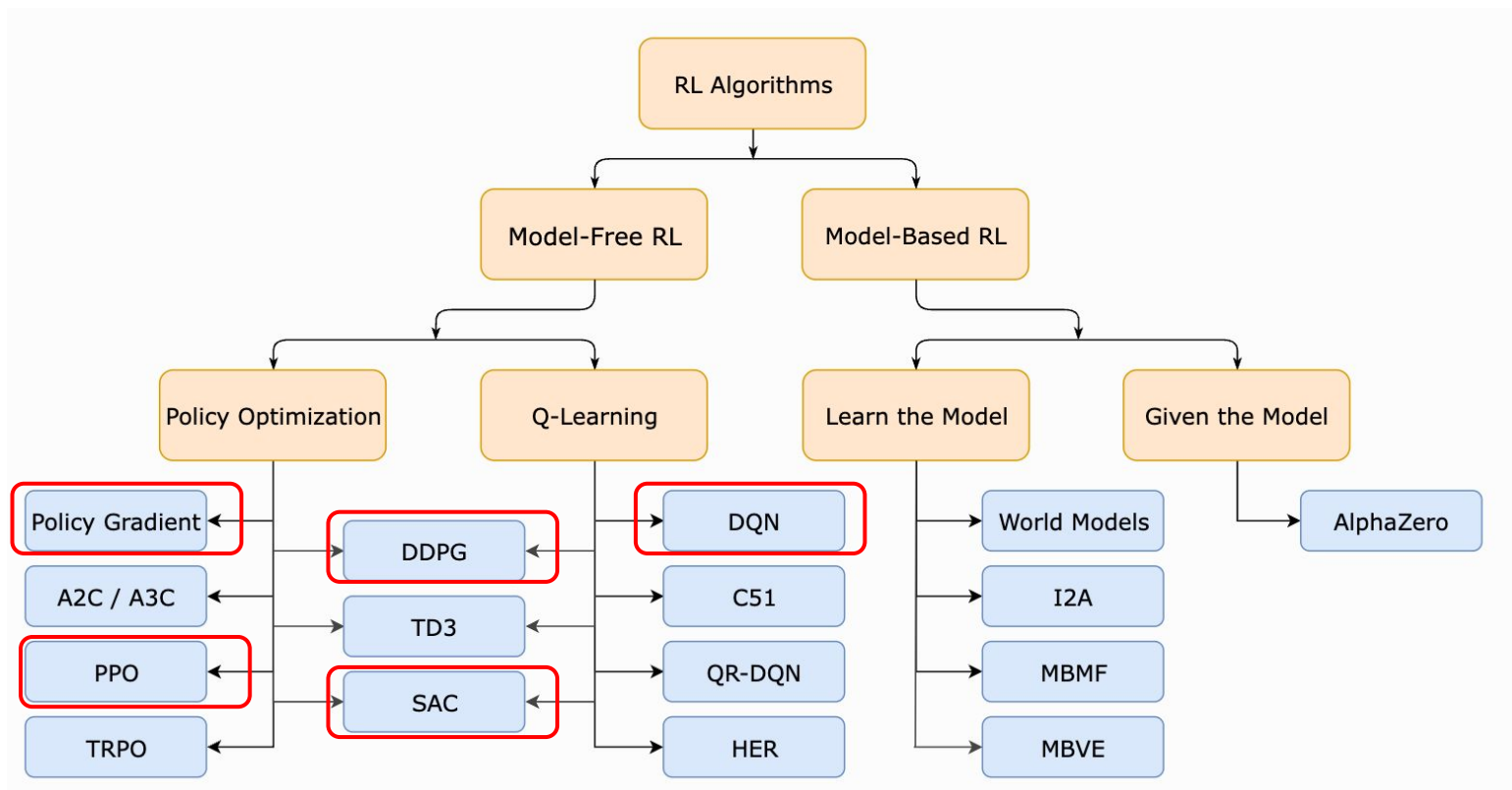
RL Algorithm Landscape (Do we have world model?)



RL Algorithm Landscape (Do we have world model?)

