



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

人工智能

第12讲：机器学习-强化学习

基于策略的强化学习

张晶

2025年春季

- 参考资料：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

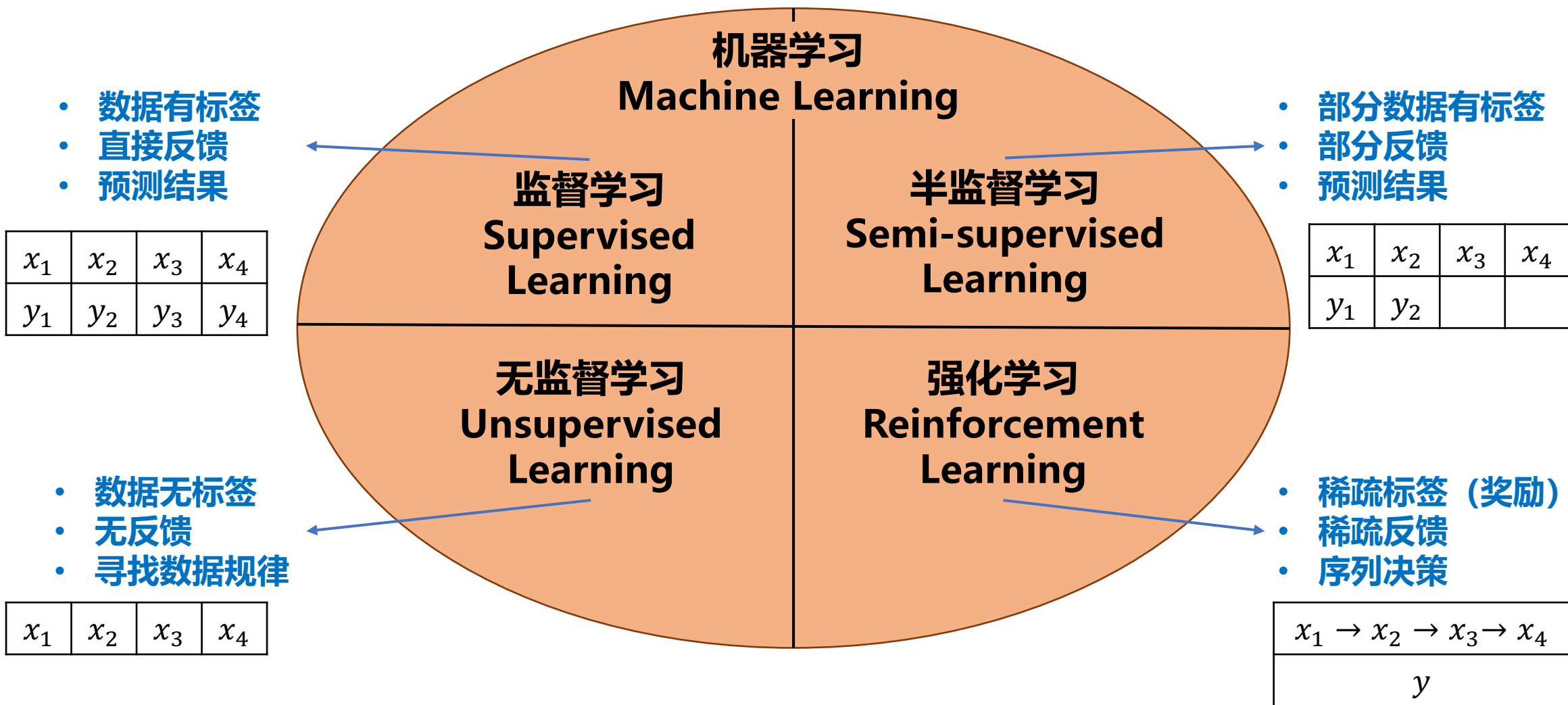
三、基于价值的强化学习

四、 Actor-Critic方法

五、其他强化学习方法



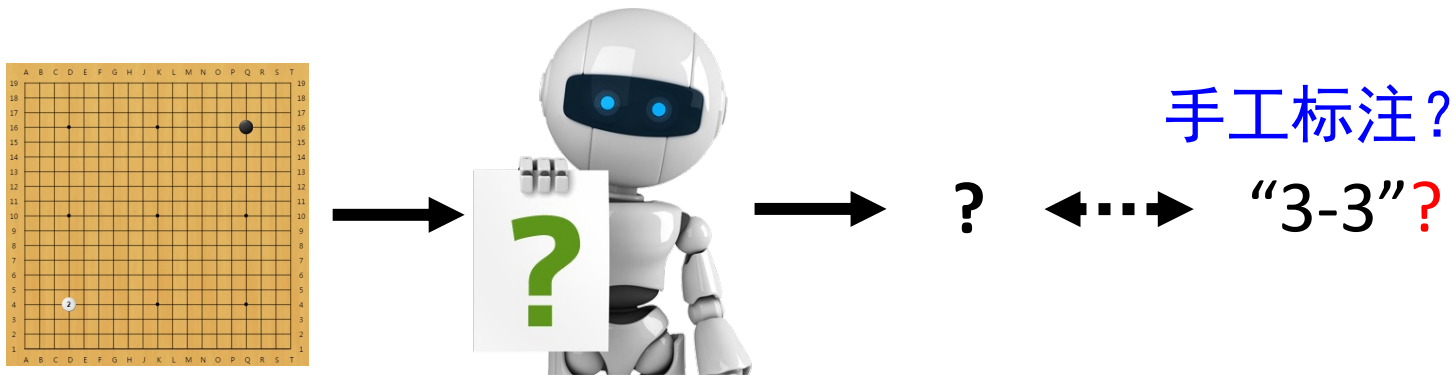
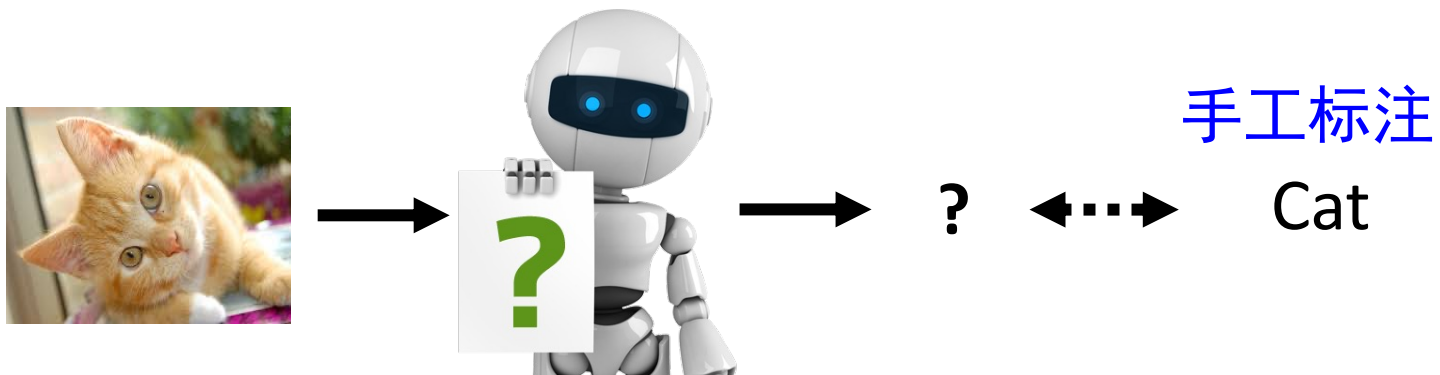
机器学习的分类（按数据标注情况分类）





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

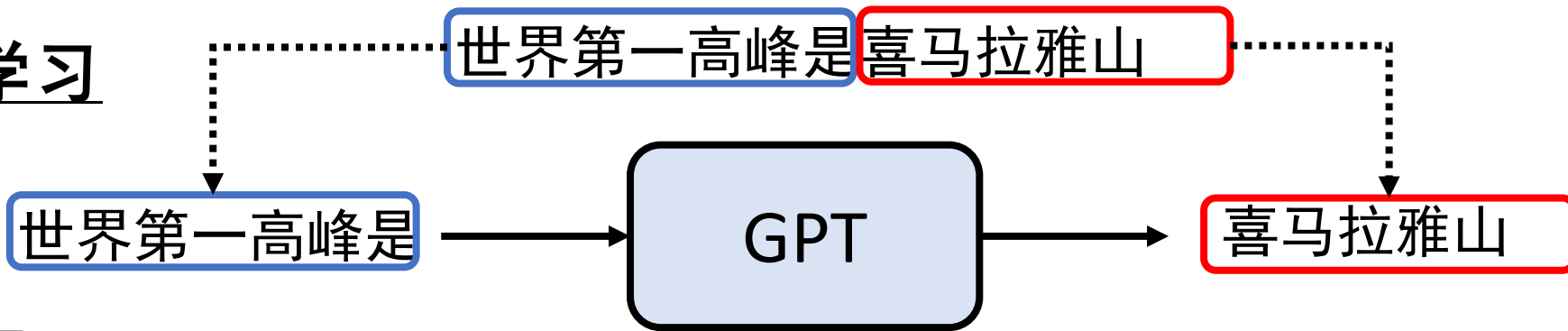
…… 但是可以告诉机器某个结果是好是坏



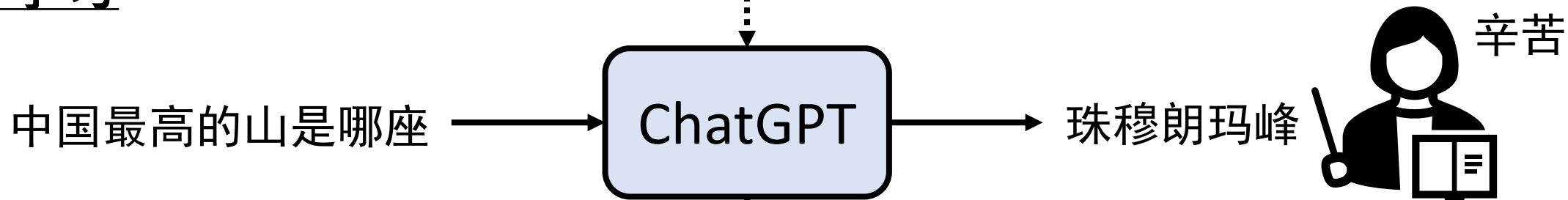
北京航空航天大学
COLLEGE OF SOFTWARE
BEIHANG UNIVERSITY 软件学院

