南京大学
NANJING UNIVERSITY

人工智能导论

强化学习
Reinforcement Learning

郭兰哲

南京大学 智能科学与技术学院

https://www.lamda.nju.edu.cn/guolz/IntroAI/fall2025/index.html

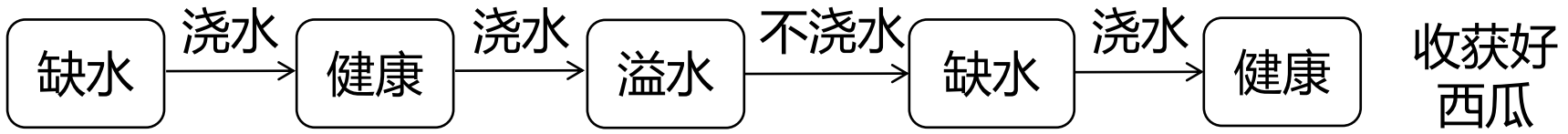
Email: guolz@nju.edu.cn

大纲

- 强化学习简介
- 有模型学习
  - 马尔科夫决策过程
  - 贝尔曼方程
  - 值迭代
  - 策略迭代
- 无模型学习
  - 蒙特卡洛强化学习
  - 时序差分学习

例子：瓜农种西瓜

种下瓜苗后：(为简便，仅考虑浇水和不浇水两个动作，不考虑施肥、除草等)



- 多步决策过程
- 过程中包含状态、动作、反馈 (奖赏) 等
- 需多次种瓜，在过程中不断摸索，才能总结出较好的种瓜策略

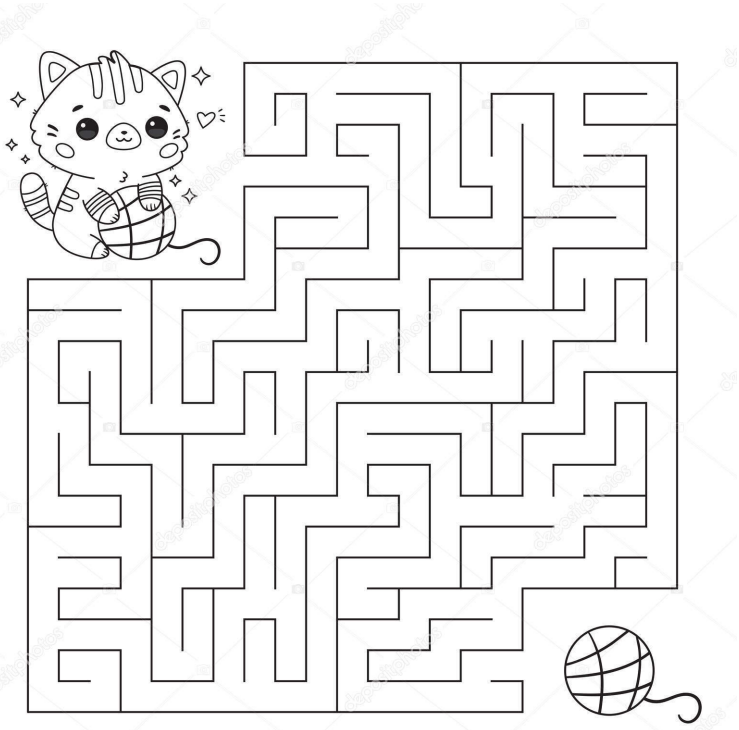
抽象该过程：序列决策任务

从搜索到策略学习

Search

确定环境

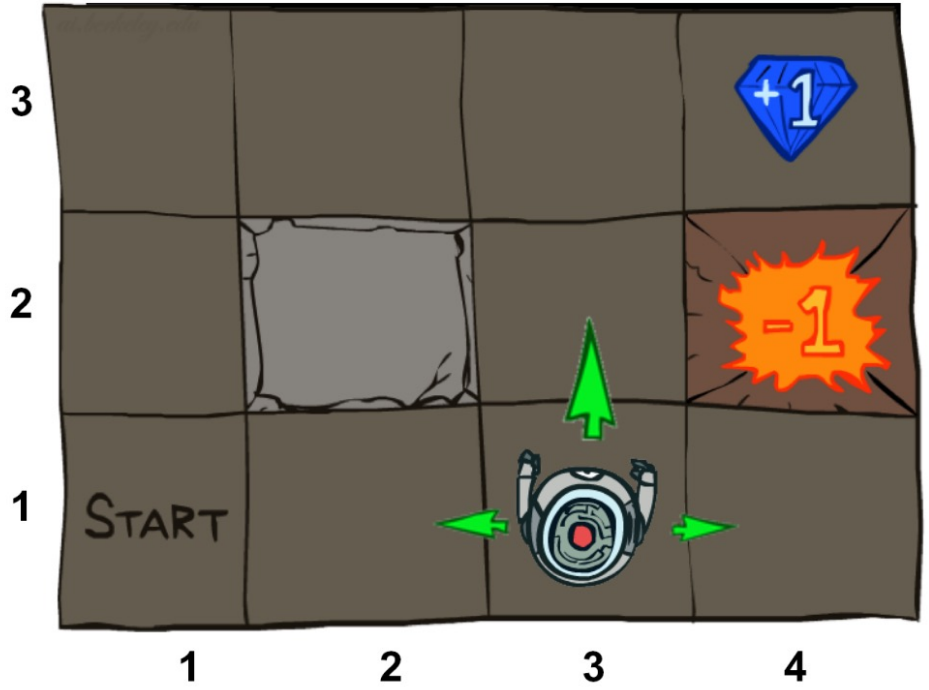
从起点到终点动作序列



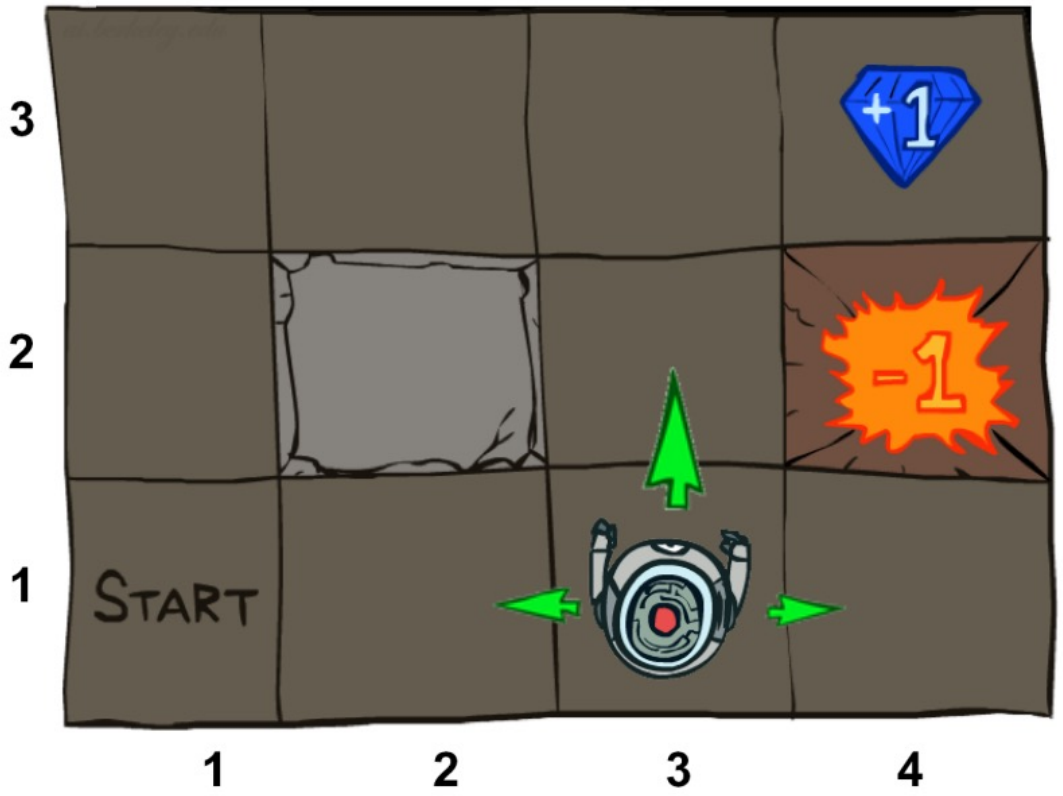
Reinforcement Learning

非确定环境

学会一个策略，能够在任意状态下找到最佳动作

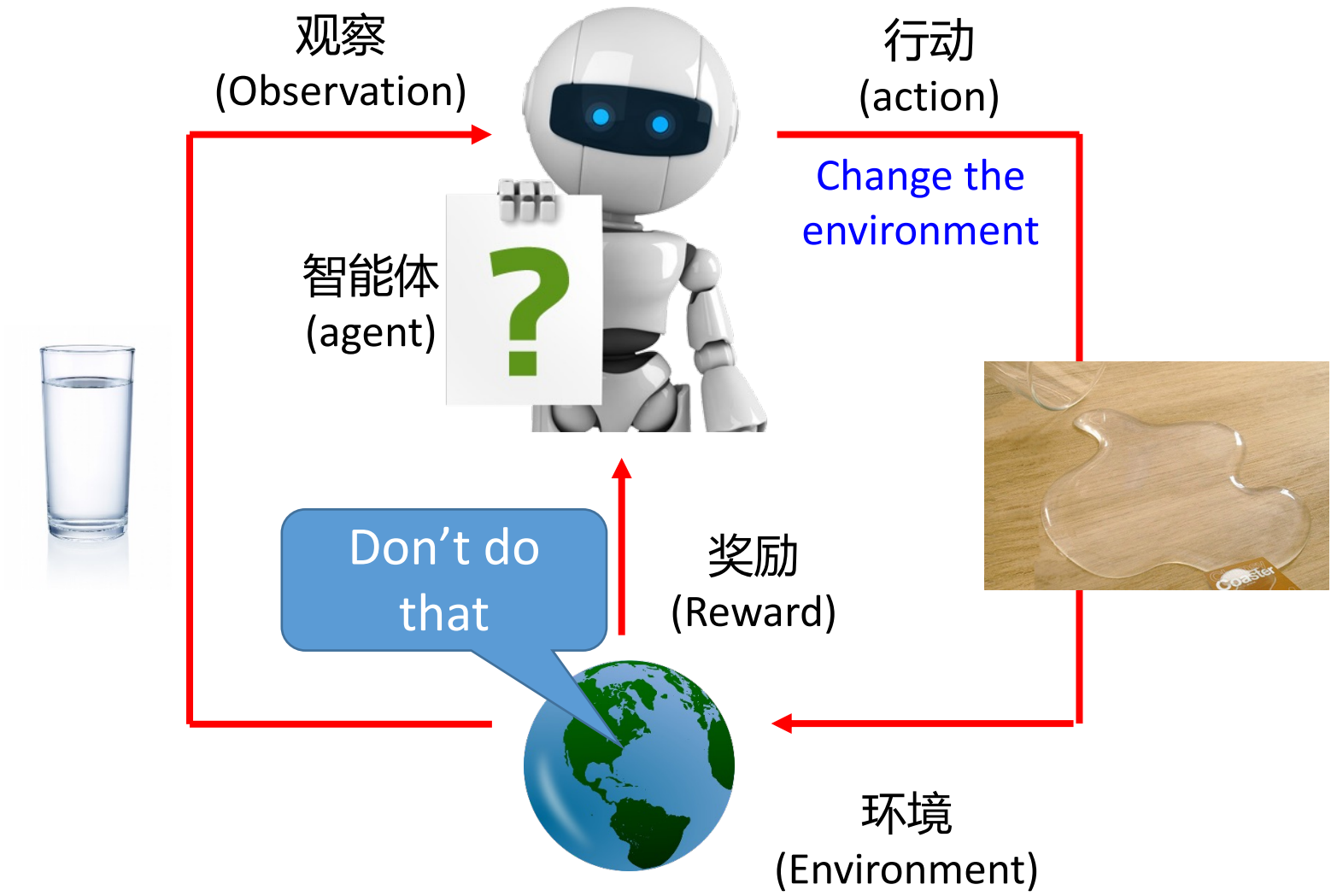


序列决策任务



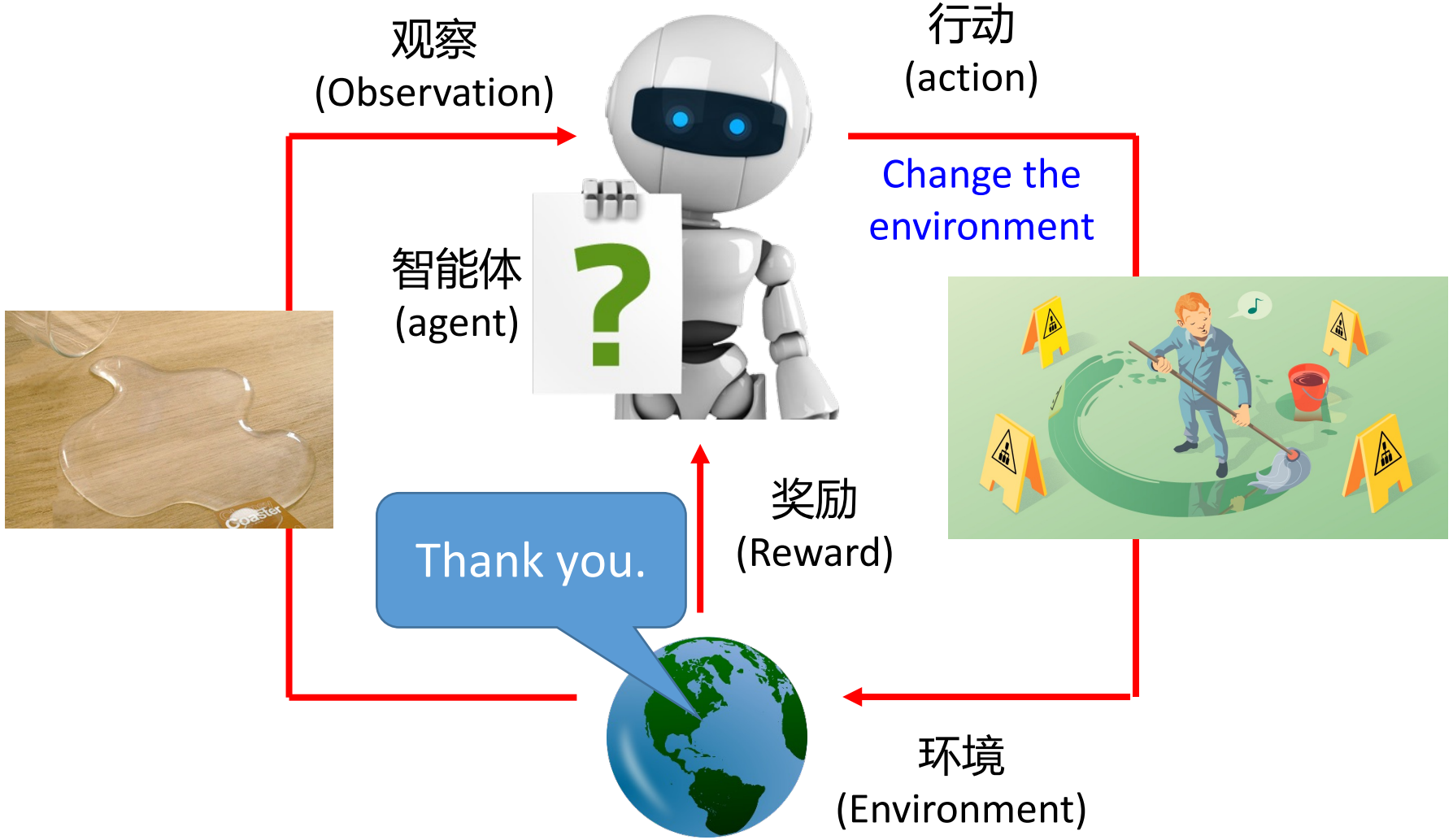
如何学会一个策略，能够在任意状态下找到最佳动作

强化学习(reinforcement learning)