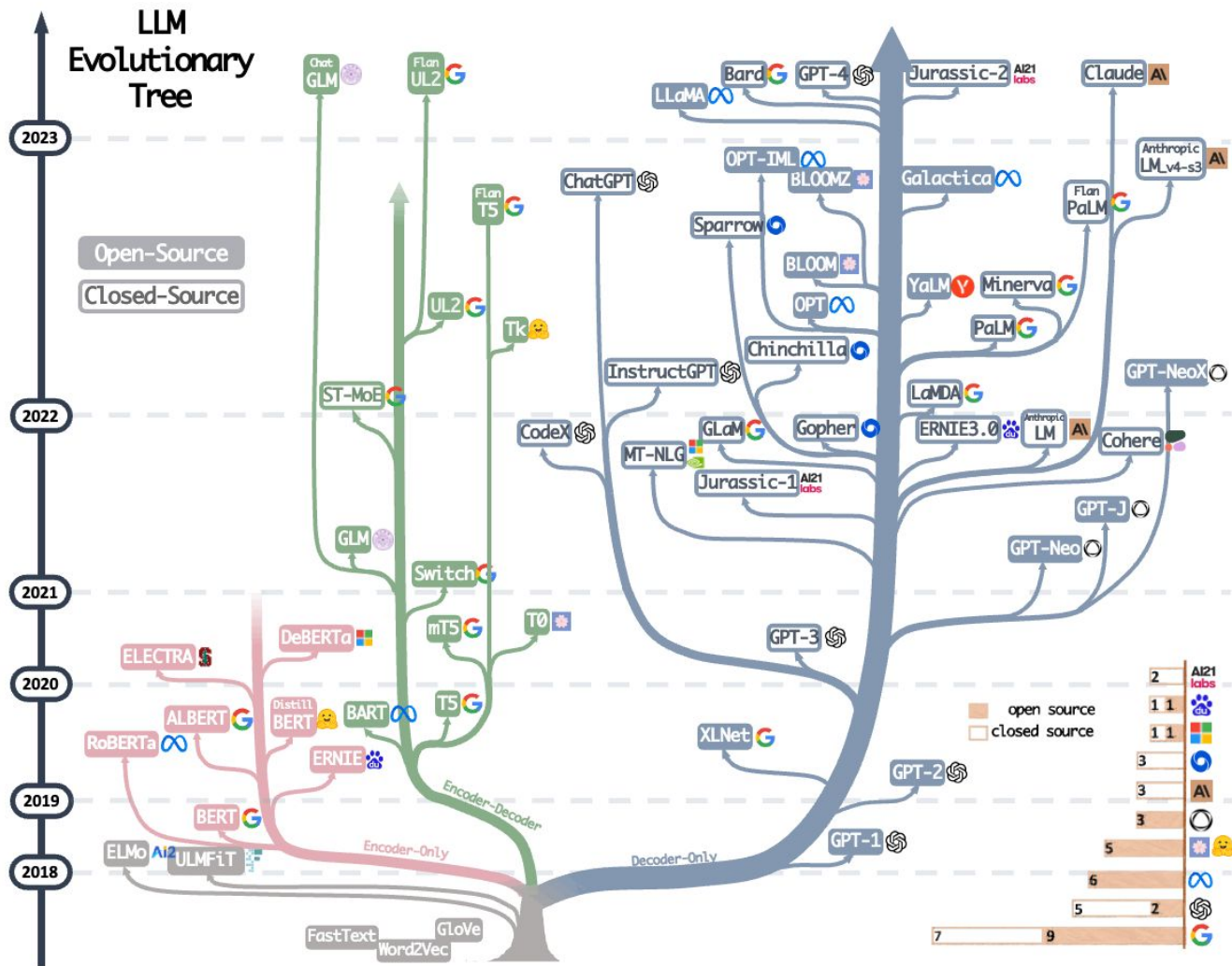


RL Algorithms (Key algorithms)

