



北京航空航天大学  
COLLEGE OF SOFTWARE 软件学院  
BEIHANG UNIVERSITY

# 人工智能

## 第13讲：机器学习-强化学习 I

张晶

2023年春季

- 参考教材： 吴飞，《人工智能导论：模型与算法》，高等教育出版社
- 在线课程：<https://www.icourse163.org/course/ZJU-1003377027?from=searchPage>
- 本部分参考：李宏毅，《机器学习》课程，台湾大学



北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 提纲

## 一、强化学习问题定义

## 二、基于策略的强化学习

## 三、基于价值的强化学习

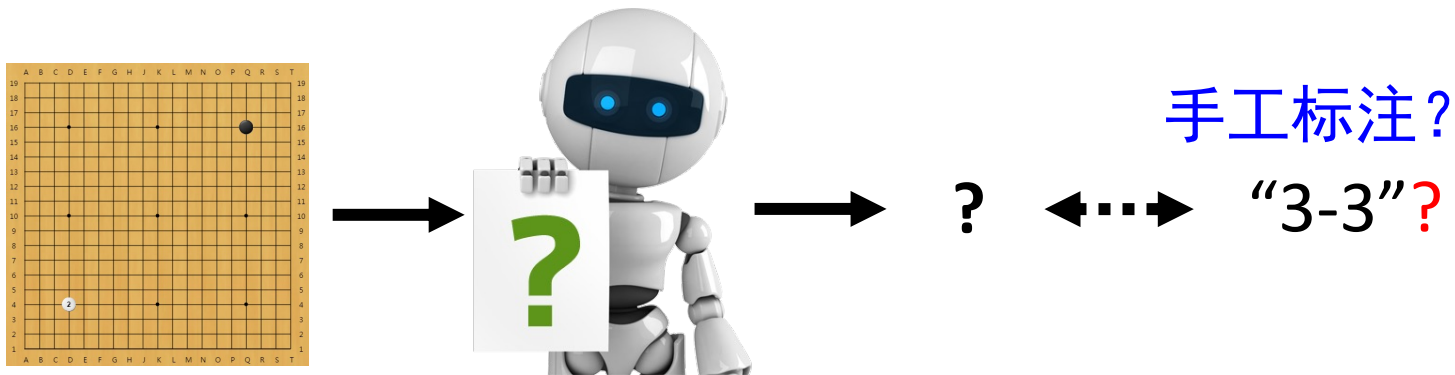
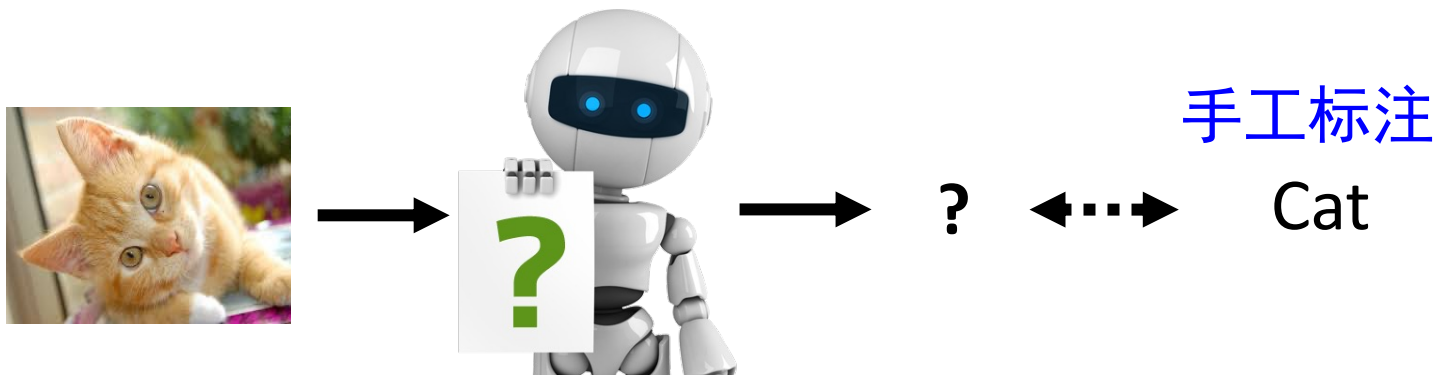
## 四、 Actor-Critic方法

## 五、其他强化学习方法



北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 监督学习→强化学习



很多任务中标注数据非常困难

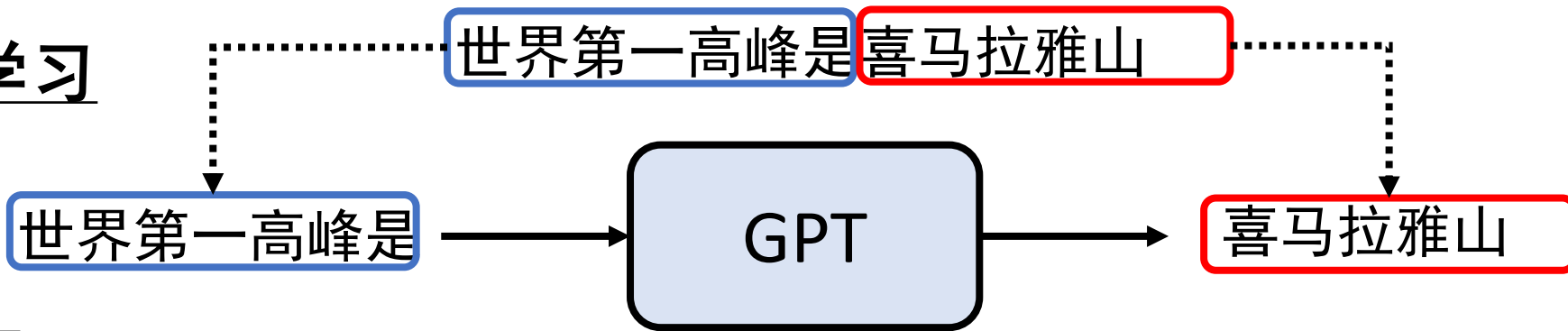
…… 但是可以告诉机器某个结果是好是坏



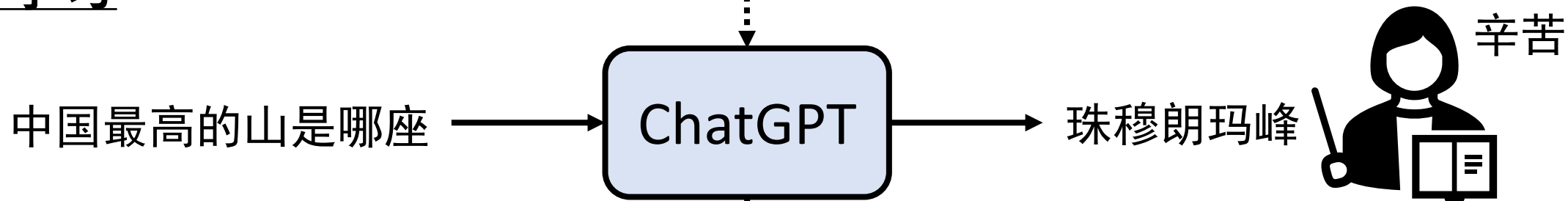
北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# ChatGPT: 自监督预训练 + 强化学习微调

## 自监督学习



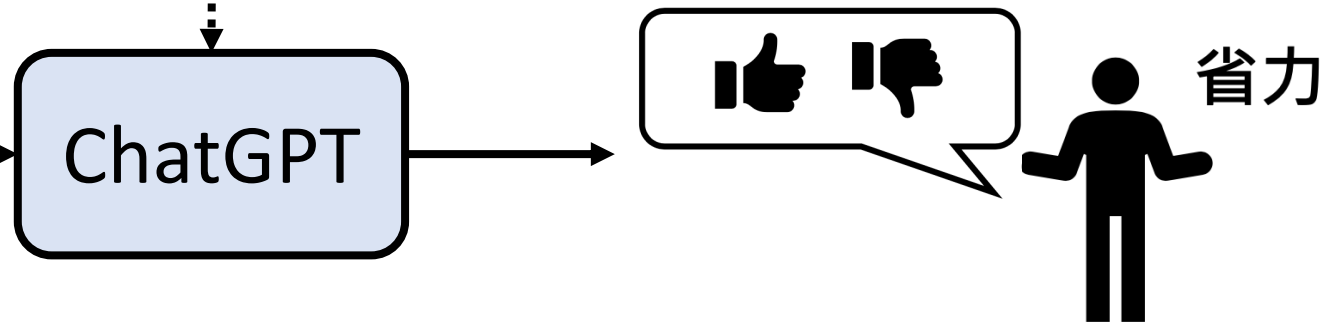
## 监督学习



## 强化学习

Reinforcement Learning, RL

中国最高的山是珠穆朗玛峰  
请帮我写诗赞美AI  
...





北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 监督学习→强化学习

- 监督学习：根据监督学习



Next move:  
"5-5"



Next move:  
"3-3"

- 强化学习：根据经验学习

First move ➡ ..... many moves ..... ➡ Win!

(Two agents play with each other.)