强化学习(reinforcement learning)

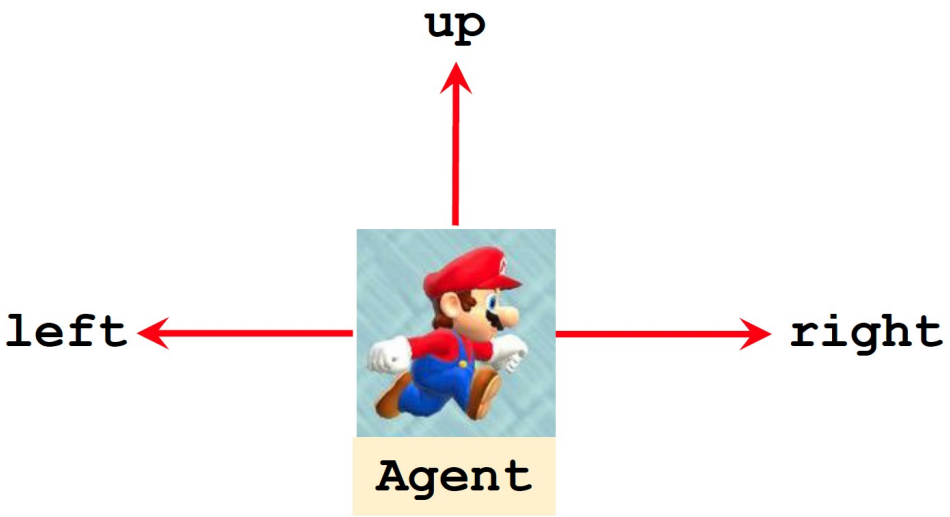
智能体从交互中学习得到最大化累计奖励的行动



强化学习的关键要素

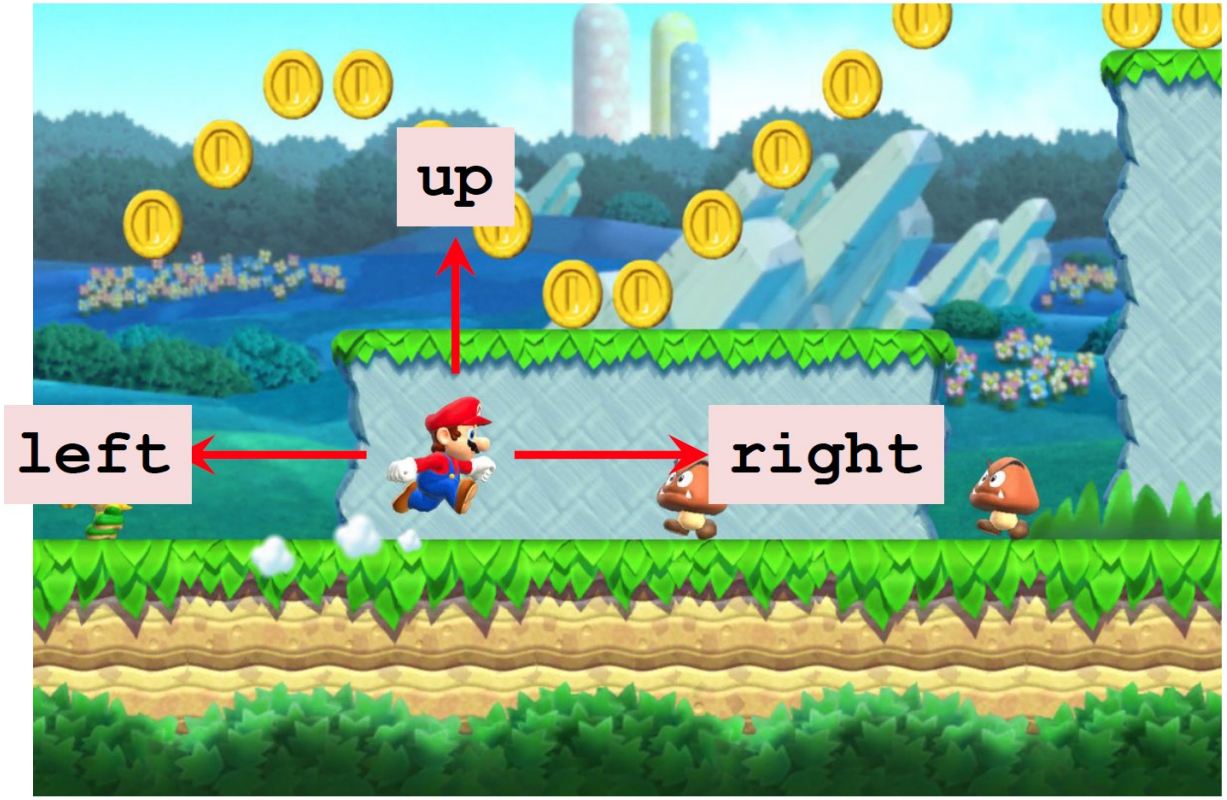
- 状态(State)：$s \in S$ 是对当前环境的描述
  - 围棋任务中，当前的局面
  - 超级马里奥中，当前的游戏画面
  - 网格世界中，当前的位置和状态
  - …

- 动作(Action)：$a \in A$对智能体采取的行动
  - 围棋任务中，选择一个位置落子
  - 超级马里奥中，向左、向右、向上
  - 网格世界中，选择一个方向移动
  - …



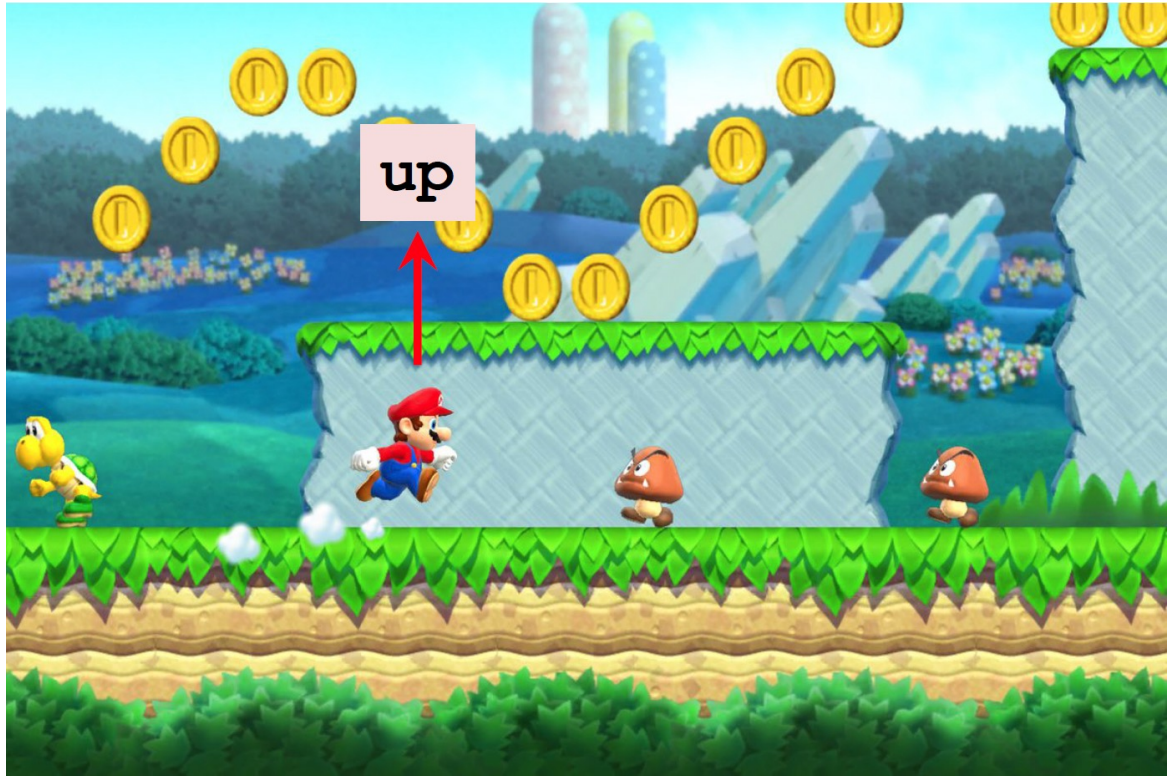
强化学习的关键要素

- 策略(Policy) π： 智能体如何根据观察到的状态做决策，即从动作空间中选择一个动作
- **强化学习最终学得的就是一个策略函数**
- 确定性策略：π：$S \to A$
- 随机性策略：π：$S \times A \to \mathbb{R}$



强化学习的关键要素

- 状态转移(state transition)P： $S \times A \times S \to \mathbb{R}$
- 智能体从当前t时刻状态s_t转移到下一个时刻的状态s_{t+1}



强化学习的关键要素

- 状态转移(state transition)P：$S \times A \times S \to \mathbb{R}$
- 智能体从当前t时刻状态s_t转移到下一个时刻的状态s_{t+1}

考虑了一般情况：非确定性环境
(Non-Deterministic Environment)

强化学习的关键要素

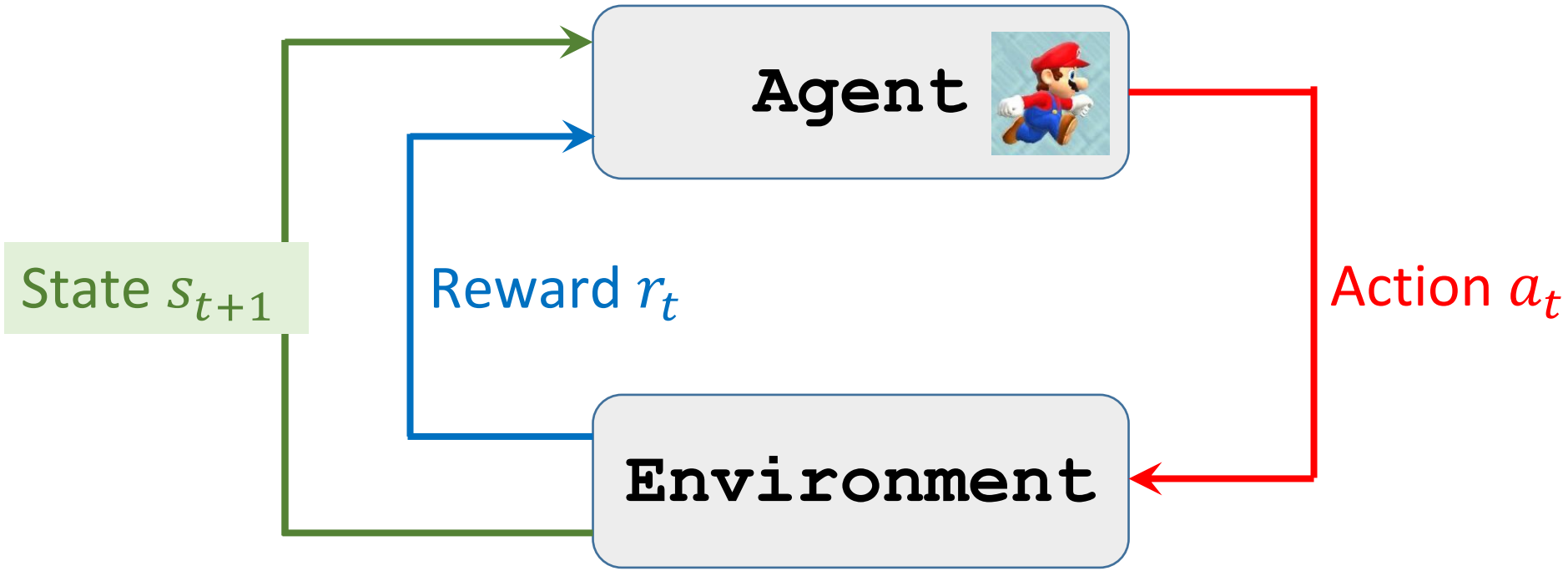
- 奖励(Reward)$R: S \times A \times S \mapsto \mathbb{R}$ （或$R: S \times S \to \mathbb{R}$）
- 智能体执行一个动作之后，环境返回给智能体一个数值

- 通常需要人来定义

- 奖励函数的好坏影响强化学习的结果

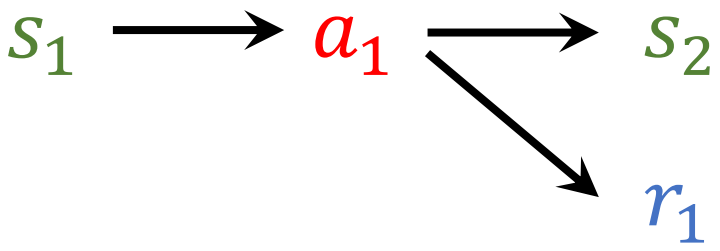
- 收集硬币：R=+1
- 被板栗仔撞到：R=-1000
- 通关：R=+1000
- 无事发生：R=0