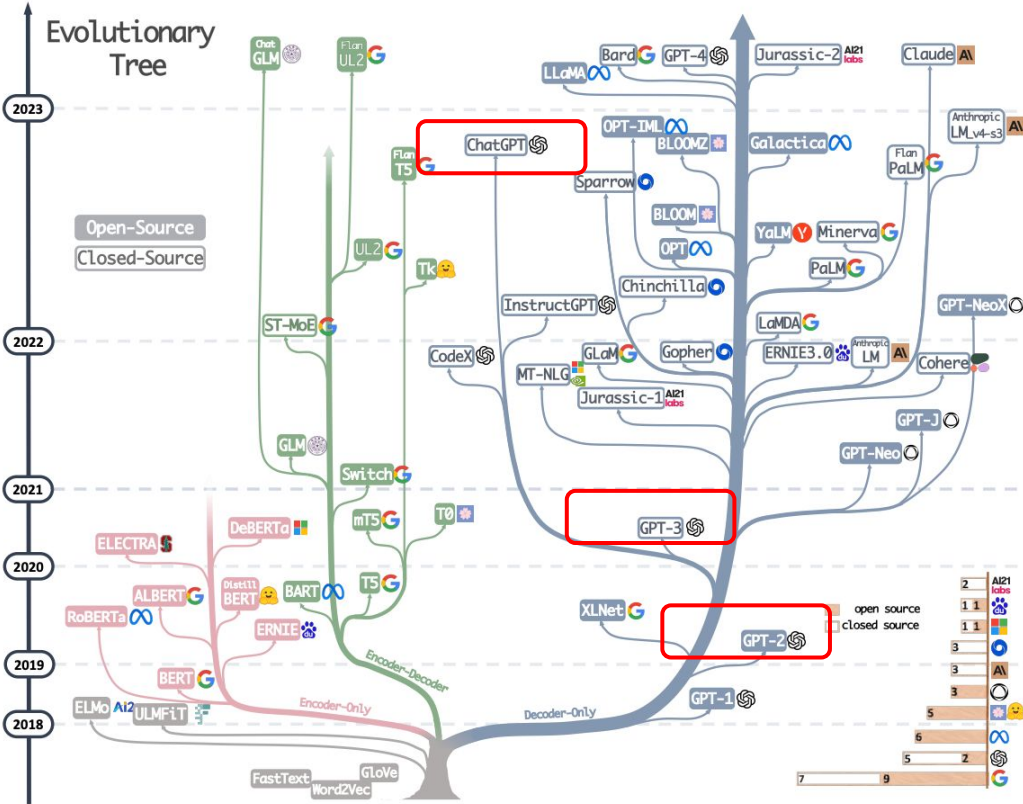


Part II: Application

RLHF in ChatGPT



From GPT-2, GPT-3 to ChatGPT



LLM Training Stages

Pre-training

Develops a broad set of skills, knowledge, pattern recognition, reasoning abilities. [1]

Post-training

Tuning LLM to better interacting with human [2]

[1] [GPT3 paper](#) Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

[2] Zhou, Chunting, et al. "Lima: Less is more for alignment." *Advances in Neural Information Processing Systems* 36 (2024).

LLM Training Stages

Pre-training

Develops a broad set of skills, knowledge, pattern recognition, reasoning abilities.

Post-training

Tuning LLM to better interacting with human

Supervised
fine-tuning

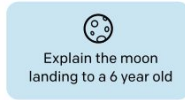
RLHF

RLHF Workflow

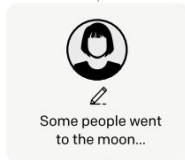
Step 1

Collect demonstration data, and train a supervised policy.

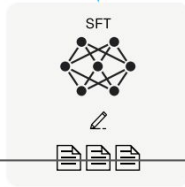
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.

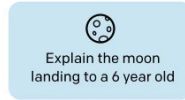


Supervised fine-tuning

Step 2

Collect comparison data, and train a reward model.

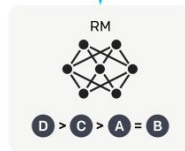
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Reinforcement Learning from Human Feedback

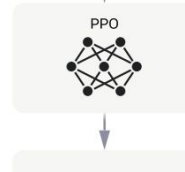
Step 3

Optimize a policy against the reward model using reinforcement learning.

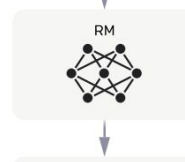
A new prompt is sampled from the dataset.