ChatGPT: 自监督预训练 + 强化学习微调

自监督学习

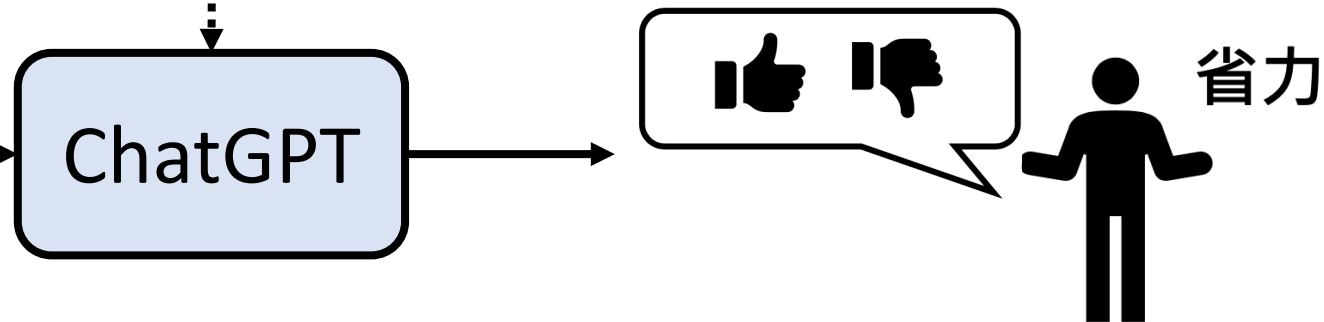


监督学习



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

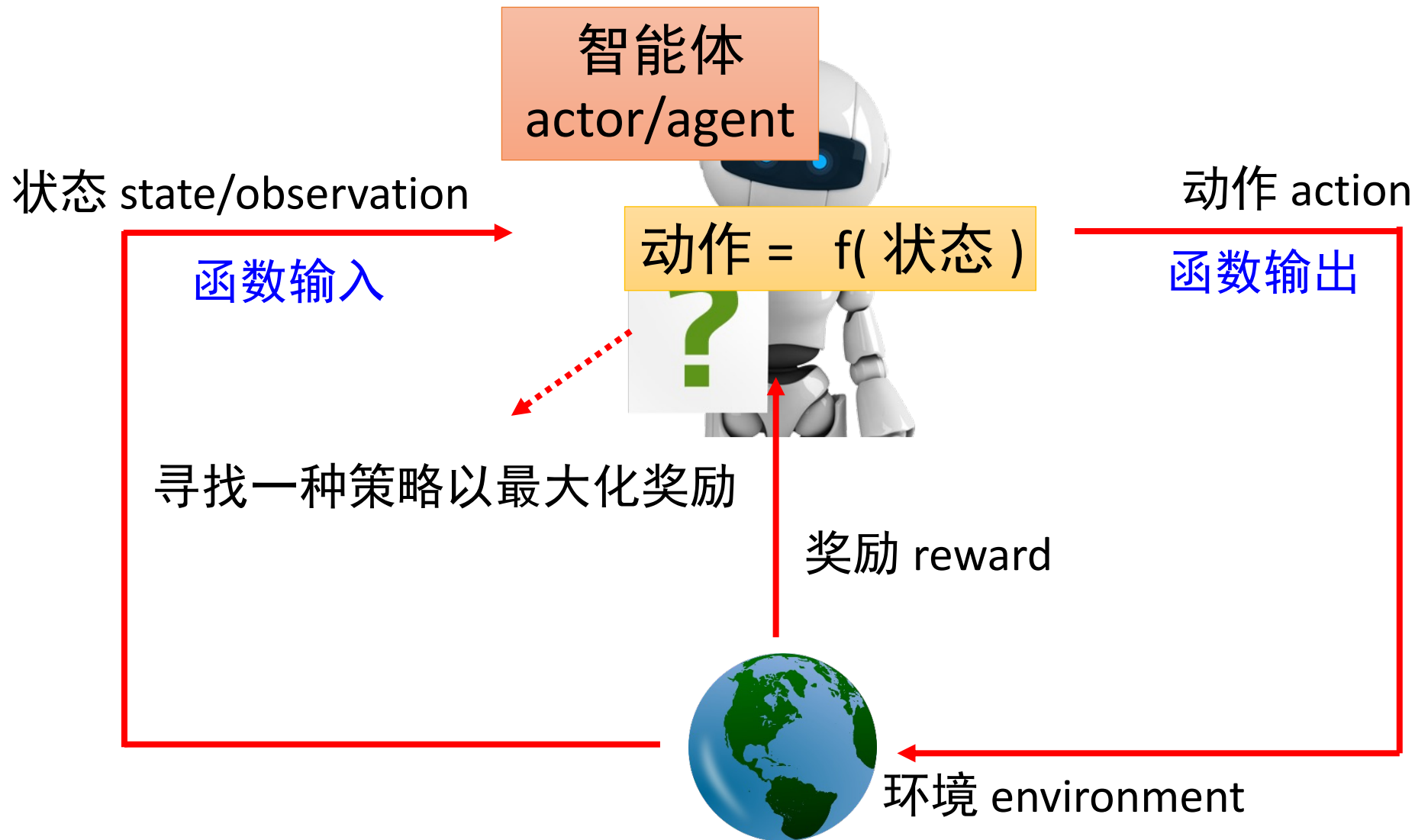
四、 Actor-Critic方法

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





强化学习定义

- 通过从交互中学习来实现目标的计算方法



- 三个方面：

- 感知：在某种程度上感知环境的状态
- 行动：可以采取行动来影响状态或者达到目标
- 目标：随着时间推移最大化累积奖励



例子：视频游戏

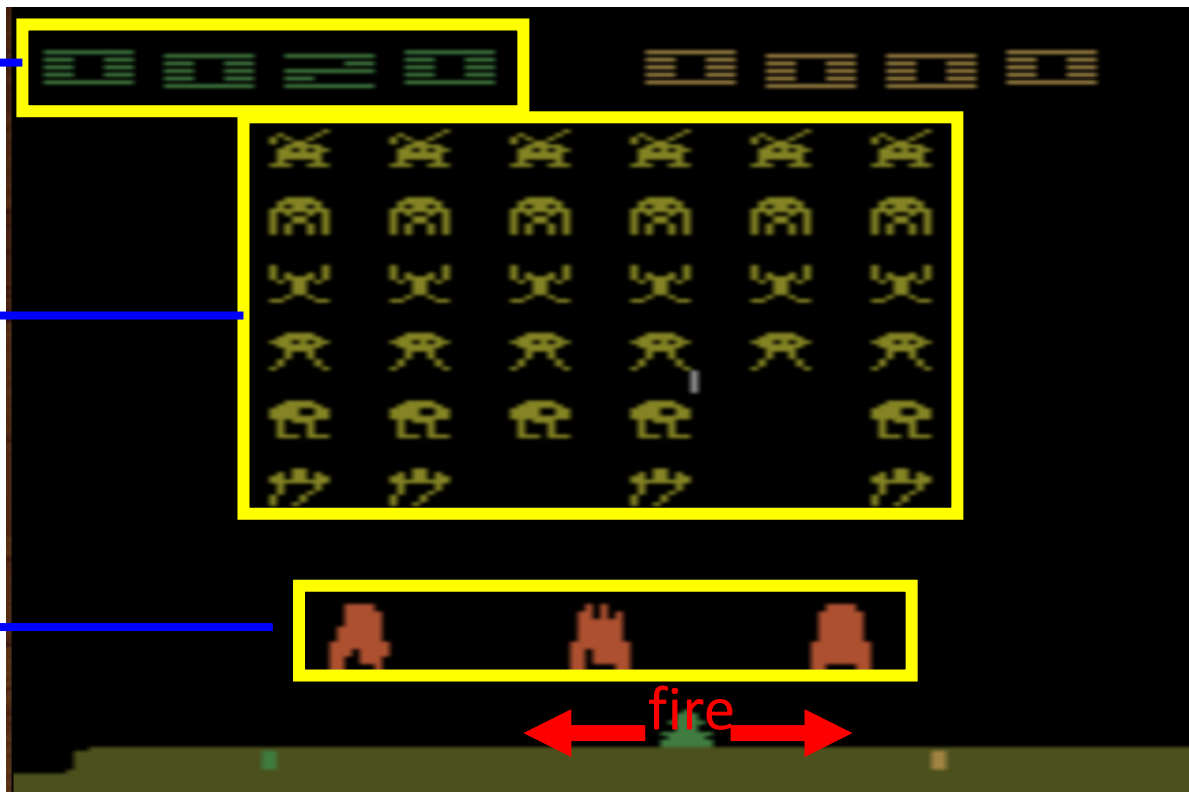
- 太空侵略者

游戏终止：所有的外星人被杀，
或太空船损坏。

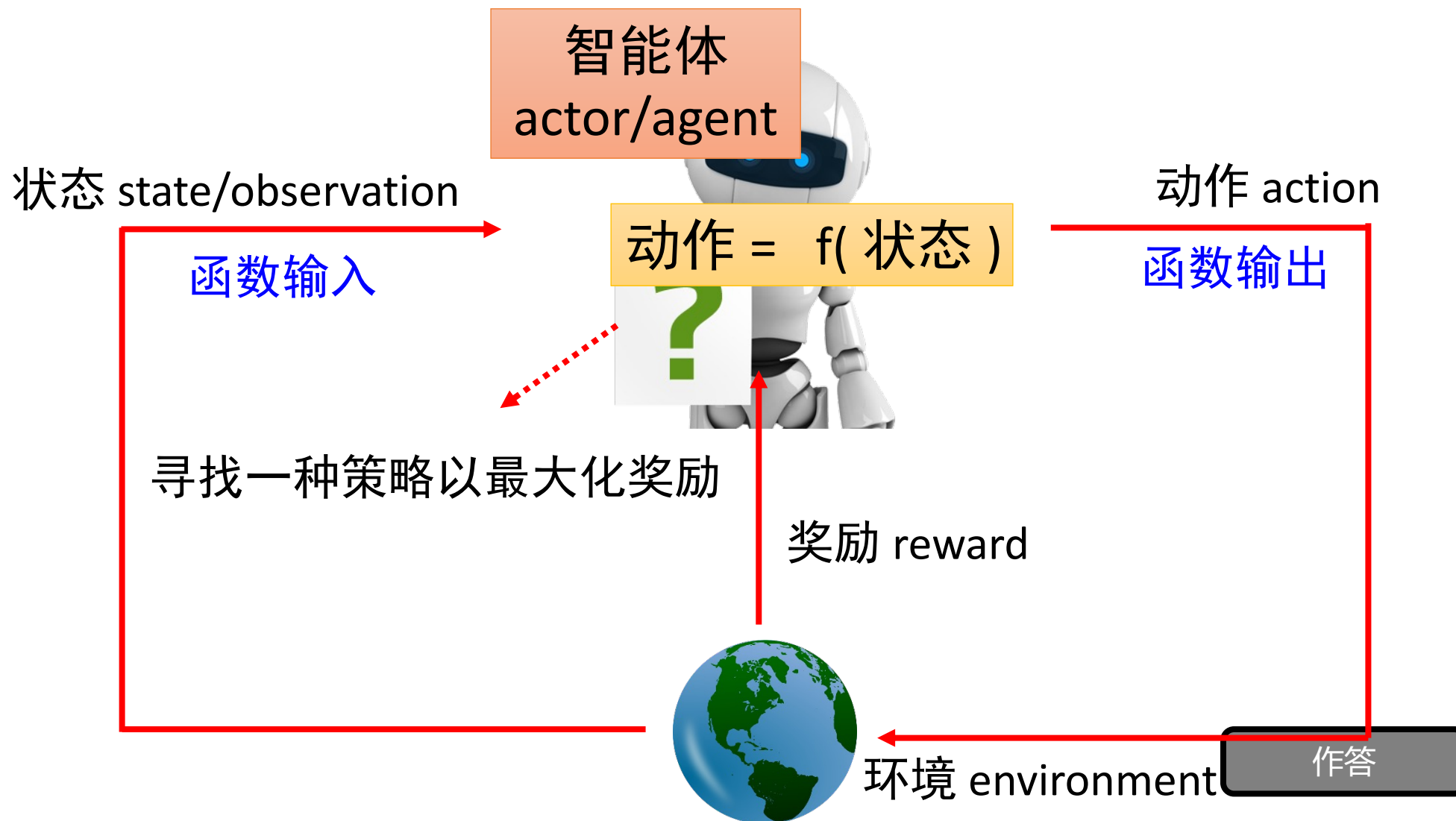
得分（奖励）←

杀死外星人←

防护盾←



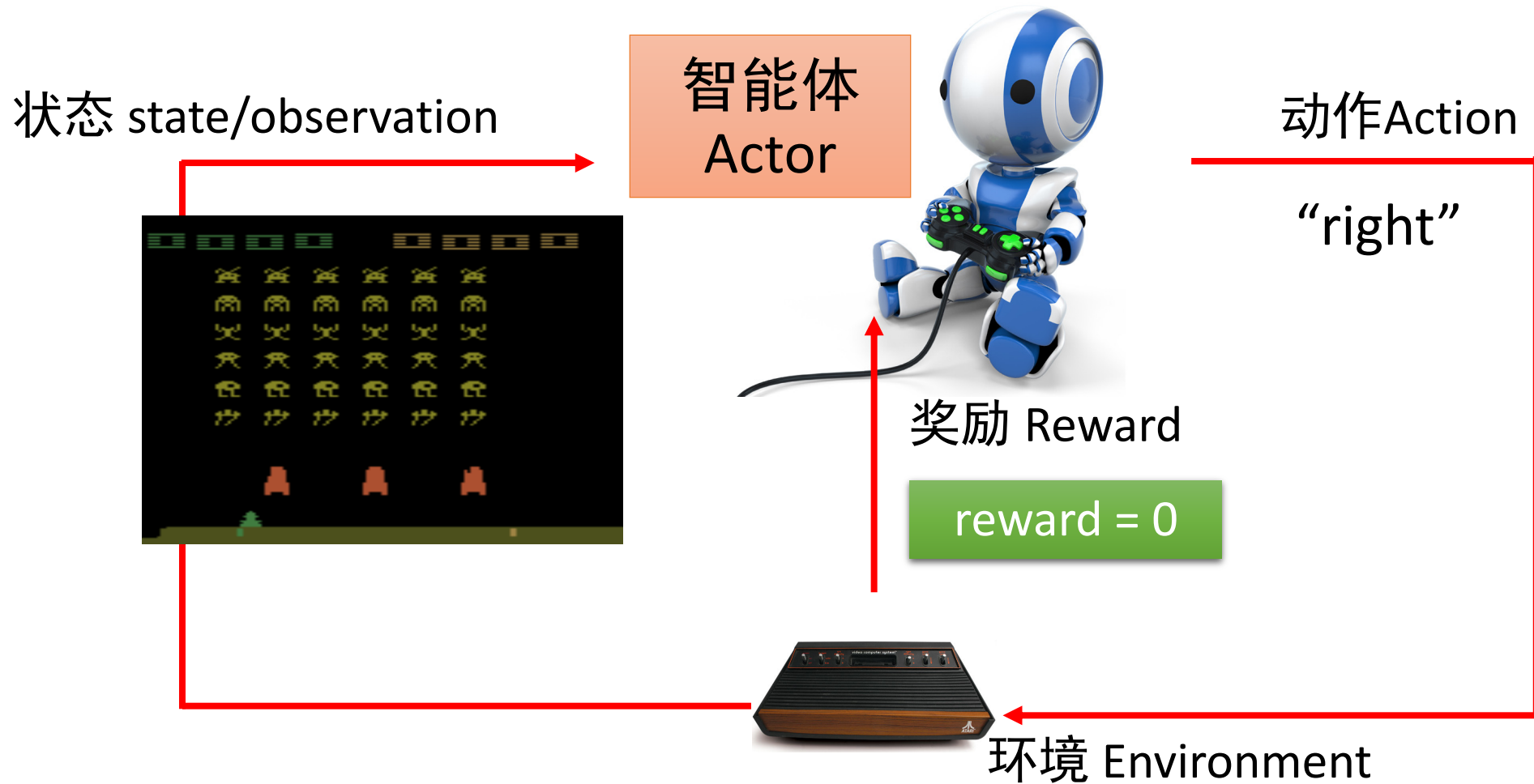
请思考，在太空侵略者游戏中，如何定义：状态、环境、动作、奖励？





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

例子：视频游戏

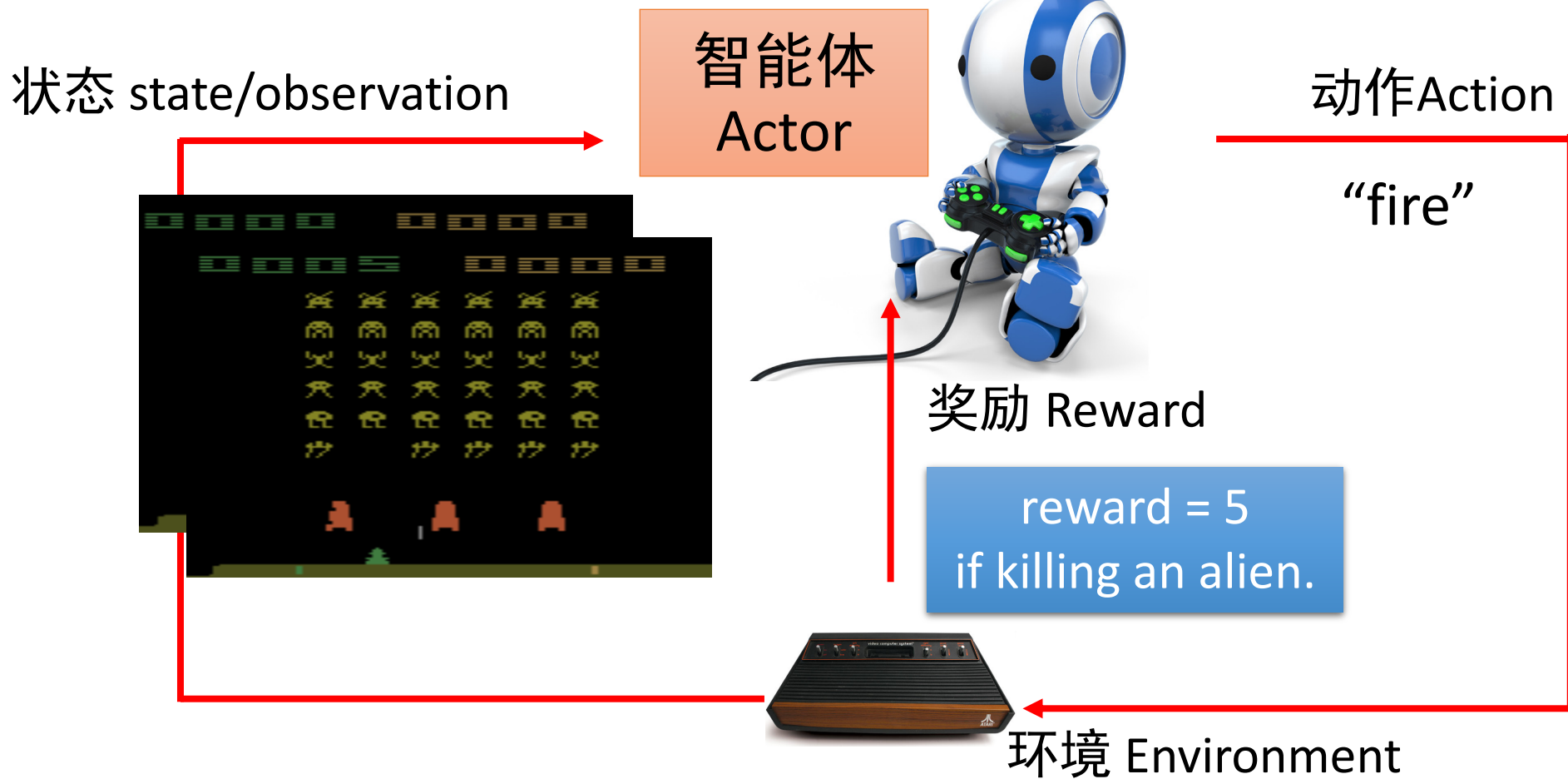




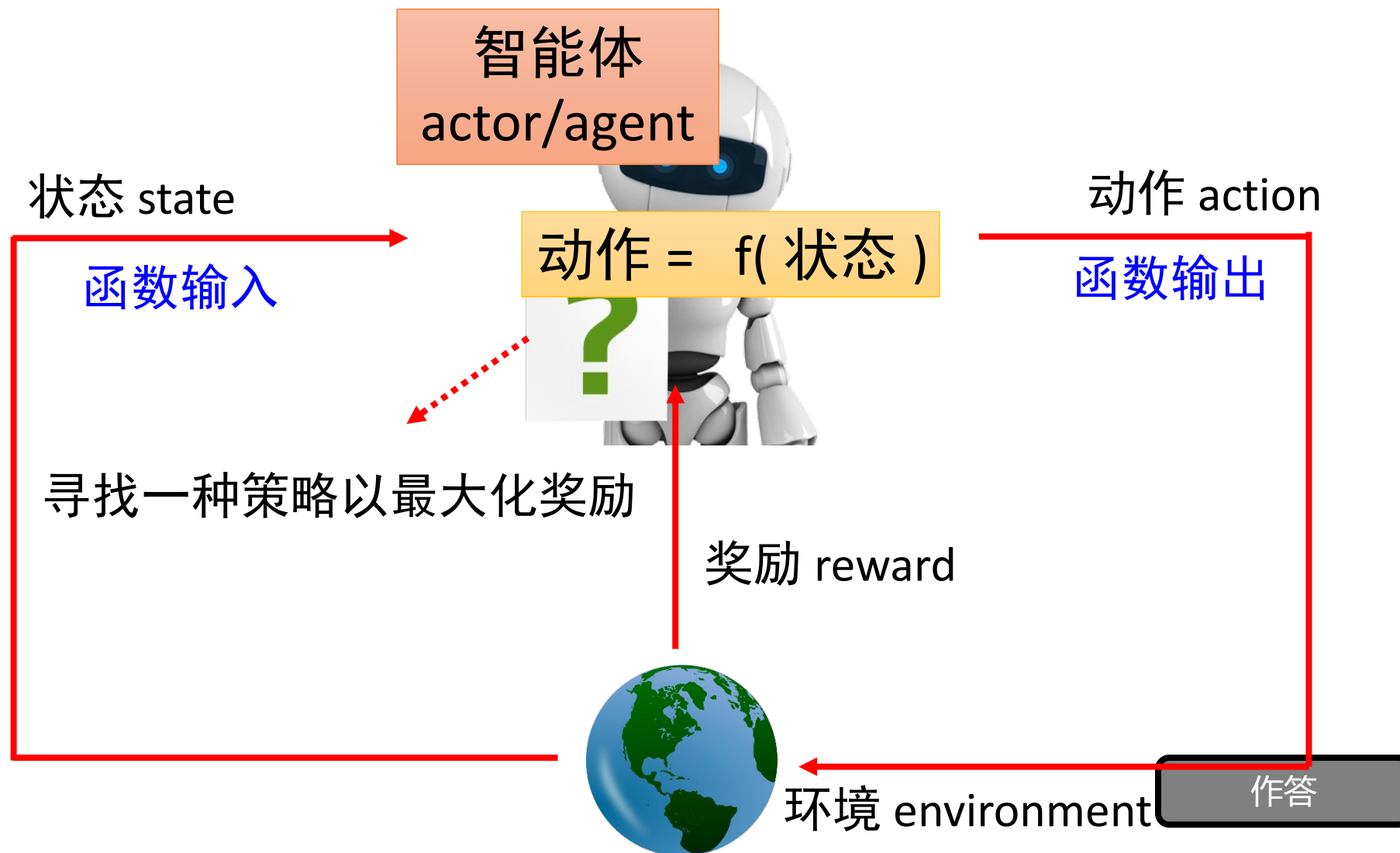
北京航空航天大学
COLLEGE OF SOFTWARE
BEIHANG UNIVERSITY 软件学院

例子：视频游戏