强化学习的交互过程



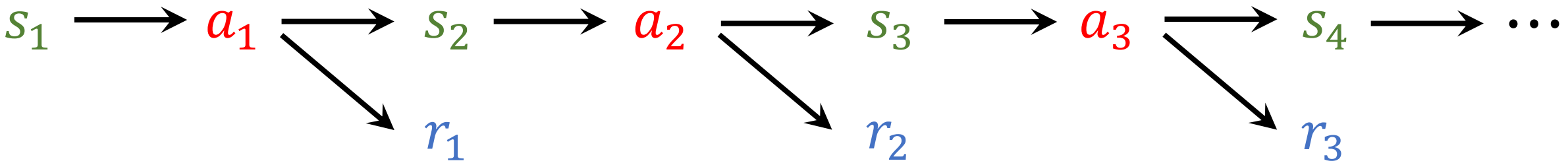
强化学习的交互过程

- 观察到状态s_t，根据策略$\pi(s_t)$选择动作a_t
- 执行动作a_t，状态转移至s_{t+1}，获得奖励r_t



强化学习的交互过程

- 观察到状态s_t，根据策略$\pi(s_t)$选择动作a_t
- 执行动作a_t，状态转移至s_{t+1}，获得奖励r_t



- 轨迹(trajectory)：智能体观测到的所有状态、动作、奖励的序列

$$s_1, a_1, r_1,\ s_2, a_2, r_2,\ \cdots,\ s_n, a_n, r_n$$

什么是好的策略

- 回报(return)：从当前时刻开始到本回合结束的所有奖励总和

$$G_t = R_t + R_{t+1} + R_{t+2} + R_{t+3} + \cdots$$

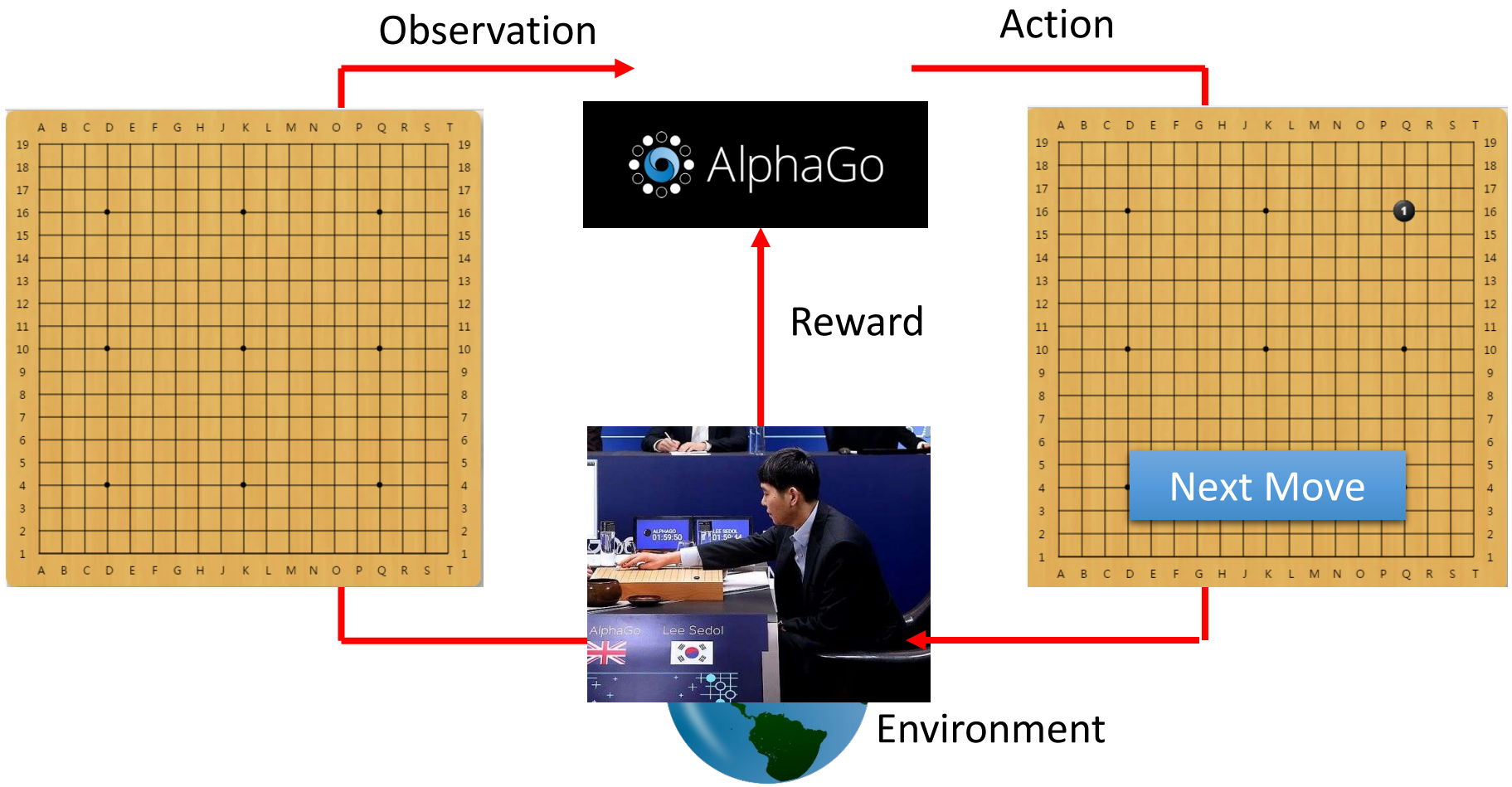
- **强化学习的目标是最大化回报，即累计奖励，而不是最大化当前奖励**（好比下棋的时候目标是赢得比赛，而不是吃掉对方的棋子）

- 通常考虑折扣汇报： $\gamma \in [0,1]$是未来奖励的折扣因子，使得和未来奖励相比起来智能体更重视即时奖励

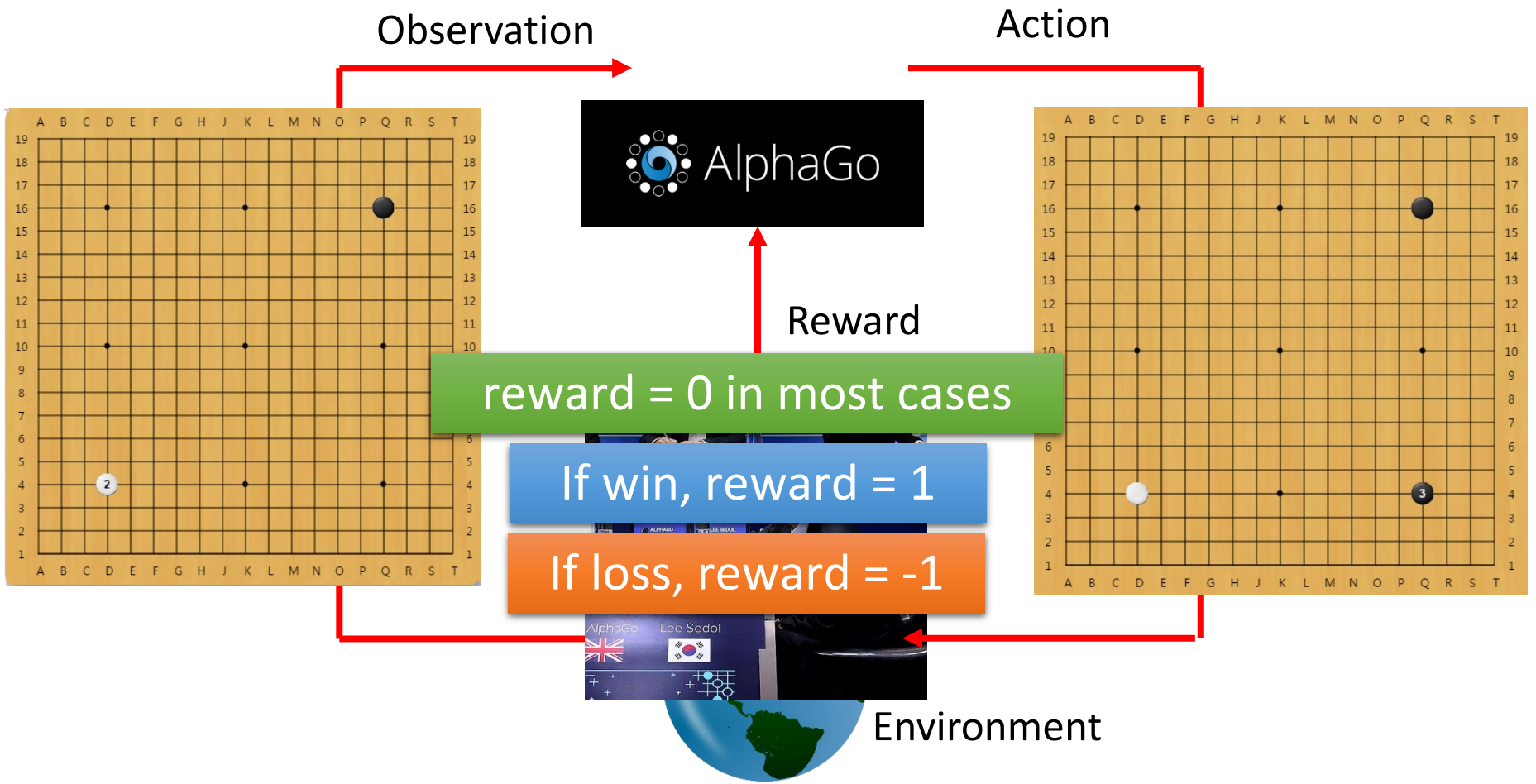
$$G_t = R_t + \gamma\, R_{t+1} + \gamma^2\, R_{t+2} + \gamma^3\, R_{t+3} + \cdots$$

- 以金融为例，今天的$1比明天的$1更有价值

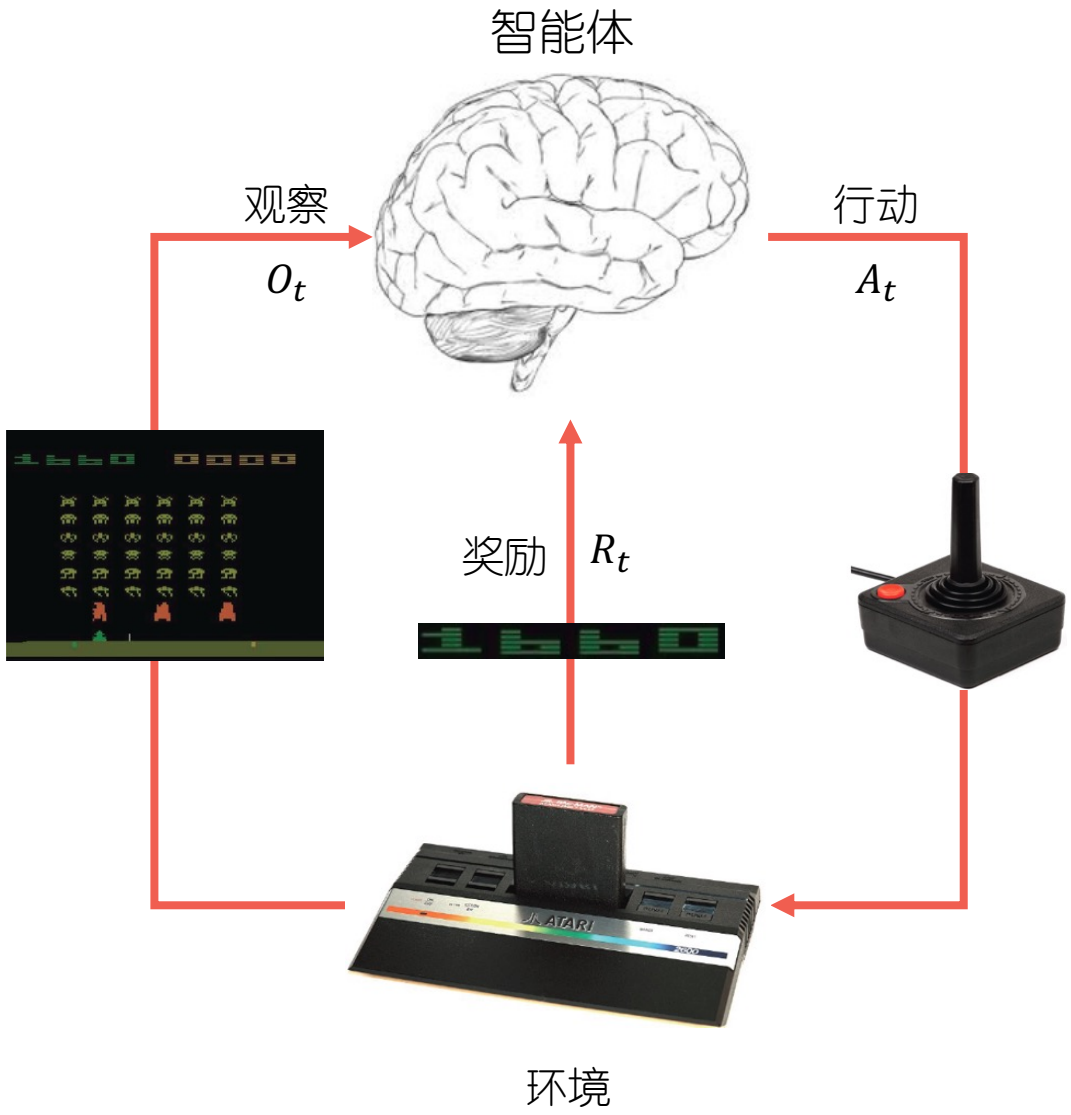
强化学习-围棋



强化学习-围棋


Observation
Action
AlphaGo
Reward
reward = 0 in most cases
If win, reward = 1
If loss, reward = -1
Environment