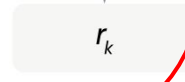
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

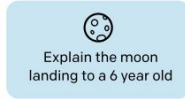


RLHF Workflow

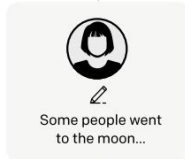
Step 1

Collect demonstration data, and train a supervised policy.

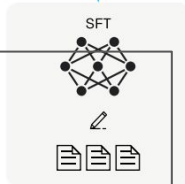
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



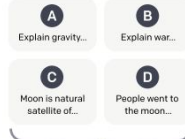
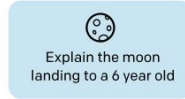
Providing a good initialization point

Reinforcement Learning from Human Feedback

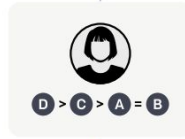
Step 2

Collect comparison data, and train a reward model.

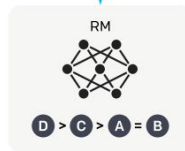
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



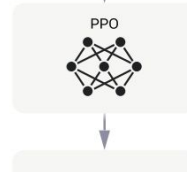
Step 3

Optimize a policy against the reward model using reinforcement learning.

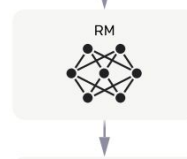
A new prompt is sampled from the dataset.



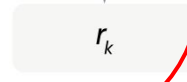
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



RLHF Workflow

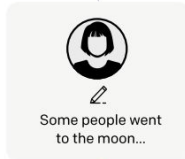
Step 1

Collect demonstration data, and train a supervised policy.

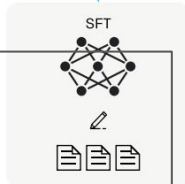
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.



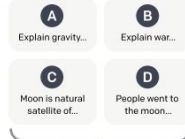
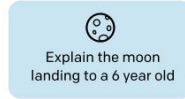
Providing a good initialization point

Reinforcement Learning from Human Feedback

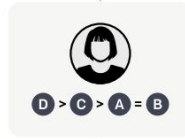
Step 2

Collect comparison data, and train a reward model.

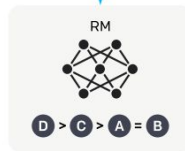
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



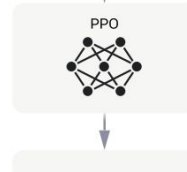
Step 3

Optimize a policy against the reward model using reinforcement learning.

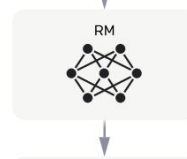
A new prompt is sampled from the dataset.



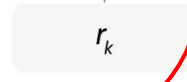
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



Why SFT alone is not sufficient?

- From the perspective of cost

Collecting SFT demonstration data is expensive. We need very high quality data samples.

- From the goal of alignment

Human preference is implicit and complicated. Training on SFT data is not maximizing human preference.

Reward model as a Human preference proxy

- Each prompts, we have K response to rank. It produces $\binom{K}{2}$ comparisons.

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} [\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))]$$

Prompt x

Rejected response y_l

Preferred response y_w

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

InstructGPT

People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

[1] [GPT3 paper](#). Brown, Tom, et al. "Language models are few-shot learners." *Advances in neural information processing systems* 33 (2020): 1877-1901.

Reward model as a Human preference proxy

- Each prompts, we have K response to rank. It produces $\binom{K}{2}$ comparisons.

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} [\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))]$$

Logistic function:

$$\sigma(r(y_w) - r(y_l)) = \frac{1}{1 + e^{r(y_l) - r(y_w)}}$$

Optimize reward so that preferred response has higher score than less preferred response.