AlphaGo: 监督学习+强化学习  
ChatGPT: 自监督学习+强化学习



北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 提纲

## 一、强化学习问题定义

## 二、基于策略的强化学习

## 三、基于价值的强化学习

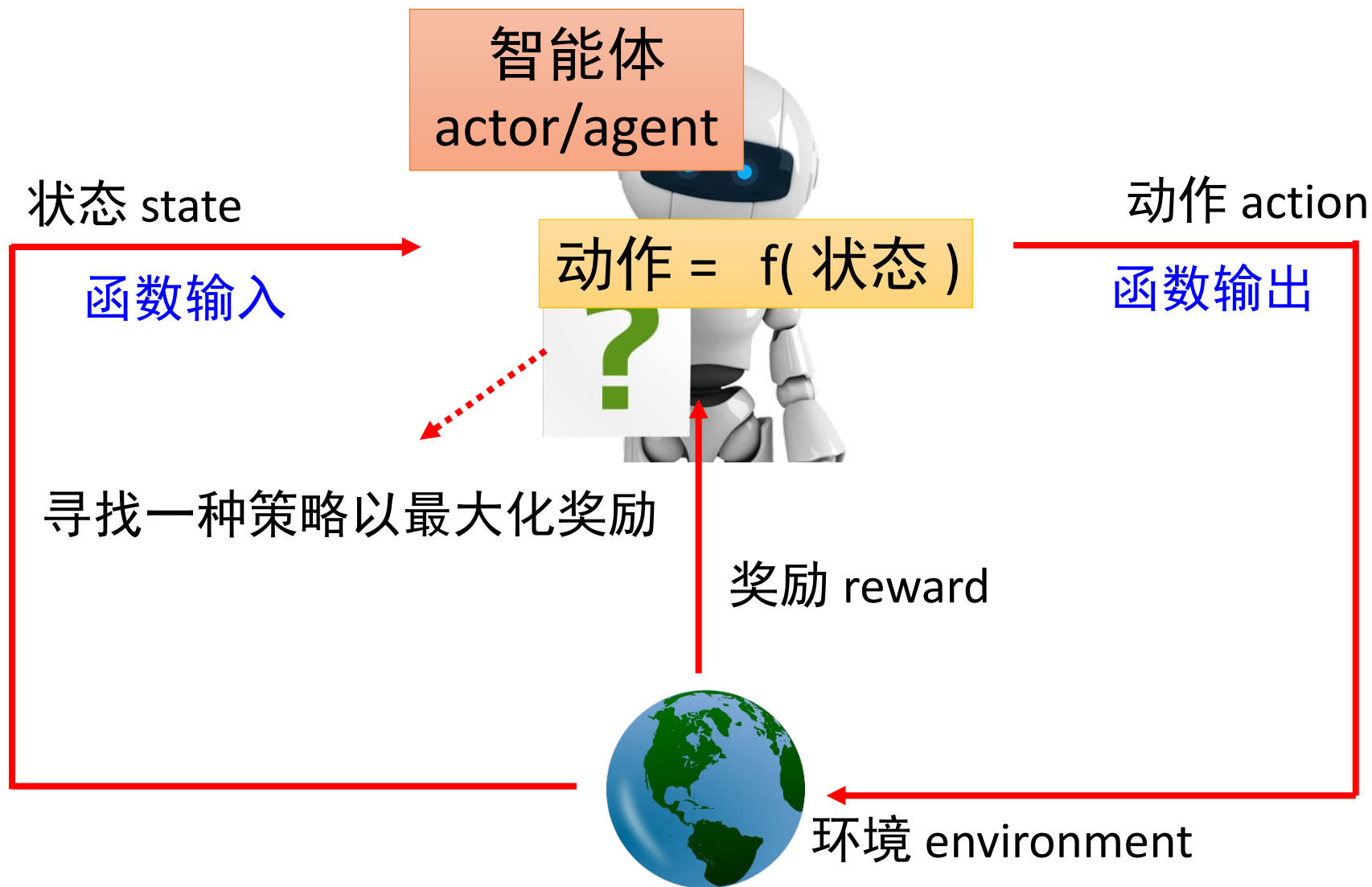
## 四、 Actor-Critic方法

## 五、其他强化学习方法



北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 机器学习≈函数拟合





# 例子：视频游戏

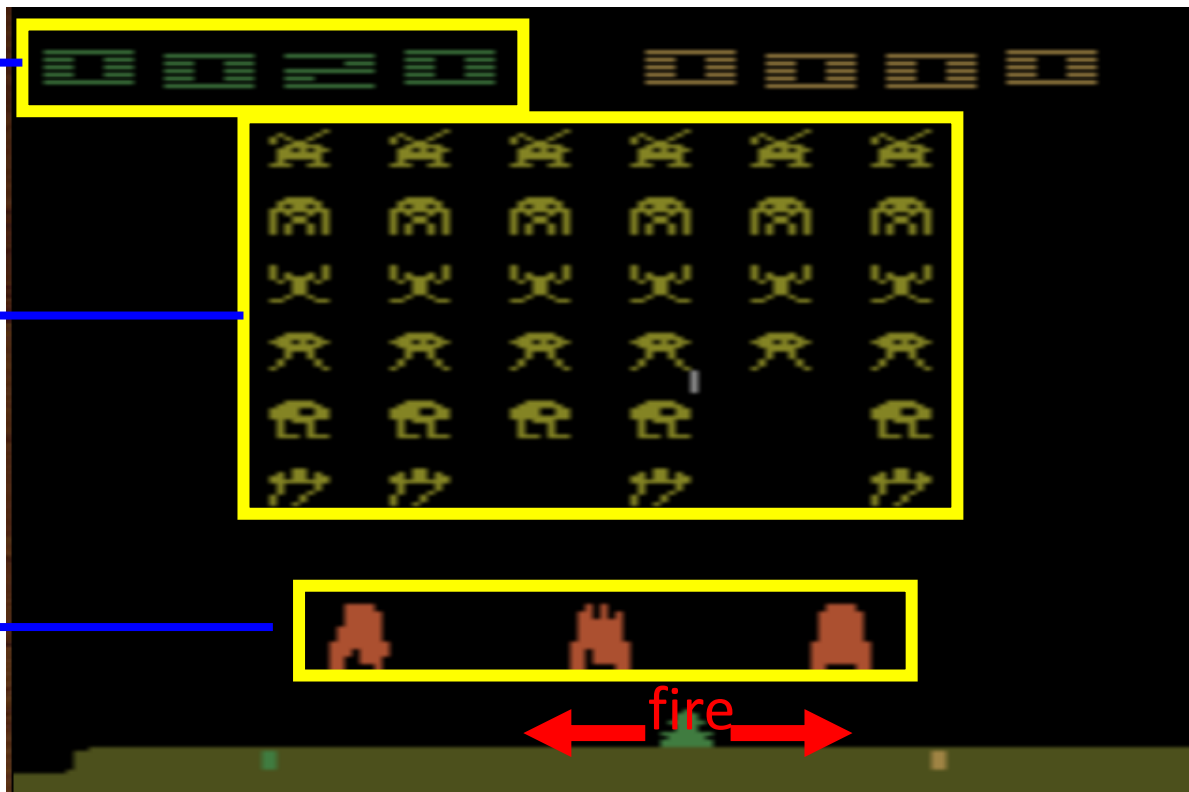
- 太空侵略者

游戏终止：所有的外星人被杀，  
或太空船损坏。

得分（奖励）←

杀死外星人←

防护盾←







北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 例子：视频游戏





北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 例子：视频游戏

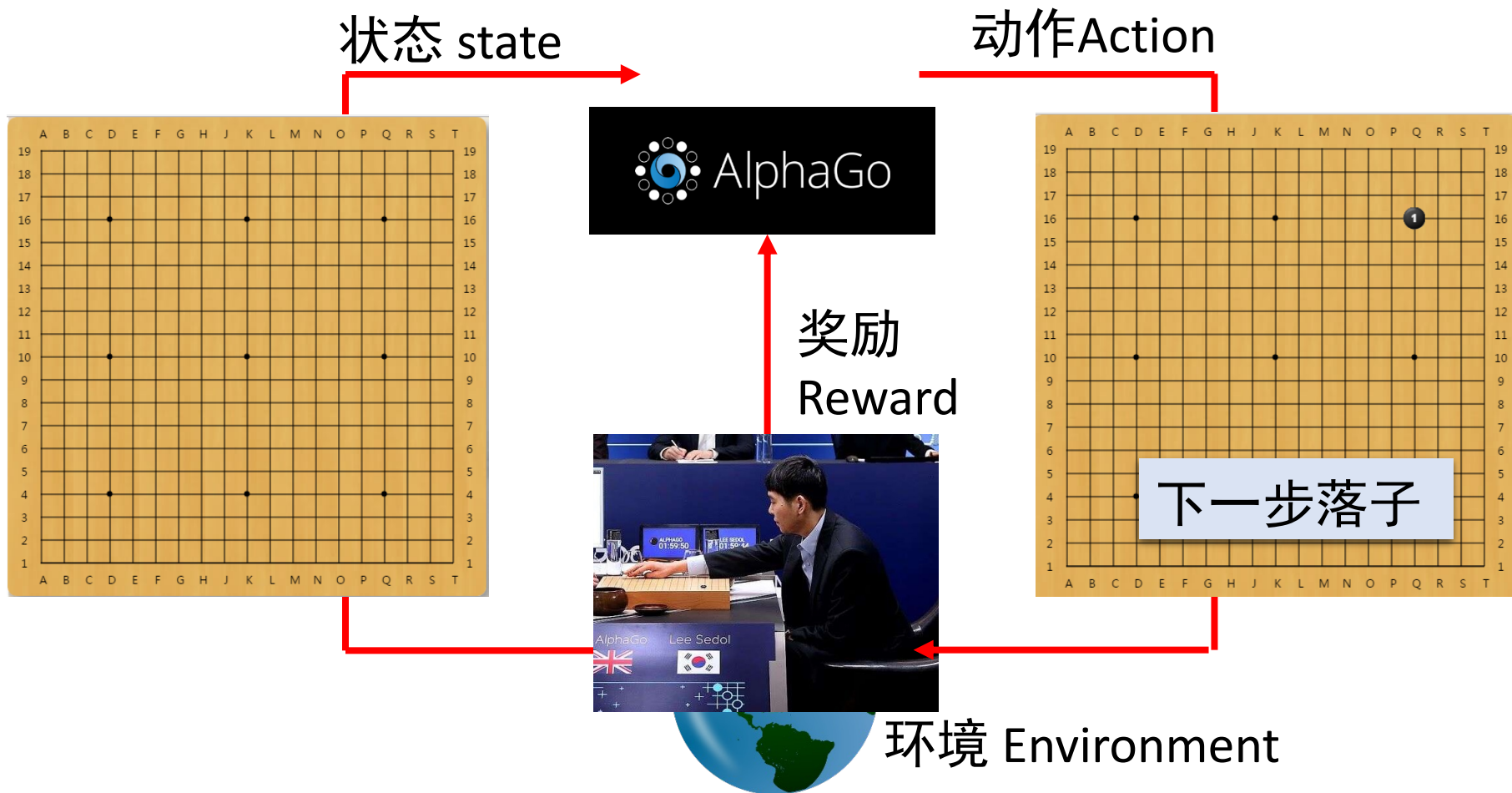
寻找能够最大化期望奖励的智能体





北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 例子：围棋游戏 AlphaGO

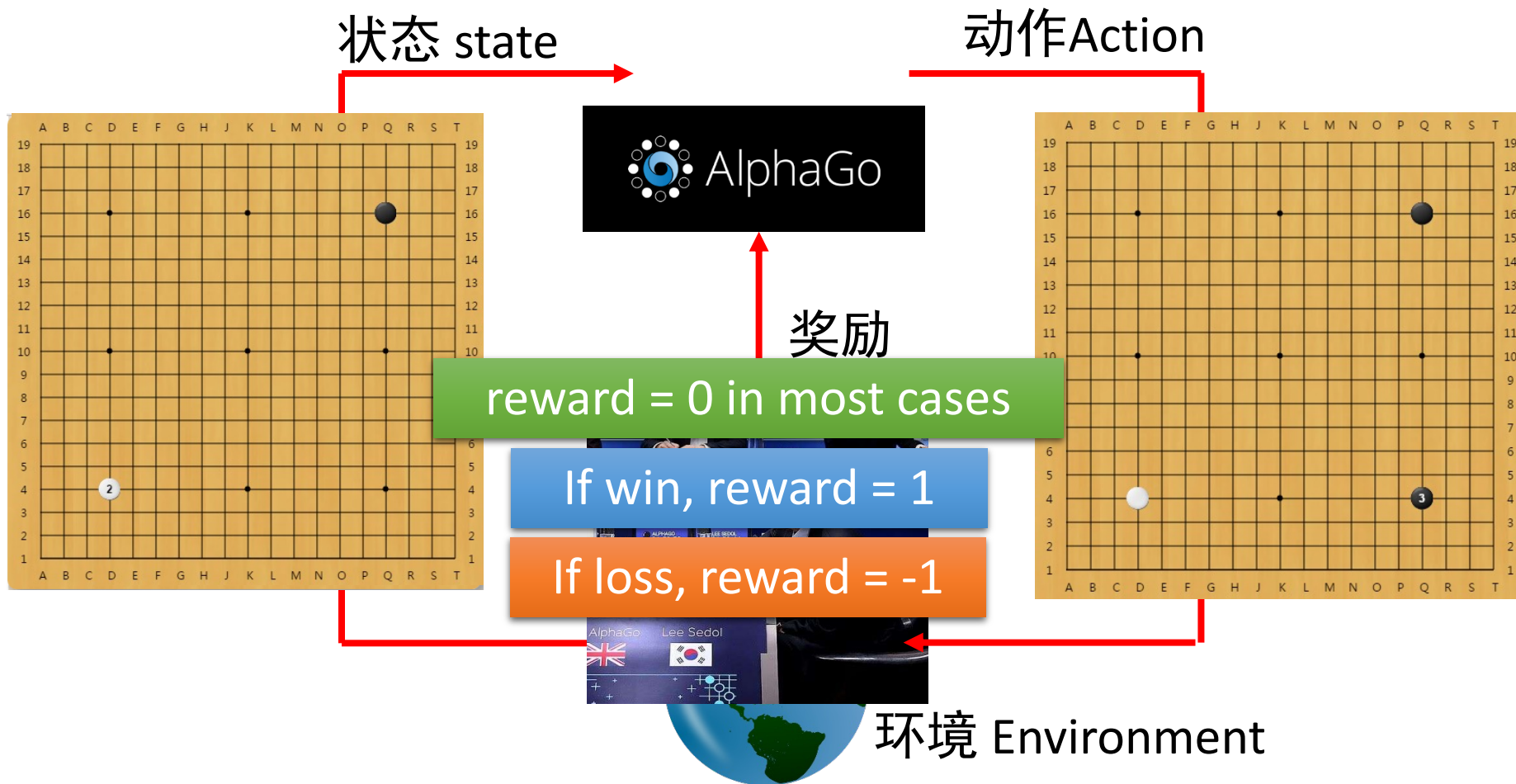




北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 例子：围棋游戏 AlphaGO

寻找能够最大化期望奖励的智能体

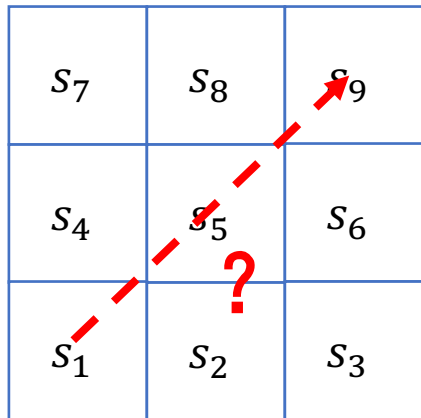




# 例子：机器人走迷宫

（序列优化）问题：

- 在下图网格中，假设有一个机器人位于 $s_1$ ，其每一步只能向上或向右移动一格，跃出方格会被惩罚（且游戏停止）
- 如何使用强化学习找到一种策略，使机器人从 $s_1$ 到达 $s_9$ ？



刻画解该问题的因素

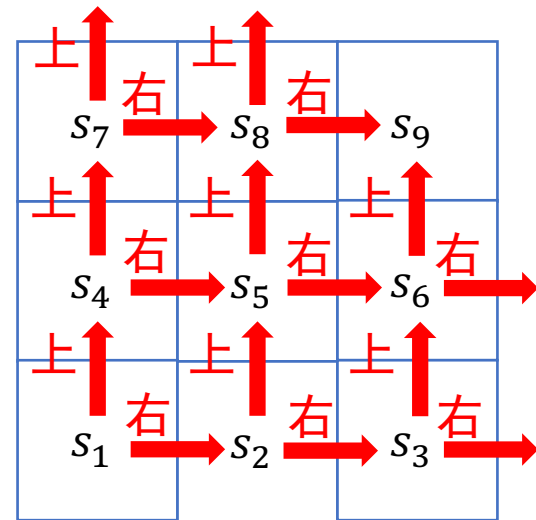
智能主体	迷宫机器人
环境	3×3方格
状态	机器人当前时刻所处方格
动作	每次移动一个方格
奖励	到达 $s_9$ 时给予奖励；越界时给予惩罚



# 马尔可夫决策过程 (Markov Decision Process)

使用离散马尔可夫过程描述机器人移动问题

- 随机变量序列 $\{S_t\}_{t=0,1,2,\dots}$ :  $S_t$ 表示机器人第 $t$ 步所在位置 (即状态), 每个随机变量 $S_t$ 的取值范围为 $S = \{s_1, s_2, \dots, s_9, s_d\}$
- 动作集合:  $A = \{\text{上}, \text{右}\}$
- 状态转移概率 $Pr(S_{t+1}|S_t, a_t)$ : 满足马尔可夫性, 其中 $a_t \in A$ 。状态转移如图所示。
- 奖励函数:  $R(S_t, a_t, S_{t+1})$
- 衰退系数:  $\gamma \in [0, 1]$



下一状态只取决于当前状态

- 动作集合 $A$ 可以是有限的, 也可以是无限的
- 状态转移可是确定 (deterministic) 的, 也可以是随机概率性 (stochastic) 的。
- 确定状态转移相当于发生从 $S_t$ 到 $S_{t+1}$ 的转移概率为1

综合以上信息, 可通过 $MDP = \{S, A, Pr, R, \gamma\}$ 来刻画马尔科夫决策过程