强化学习-游戏



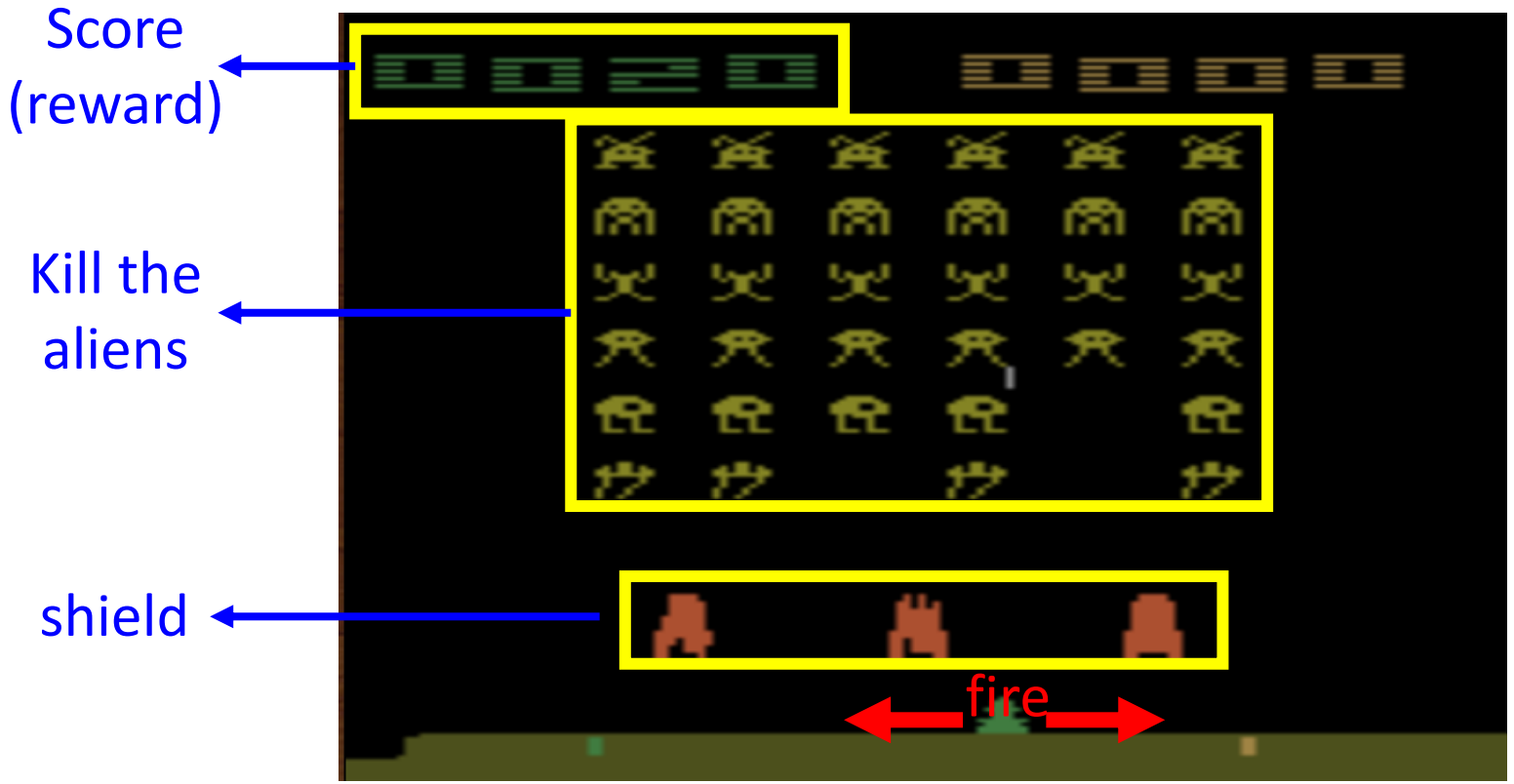
从交互游戏中进行学习

在操纵杆上选择行动并查看分数和像素画面

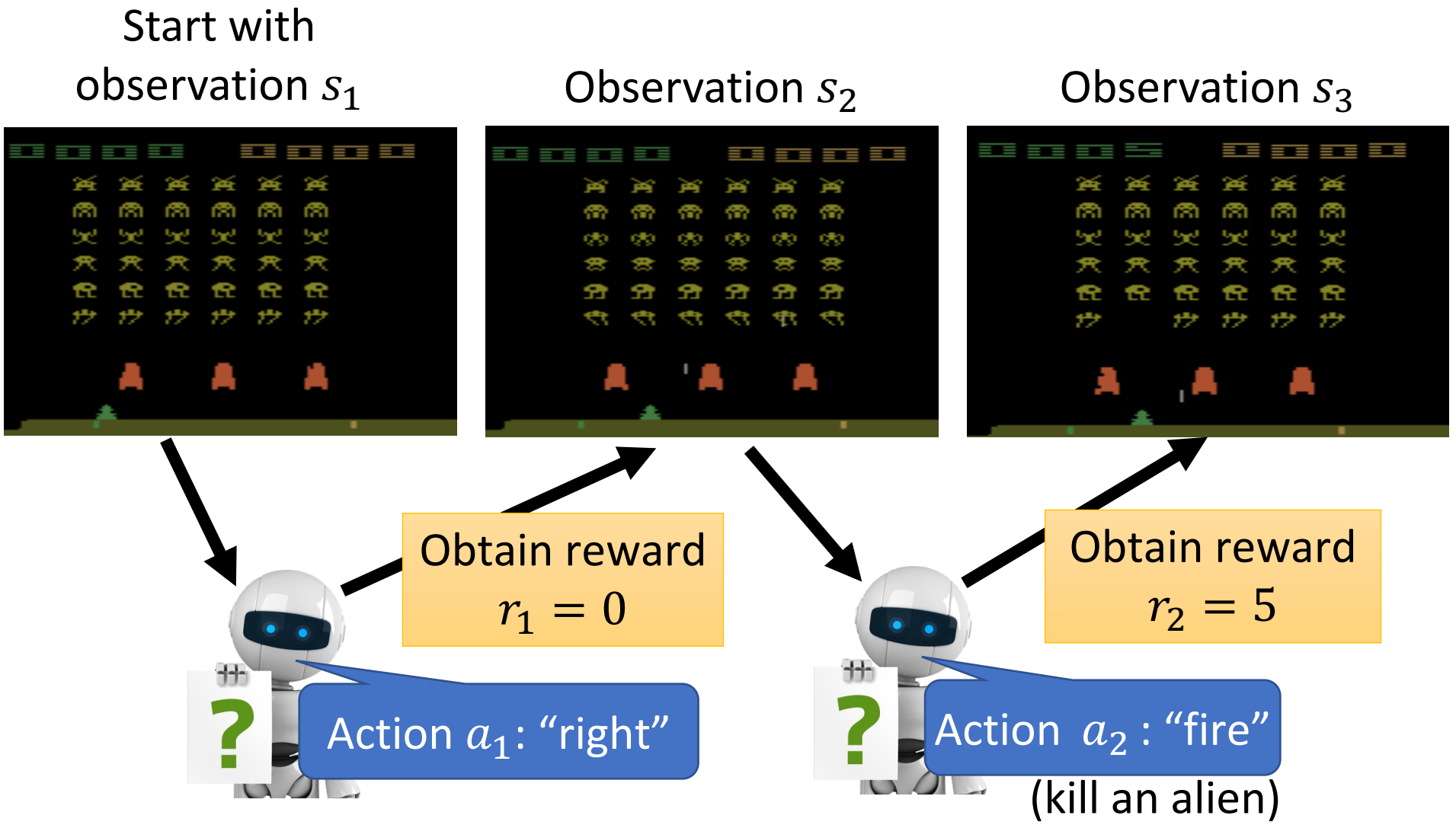
强化学习-游戏

- Space invader

Termination: all the aliens are killed, or your spaceship is destroyed.



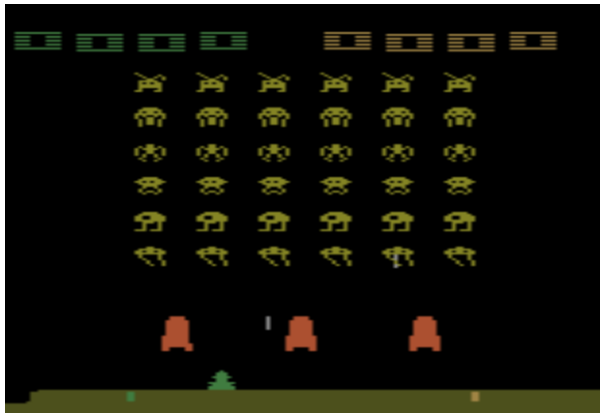
强化学习-游戏


Start with observation s_1
Observation s_2
Observation s_3
Obtain reward $r_1 = 0$
Action a_1: "right"
Obtain reward $r_2 = 5$
Action a_2 : "fire"
(kill an alien)

强化学习-游戏

Start with observation s_1

Observation s_2



Observation s_3

After many turns

This is an ***episode***.

Action a_T

Obtain reward r_T

强化学习应用案例


INNOVATIONS IN
The microbiome
nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
LEARNING CURVE
Self-taught AI software attains human-level performance in video games
PAGES 406 & 529
EPIDEMIOLOGY
SHARE DATA IN OUTBREAKS
Forge open access to sequences and more
PAGE 477
COSMOLOGY
A GIANT IN THE EARLY UNIVERSE
A supermassive black hole at a redshift of 6.3
PAGES 490 & 512
QUANTUM PHYSICS
TELEPORTATION FOR TWO
Transferring two properties of a single photon
PAGES 491 & 516
NATURE.COM/NATURE
26 February 2015
Vol. 518, No. 7540


nature
THE INTERNATIONAL WEEKLY JOURNAL OF SCIENCE
At last — a computer program that can beat a champion Go player PAGE 484
ALL SYSTEMS GO
CONSERVATION
SONGBIRDS À LA CARTE
Illegal harvest of millions of Mediterranean birds
PAGE 452
RESEARCH ETHICS
SAFEGUARD TRANSPARENCY
Don't let openness backfire on individuals
PAGE 459
POPULAR SCIENCE
WHEN GENES GOT 'SELFISH'
Dawkins's calling card 40 years on
PAGE 462
NATUREASIA.COM
28 January 2016
Vol. 529, No. 7587

强化学习应用案例：电子游戏

Playing Atari Games

Human-level control through deep reinforcement learning. Nature 2015

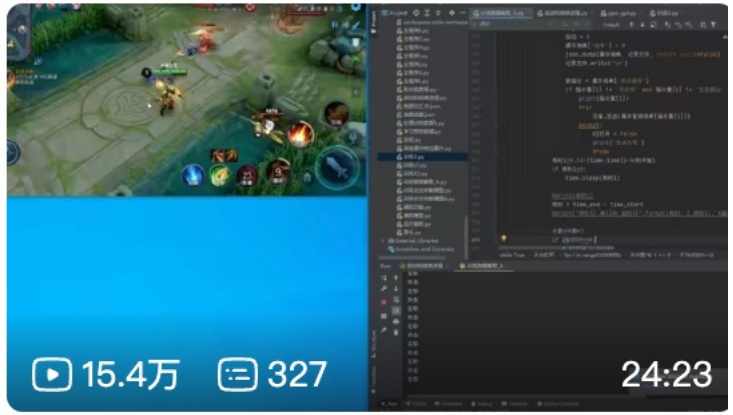
强化学习应用案例：机械臂操作

Finetuning Offline World Models in the Real World. CoRL 2023. https://arxiv.org/pdf/2310.16029

强化学习应用案例：无人驾驶小车

Alex Kendall et.al, Learning to Drive in a Day. ICRA 2019: 8248-8254
https://www.youtube.com/watch?v=eRwTbRtnT1I

更多应用



用强化学习训练AI玩王者荣耀

强化学习超级马里奥（stablebaseline3框架gym游戏包）

黑神话悟空 AI vs 广智 强化学习训练 直播实录

【DQN强化学习】700行代码，让AI学习如何玩原神

更多应用

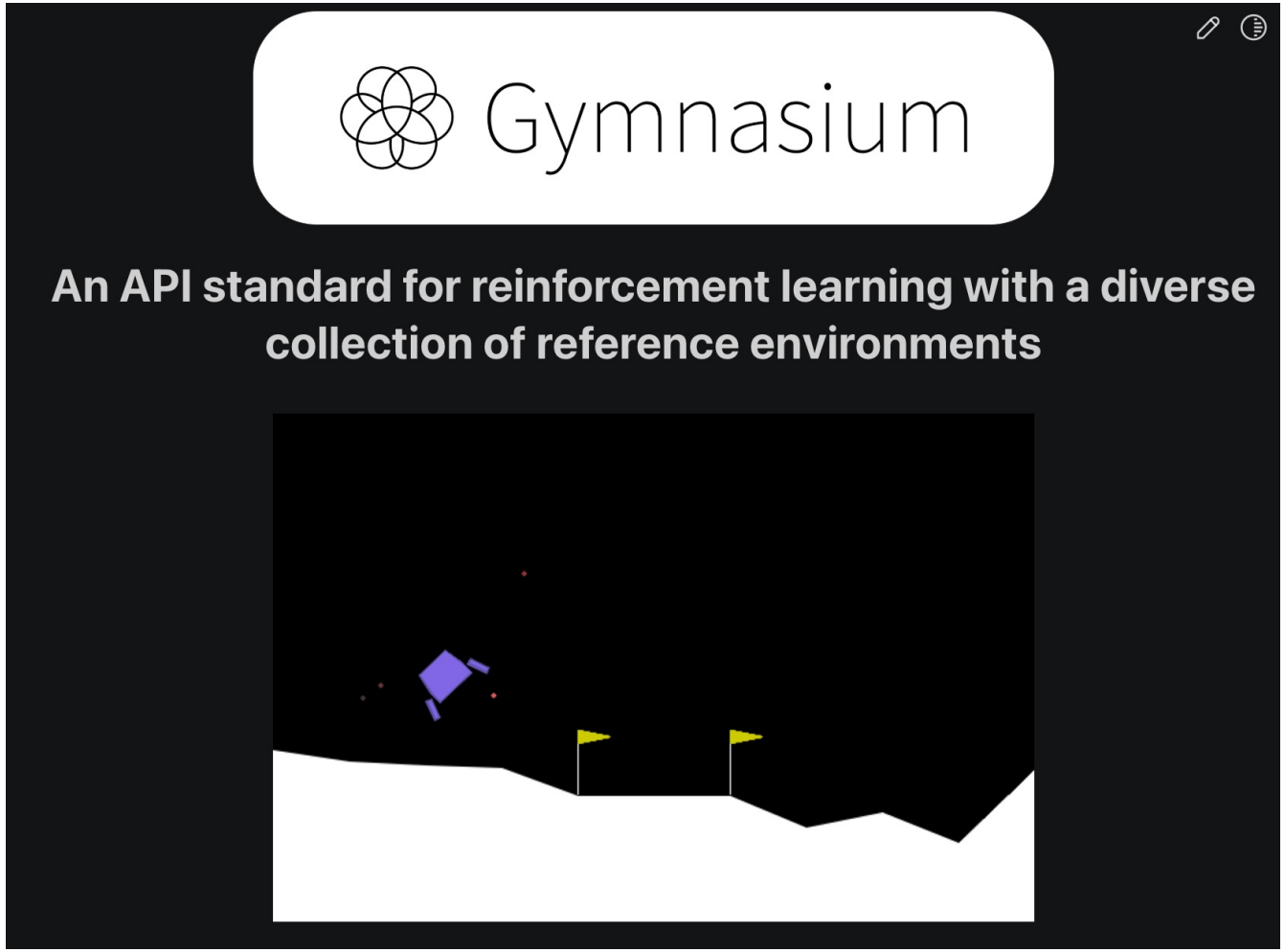
- Flying Helicopter
  - https://www.youtube.com/watch?v=0JL04JJjocc
- Driving
  - https://www.youtube.com/watch?v=0xo1Ldx3L5Q
- Robot
  - https://www.youtube.com/watch?v=370cT-OAzzM
- Text generation
  - https://www.youtube.com/watch?v=pbQ4qe8EwLo