寻找能够最大化期望奖励的智能体



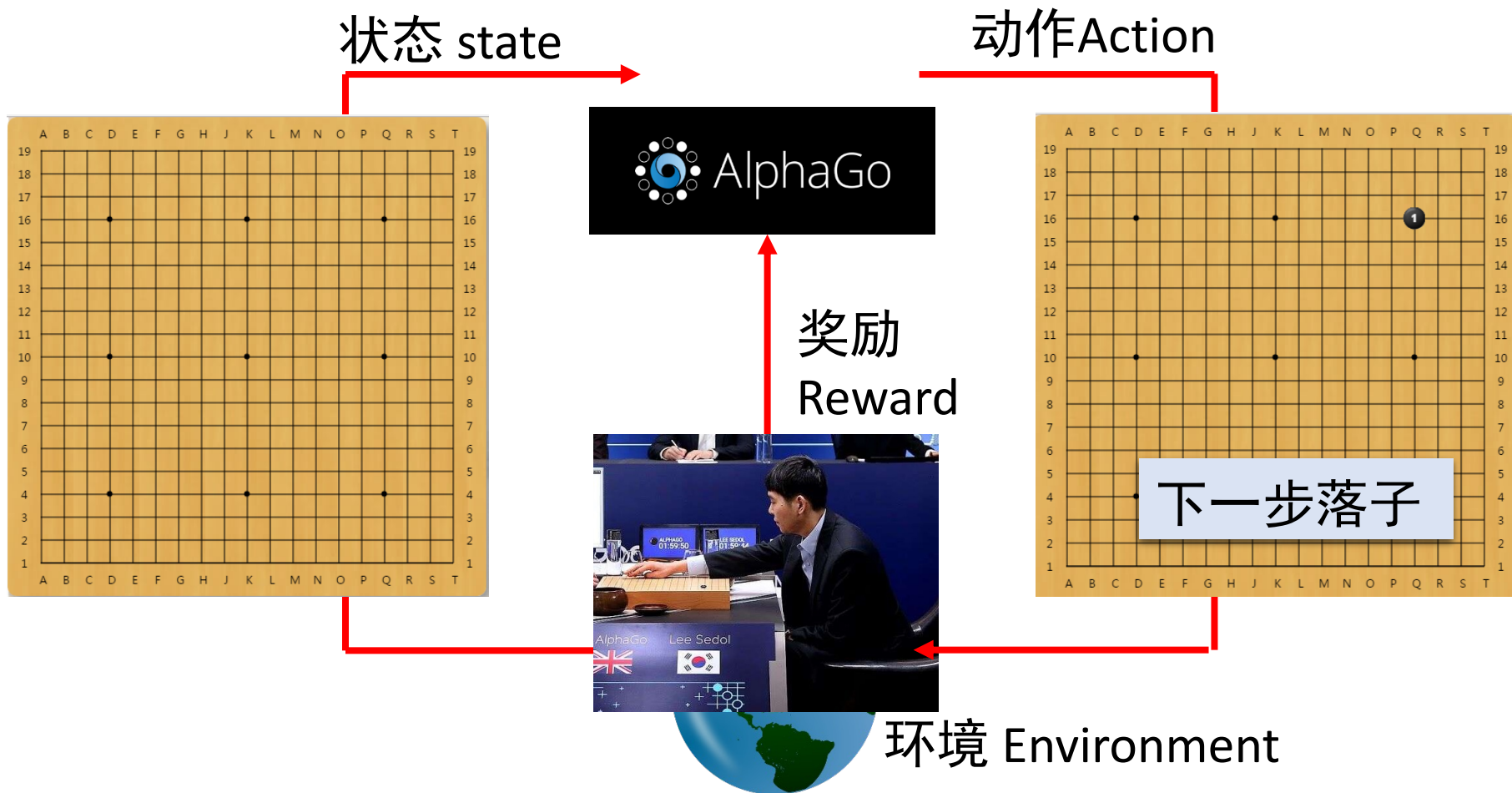
请思考，在围棋游戏中，如何定义：状态、环境、动作、奖励？





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

例子：围棋游戏 AlphaGO

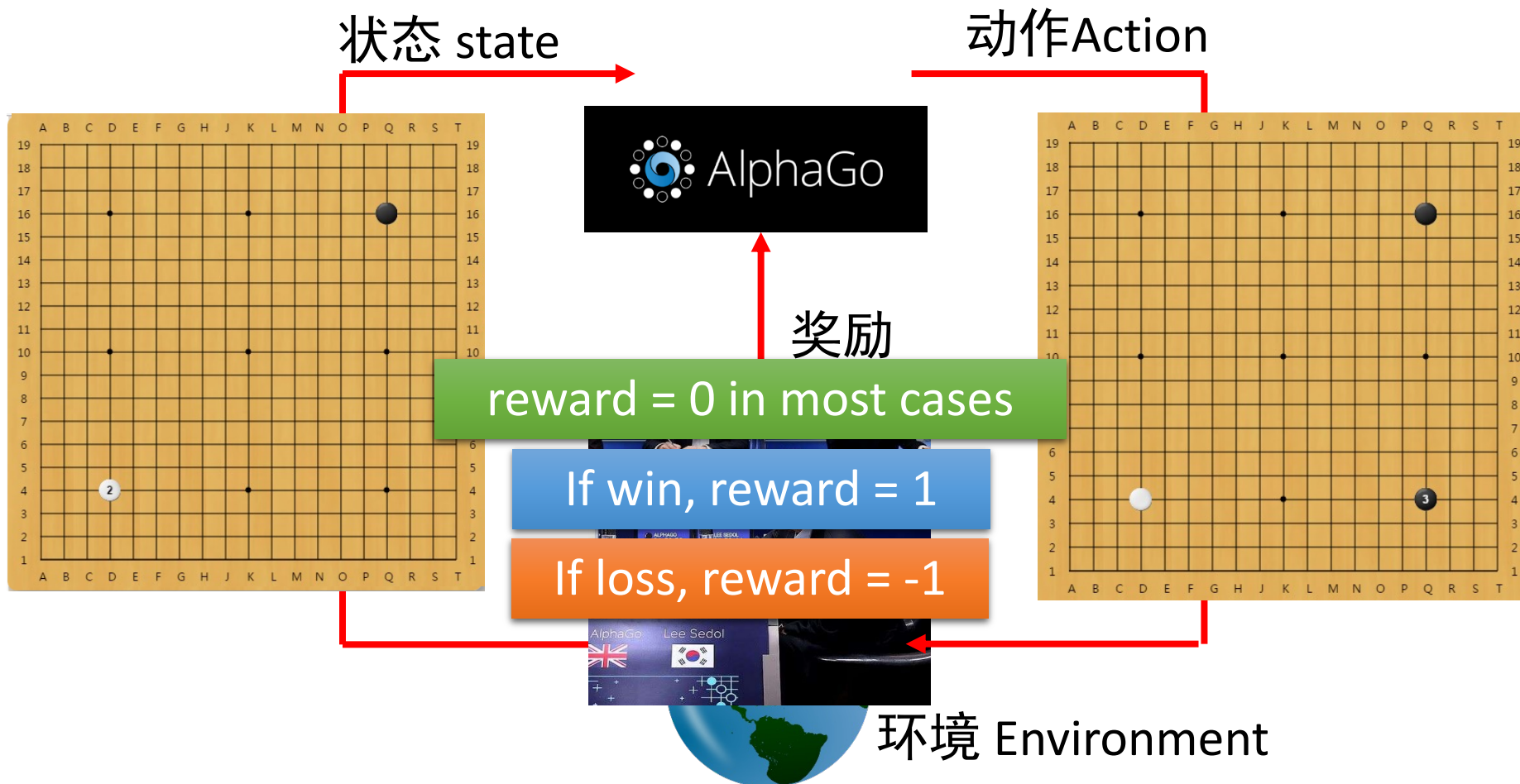




北京航空航天大学
COLLEGE OF SOFTWARE
BEIHANG UNIVERSITY 软件学院

例子：围棋游戏 AlphaGO

寻找能够最大化期望奖励的智能体



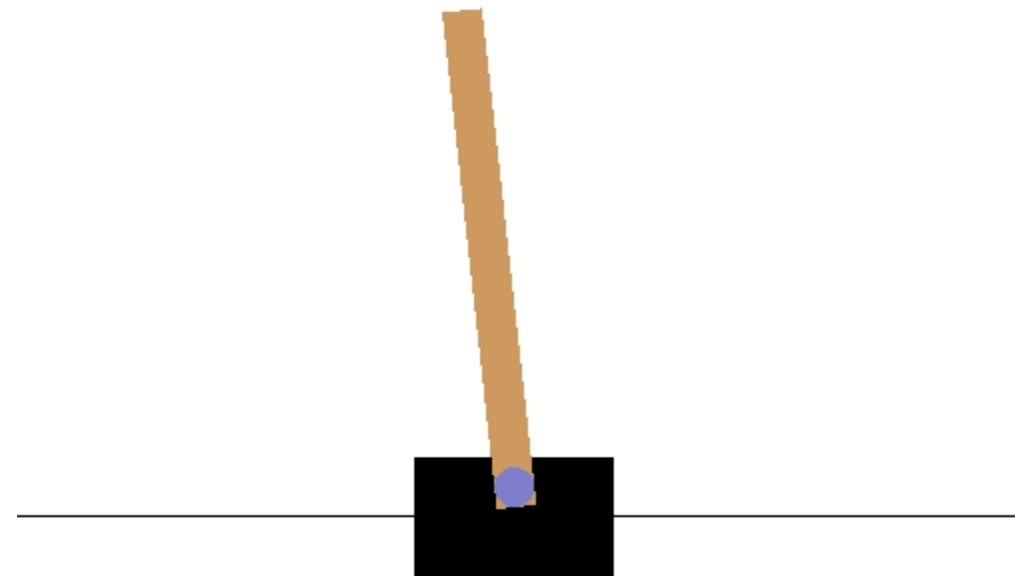


北京航空航天大学
COLLEGE OF SOFTWARE
BEIHANG UNIVERSITY 软件学院

例子：CartPole 推车杆游戏

• 游戏目标

- 通过左右移动小车，防止杆子倾斜超过一定角度（比如 $\pm 12^\circ$ ）或小车超出轨道范围（如 ± 2.4 单位长度）。
- 成功标准：连续平衡杆子 200 步（每步奖励+1，累计200分即通关）。





例子：CartPole 推车杆游戏

- **状态 (Observation)**

- 小车位置 (Cart Position) : $x \in [-4.8, 4.8]$
- 小车速度 (Cart Velocity) : $v \in (-\infty, \infty)$
- 杆子角度 (Pole Angle) : $\theta \in [-24^\circ, 24^\circ]$
- 杆子角速度 (Pole Angular Velocity) : $\omega \in (-\infty, \infty)$