# 马尔可夫决策过程 (Markov Decision Process)

在机器人移动问题中：状态、行为、衰退系数、起始/终止状态、回报、状态转移概率矩阵的定义如下

$$S = \{s_1, s_2, \dots, s_9, s_d\}$$

$$A = \{\text{上}, \text{右}\}$$

$$\gamma = 0.99$$

起始状态:  $S_0 = s_1$

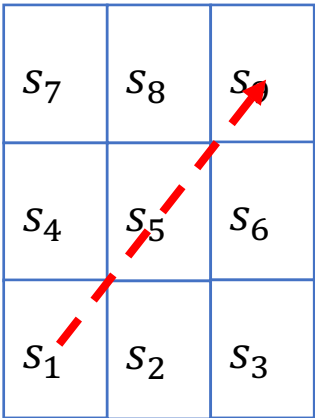
终止状态:  $S_T \in \{s_9, s_d\}$

$$R(S_t, a_t, S_{t+1}) = \begin{cases} 1, & \text{如果 } S_{t+1} = s_9 \\ -1, & \text{如果 } S_{t+1} = s_d \\ 0, & \text{其他情况} \end{cases}$$

图7.4 机器人寻路问题的状态转移函数

$Pr(S_{t+1} S_t, a_t = \text{右})$						
$S_{t+1} \backslash S_t$	$s_1$	$s_2$	$s_3$	$s_9$	$s_d$	
$s_1$	0	1	0	...	0	0
$s_2$	0	0	1	...	0	0
$s_3$	0	0	0	...	0	1
...				...		
$s_8$	0	0	0	...	1	0
$s_9$	0	0	0	...	0	1

$Pr(S_{t+1} S_t, a_t = \text{上})$						
$S_{t+1} \backslash S_t$	$s_1$	$s_4$	$s_7$	$s_9$	$s_d$	
$s_1$	0	1	0	...	0	0
$s_4$	0	0	1	...	0	0
$s_7$	0	0	0	...	0	1
...				...		
$s_6$	0	0	0	...	1	0
$s_9$	0	0	0	...	0	1



如何从起始状态  
到终止状态？





# 强化学习中的策略学习

马尔可夫决策过程 $MDP = \{S, A, Pr, R, \gamma\}$ 对环境进行了描述，那么智能主体如何与环境交互而完成任务？需要进行策略学习

## 策略函数：

- 策略函数 $\pi: S \times A \mapsto [0, 1]$ ，其中 $\pi(s, a)$ 的值表示在状态 $s$ 下采取动作 $a$ 的概率。
- 策略函数的输出可以是确定的，即给定 $s$ 情况下，只有一个动作 $a$ 使得概率 $\pi(s, a)$ 取值为1。  
对于确定的策略，记为 $a = \pi(s)$ 。
- 如何进行策略学习：一个好的策略是在当前状态下采取了一个行动后，该行动能够在未来收到最大化的回报Return：

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$





# 强化学习分类

- 基于策略的强化学习 (Policy-based)
  - 显式学习：策略函数
  - 无价值函数
- 基于价值的强化学习 (Value-based)
  - 显式学习：价值函数
  - 隐式得到：策略（可以通过价值函数得出策略）
- 行动器-判别器方法 (Actor-Critic)
  - 显式学习：策略函数和价值函数



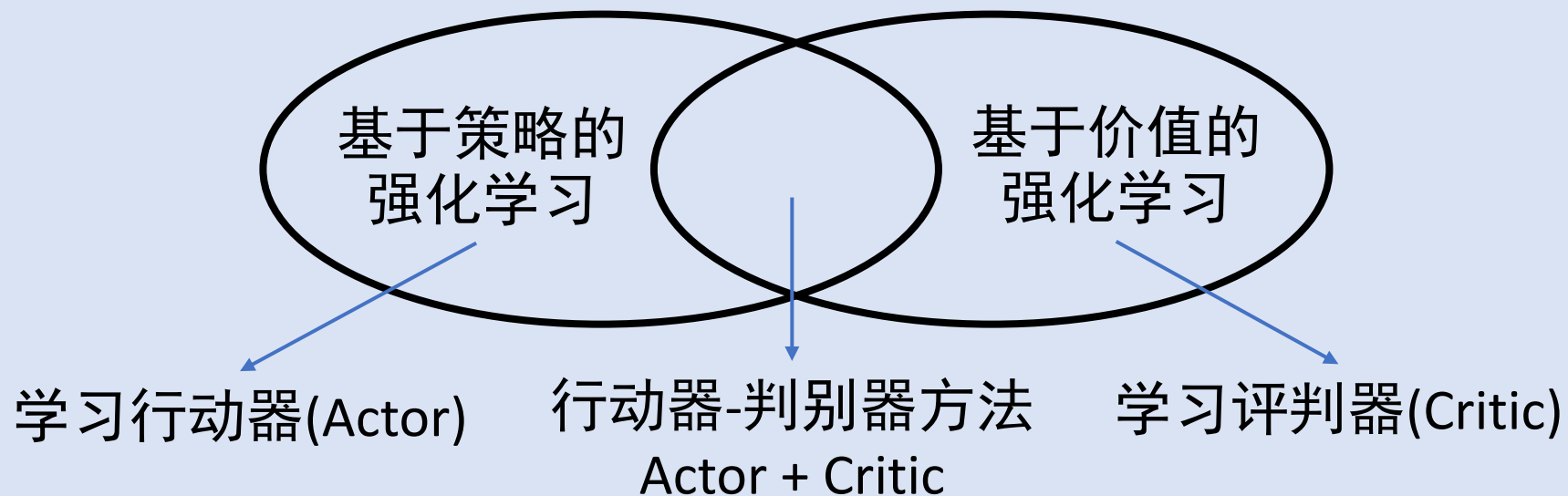
# 强化学习分类

- 基于环境模型的强化学习 (Model-based)
  - 显式学习：环境模型
  - 不一定需要策略函数或价值函数
- 无模型的强化学习 (Model-free)
  - 显式学习：价值函数或（和）策略函数
  - 无环境模型