扩展阅读

- Sutton’s book:
  - http://incompleteideas.net/sutton/book/the-book.html
- David Silver’s Lecture:
  - https://www.davidsilver.uk/teaching/
- Sergey Levine
  - https://rail.eecs.berkeley.edu/deeprlcourse/
- 动手学强化学习：
  - https://hrl.boyuai.com/chapter/intro

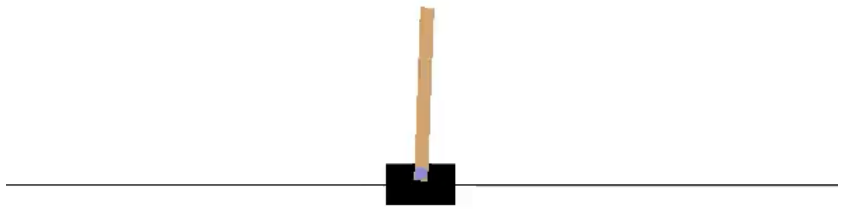
Open AI Gym

https://gymnasium.farama.org/

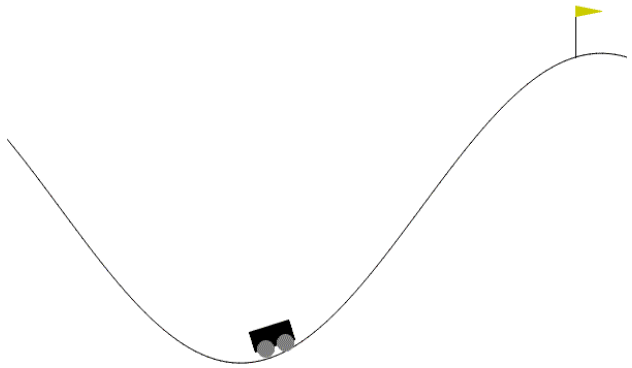


Open AI Gym

经典控制问题



Cart Pole



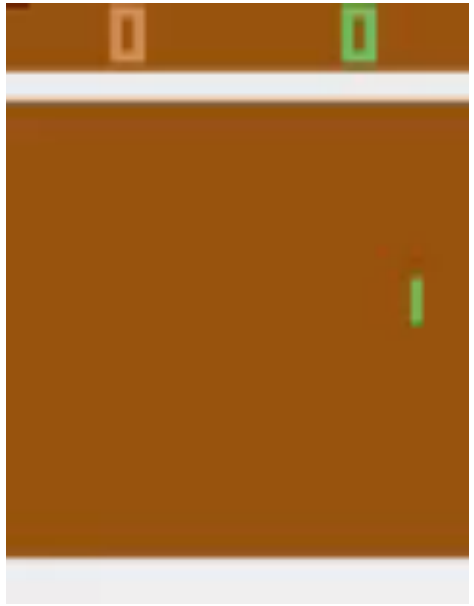
Mountain Car

Num	Observation	Min	Max
0	Cart Position	-4.8	4.8
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -0.418 rad (-24°)	~ 0.418 rad (24°)
3	Pole Angular Velocity	-Inf	Inf

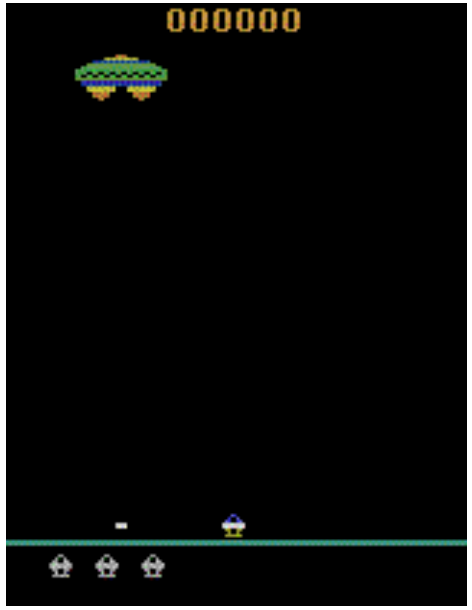
Num	Action
0	Push cart to the left
1	Push cart to the right

Open AI Gym

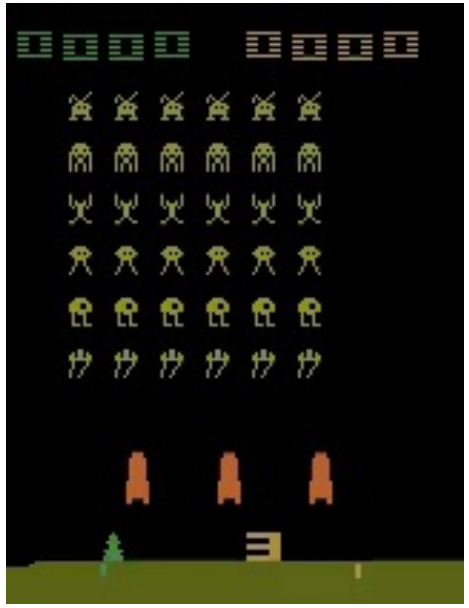
Atari游戏



Pong



Assault

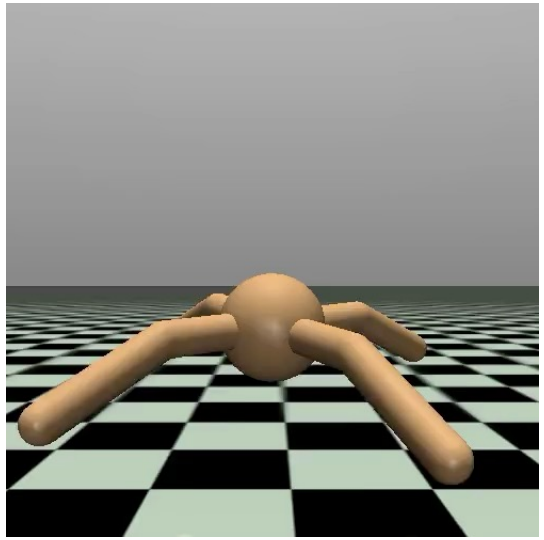


Space Invader

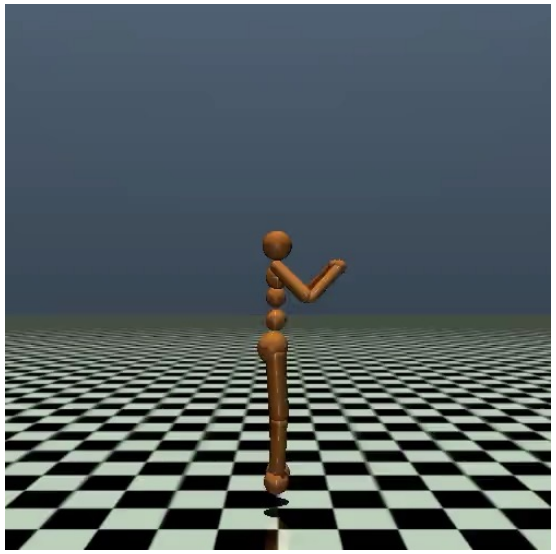
Breakout

Open AI Gym

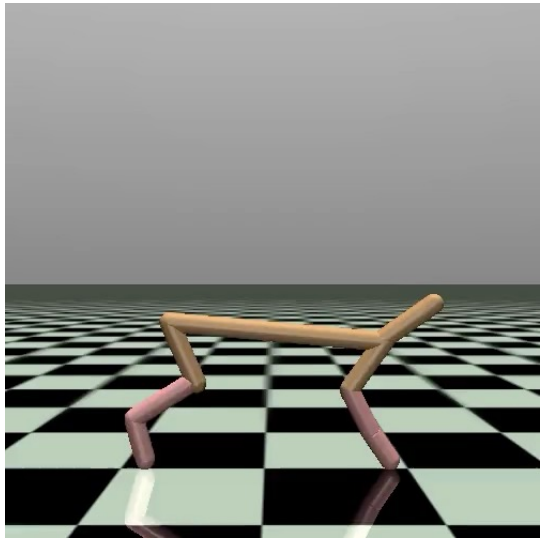
机器人的连续控制问题



Ant

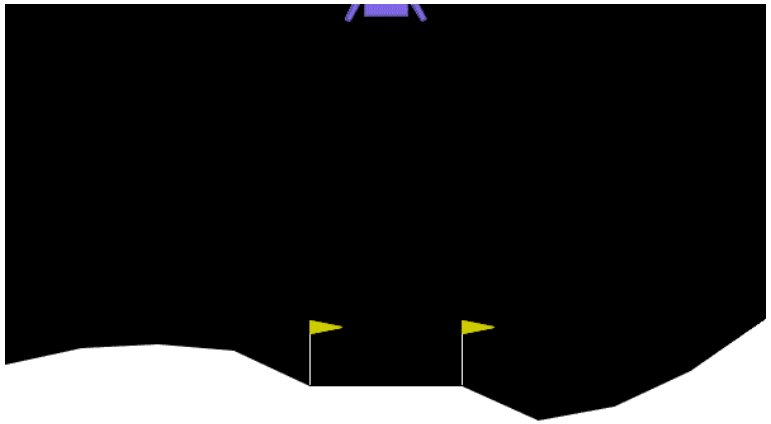


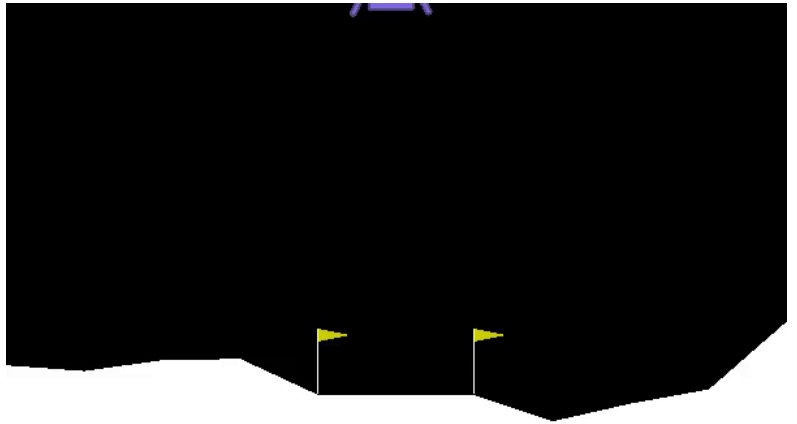
Humanoid



Half Cheetah

Open AI Gym





```python
import gym
env = gym.make("LunarLander-v2", render_mode="human")
env.action_space.seed(42)

observation, info = env.reset(seed=42)

for _ in range(1000):
    observation, reward, terminated, truncated, info = env.step(env.action_space.sample())

    if terminated or truncated:
        observation, info = env.reset()

env.close()
```

大纲

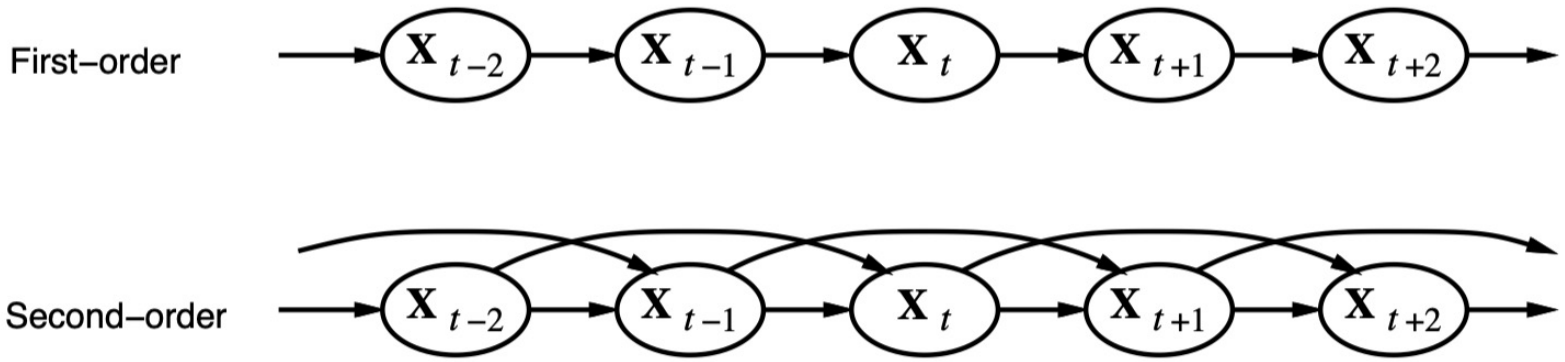
- 强化学习简介
- 有模型学习
  - 马尔科夫决策过程
  - 贝尔曼方程
  - 值迭代
  - 策略迭代
- 无模型学习
  - 蒙特卡洛强化学习
  - 时序差分学习

马尔可夫模型(Markov Model)

- 状态变量：给定时刻t，随机变量X的取值$X_1, \cdots, X_t$
- 初始状态概率：随机变量X的先验概率分布
- 转移模型(transition model)：指定世界如何随时间演变，给定过去的状态变量取值之后，确定最新状态的概率分布$P(X_t|X_{1,\cdots t-1})$