PPO for LLM

Objective function

PPO clipped loss

KL regularization

$$\text{objective}(\phi) = E_{(x,y) \sim D_{\pi_{\phi}^{\text{RL}}}} \left[r_{\theta}(x, y) - \beta \log \left(\pi_{\phi}^{\text{RL}}(y | x) / \pi^{\text{SFT}}(y | x) \right) \right] + \\ \gamma E_{x \sim D_{\text{pretrain}}} \left[\log(\pi_{\phi}^{\text{RL}}(x)) \right]$$

Pretraining loss to alleviate
the alignment tax

PPO for Large Language Model

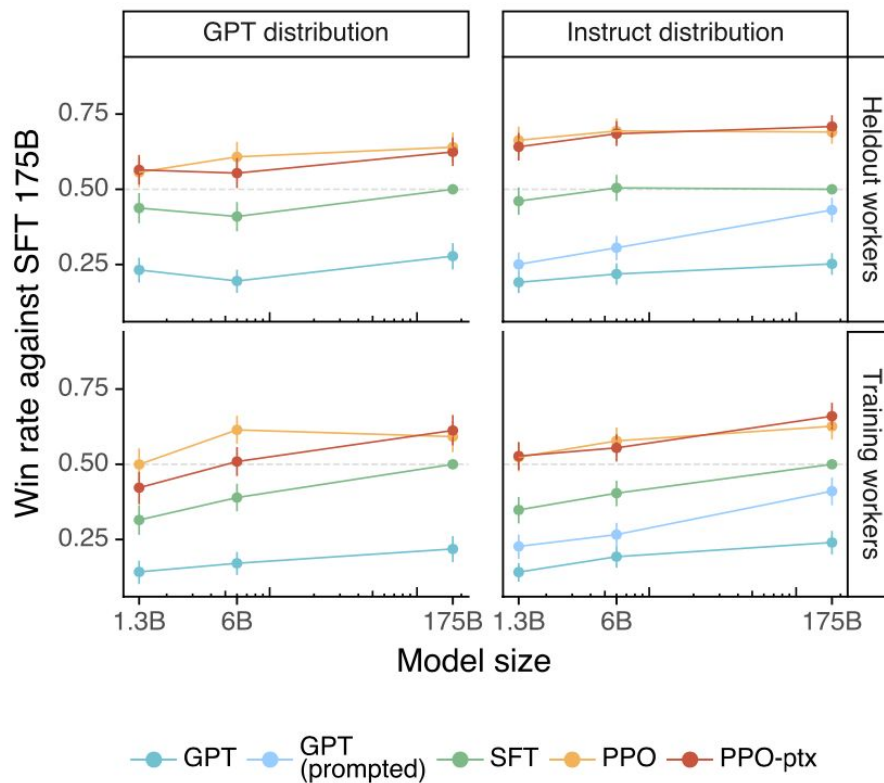
Objective function

$$\text{objective}(\phi) = E_{(x,y) \sim D_{\pi_{\phi}^{\text{RL}}}} \left[r_{\theta}(x,y) - \beta \log \left(\frac{\pi_{\phi}^{\text{RL}}(y | x)}{\pi^{\text{SFT}}(y | x)} \right) \right] + \\ \gamma E_{x \sim D_{\text{pretrain}}} \left[\log(\pi_{\phi}^{\text{RL}}(x)) \right]$$

What is state, action, defined here?

What is the environment?

Win rate Pre-trained vs SFT vs RL



Limitation & Challenges

- Collecting preference data is expensive.
 - This step is hard to avoid as this is supervision of human preference. But,
 - How do we best select our data for collecting human feedback.
 - To what extent we can rely on AI feedback?
- RL optimization is hard.
 - Reward score not increasing, Performance collapse, Over optimization of reward.
- Reward hacking.
 - It can prefer long response instead of actually high quality response.
- Multi-objectives
 - Multiple preferences might conflicting with each other.
- Iterative process
 - When do we start the next iteration.

References

Books

- Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare and Joelle Pineau (2018), “An Introduction to Deep Reinforcement Learning”, Foundations and Trends in Machine Learning: Vol. 11, No. 3-4. DOI: 10.1561/22000000071.
- Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- Bertsekas, Dimitri, and John N. Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

Online Courses

- Sergey Levine, UCB DRL course <https://rail.eecs.berkeley.edu/deeprlcourse/>
- Pieter Abbeel, Foundations of Deep RL 6 lectures series, [Youtube Videos and Slides](#)
- David Silver RL <https://www.davidsilver.uk/teaching/>

Code Repo & Tutorial

- [Google Dopamine](#)
- [OpenAI SpinningUp](#)

Q&A

- Feel free to email kaixianglin.cs@gmail.com for any questions regarding this slides