# 强化学习分类

## 无模型强化学习



## 基于环境模型的强化学习



北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 提纲

## 一、强化学习问题定义

## 二、基于策略的强化学习

## 三、基于价值的强化学习

## 四、 Actor-Critic方法

## 五、其他强化学习方法



北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 机器学习步骤

步骤1: 定义  
带有未知参  
数的函数



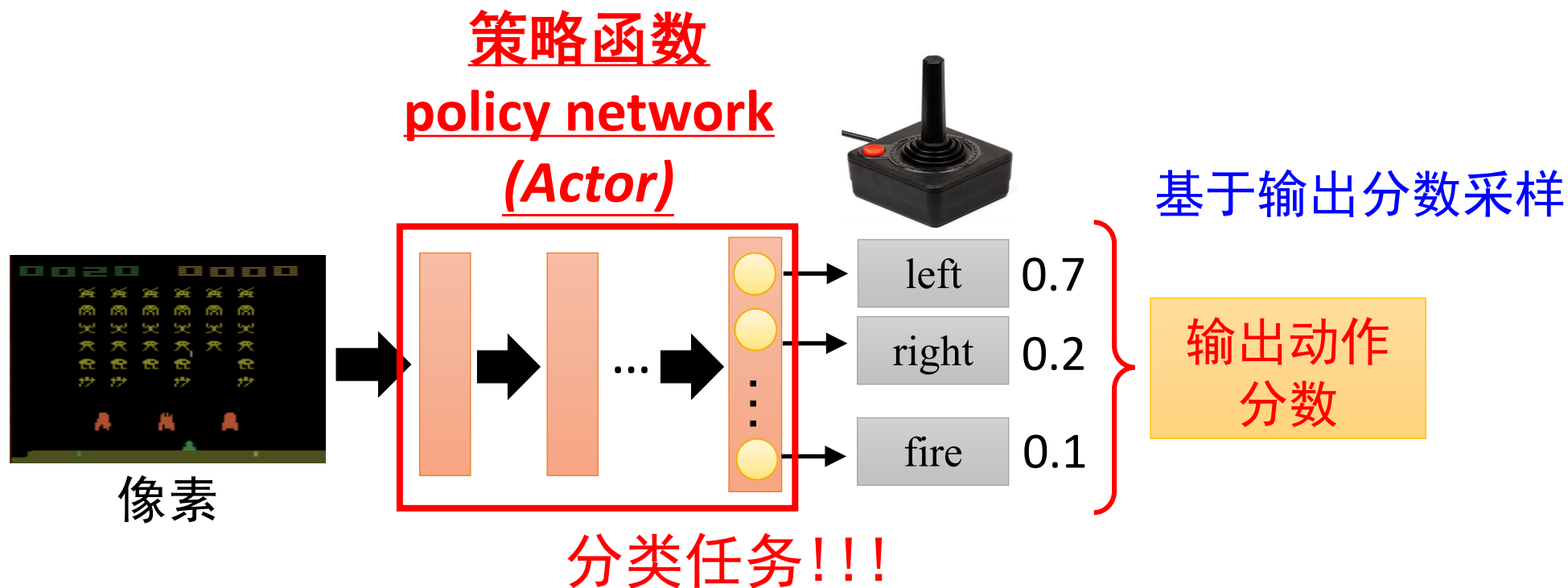
步骤2: 从训  
练数据中定  
义损失函数



步骤3: 优化



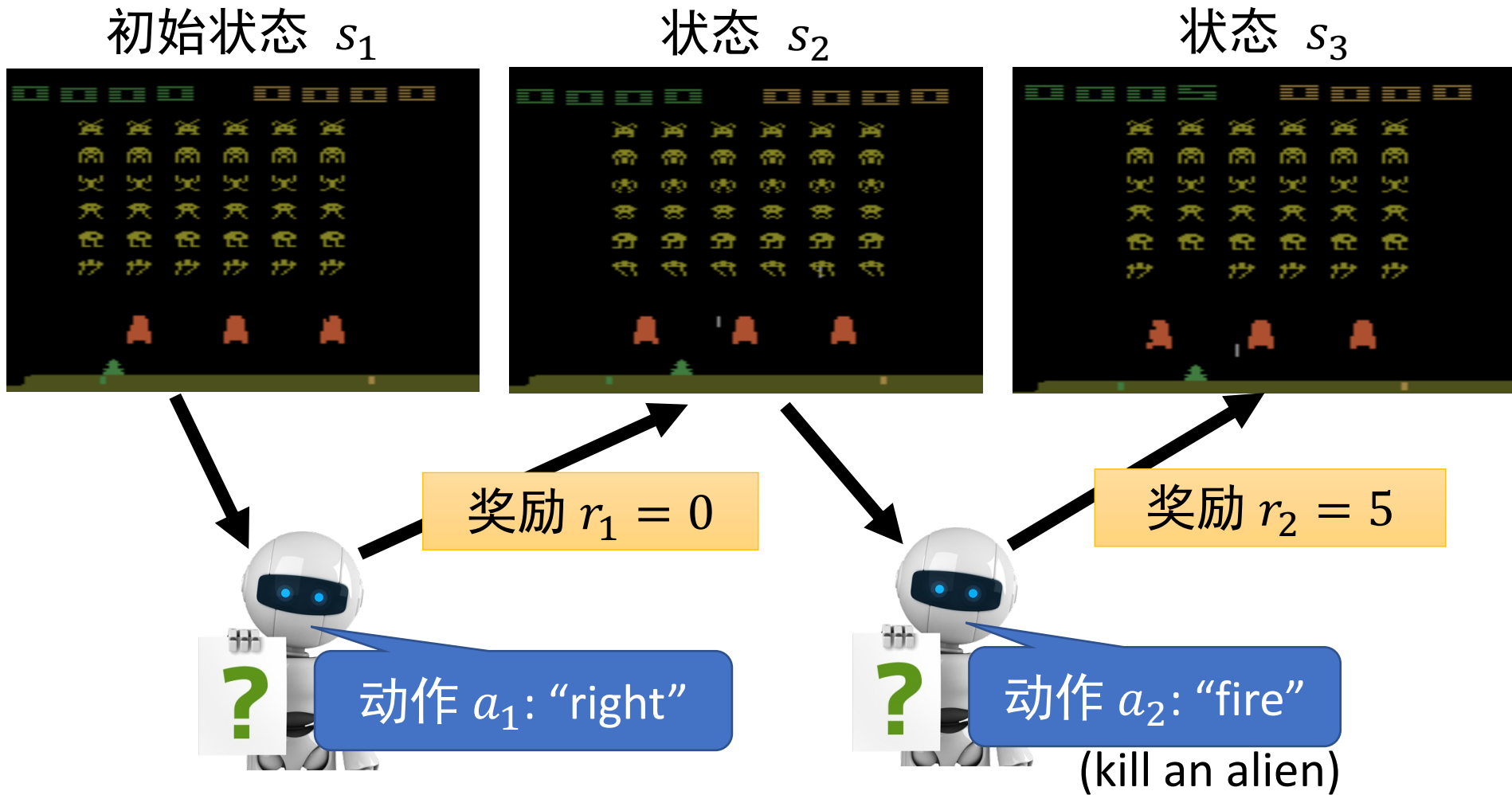
# 步骤 1: 定义带有未知参数的函数



- 神经网络的输入：向量或矩阵形式的状态数据
- 神经网络的输出：每个神经元对应的动作



## 步骤 2: 定义损失函数





## 步骤 2: 定义损失函数

初始状态  $s_1$



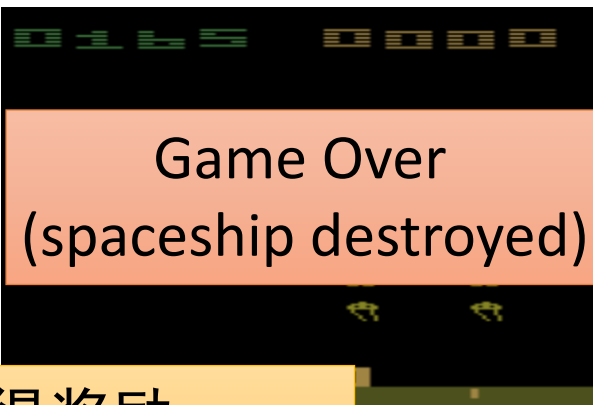
状态  $s_2$



状态  $s_3$



After many turns



获得奖励  $r_T$

动作  $a_T$

This is an episode (片段).

总奖励 (回报):