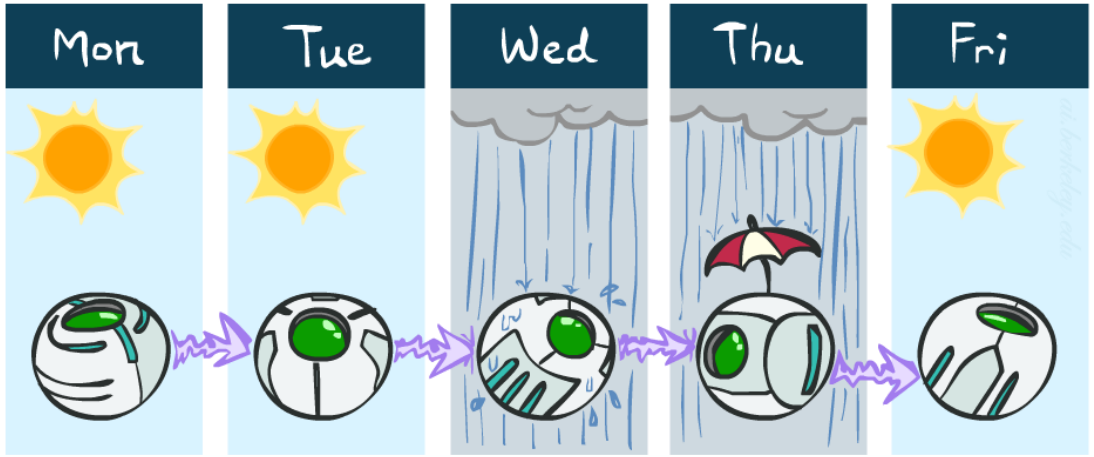
- 马尔可夫假设：当前状态只依赖于有限的固定数量的过去状态

状态+状态转移=马尔科夫过程

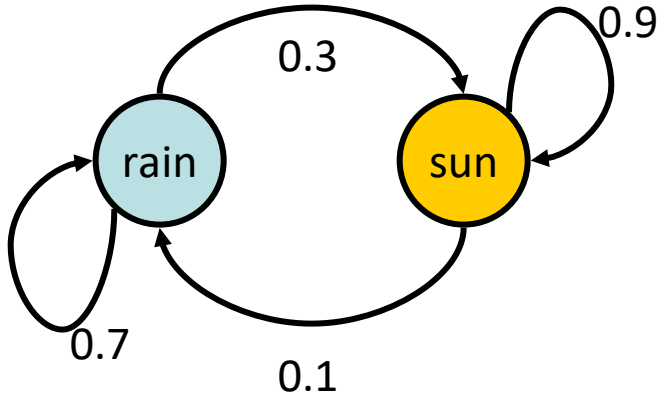


马尔可夫模型示例：天气预报

- 状态：$X = \{rain, sun\}$
- 起始分布: $[1.0\ sun, 0.0\ rain]$
- 状态转移：$P(X_t|X_{t-1})$



X_{t-1}	X_t	$P(X_t \mid X_{t-1})$
sun	sun	0.9
sun	rain	0.1
rain	sun	0.3
rain	rain	0.7



马尔可夫决策过程(Markov Decision Process, MDP)

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

- 状态S_t是马尔可夫的，当且仅当

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

- 马尔可夫决策过程（Markov Decision Process， MDP）
  - 一套为在结果部分随机、部分在决策者的控制下的决策过程建模的数学框架

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

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

马尔科夫过程+动作+奖励

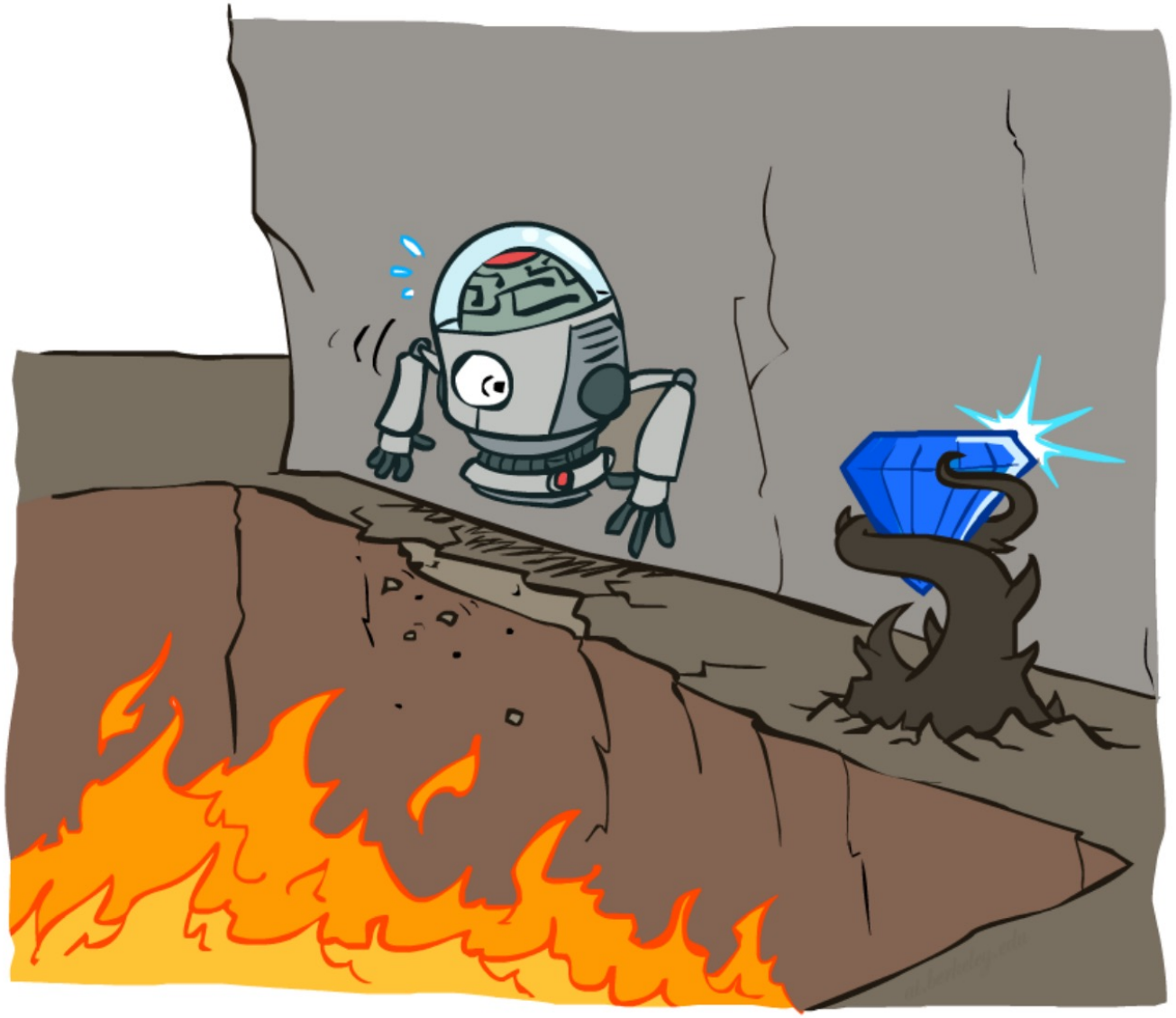
马尔可夫决策过程(Markov Decision Process, MDP)

- 强化学习对应了马尔可夫四元组：(S, A, P, R)
- 有模型学习(model-based learning)：假设S, A, P, R均已知
- MDP模型的动态转换如下所示：
  - 从状态s_0开始，智能体选择某个动作$a_0 \in A$，智能体得到奖励r_0
  - MDP随机转移到下一个状态$s_1 \sim P_{s_0 a_0}$
  - 这个过程不断进行

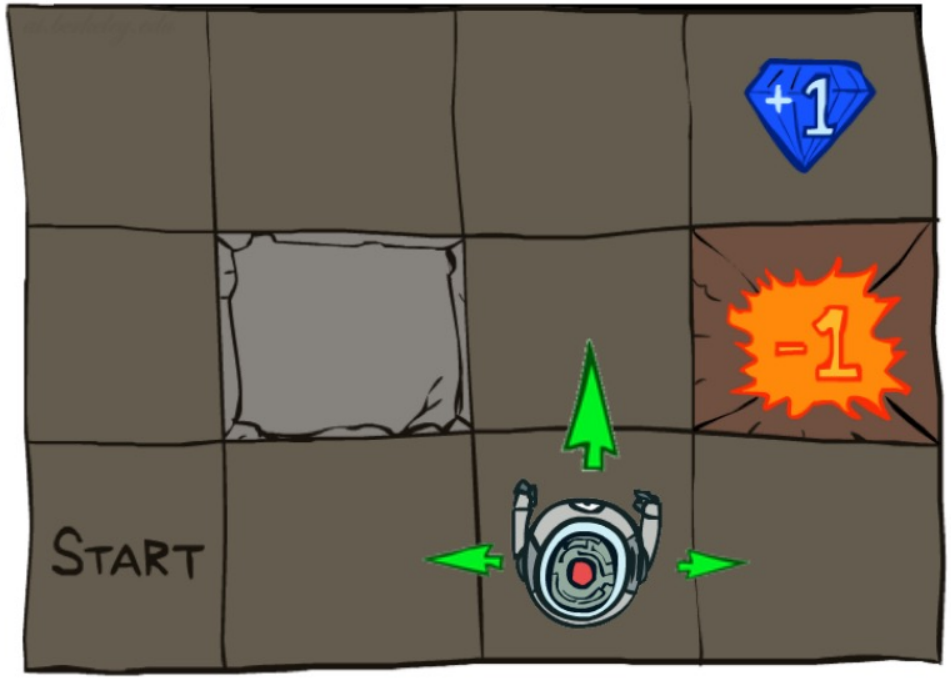
$$s_0 \xrightarrow{a_0, r_0} s_1 \xrightarrow{a_1, r_1} s_2 \xrightarrow{a_2, r_2} s_3 \cdots$$

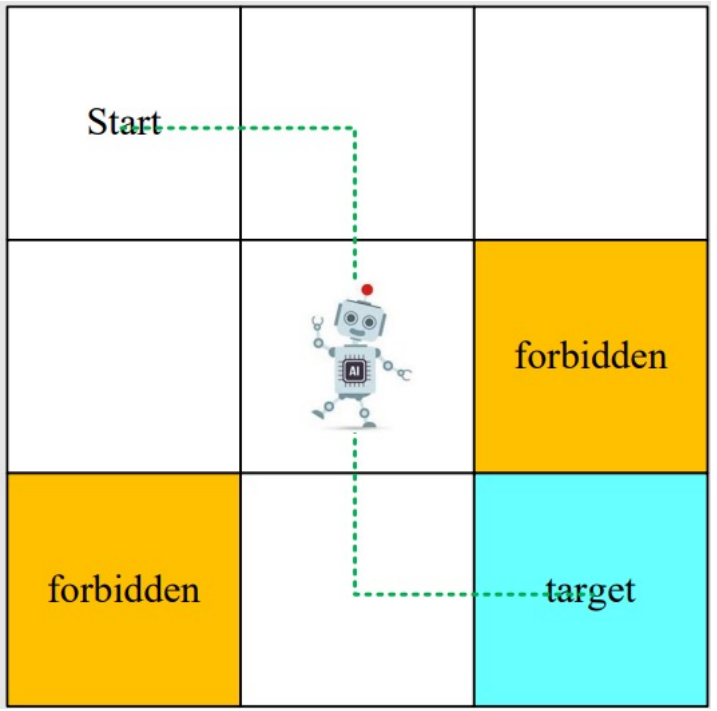
  - 智能体的总回报为 $r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots$

一个例子：网格世界(Grid World)



Grid World

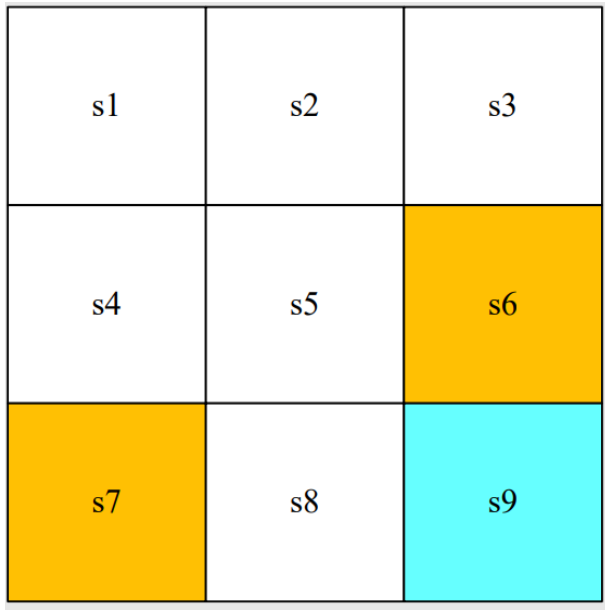




- 环境：由若干网格单元组成的网格世界
- 可通行单元 / 禁止进入单元 / 目标单元 / 边界
- 任务：给定任意起始位置，寻找一条通往目标的“较好”路径
- **如何定义“较好”？**

Grid World

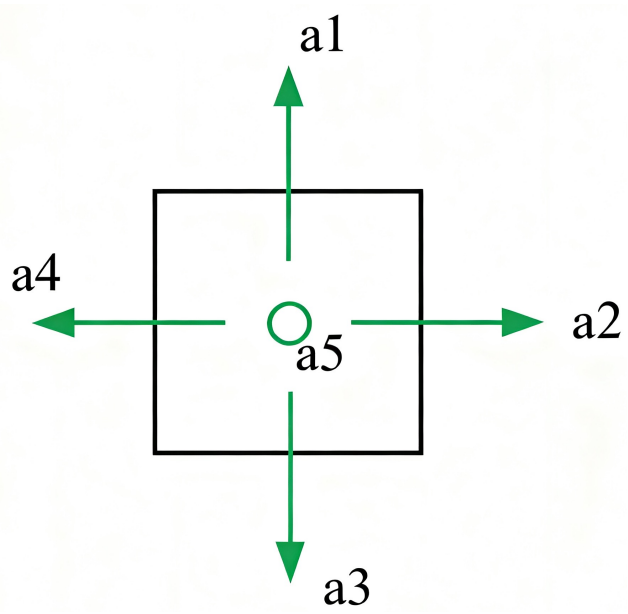
- **状态 (State)**： agent在环境中的状况
- 在grid-world中，状态就是agent所在的位置
- 共有 9 个可能的位置，那么就有 9 个状态 $S_1, S_2, \cdots, S_9$