- **动作 (Action)**

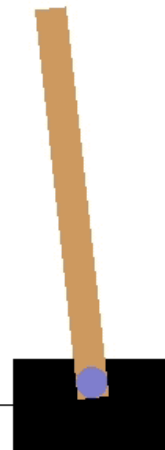
- 离散动作空间：0（向左推）或 1（向右推）

- **奖励 (Reward)**

- 每存活一步奖励 +1，包括终止步

- **终止条件**

- 杆子倾斜超限或小车越界；或达到200分





例子：CartPole 推车杆游戏

- **所有状态变量通过物理引擎相互耦合：**

- 动作（力） → 影响小车加速度 → 改变小车速度和位置。
- 小车加速度 → 通过惯性力矩影响杆子角加速度 → 改变杆子角度和角速度。
- 杆子角度 → 影响重力矩 → 进一步改变角速度。



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

例子：CartPole 推车杆游戏

寻找能够最大化期望奖励的智能体

状态 state/observation

智能体
Actor

动作 Action

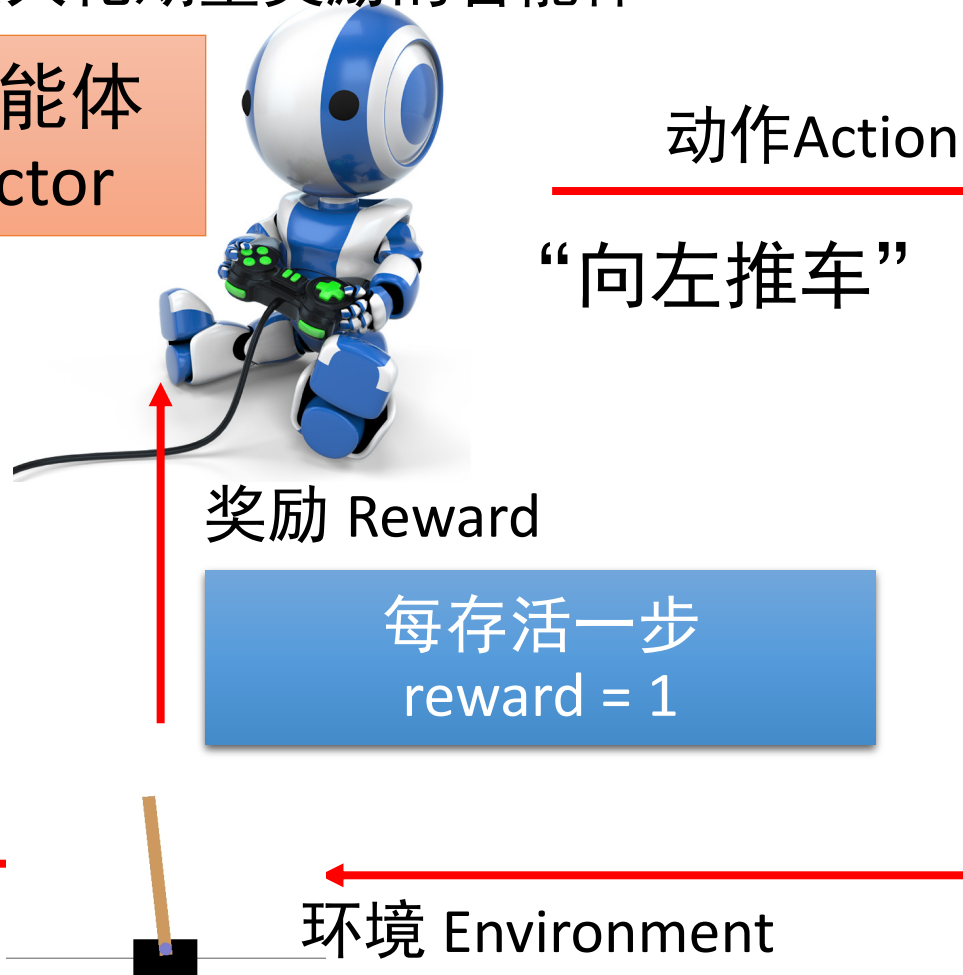
“向左推车”

小车位置 (Cart Position) : $x \in [-4.8, 4.8]$
小车速度 (Cart Velocity) : $v \in (-\infty, \infty)$
杆子角度 (Pole Angle) : $\theta \in [-24^\circ, 24^\circ]$
杆子角速度 (Pole Angular Velocity) : $\omega \in (-\infty, \infty)$

奖励 Reward

每存活一步
reward = 1

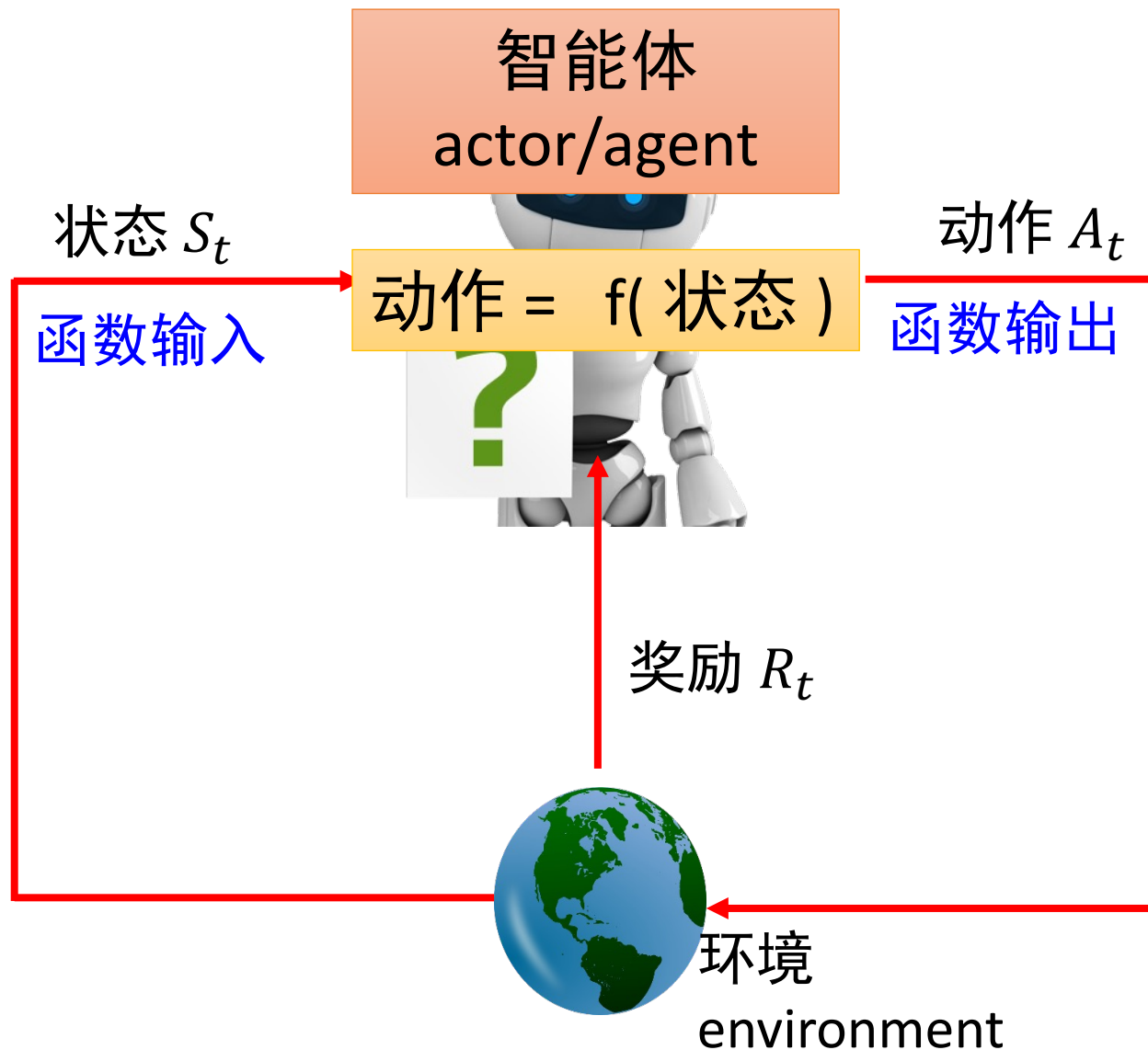
环境 Environment





强化学习交互过程

- 在每一步 t ,
- 智能体：
 - 获得状态 S_t
 - 获得奖励 R_t
 - 执行行动 A_t
- 环境：
 - 获得行动 A_t
 - 给出状态 S_{t+1}
 - 给出奖励 R_{t+1}
- t 在环境这一步增加





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

在与动态环境的交互中学习

有监督、无监督学习

Model ←



Fixed Data

强化学习

Agent ↔



Dynamic Environment

Agent不同, 交互出的数据也不同!



强化学习系统要素

- 状态 (State) → “你现在在哪？”
 - 环境的完整内部描述，包含所有决定未来动态的信息： S_t
- 策略 (Policy) → “你应该做什么？”
 - 是学习智能体在特定时间的行为方式
 - 是从状态到动作的映射
 - 确定性策略 (Deterministic Policy) :
$$a = \pi(s)$$
 - 随机策略 (Stochastic Policy) :

$$\pi(a|s) = P(A_t = a|S_t = s)$$



强化学习系统要素

- **奖励 (Reward)** → “你做这个动作，能得多少分？”
 - 一个定义强化学习目标的标量： $R(s, a)$
 - 能立即感知到什么是“好”的
- **回报 (Return)** → “某个时刻之后总共能得多少分？”
 - 回报即累积奖励，描述了一种长期的收益
 - 无折扣回报： $G_t := R_{t+1} + R_{t+2} + R_{t+3} + \dots$
 - 折后回报： $G_t := R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$



强化学习系统要素

• 价值函数 (Value Function)

- 状态价值是一个标量，用于定义对于长期来说什么是“好”的
- 价值函数是对于未来累积奖励的预测，用于评估在给定策略 π 时，某个状态 s 的好坏

$$\begin{aligned} V_{\pi}(s) &= \mathbb{E}_{\pi}[G_t | S_t = s] = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma V_{\pi}(s') | S_t = s] \end{aligned}$$



$V_{\pi}(s_1)$



$V_{\pi}(s_2)$



强化学习系统要素

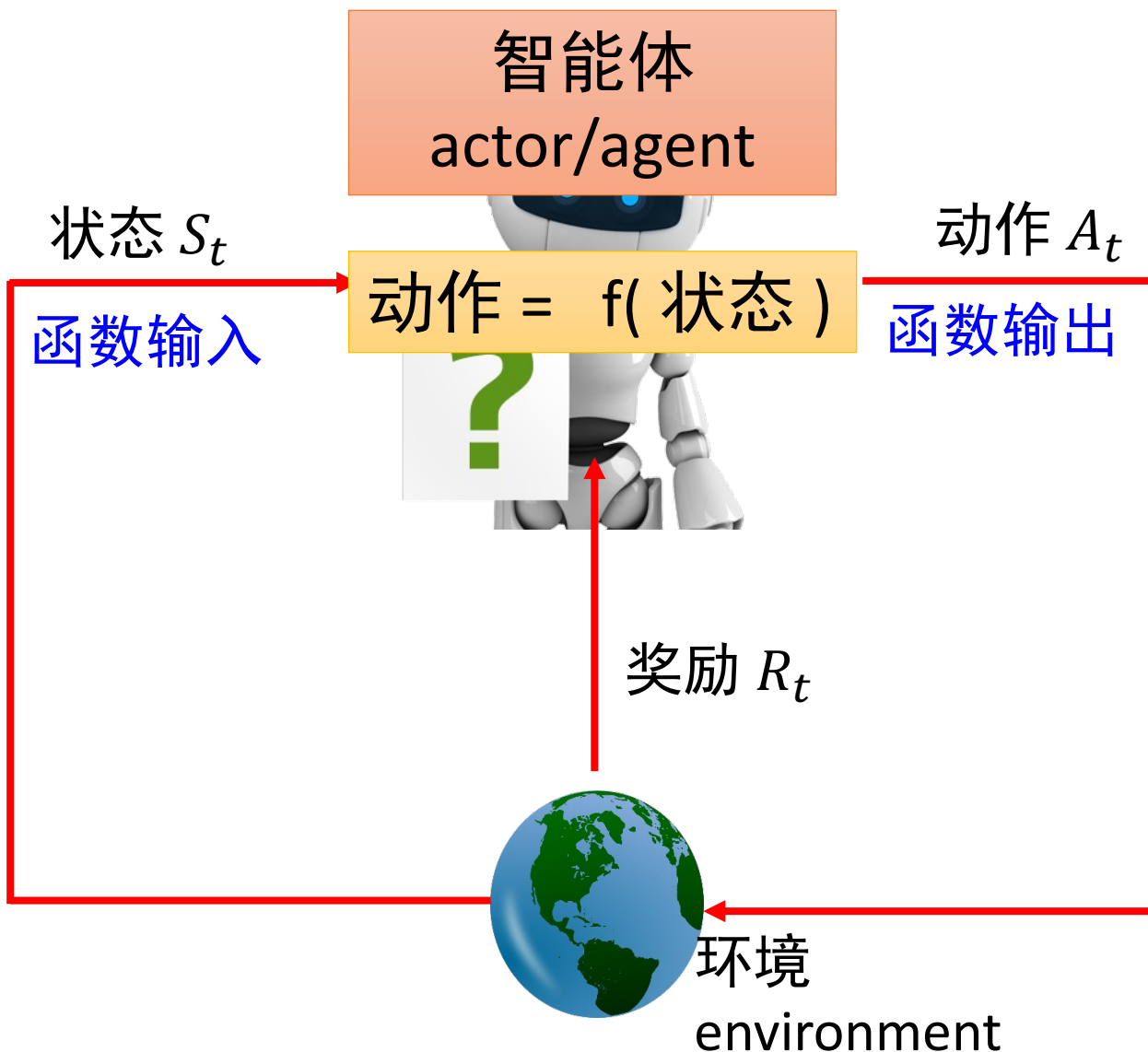
- 环境的模型 (Model) 用于模拟环境的行为

- 预测下一个状态

$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a]$$

- 预测下一个 (立即) 奖励

$$\mathcal{R}_s^a = \mathbb{E}[R_{t+1} | S_t = s, A_t = a]$$





马尔科夫过程

- 马尔可夫过程（Markov Process）是具有**马尔可夫性质**的随机过程

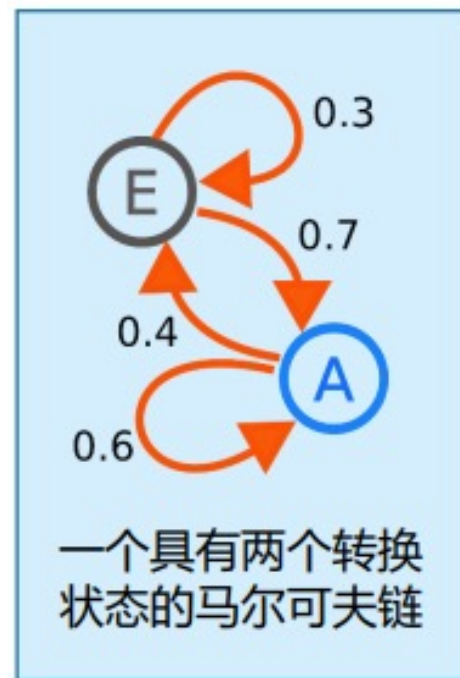
- 定义：

- 状态 S_t 是马尔可夫的，当且仅当

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

- 性质：

- 状态从历史（history）中捕获了所有相关信息
 - 当状态已知的时候，可以抛开历史不管
 - 也就是说，**当前状态是未来的充分统计量**





马尔科夫决策过程

- 马尔可夫决策过程（Markov Decision Process, MDP）
 - 为在结果部分随机、部分在决策者的控制下的决策过程建模的数学框架

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, \dots, S_t]$$

$$\mathbb{P}[S_{t+1}|S_t, A_t]$$

- MDP形式化地描述了一种强化学习的环境
 - 环境完全可观测
 - 即，当前状态可以完全表征过程（马尔可夫性质）



MDP五元组 $MDP = \{S, A, P_{sa}, \gamma, R\}$

- S 是状态的集合 \rightarrow “你现在在哪？”
 - 比如，视频游戏中的当前屏幕显示；CartPole游戏中的 (x, v, θ, ω)
- A 是动作的集合 \rightarrow “你能做什么？”
 - 比如，视频游戏中手柄操纵杆方向和按钮；CartPole游戏中 $A = \{\text{左}, \text{右}\}$
- P_{sa} 是状态转移概率 \rightarrow “你做了某个动作后，会发生什么？”
 - 对每个状态和动作， $P_{sa} = Pr(S_{t+1}|S_t, a_t)$ 是下一个状态在 S 中的概率分布
- $\gamma \in [0, 1]$ 是对未来奖励的折扣因子
- R 是奖励函数 \rightarrow “你做这个动作，能得多少分？”
 - $R(S_t, a_t, S_{t+1}): S \times A \mapsto \mathbb{R}$