$$G = \sum_{t=1}^T r_t$$

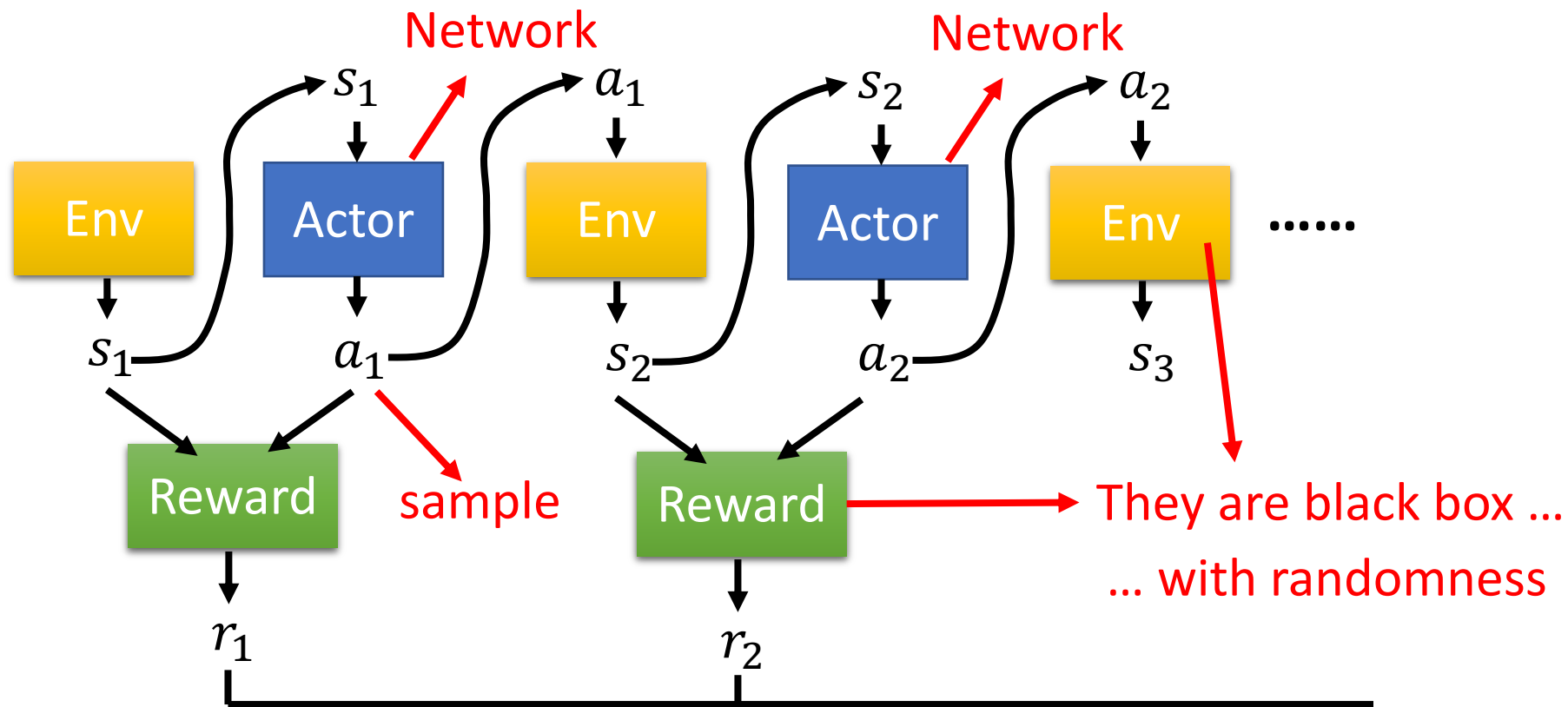
最大化回报






## 步骤 3: 优化

Trajectory (轨迹)  $\tau = \{s_1, a_1, s_2, a_2, \dots\}$



如何进行模型的优化是RL的主要挑战

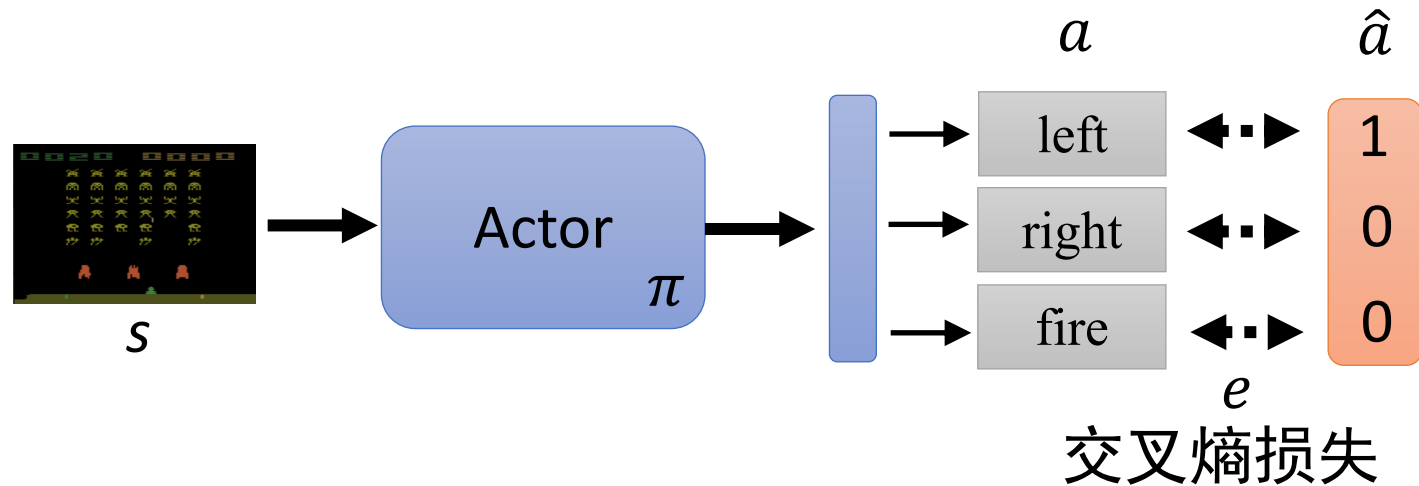
c.f. GAN

$$R(\tau) = \sum_{t=1}^T r_t$$




# 如何学习策略（学习actor）

- 对于某个状态 $s$ ，执行（或不执行）某个动作  $\hat{a}$



执行动作 $\hat{a}$ :  $L = e$

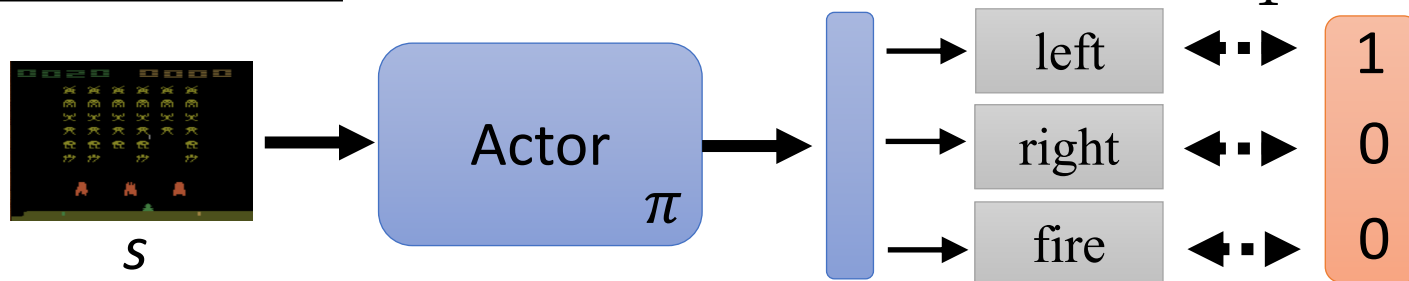
不执行动作 $\hat{a}$ :  $L = -e$

$$\pi^* = \arg \min_{\pi} L$$

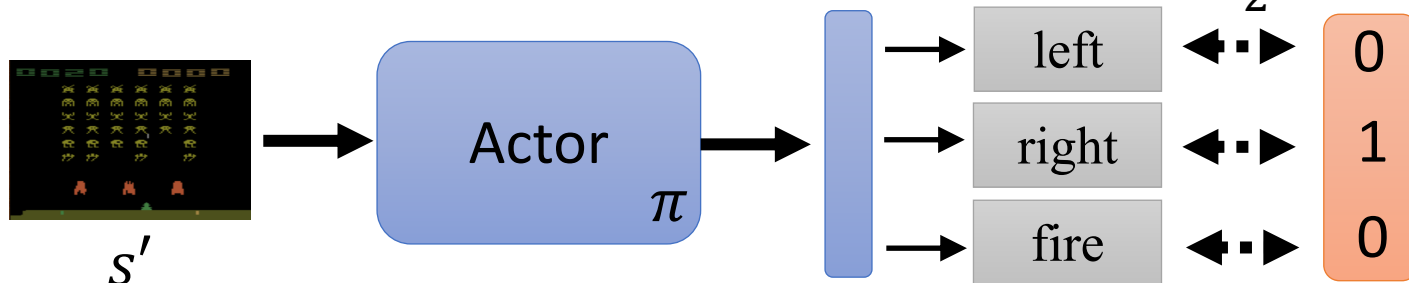


# 如何学习策略（学习actor）

对于  $s$  执行动作  $\hat{a}$



对于  $s'$  不执行动作  $\hat{a}'$



$$L = e_1 - e_2 \quad \pi^* = \arg \min_{\pi} L$$



# 如何学习策略（学习actor）

## Training Data

$\{s_1, \hat{a}_1\}$	+1	Yes
$\{s_2, \hat{a}_2\}$	-1	No
$\{s_3, \hat{a}_3\}$	+1	Yes
$\vdots$	$\vdots$	
$\{s_N, \hat{a}_N\}$	-1	No



$$L = +e_1 - e_2 + e_3 \cdots - e_N$$

$$\pi^* = \arg \min_{\pi} L$$



# 如何学习策略（学习actor）

## Training Data

$\{s_1, \hat{a}_1\}$	$A_1$	+1.5
$\{s_2, \hat{a}_2\}$	$A_2$	-0.5
$\{s_3, \hat{a}_3\}$	$A_3$	+0.5
$\vdots$	$\vdots$	
$\{s_N, \hat{a}_N\}$	$A_N$	-10
?	?	