状态空间(State Space)：所有状态的集合 $\{S_1, S_2, \cdots, S_9\}$

Grid World

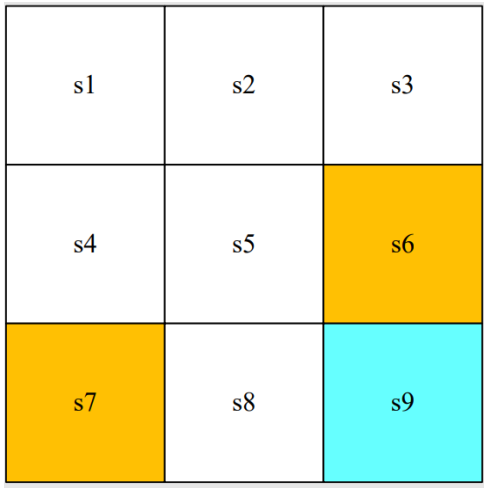
- **动作(Action)**： agent在每个状态可选的行动
- 向上、向右、向下、向左、保持不动



s1	s2	s3
s4	s5	s6
s7	s8	s9

动作空间(Action Space)：所有可行动作的集合

Grid World



禁止区域（Forbidden area）：

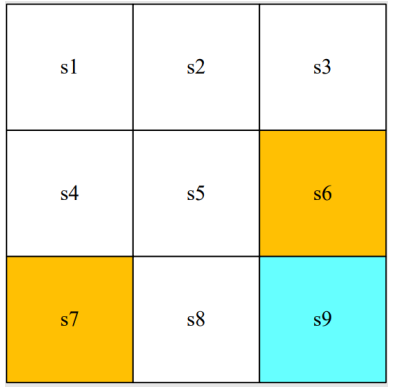
在状态S_5下，如果我们选择动作 a_2，下一个状态会是什么？存在两种情形：

- 情形 1（禁止区可进入但有惩罚）
- 情形 2（禁止区不可进入，例如被墙包围）

考虑第一种，让算法学会自己避免进入禁止区

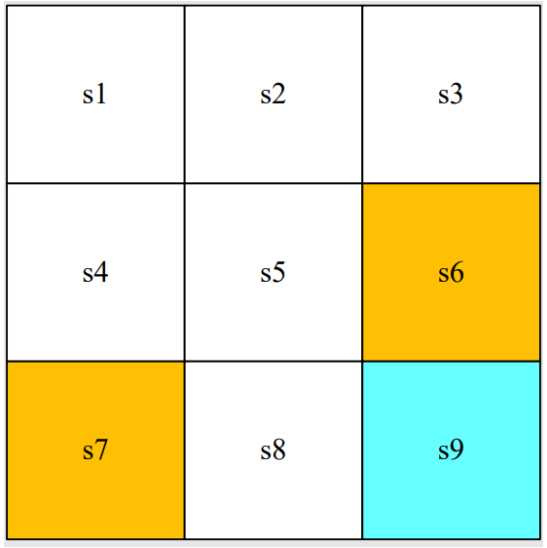
Grid World

- 表格表示(Tabular Representation)



	a_1 (upwards)	a_2 (rightwards)	a_3 (downwards)	a_4 (leftwards)	a_5 (unchanged)
s_1	s_1	s_2	s_4	s_1	s_1
s_2	s_2	s_3	s_5	s_1	s_2
s_3	s_3	s_3	s_6	s_2	s_3
s_4	s_1	s_5	s_7	s_4	s_4
s_5	s_2	s_6	s_8	s_4	s_5
s_6	s_3	s_6	s_9	s_5	s_6
s_7	s_4	s_8	s_7	s_7	s_7
s_8	s_5	s_9	s_8	s_7	s_8
s_9	s_6	s_9	s_9	s_8	s_9

Grid World



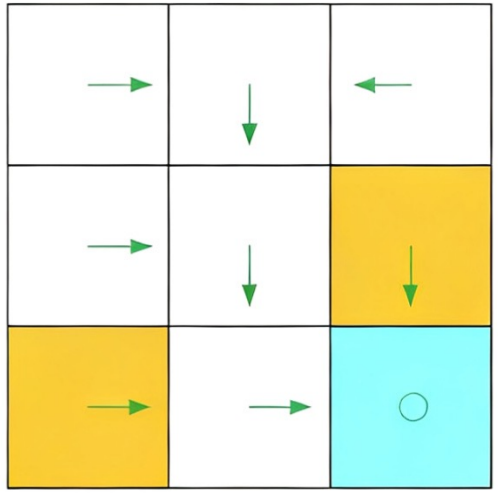
- 状态转移概率(State Transition Probability)
- Deterministic or Stochastic

$$P(S_2|S_1, a_2) = 1$$

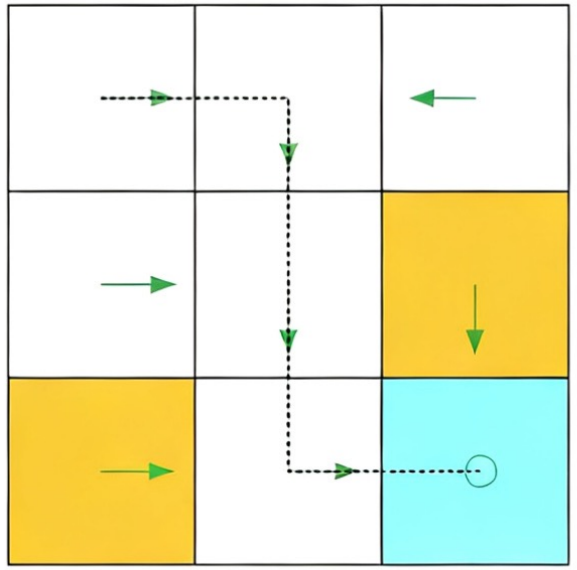
$$P(S_i|S_1, a_2) = 0, \forall i \neq 2$$

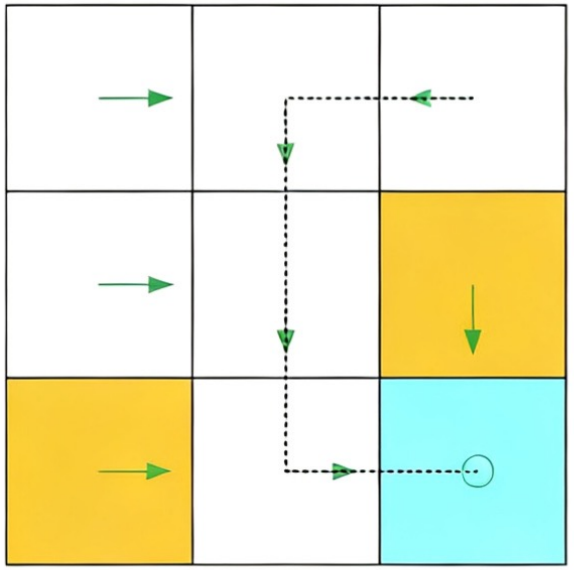
Grid World

Policy：告诉Agent每个状态下应该选择什么动作

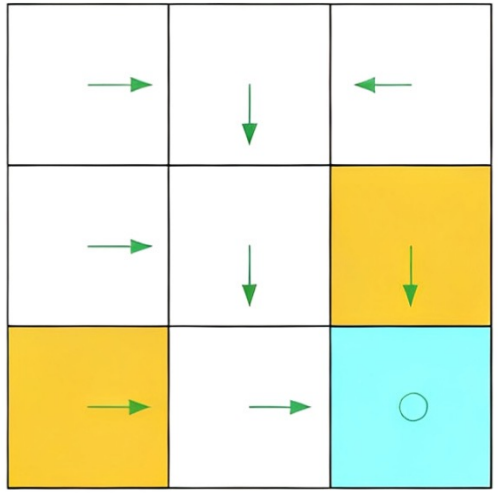


基于Policy可以得到任意起点出发的路径





Grid World



Deterministic Policy

$$\pi(a_1|s_1) = 0$$

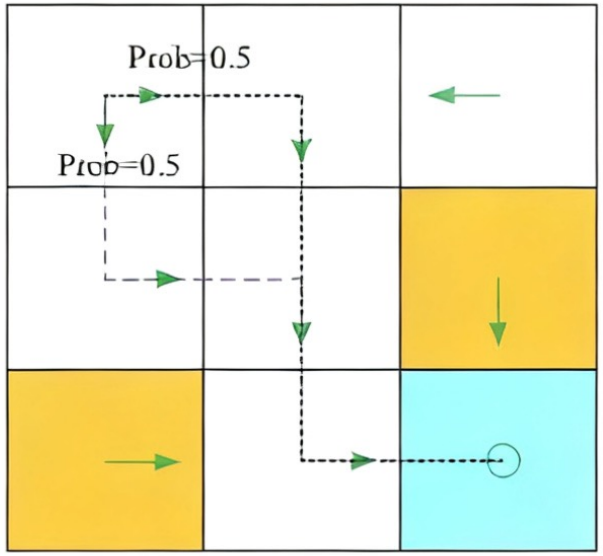
$$\pi(a_2|s_1) = 0$$

$$\pi(a_3|s_1) = 0$$

$$\pi(a_4|s_1) = 0$$

$$\pi(a_5|s_1) = 0$$

Grid World



Stochastic Policy

$$\pi(a_1|s_1) = 0$$

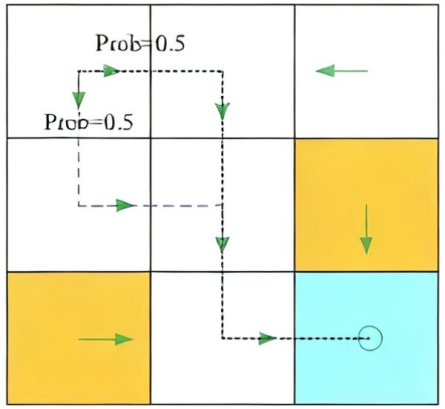
$$\pi(a_2|s_1) = 0.5$$

$$\pi(a_3|s_1) = 0.5$$

$$\pi(a_4|s_1) = 0$$

$$\pi(a_5|s_1) = 0$$

Grid World

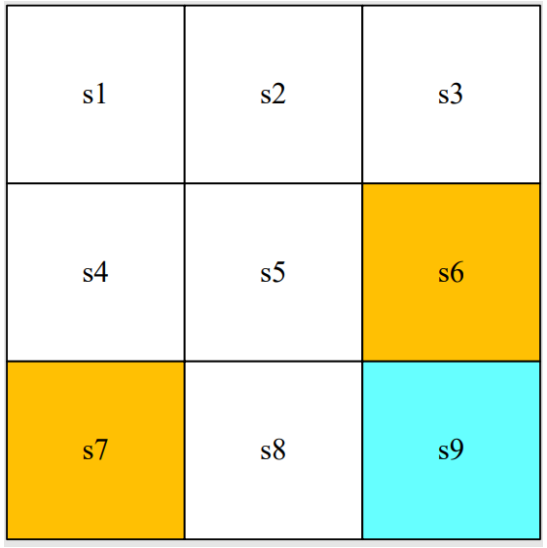


策略的表格表示

	a_1 (upwards)	a_2 (rightwards)	a_3 (downwards)	a_4 (leftwards)	a_5 (unchanged)
s_1	0	0.5	0.5	0	0
s_2	0	0	1	0	0
s_3	0	0	0	1	0
s_4	0	1	0	0	0
s_5	0	0	1	0	0
s_6	0	0	1	0	0
s_7	0	1	0	0	0
s_8	0	1	0	0	0
s_9	0	0	0	0	1

Grid World

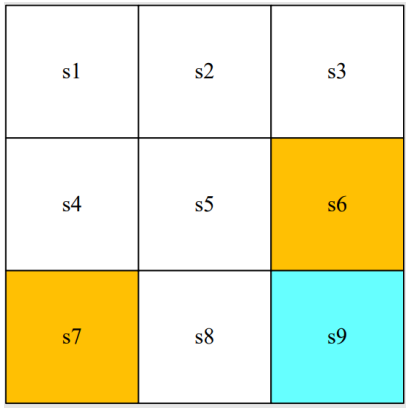
如何设计奖励函数?



在网格世界 (grid-world) 示例中，奖励函数的设计如下：

- 若Agent尝试越出边界，则奖励 $r_{bound} = -1$
- 若Agent尝试进入禁止单元，则奖励 $r_{forbid} = -1$
- 若Agent到达目标单元，则奖励 $r_{target} = 1$
- 其他情况下，奖励 $r = 0$

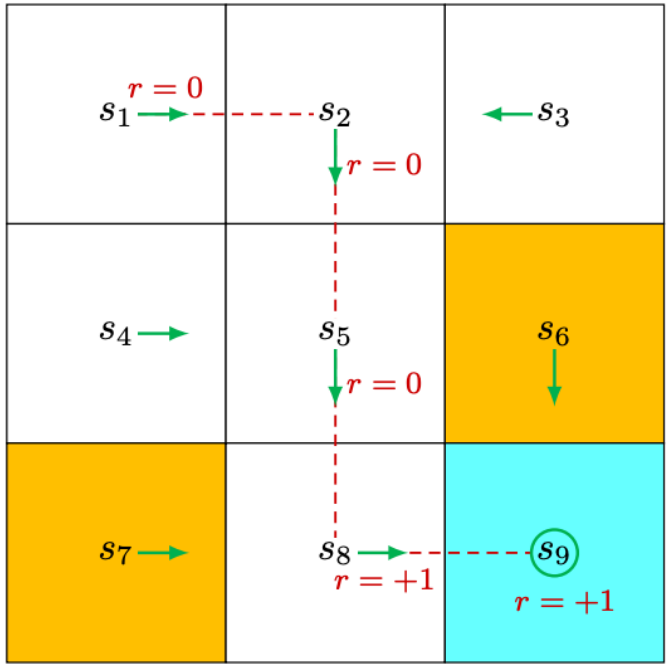
Grid World

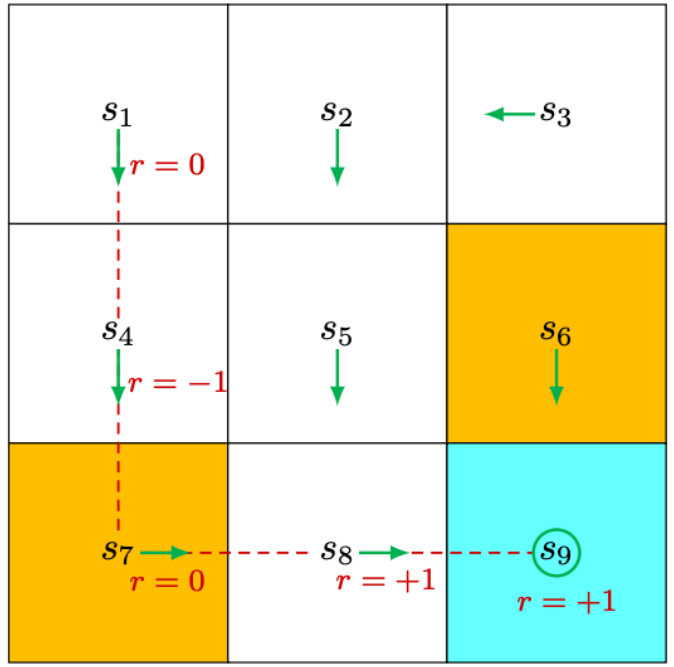


奖励的表格表示

	a_1 (upward)	a_2 (rightward)	a_3 (downward)	a_4 (leftward)	a_5 (still)
s_1	r_{boundary}	0	0	r_{boundary}	0
s_2	r_{boundary}	0	0	0	0
s_3	r_{boundary}	r_{boundary}	$r_{\text{forbidden}}$	0	0
s_4	0	0	$r_{\text{forbidden}}$	r_{boundary}	0
s_5	0	$r_{\text{forbidden}}$	0	0	0
s_6	0	r_{boundary}	r_{target}	0	$r_{\text{forbidden}}$
s_7	0	0	r_{boundary}	r_{boundary}	$r_{\text{forbidden}}$
s_8	0	r_{target}	r_{boundary}	$r_{\text{forbidden}}$	0
s_9	$r_{\text{forbidden}}$	r_{boundary}	r_{boundary}	0	r_{target}

Grid World: Trajectory and Return





$$s_1 \xrightarrow[r=0]{a_2} s_2 \xrightarrow[r=0]{a_3} s_5 \xrightarrow[r=0]{a_3} s_8 \xrightarrow[r=1]{a_2} s_9$$

$$s_1 \xrightarrow[r=0]{a_3} s_4 \xrightarrow[r=-1]{a_3} s_7 \xrightarrow[r=0]{a_2} s_8 \xrightarrow[r=+1]{a_2} s_9$$

return = 0 + 0 + 0 + 1 = 1

return = 0 - 1 + 0 + 1 = 0

有模型学习

- 目标：选择能够最大化累积奖励期望的策略$\pi(s): S \rightarrow A$

$$\mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots]$$

- 三个步骤：
  - 策略评估
  - 策略改进
  - 策略迭代

策略评估：给定一个策略，如何评估好坏?