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



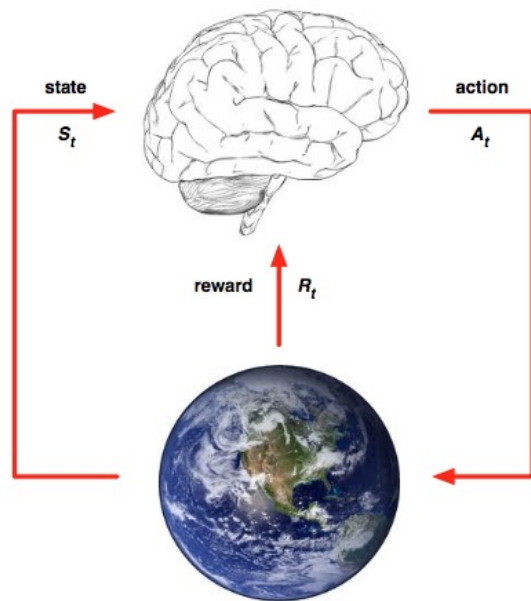
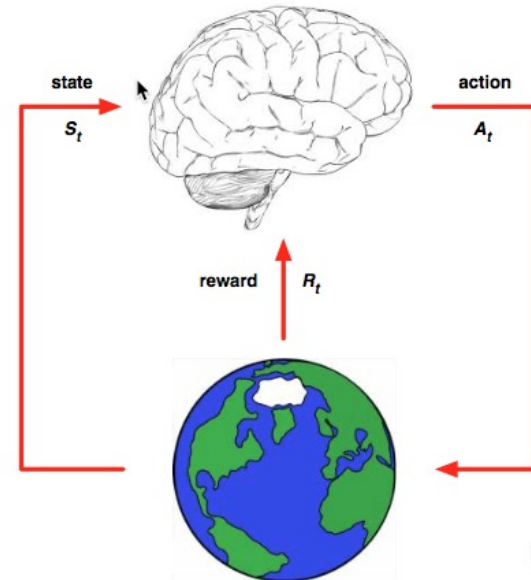
强化学习分类

- 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)

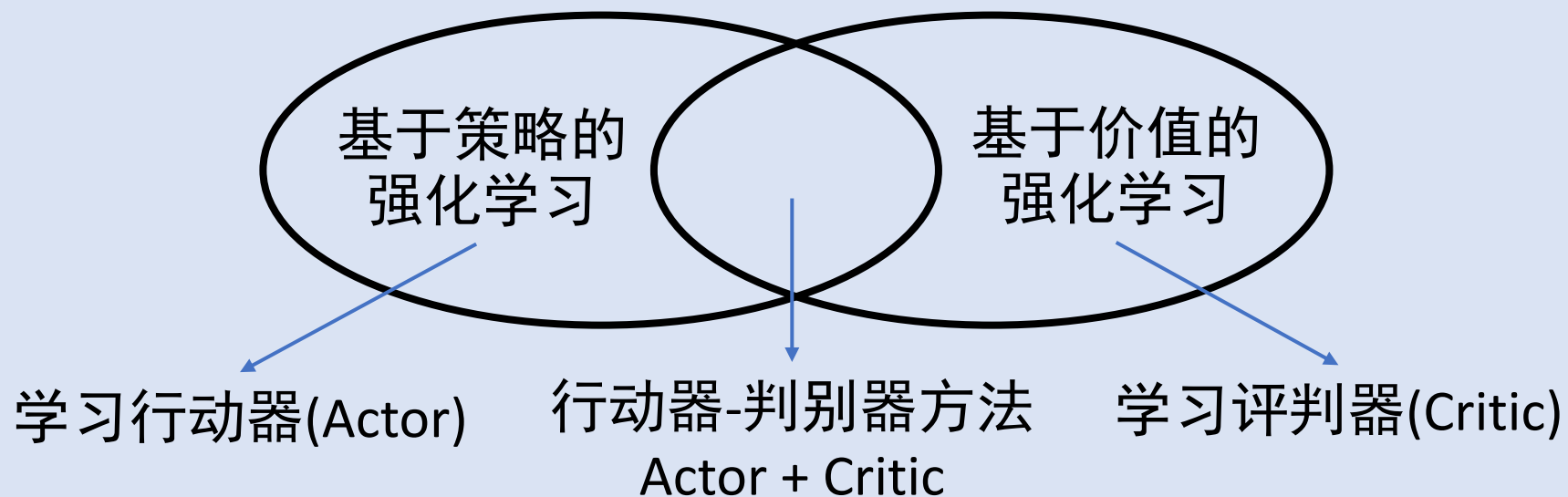
- 显式学习：价值函数或（和）策略函数
- 无环境模型





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

Sora，世界模型？

何以AI



- 人类的生物特征
- 丰富的人文环境
- 文化的基本形态
- 人与世界的互动
- 人类行走的姿态
- 社会的运行方式



对世界的细致理解

➤ 早在 2016 年，OpenAI 就明确提出**生成模型**是让计算机理解世界最有潜力的方向。



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



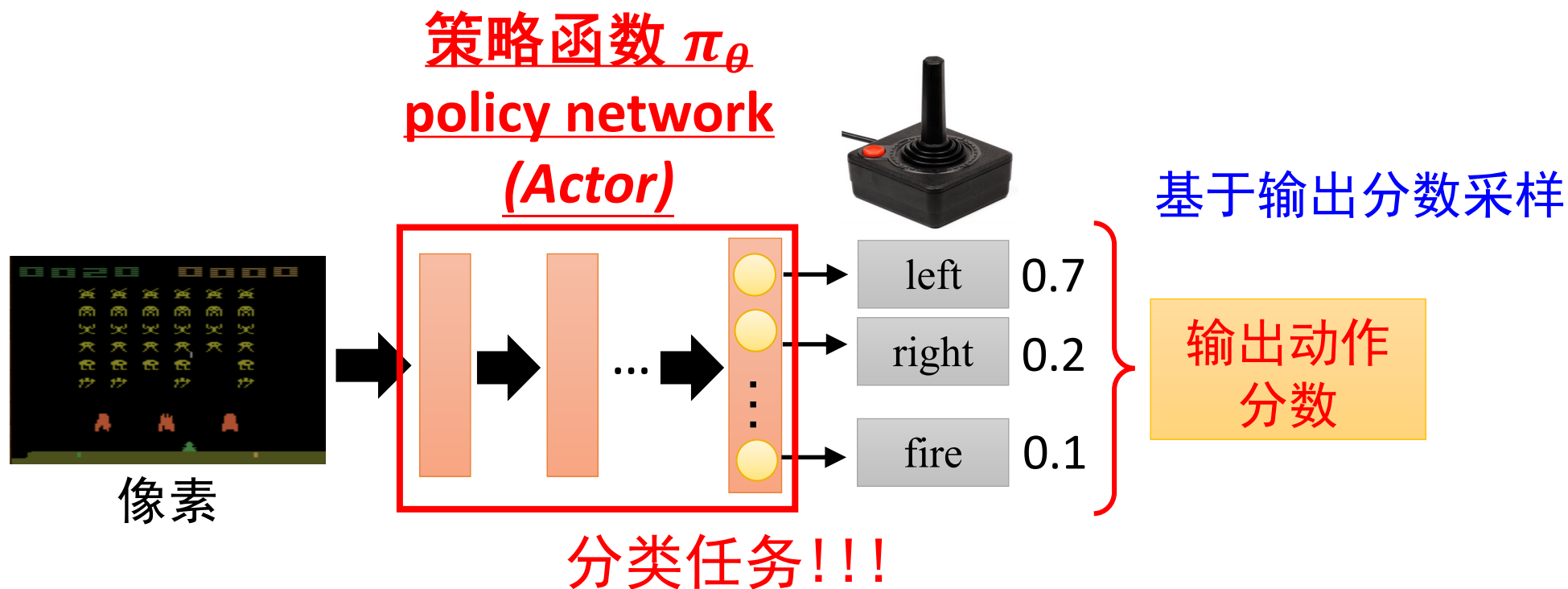
步骤2: 从训
练数据中定
义损失函数



步骤3: 优化



步骤 1: 定义带有未知参数的函数

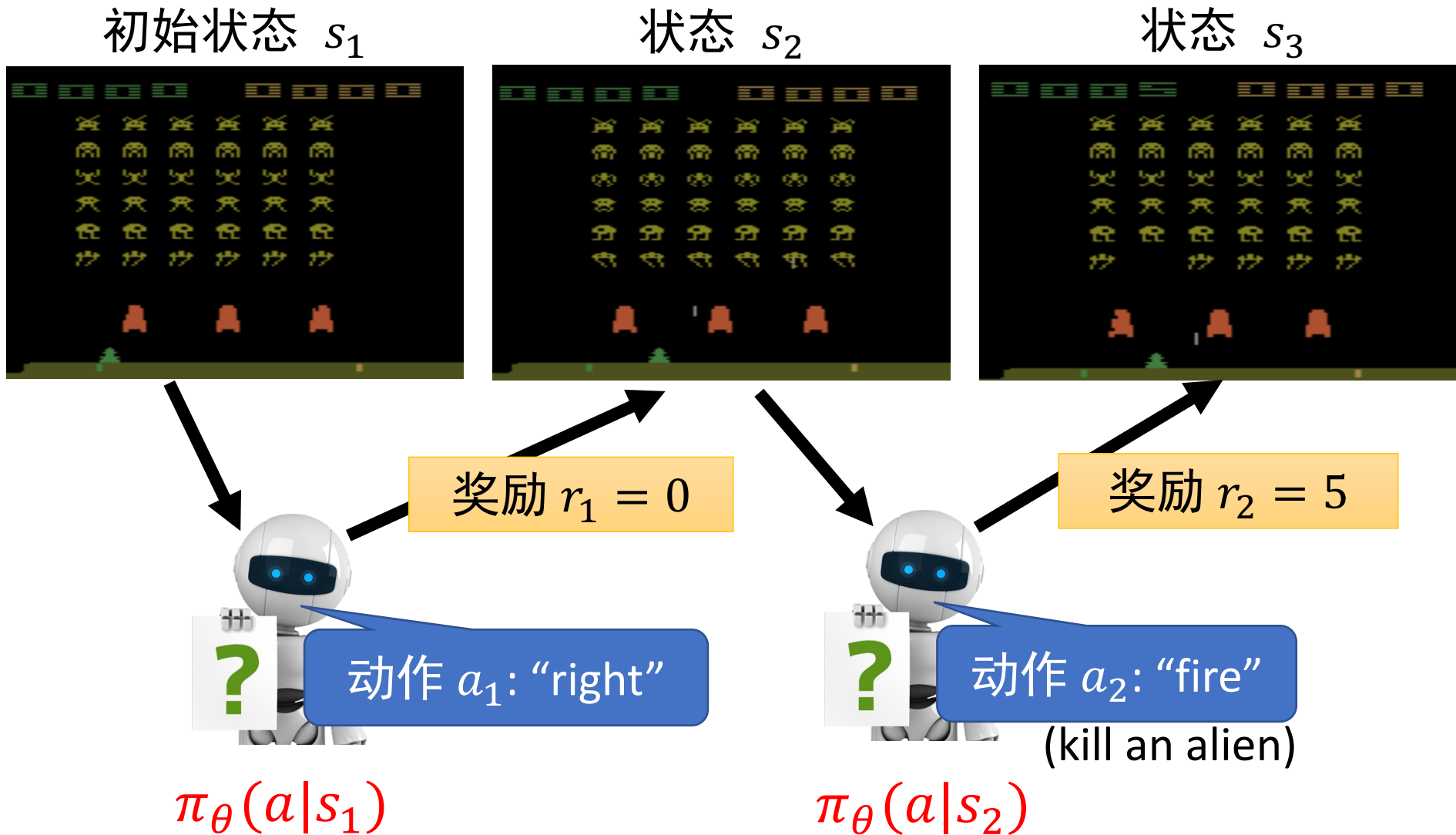


- 神经网络的输入：向量或矩阵形式的状态数据
- 神经网络的输出：每个神经元对应的动作（基于输出分数采样）



步骤 2: 定义损失函数

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





步骤 2: 定义损失函数

初始状态 s_1



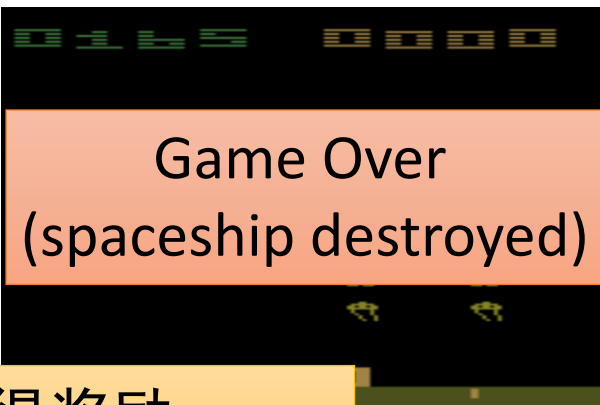
状态 s_2



状态 s_3



After many turns



获得奖励 r_T

动作 a_T

This is an episode (片段).