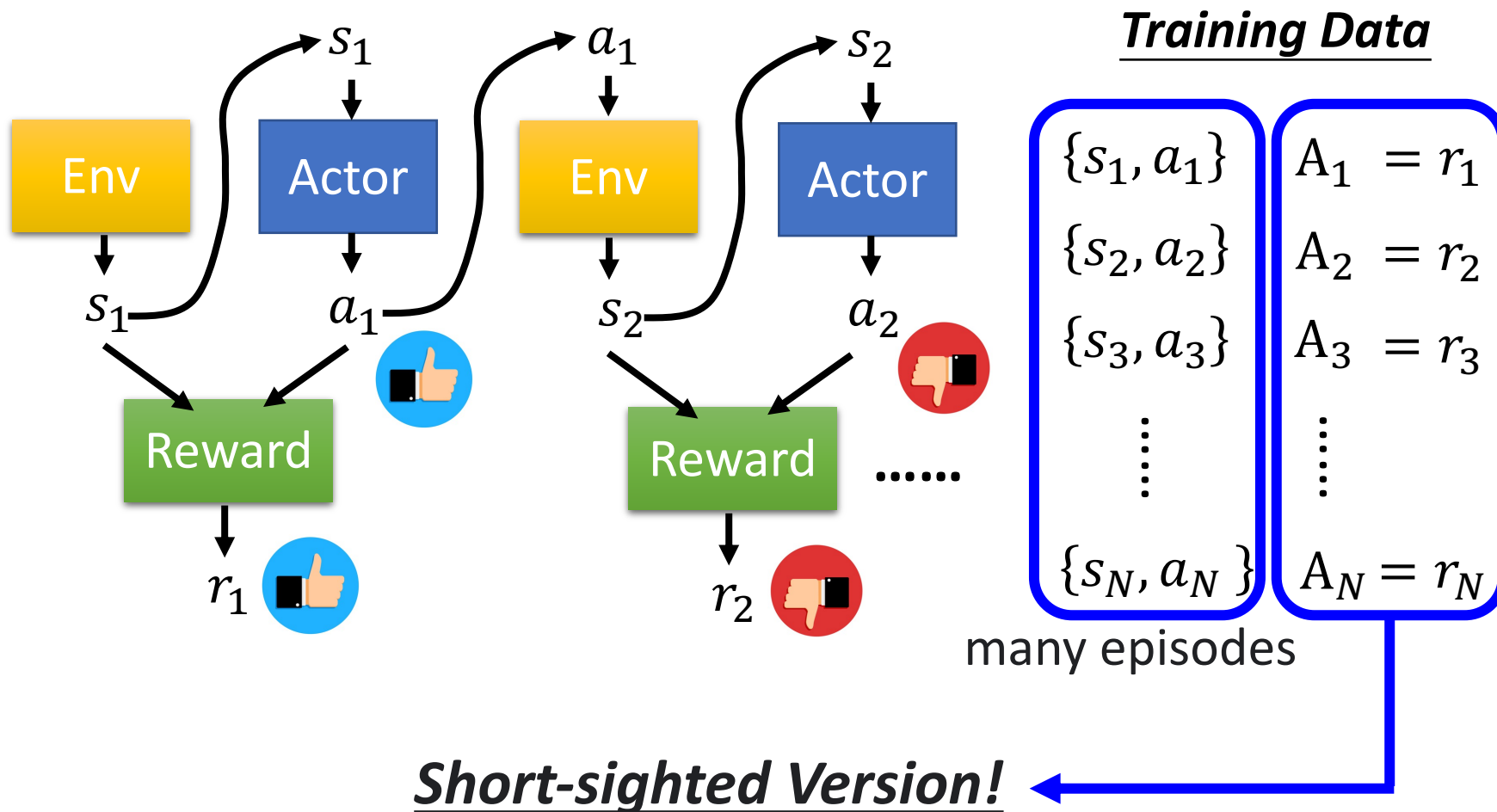


$$L = \sum A_n e_n$$

$$\pi^* = \arg \min_{\pi} L$$

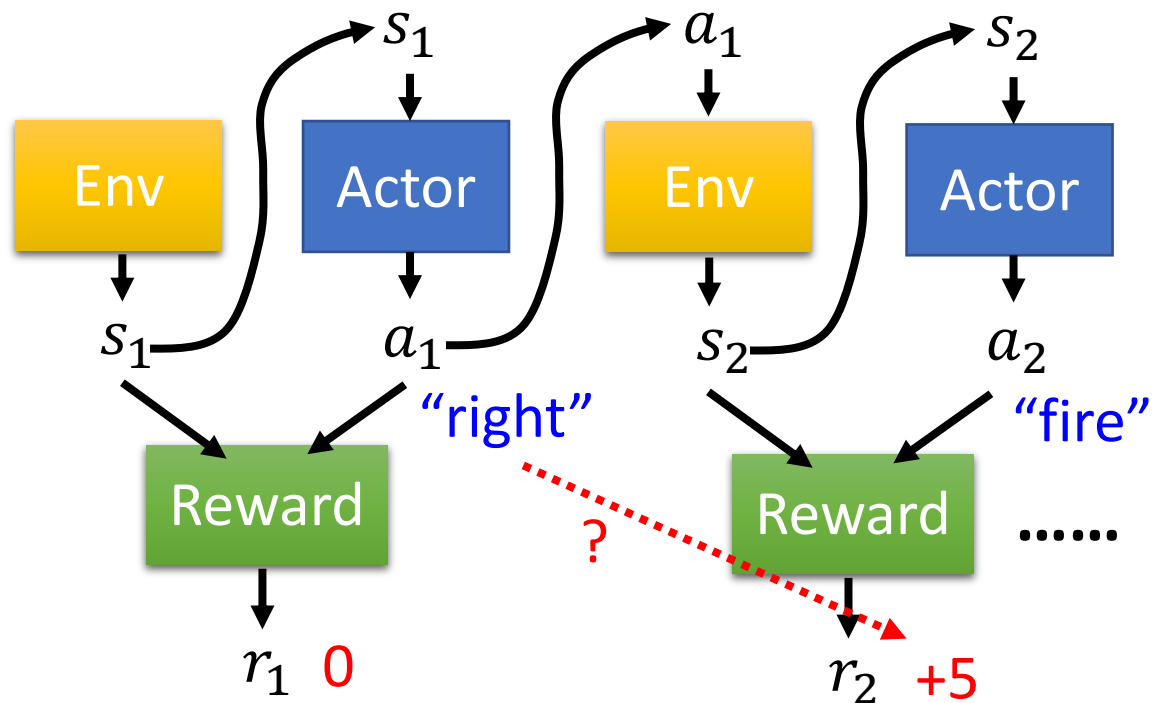


# 版本 0





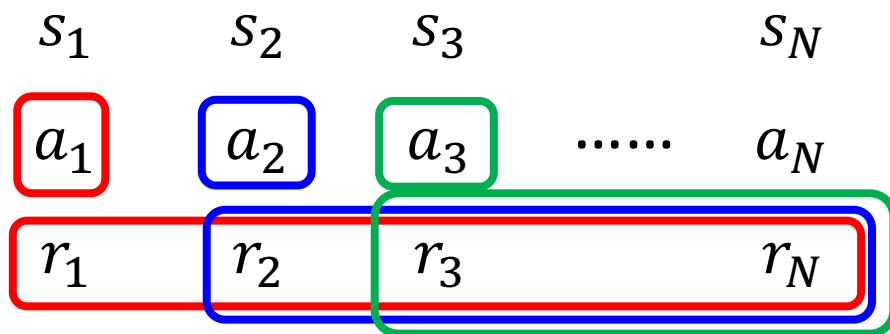
# 版本 0



- 某个动作的执行会影响后续状态，进而影响后续的奖励
- 奖励延迟：智能体需要牺牲立即奖励，以获得长远奖励
- 在太空入侵者游戏中，只有“fire”会引起正向奖励，因此版本0会让智能体不断开火。



# 版本 1



$$G_1 = r_1 + r_2 + r_3 + \dots + r_N$$

$$G_2 = r_2 + r_3 + \dots + r_N$$

$$G_3 = r_3 + \dots + r_N$$

累积奖励 (*cumulated reward*)

## Training Data

$$\{s_1, a_1\} \quad A_1 = G_1$$

$$\{s_2, a_2\} \quad A_2 = G_2$$

$$\{s_3, a_3\} \quad A_3 = G_3$$

$\vdots$

$\vdots$

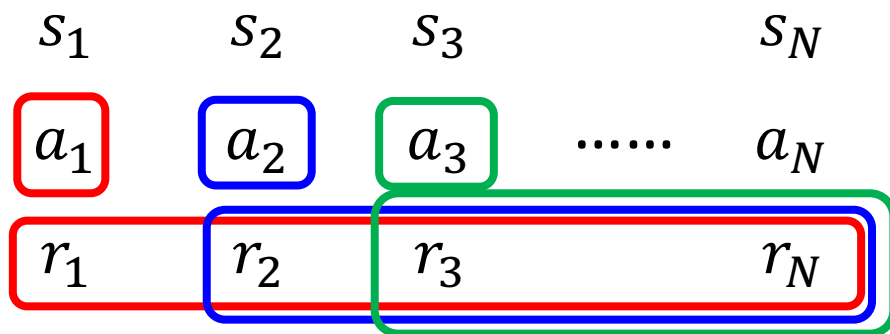
$$\{s_N, a_N\} \quad A_N = G_N$$

$$G_t = \sum_{n=t}^N r_n$$





# 版本 2



Also the credit of  $a_1$  ?

$$G_1 = r_1 + r_2 + r_3 + \dots + r_N$$

$$G'_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$$

衰退系数 (Discount factor)  $\gamma < 1$

## Training Data

$$\{s_1, a_1\} \quad A_1 = G'_1$$

$$\{s_2, a_2\} \quad A_2 = G'_2$$

$$\{s_3, a_3\} \quad A_3 = G'_3$$

$\vdots$

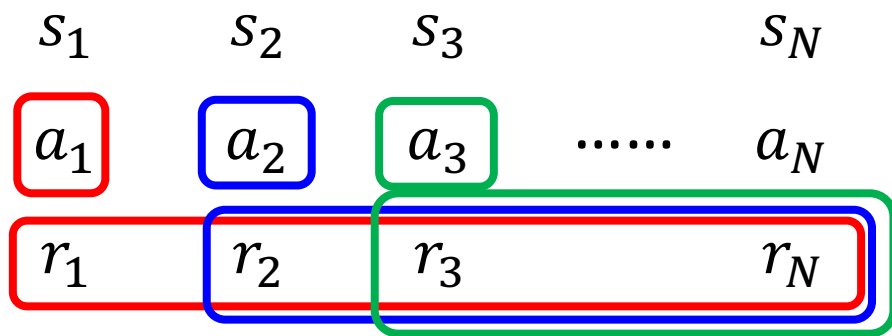
$\vdots$

$$\{s_N, a_N\} \quad A_N = G'_N$$

$$G'_t = \sum_{n=t}^N \gamma^{n-t} r_n$$



# 版本 3



奖励的好坏是“相对的”

如果所有的  $r_n \geq 10$ , 则  $r_n = 10$  是负奖励...

减去一个基线值  $b$

???

使得  $G'_t$  具有正负奖励值

## Training Data

$$\{s_1, a_1\} \quad A_1 = G'_1 - b$$

$$\{s_2, a_2\} \quad A_2 = G'_2 - b$$

$$\{s_3, a_3\} \quad A_3 = G'_3 - b$$

$\vdots$

$\vdots$

$$\{s_N, a_N\} \quad A_N = G'_N - b$$

$$G'_t = \sum_{n=t}^N \gamma^{n-t} r_n$$



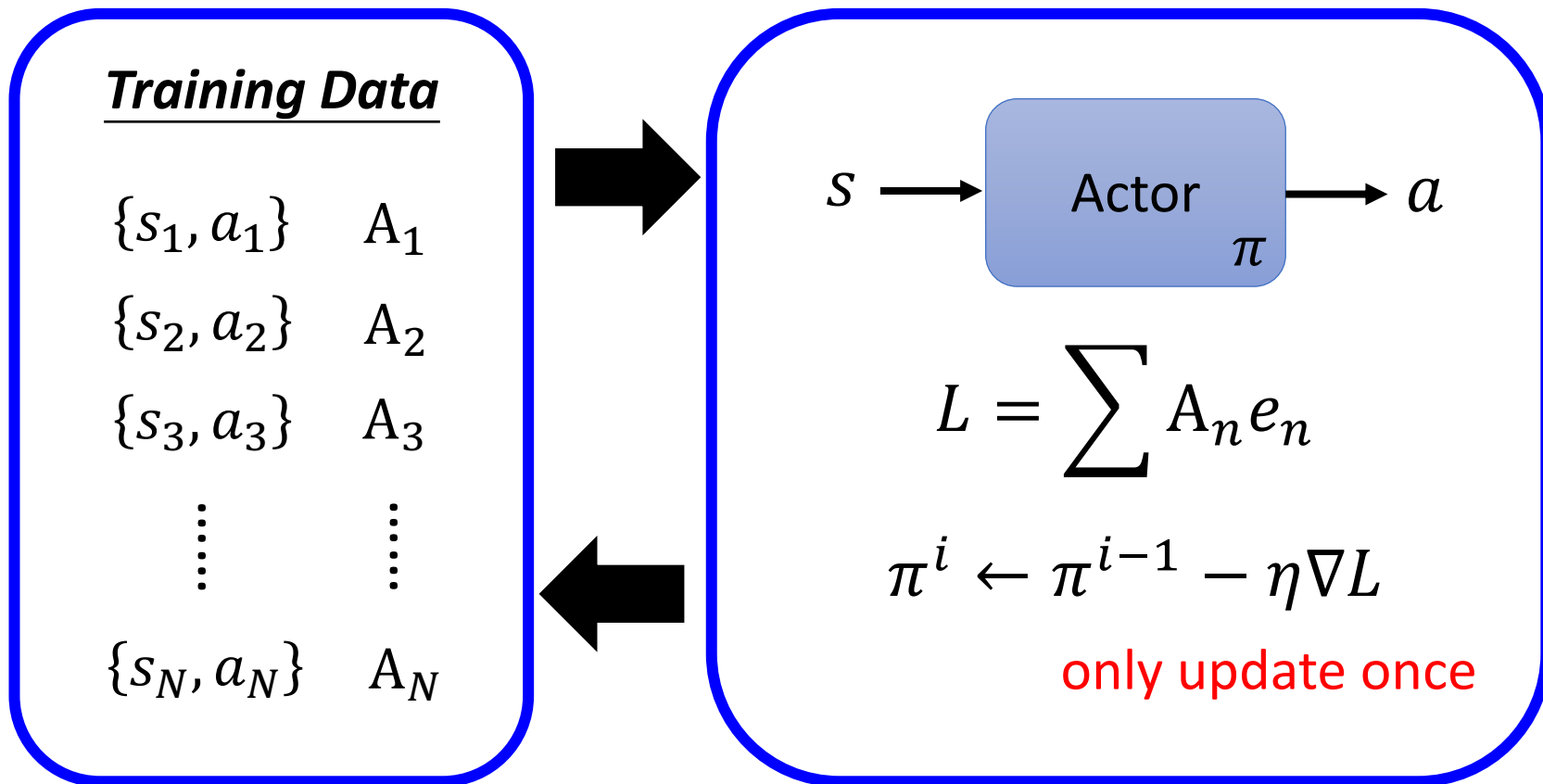
# 策略梯度 policy gradient

- 初始化策略网络参数  $\pi^0$
- 迭代训练  $i = 1$  to  $T$ 
  - 用策略网络  $\pi^{i-1}$  进行交互
  - 获得训练数据  $\{s_1, a_1\}, \{s_2, a_2\}, \dots, \{s_N, a_N\}$
  - 计算  $A_1, A_2, \dots, A_N$
  - 计算损失  $L$
  - $\pi^i \leftarrow \pi^{i-1} - \eta \nabla L$

数据采集是在训练迭代过程的“for 循环”内部完成.