累积奖励 (回报):

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

最大化回报



步骤 2: 定义损失函数

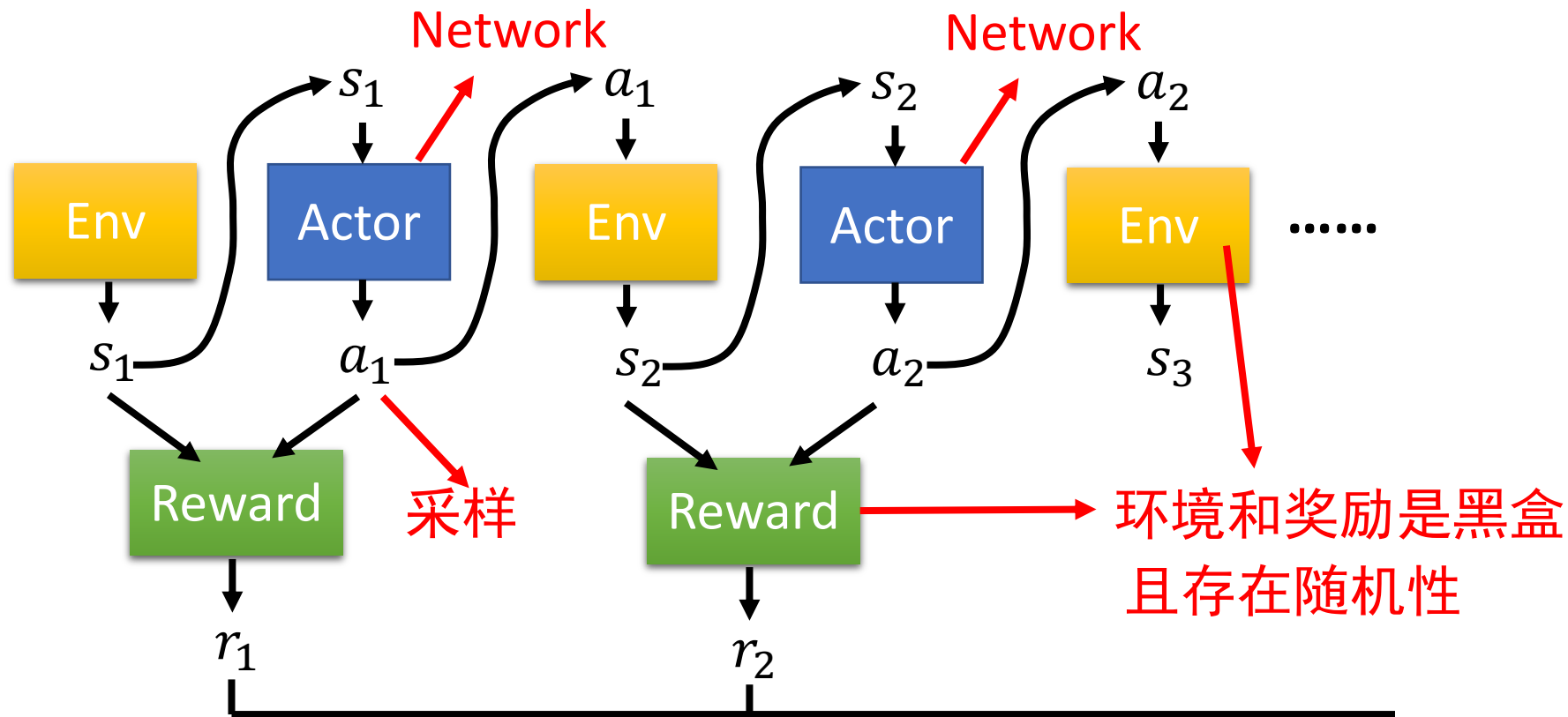
- 一个episode（片段）可以得到一个轨迹 $\tau = \{s_1, a_1, s_2, a_2, \dots\}$
- 一个轨迹的回报（累积奖励）： $G(\tau) = \sum_{t=1}^T r_t$
- 当前的Actor产生每个轨迹的概率为： $p_{\theta}(\tau)$
- 目标函数：最大化回报 G 的期望。用当前的Actor与环境交互多个 episodes，得到 M 个轨迹，近似回报的期望：

$$\overline{G_{\theta}} = \sum_{\tau} G(\tau) p_{\theta}(\tau)$$




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

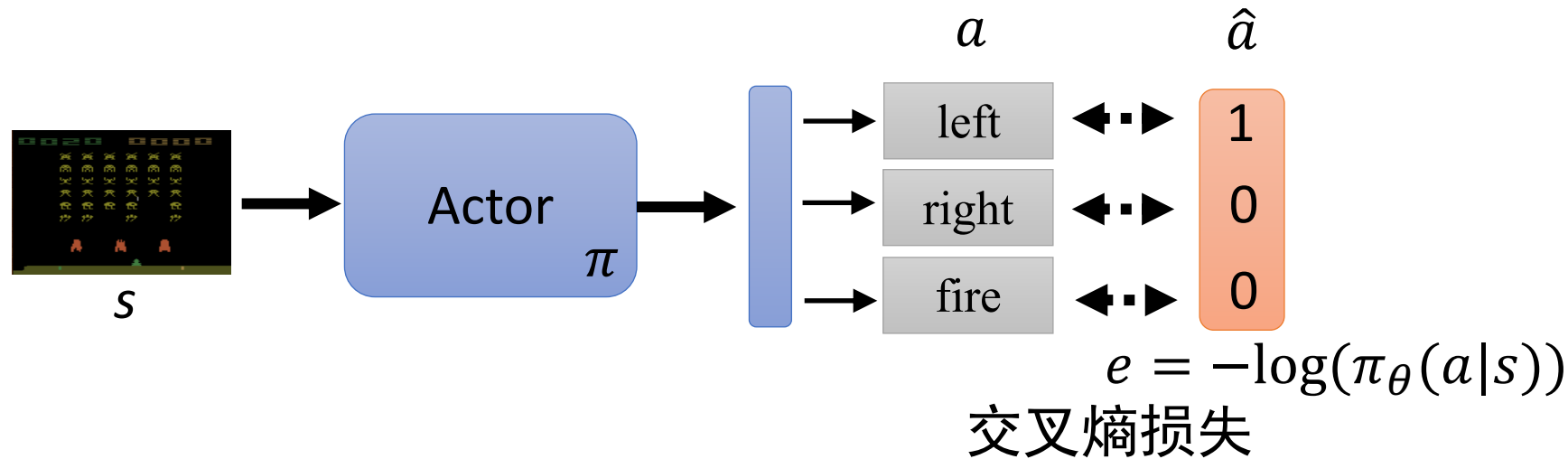
c.f. GAN

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




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} : $L = e$

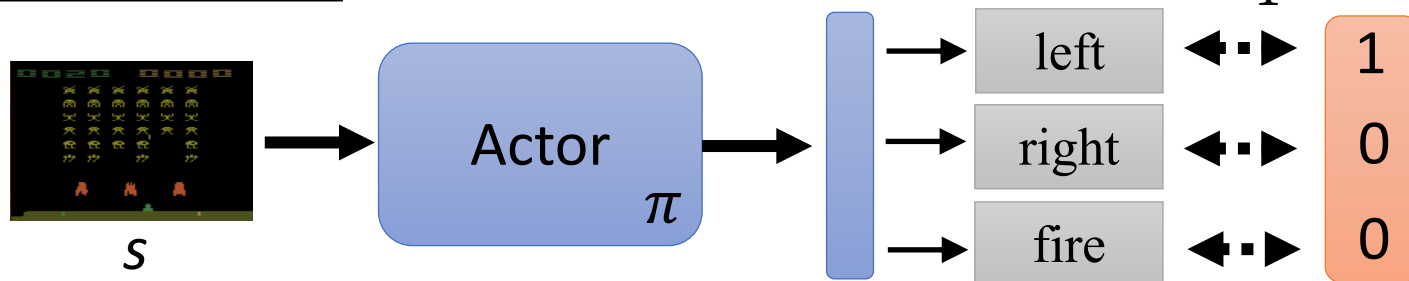
不执行动作 \hat{a} : $L = -e$

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

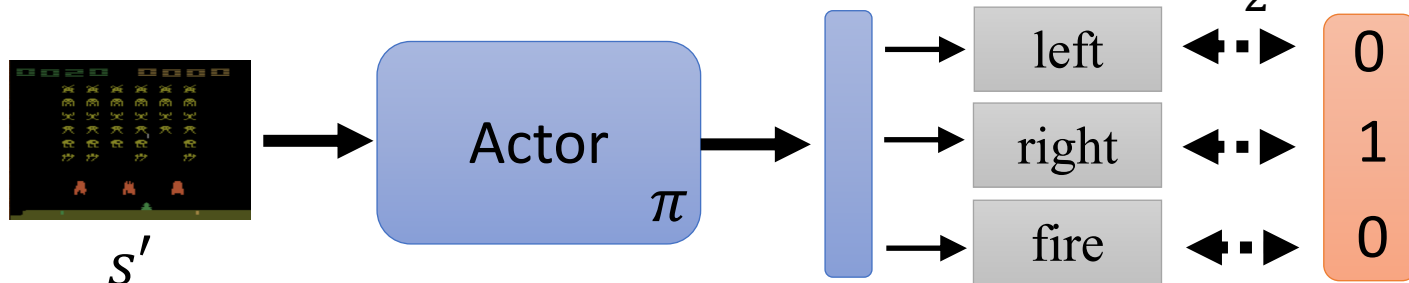


如何学习策略（学习actor）

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	
$\{s_T, \hat{a}_T\}$	-1	No



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

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



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	
$\{s_T, \hat{a}_T\}$	A_T	-10
?	?	

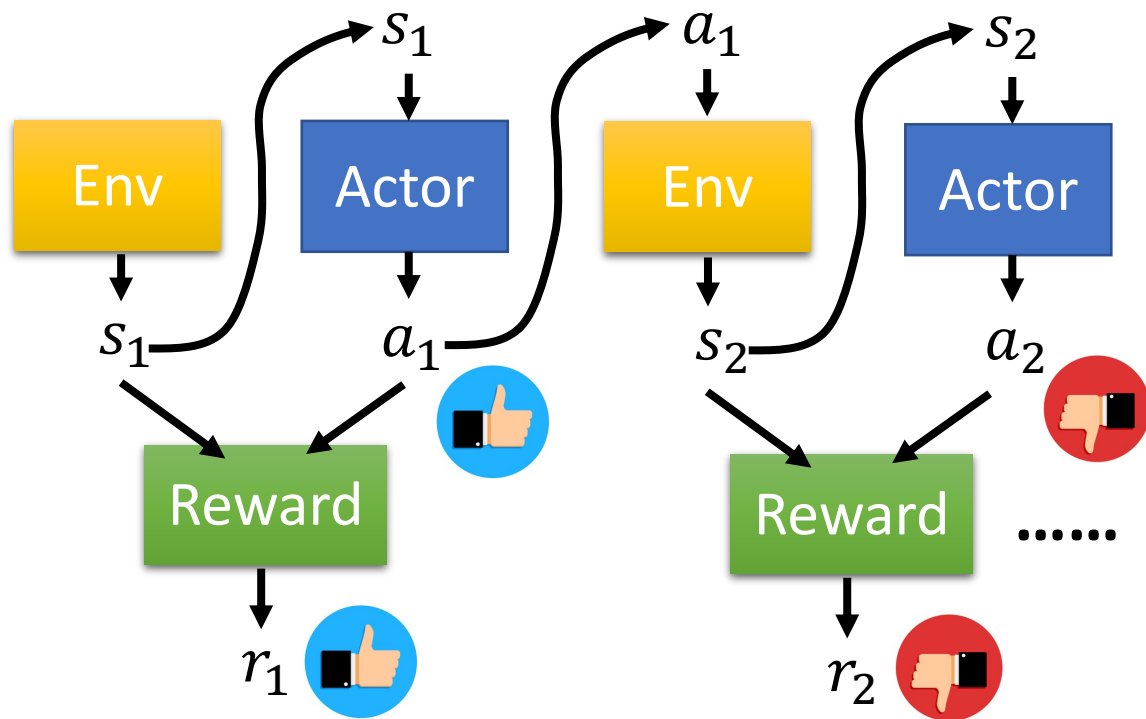


$$L = \sum A_t e_t$$

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



版本 0



Training Data

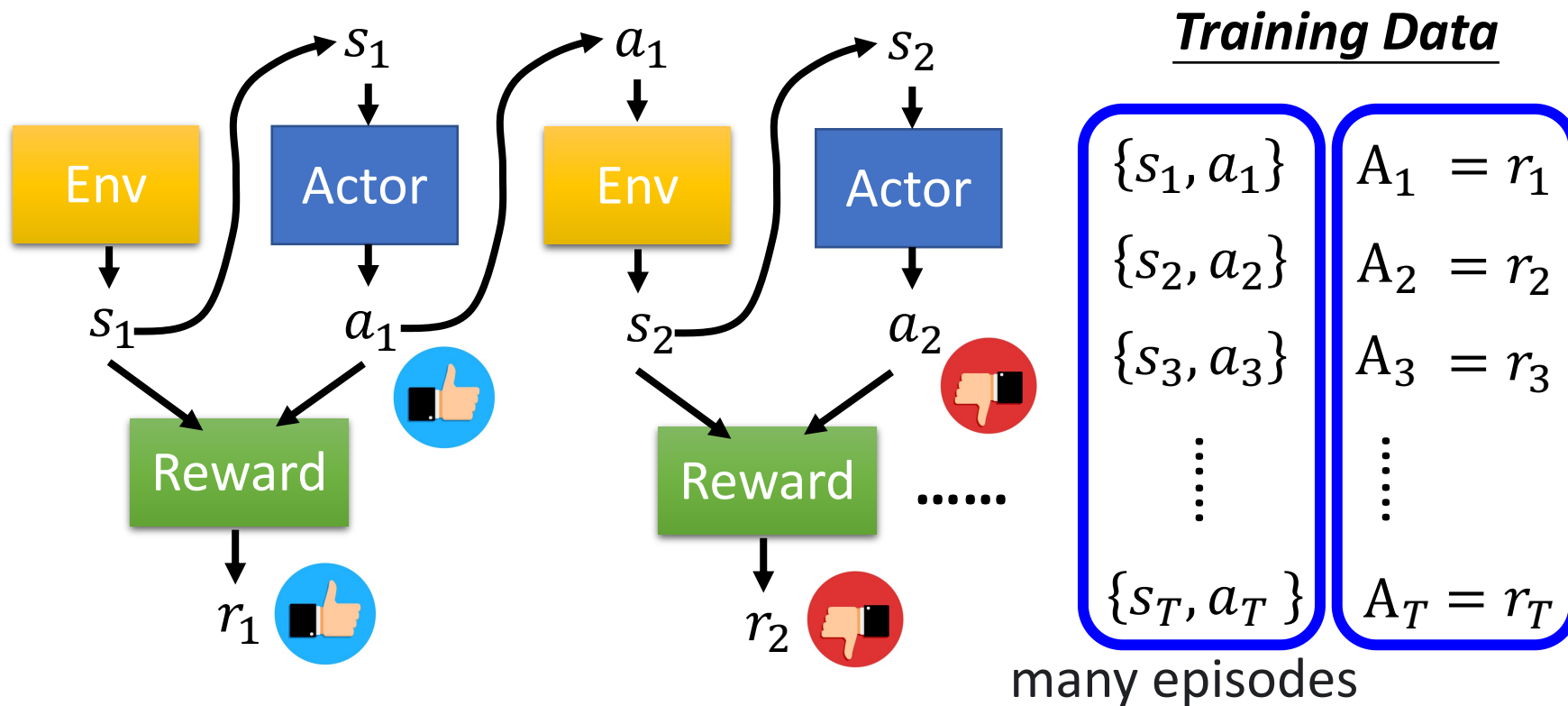
$\{s_1, a_1\}$	$A_1 = r_1$
$\{s_2, a_2\}$	$A_2 = r_2$
$\{s_3, a_3\}$	$A_3 = r_3$
\vdots	\vdots
$\{s_T, a_T\}$	$A_T = r_T$