# 策略梯度 policy gradient



每次模型的更新都需要重新采集整个训练数据集



# 策略梯度 policy gradient

- 初始化策略网络参数  $\pi^0$

- 迭代训练  $i = 1$  to  $T$

- 用策略网络  $\pi^{i-1}$  进行交互
- 获得训练数据  $\{s_1, a_1\}, \{s_2, a_2\}, \dots, \{s_N, a_N\}$
- 计算  $A_1, A_2, \dots, A_N$

- 计算损失  $L$

- $\pi^i \leftarrow \pi^{i-1} - \eta \nabla L$

不一定适合  $\pi^i$

$\pi^{i-1}$  学习的经验



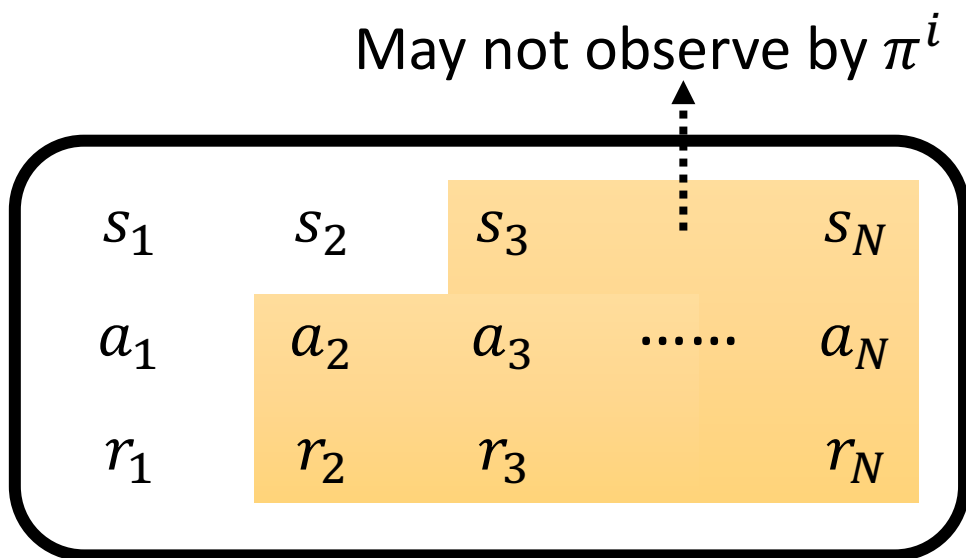
One man's meat is another man's poison.



# 策略梯度 policy gradient

- 初始化策略网络参数  $\pi^0$
- 迭代训练  $i = 1$  to  $T$ 
  - 用策略网络  $\pi^{i-1}$  进行交互
  - 获得训练数据  $\{s_1, a_1\}, \{s_2, a_2\}, \dots, \{s_N, a_N\}$
  - 计算  $A_1, A_2, \dots, A_N$
  - 计算损失  $L$
  - $\pi^i \leftarrow \pi^{i-1} - \eta \nabla L$

Trajectory of  $\pi^{i-1}$

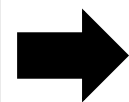
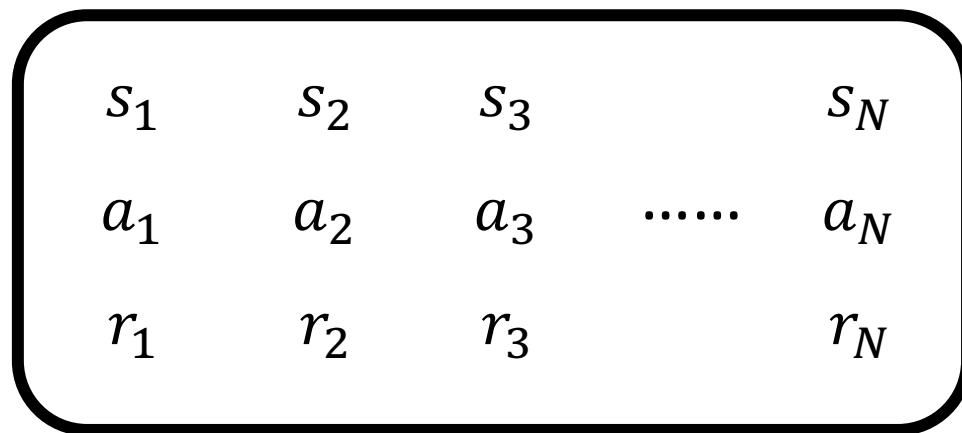




# 在线策略 On-policy v.s. 离线策略 Off-policy

- 待训练策略和交互策略相同  $\rightarrow$  On-policy
- 待训练策略和交互策略不同  $\rightarrow$  Off-policy  $\rightarrow$  Proximal Policy Optimization (PPO)

Trajectory of  $\pi^{i-1}$

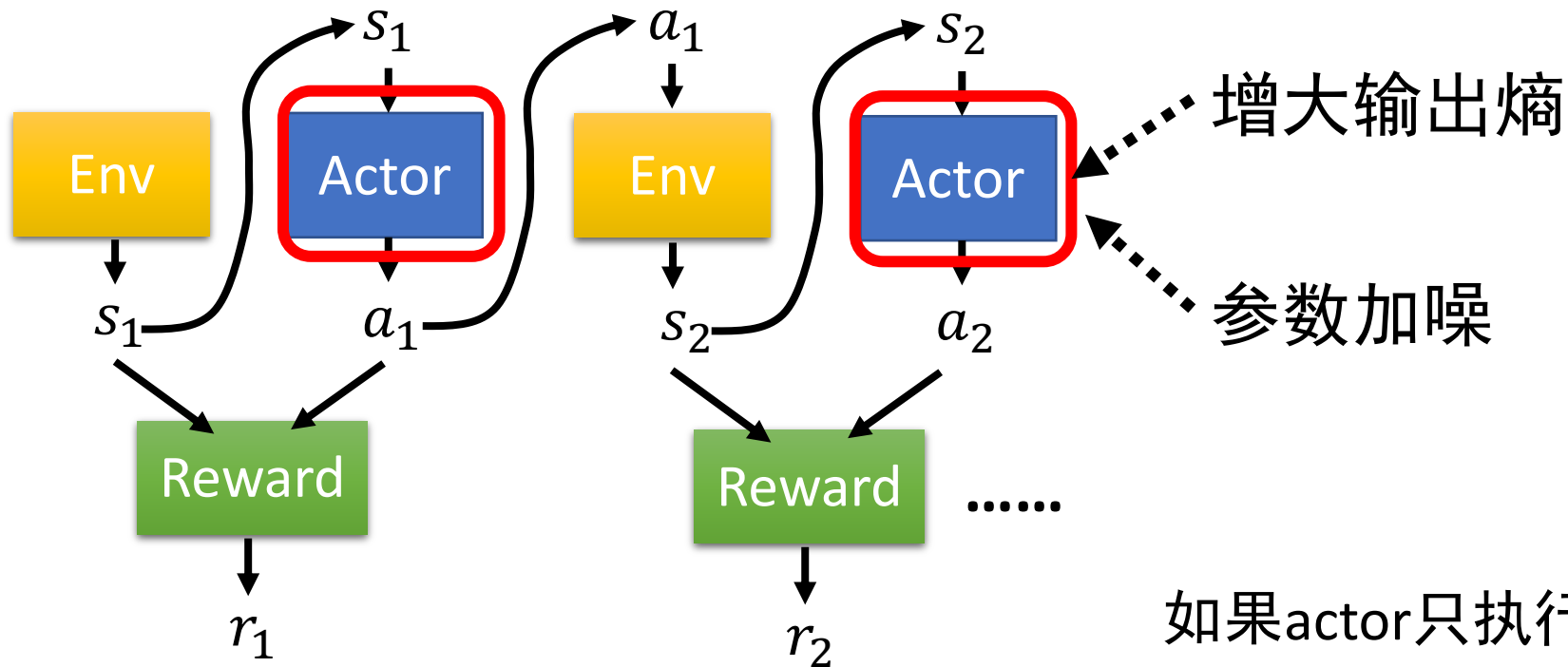


To train  $\pi^i$

因此，无需每次更新一次参数后都重新采集数据.



# 训练数据采集：探索



actor在数据采集过程中应该具有一定随机性.  
也就是要对数据进行采样的主要原因. 😊

如果actor只执行“left”这个行为.  
将永远无法得知采取“fire”行为  
会得到何种奖励.





北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# DeepMind – PPO(off-policy)

<https://youtu.be/gn4nRCC9TwQ>

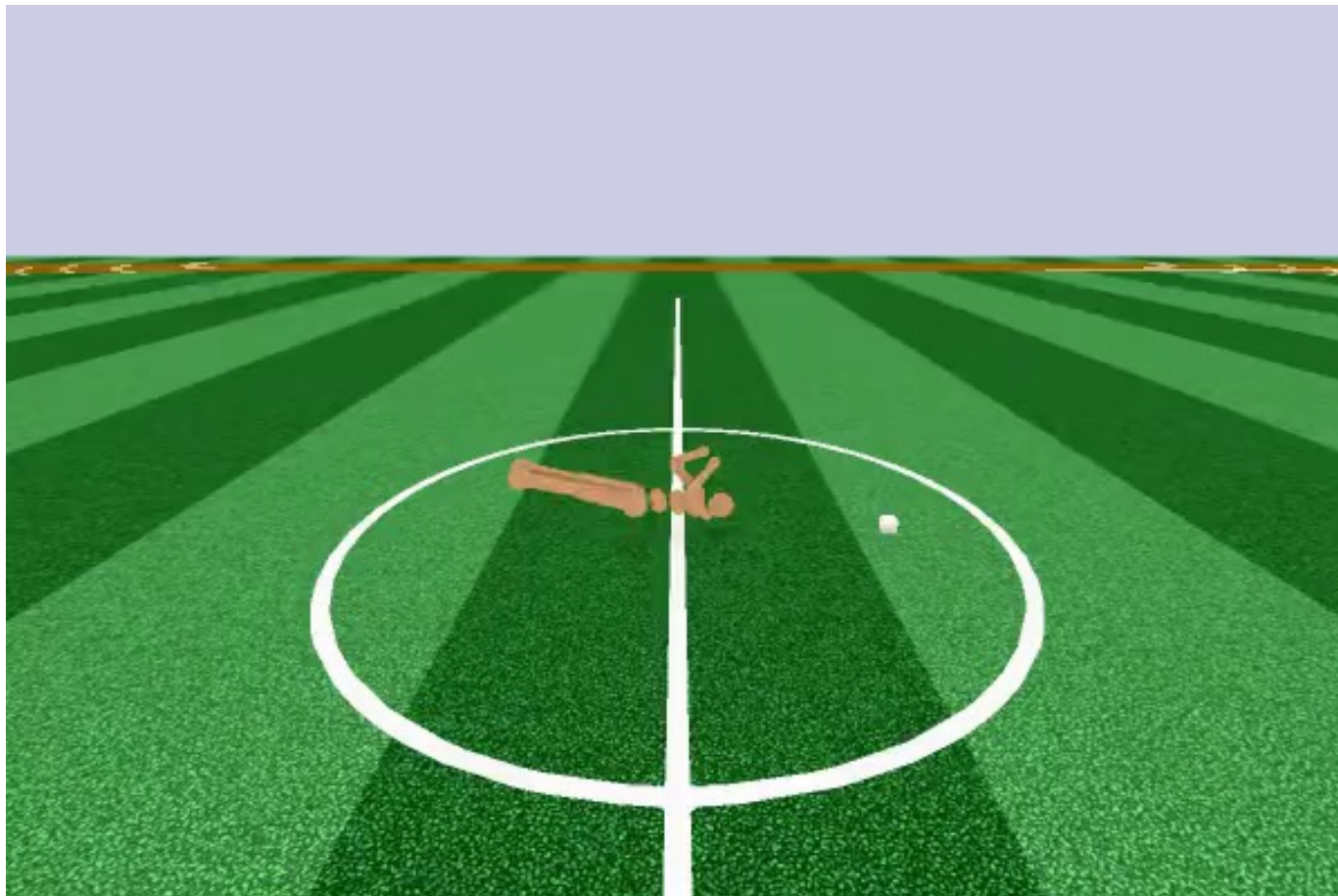




北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# OpenAI – PPO(off-policy)

<https://blog.openai.com/openai-baselines-ppo/>





北京航空航天大学  
COLLEGE OF SOFTWARE  
BEIHANG UNIVERSITY 软件学院

# 提纲

## 一、强化学习问题定义

## 二、基于策略的强化学习

## 三、基于价值的强化学习

## 四、Actor-Critic方法

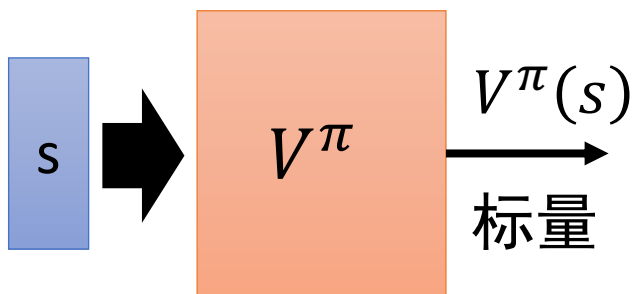
## 五、其他强化学习方法



## Critic (评判器)

$$G'_1 = r_1 + \gamma r_2 + \gamma^2 r_3 + \dots$$

- Critic: 对于策略 $\pi$ , 评判状态 $s$  (且执行动作 $a$ )的好坏
- 价值函数 $V^\pi(s)$ :
  - 使用 $\pi$ 时, 观察到状态 $s$ 后未来获得的折扣累积奖励(discounted *cumulated* reward)的期望。



$V^\pi(s)$  大



$V^\pi(s)$  小

Critic的输出值与当前策略 $\pi$ 有关



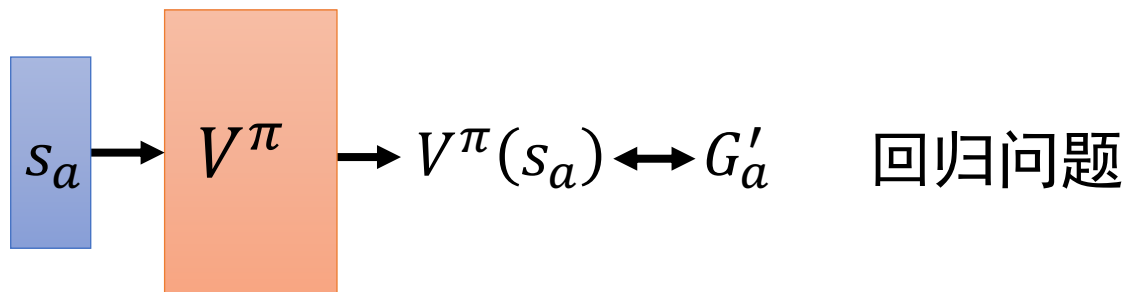
# 估计 $V^\pi(s)$

- 蒙特卡洛法 Monte-Carlo (MC)

- Critic 观看  $\pi$  与环境进行交互。

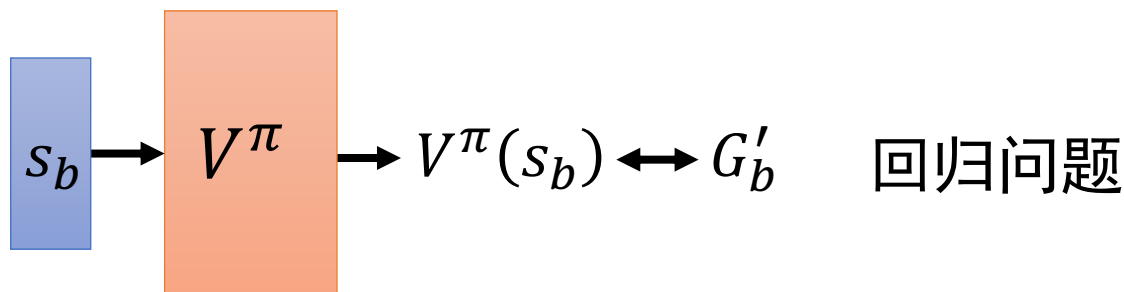
观察到  $s_a$  后,

在一个片段(episode)结束后,  
累积奖励为  $G'_a$



观察到  $s_b$  后,

在一个片段(episode)结束后,  
累积奖励为  $G'_b$





# 估计 $V^\pi(s)$

## • 时序差分法 Temporal-difference (TD)

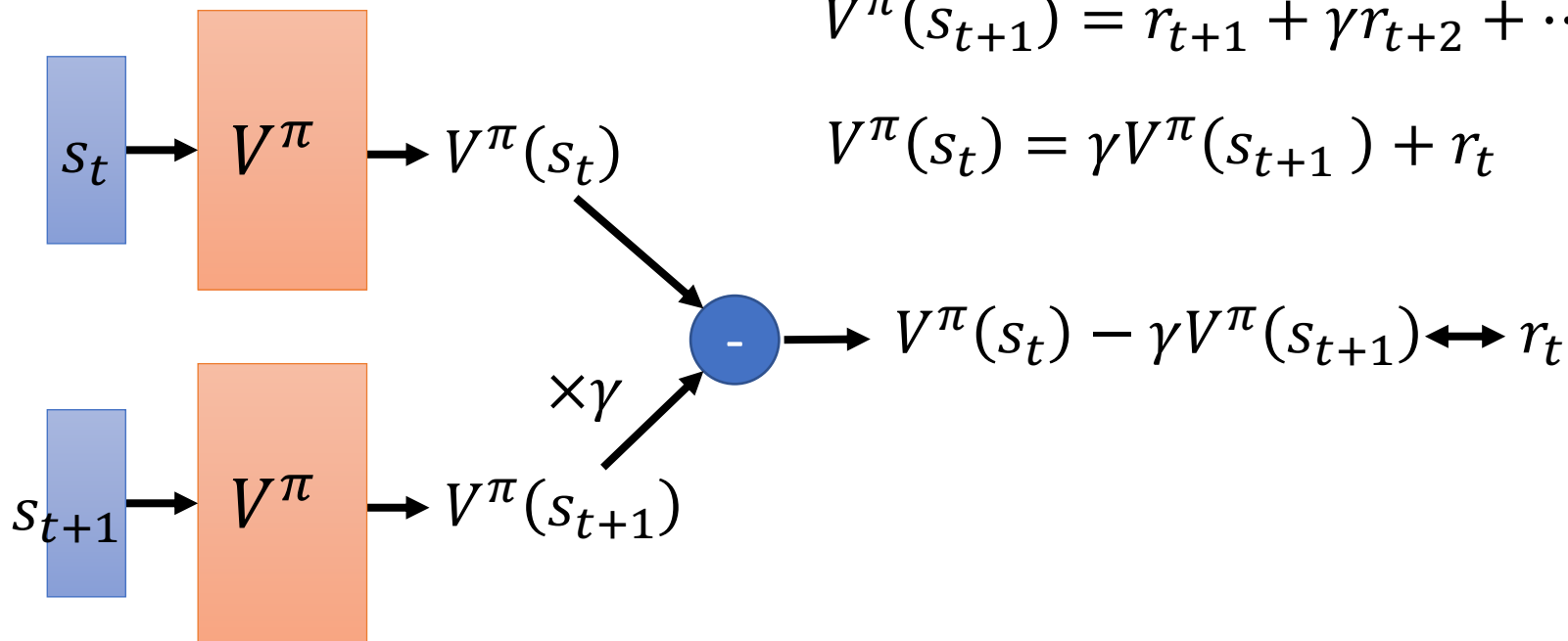
$\cdots S_t, a_t, r_t, S_{t+1} \cdots$

(ignore the expectation here)

$$V^\pi(s_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdots$$

$$V^\pi(s_{t+1}) = r_{t+1} + \gamma r_{t+2} + \cdots$$

$$V^\pi(s_t) = \gamma V^\pi(s_{t+1}) + r_t$$





# MC v.s. TD

[Sutton, v2, Example 6.4]

- 假设critic观察到了以下8个片段(episodes)

- $s_a, r = 0, s_b, r = 0, \text{END}$

- $s_b, r = 1, \text{END}$

- $s_b, r = 1, \text{END}$

- $s_b, r = 1, \text{END}$

- $s_b, r = 1, \text{END}$

- $s_b, r = 1, \text{END}$

- $s_b, r = 1, \text{END}$

- $s_b, r = 0, \text{END}$

$$V^\pi(s_b) = 3/4$$

$$V^\pi(s_a) = ? \quad 0? \quad 3/4?$$

蒙特卡洛法:  $V^\pi(s_a) = 0$

时序差分法:  $V^\pi(s_a) = V^\pi(s_b) + r$   
 $3/4 \quad 3/4 \quad 0$

(假设  $\gamma = 1$ , 并忽略动作.)