many episodes



版本 0

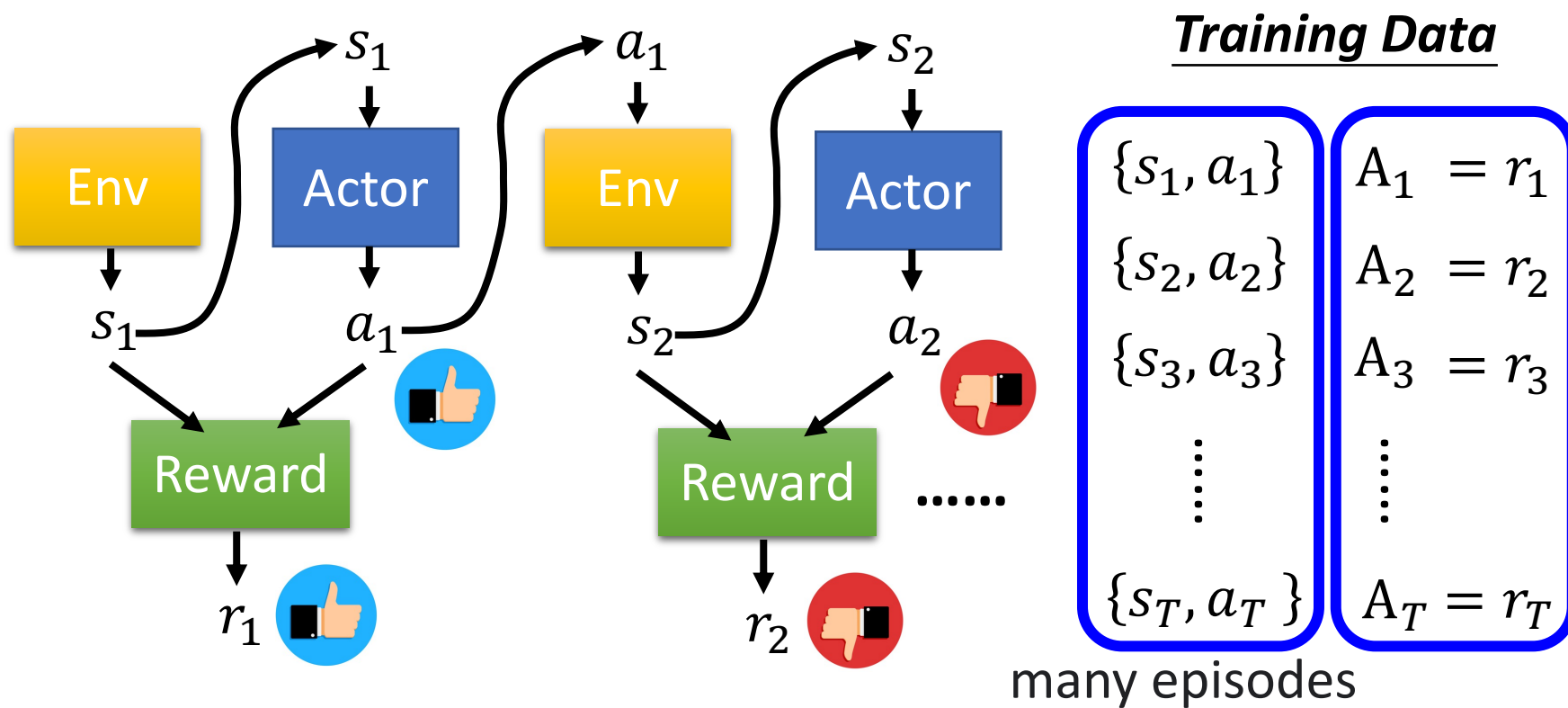
Trajectory (轨迹) $\tau = \{s_1, a_1, s_2, a_2, \dots\}$



$$L = \sum_{t=1}^T A_t e_t = - \sum_{t=1}^T A_t \log(\pi_{\theta}(a_t | s_t)) \quad \pi^* = \arg \min_{\pi} \frac{1}{N} \sum_{n=1}^N L$$

N 为episodes 个数

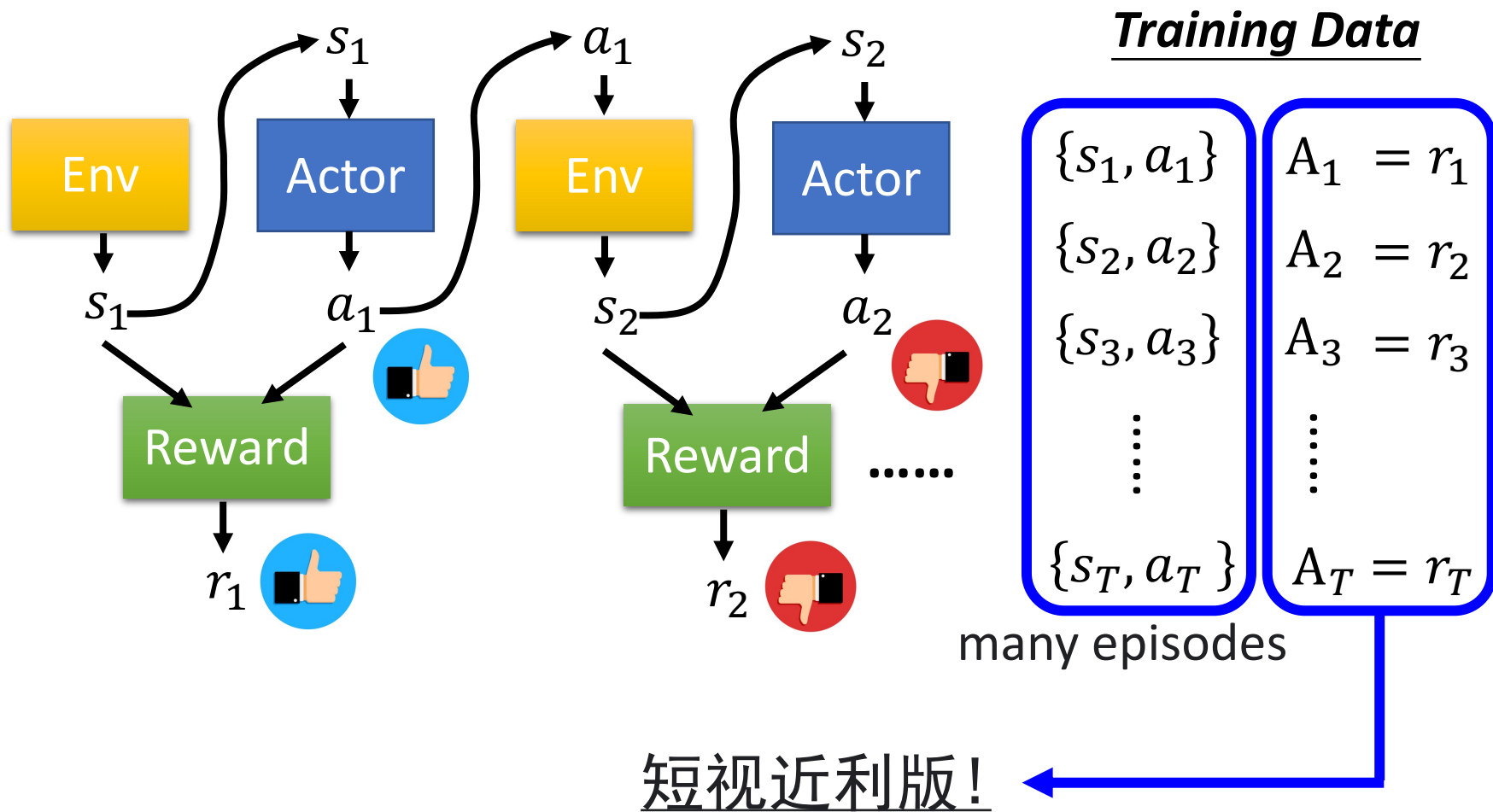
版本0存在什么问题？



作答

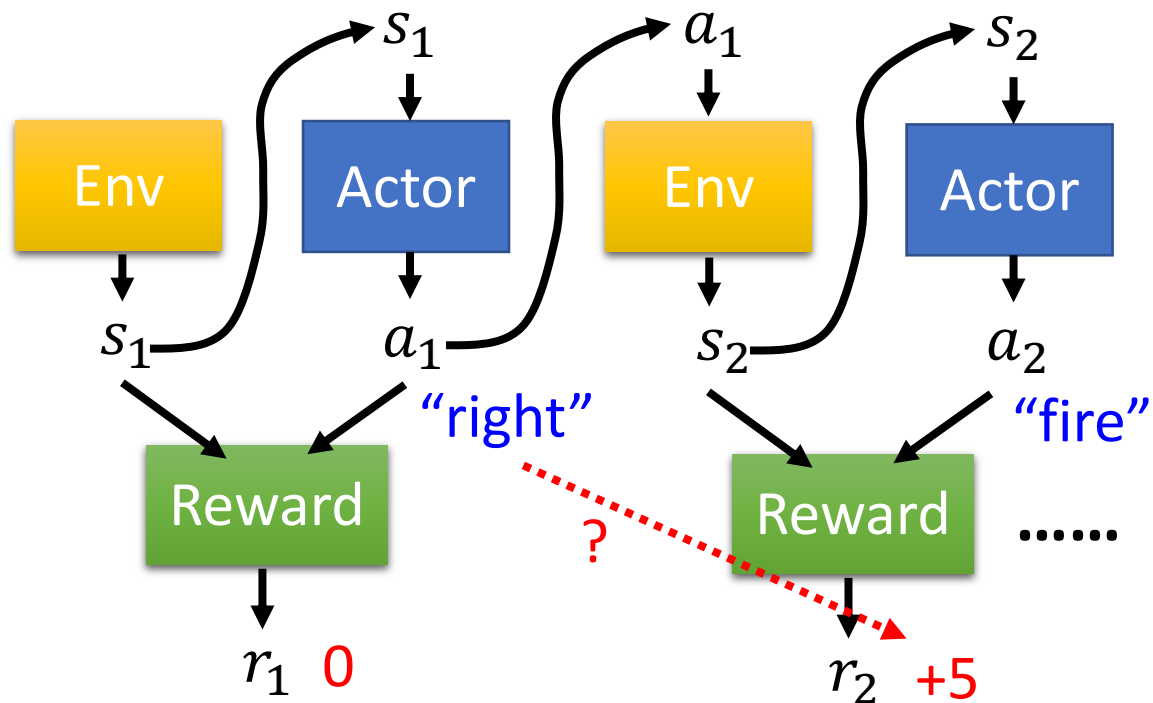


版本 0





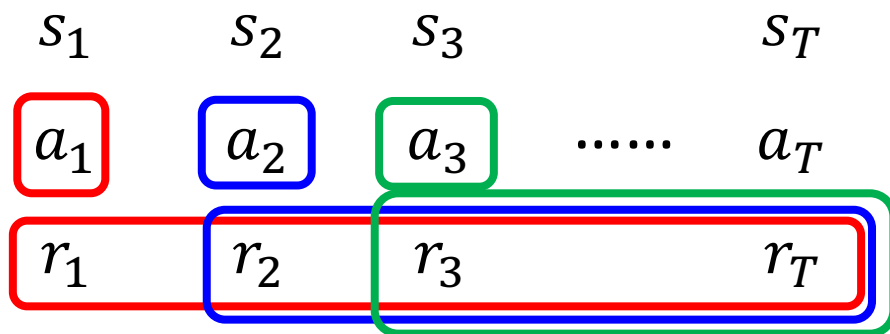
版本 0



- 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



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

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

$$G_3 = r_3 + \dots + r_T$$

累积奖励 (cumulated reward)

Training Data

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

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

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

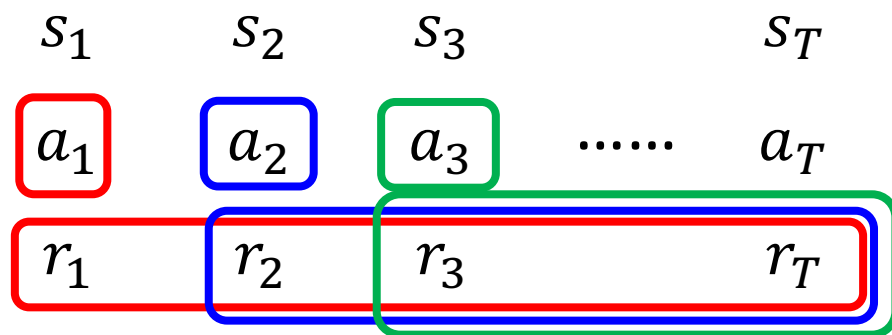
\vdots

\vdots

$$\{s_T, a_T\} \quad A_T = G_T$$

$$G_t = \sum_{t'=t}^T r_{t'}$$

版本1存在什么问题?



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

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

$$G_3 = r_3 + \dots + r_T$$

累积奖励 (cumulated reward)

Training Data

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

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

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

⋮

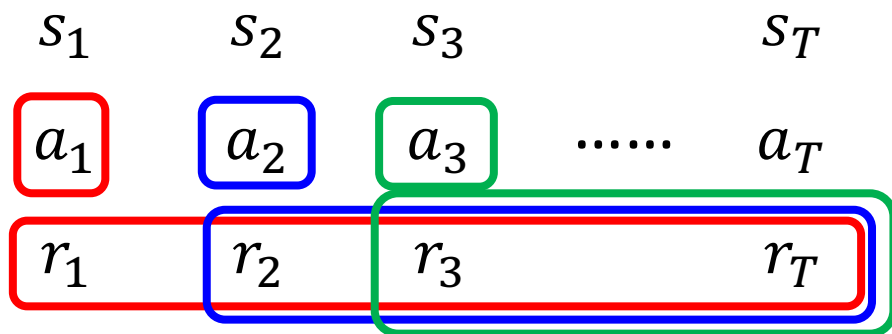
$$\{s_T, a_T\} \quad A_T = G_T$$

$$G_t = \sum_{t'=t}^T r_{t'}$$

作答



版本 2: REINFORCE算法的核心



Also the credit of a_1 ?

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

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

折扣因子 (Discount factor) $\gamma < 1$

Training Data

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

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

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

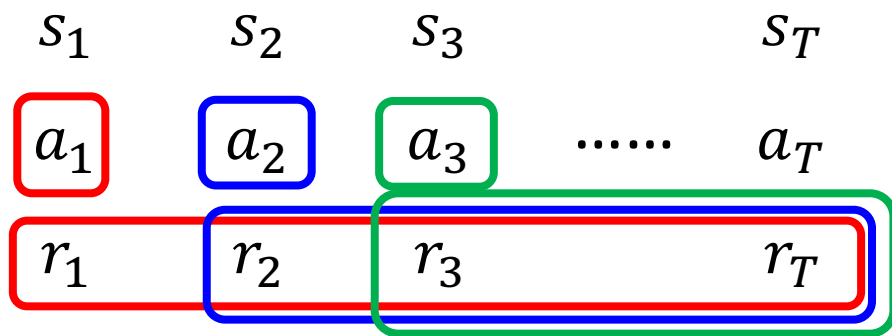
\vdots

\vdots

$$\{s_T, a_T\} \quad A_T = G'_T$$

$$G'_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$$

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

优势函数
advantage function

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

$$\{s_T, a_T\} \quad A_T = G'_T - b$$

$$G'_t = \sum_{t'=t}^T \gamma^{t'-t} r_{t'}$$



策略梯度 policy gradient

- 策略梯度的目标函数即先对每个轨迹进行目标函数的计算，再进行多个轨迹的采样，得到回报的期望：

$$L(\tau) = \sum_{t=1}^T A_t e_t = - \sum_{t=1}^T A_t^\theta \log(\pi_\theta(a_t | s_t))$$

$$\pi^* = \arg \min_{\pi} E_{\tau \sim \pi_\theta(\tau)} L(\tau) \approx \arg \min_{\pi} \frac{1}{N} \sum_{n=1}^N L$$

- 即： $L(\theta) = -E_{\tau \sim \pi_\theta(\tau)} \left[\sum_{t=1}^T A_t^\theta \log(\pi_\theta(a_t | s_t)) \right]$



策略梯度 policy gradient

- $L(\theta) = -E_{\tau \sim \pi_{\theta}(\tau)} [\sum_{t=1}^T A_t^{\theta} \log(\pi_{\theta}(a_t | s_t))]$
- 然而，由于实际在做策略梯度时，并不是给整条轨迹 τ 一样的分数，而是将每个状态-动作分开计算。实际的目标函数可以写为

$$L(\theta) = -E_{(s_t, a_t) \sim \pi_{\theta}} A_t^{\theta} \log(\pi_{\theta}(a_t | s_t))$$

- 策略梯度即：

$$\nabla L(\theta) = -E_{(s_t, a_t) \sim \pi_{\theta}} A_t^{\theta} \nabla \log(\pi_{\theta}(a_t | s_t))$$



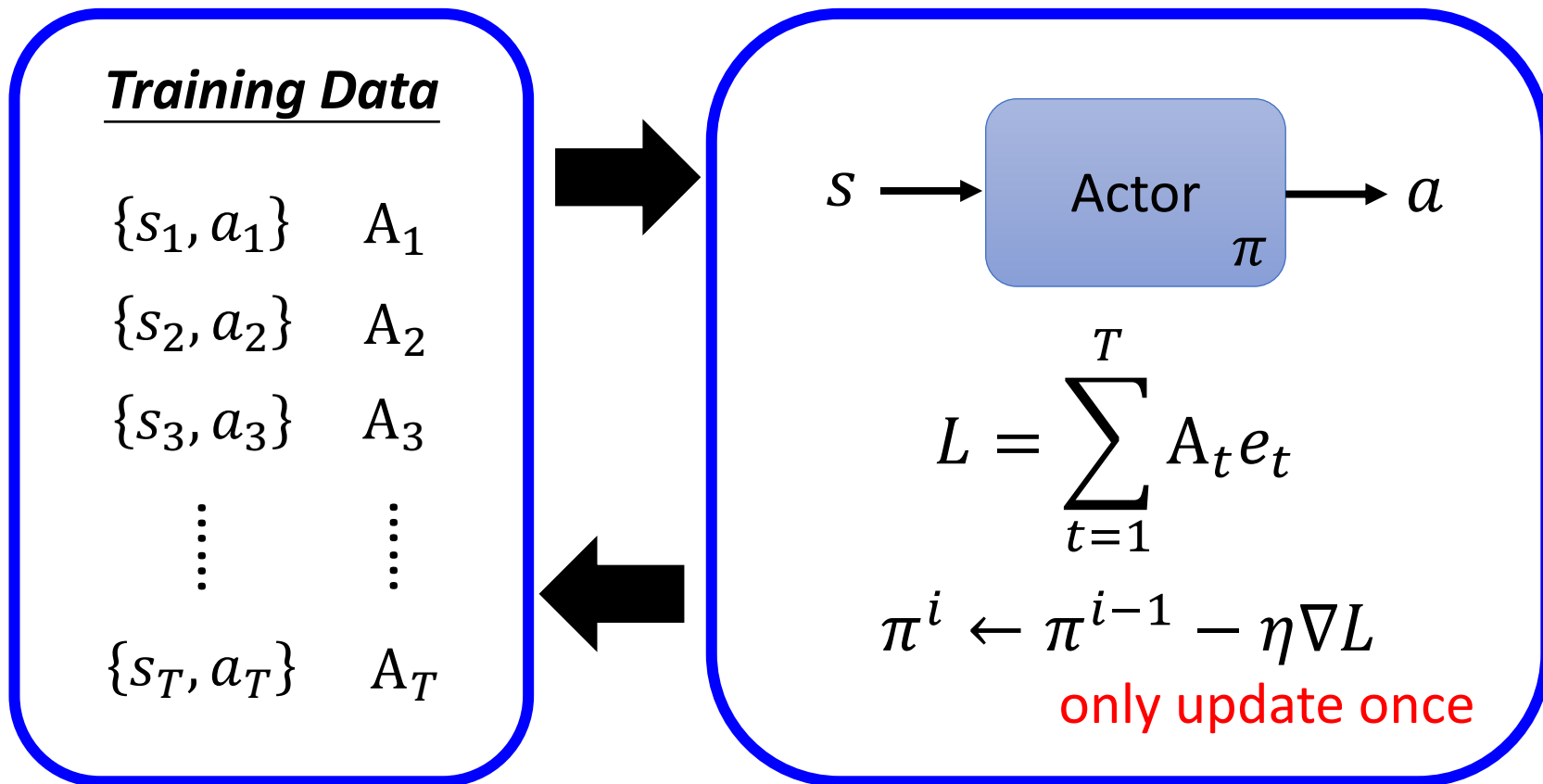
策略梯度 policy gradient

- 初始化策略网络参数 π^0
- 迭代训练 $i = 1$ to N
 - 用策略网络 π^{i-1} 进行交互
 - 获得训练数据 $\{s_1, a_1\}, \{s_2, a_2\}, \dots, \{s_T, a_T\}$
 - 计算 A_1, A_2, \dots, A_T
 - 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 π^0

- 迭代训练 $i = 1$ to N

- 用策略网络 π^{i-1} 进行交互
- 获得训练数据 $\{s_1, a_1\}, \{s_2, a_2\}, \dots, \{s_T, a_T\}$
- 计算 A_1, A_2, \dots, A_T

- 计算损失 L

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

不一定适合 π^i

π^{i-1} 学习的经验



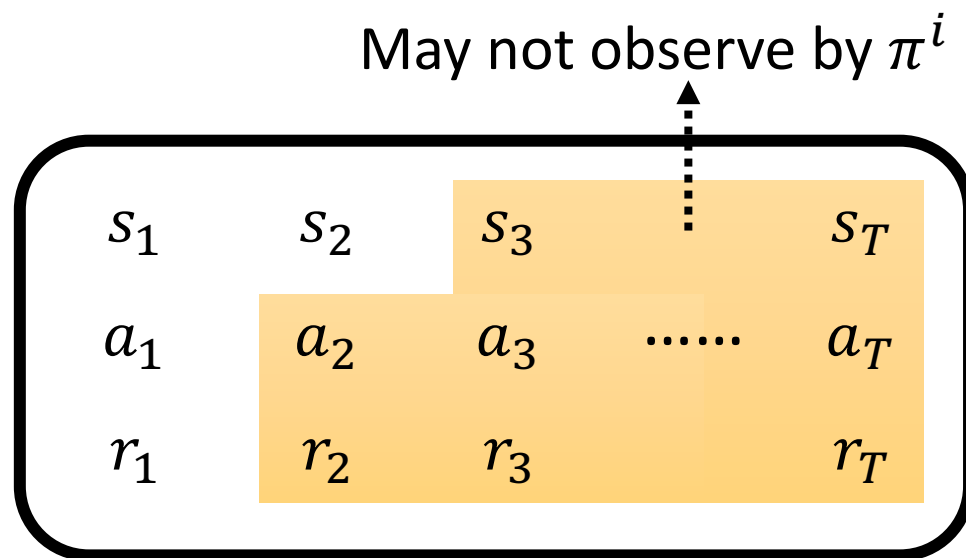
One man's meat is another man's poison.



策略梯度 policy gradient

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

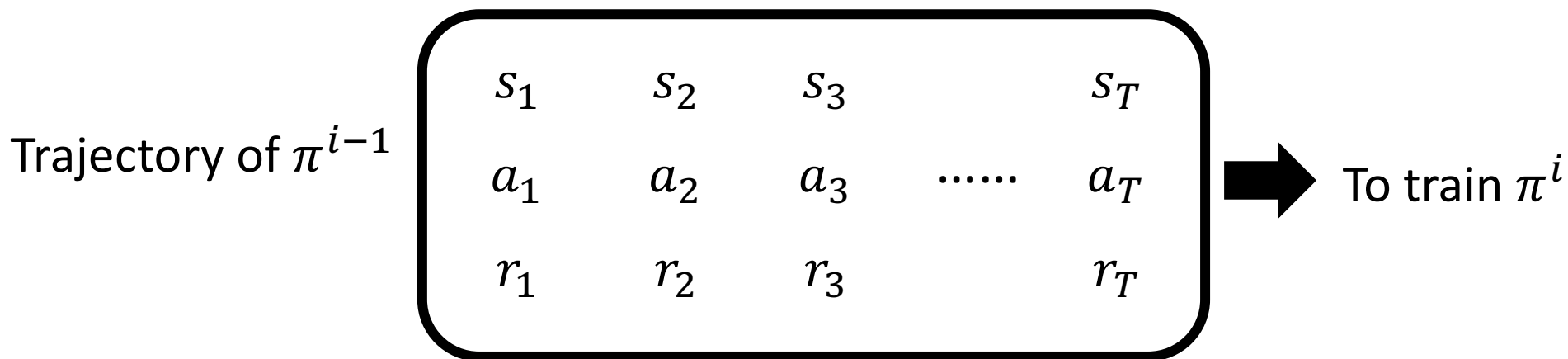
Trajectory of π^{i-1}





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)



因此，无需每次更新一次参数后都重新采集数据。



在线策略 On-policy v.s. 离线策略 Off-policy

- 待训练策略需要知道其与待交互策略的差异
- 重要性采样 Importance Sampling

$$E_{x \sim p}[f(x)] \approx \frac{1}{N} \sum_{i=1}^N \cancel{f(x^i)}$$

x^i is sampled from $p(x)$

We only have x^i sampled from $q(x)$

$$= \int f(x)p(x)dx = \int f(x) \frac{p(x)}{q(x)} q(x)dx = E_{x \sim q}[f(x) \boxed{\frac{p(x)}{q(x)}}]$$

Importance weight



在线策略 On-policy v.s. 离线策略 Off-policy (略)

- 待训练策略和交互策略相同 \rightarrow On-policy

$$\nabla L(\theta) = -E_{(s_t, a_t) \sim \pi_\theta} A_t^\theta \nabla \log(\pi_\theta(a_t | s_t))$$

- 待训练策略和交互策略不同 \rightarrow Off-policy \rightarrow Proximal Policy

Optimization (PPO)

$$\nabla L^{\theta'}(\theta) = -E_{(s_t, a_t) \sim \pi_{\theta'}} \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta'}(a_t | s_t)} \frac{\pi_\theta(s_t)}{\pi_{\theta'}(s_t)} \boxed{A_t^{\theta'}}$$

This term is from sampled data.

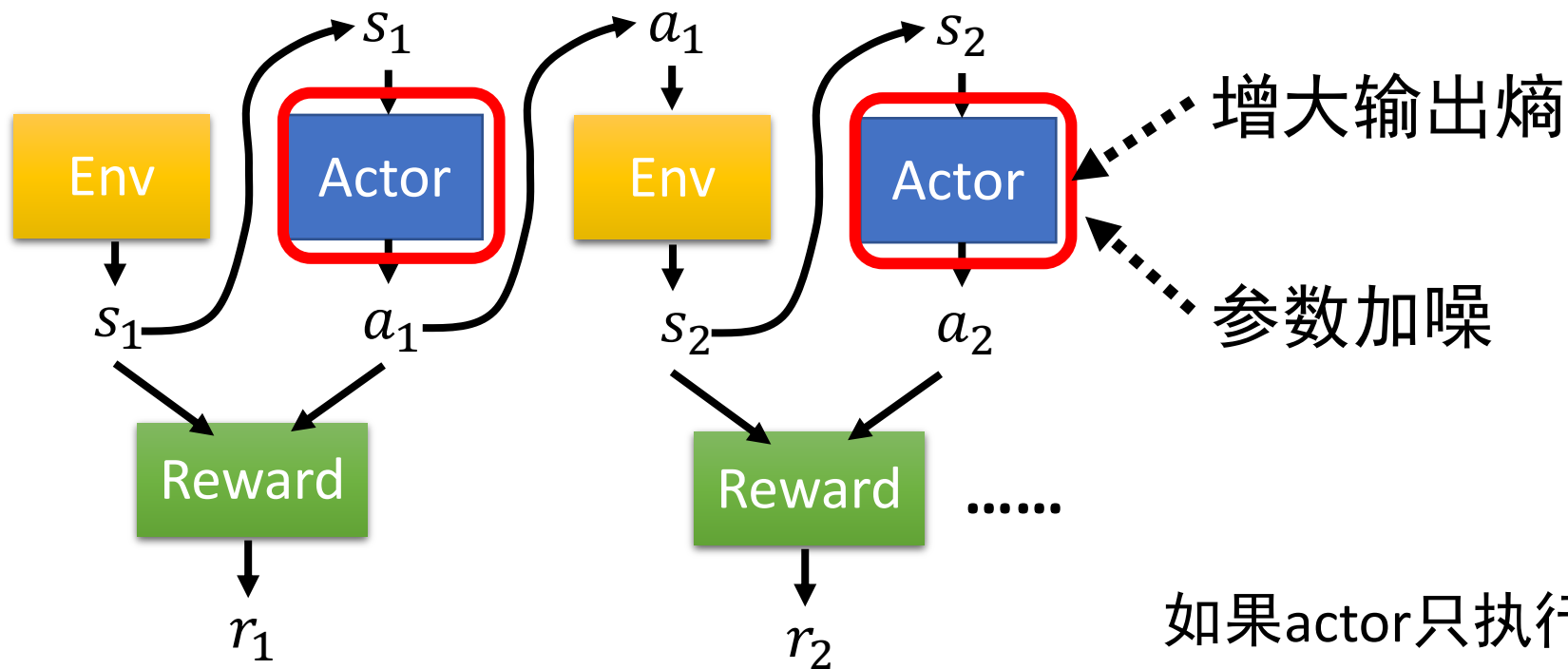
- 即 $L^{\theta'}(\theta) = -E_{(s_t, a_t) \sim \pi_{\theta'}} \frac{\pi_\theta(a_t | s_t)}{\pi_{\theta'}(a_t | s_t)} A_t^{\theta'}$

$$\nabla f(x) = f(x) \nabla \log f(x)$$

其中， θ' 是与环境交互的策略， θ 是待训练策略



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/>