价值函数(Value Function)

- 给策略π定义价值函数，价值函数是状态（或状态-动作二元组）的函数，用来评估当前智能体在给定状态（获给定状态与动作）下有多好，这里的“好”是基于未来预期的汇报（累计奖励）来定义的

- 状态值函数(State value function)

$$V^{\pi}(s_t) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s = s_t]$$

- 状态-动作值函数(State-Action value function)

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s = s_t, a = a_t]$$

- 二者的关系：

$$V^{\pi}(s_t) = \sum_a \pi(s_t, a) \cdot Q^{\pi}(s_t, a)$$

贝尔曼方程(Bellman Equation)

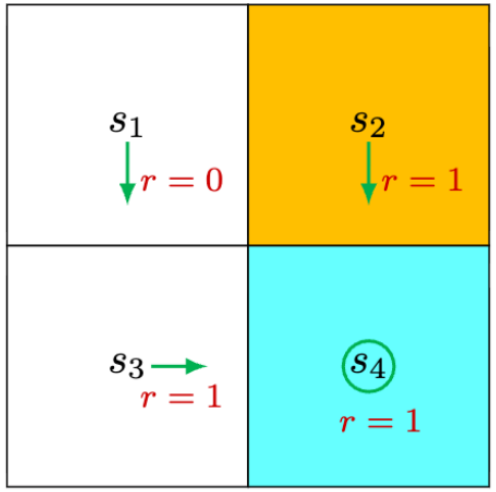
$$V^{\pi}(s) = \mathbb{E}[r_1 + \underbrace{\gamma r_2 + \gamma^2 r_3 + \cdots}_{\gamma V^{\pi}(s_{t+1})} | s_0 = s]$$

$$= \sum_{a} \boldsymbol{\pi(s,a)} \sum_{s' \in S} \boldsymbol{P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right)}$$

Bellman等式

$$\boldsymbol{Q^{\pi}(s,a) = \sum_{s' \in S} P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right)}$$

贝尔曼方程示例



贝尔曼方程： $V^{\pi}(s_t) = \sum_a \pi(s,a) \sum_{s' \in S} P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right)$ **General Case**

$$V^{\pi}(s_1) = 0 + \gamma V^{\pi}(s_3)$$
$$V^{\pi}(s_2) = 1 + \gamma V^{\pi}(s_4)$$
$$V^{\pi}(s_3) = 1 + \gamma V^{\pi}(s_4)$$
$$V^{\pi}(S_4) = 1 + \gamma V^{\pi}(s_4)$$

贝尔曼方程示例

如何求解?

$$V^{\pi}(s_1) = 0 + \gamma V^{\pi}(s_3)$$
$$V^{\pi}(s_2) = 1 + \gamma V^{\pi}(s_4)$$
$$V^{\pi}(s_3) = 1 + \gamma V^{\pi}(s_4)$$
$$V^{\pi}(S_4) = 1 + \gamma V^{\pi}(s_4)$$

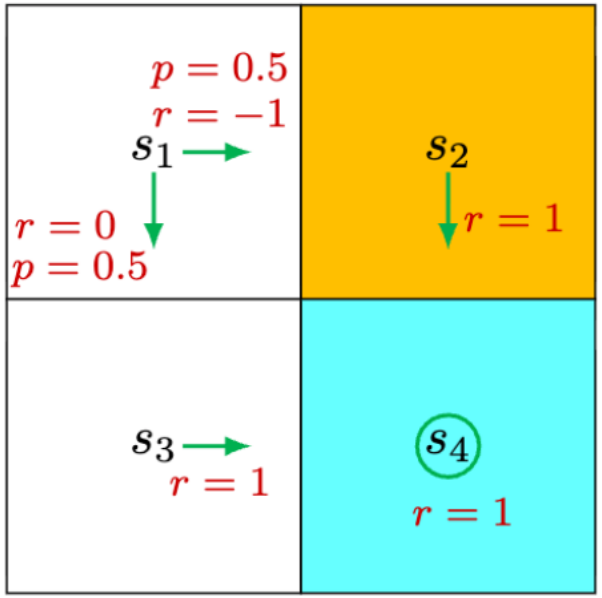
$$V^{\pi}(S_4) = \frac{1}{1-\gamma}$$
$$V^{\pi}(S_3) = \frac{1}{1-\gamma}$$
$$V^{\pi}(S_2) = \frac{1}{1-\gamma}$$
$$V^{\pi}(S_1) = \frac{\gamma}{1-\gamma}$$

贝尔曼方程示例

令$\gamma = 0.9$,

$$V^{\pi}(S_4) = \frac{1}{1-0.9} = 10$$
$$V^{\pi}(S_3) = \frac{1}{1-0.9} = 10$$
$$V^{\pi}(S_2) = \frac{1}{1-0.9} = 10$$
$$V^{\pi}(S_1) = \frac{0.9}{1-0.9} = 9$$

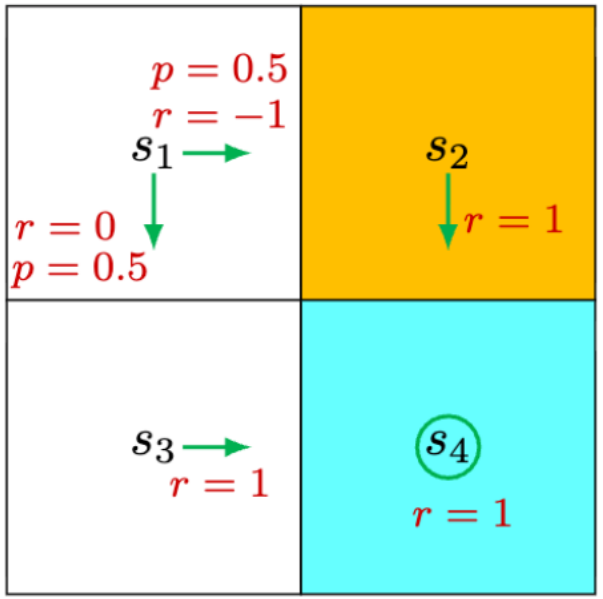
Exercise



贝尔曼方程： $V^{\pi}(s_t) = \sum_a \pi(s,a) \sum_{s' \in S} P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right)$ **General Case**

- 写出每个状态的贝尔曼方程
- 计算状态值
- 该策略和上一例子中的策略相比如何?

Solution



$$V^{\pi}(s_1) = 0.5[0 + \gamma V^{\pi}(s_3)] + 0.5[-1 + \gamma V^{\pi}(s_2)]$$

$$V^{\pi}(s_2) = 1 + \gamma V^{\pi}(s_4)$$

$$V^{\pi}(s_3) = 1 + \gamma V^{\pi}(s_4)$$

$$V^{\pi}(S_4) = 1 + \gamma V^{\pi}(s_4)$$

求解上述方程可得：

$$V^{\pi}(S_2) = V^{\pi}(S_3) = V^{\pi}(S_4) = \frac{1}{1-\gamma}$$

$$V^{\pi}(S_1) = -0.5 + \frac{\gamma}{1-\gamma}$$

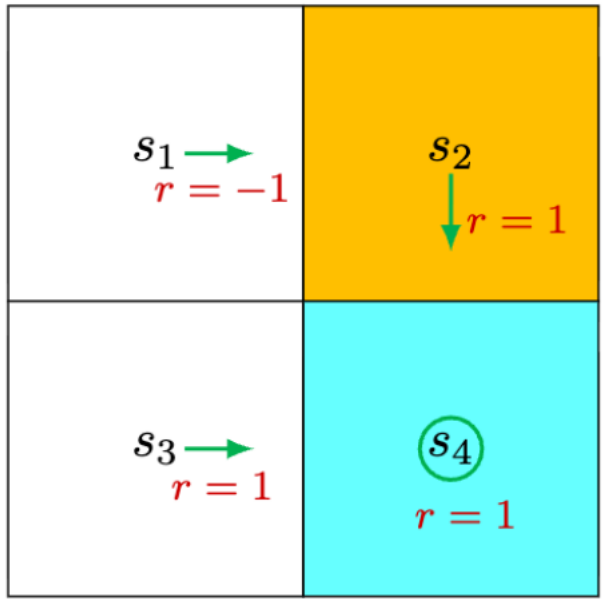
令γ=0.9可得

$$V^{\pi}(S_2) = V^{\pi}(S_3) = V^{\pi}(S_4) = 10$$

$$V^{\pi}(S_1) = -0.5 + 9 = 8.5$$

策略改进：如何改进一个策略?

示例

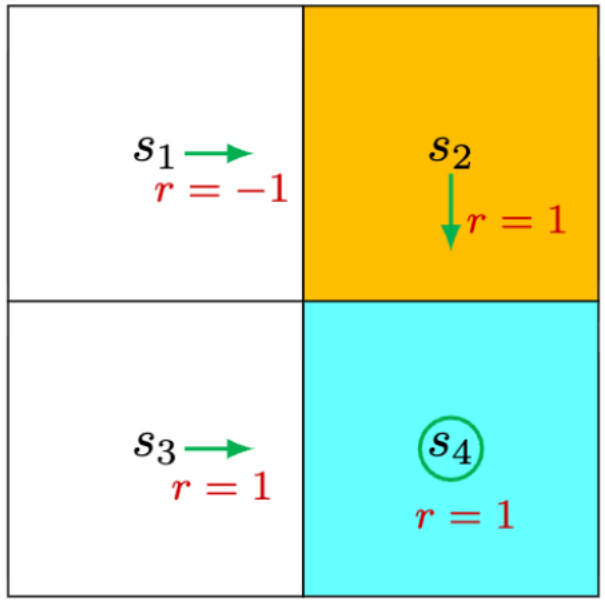


$$V^{\pi}(s_1) = -1 + \gamma V^{\pi}(s_2)$$
$$V^{\pi}(s_2) = 1 + \gamma V^{\pi}(s_4)$$
$$V^{\pi}(s_3) = 1 + \gamma V^{\pi}(s_4)$$
$$V^{\pi}(S_4) = 1 + \gamma V^{\pi}(s_4)$$

令$\gamma = 0.9$，

$V^{\pi}(s_1) = 8$，$V^{\pi}(s_2) = V^{\pi}(s_3) = V^{\pi}(4) = 10$

示例



令$\gamma = 0.9$，

$V^{\pi}(s_1) = 8$

$V^{\pi}(s_2) = V^{\pi}(s_3) = V^{\pi}(4) = 10$

$q^{\pi}(s_1, a_1) = -1 + \gamma V^{\pi}(s_1) = 6.2$

$q^{\pi}(s_1, a_2) = -1 + \gamma V^{\pi}(s_2) = 8$

$\boldsymbol{q^{\pi}(s_1, a_3) = 0 + \gamma V^{\pi}(s_3) = 9}$

$q^{\pi}(s_1, a_4) = -1 + \gamma V^{\pi}(s_1) = 6.2$

$q^{\pi}(s_1, a_5) = 0 + \gamma V^{\pi}(s_1) = 7.2$

如果我们更新策略，使之选择具有最大动作值的动作，就有望得到一个更好的策略

贝尔曼最优方程

- 最优策略对应的值函数称为最优值函数

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

- 最优价值函数的Bellman等式

$$V^*(s) = \max_{a} \sum_{s' \in S} P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right) = \max_{a} Q^{\pi *}(s,a)$$

最优Bellman等式

- 非最优策略的改进方式：将策略选择的动作改为当前最优的动作

$$\pi'(s) = \arg\max_{a \in A} Q^{\pi}(s,a)$$

价值迭代 (Value Iteration)

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$

- 价值迭代过程

1. 对每个状态s，初始化 $V(s)$=0
2. 重复以下过程直到收敛 {

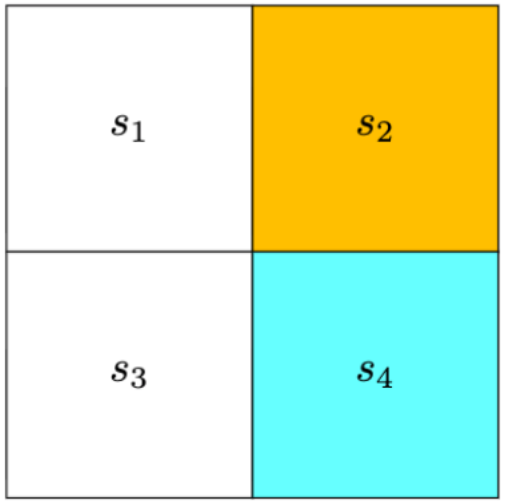
对每个状态，更新

$$V(s) = \max_a \sum_{s' \in S} P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right) = \max_a Q^{\pi}(s,a)$$

}

价值迭代 (Value Iteration)

网格世界



Q表

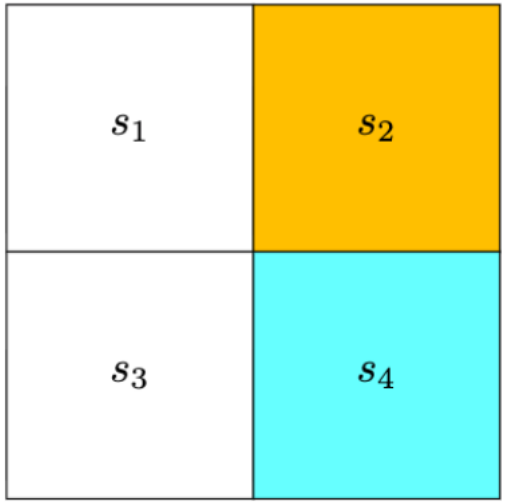
q-table	a_1	a_2	a_3	a_4	a_5
s_1	$-1+\gamma v(s_1)$	$-1+\gamma v(s_2)$	$0+\gamma v(s_3)$	$-1+\gamma v(s_1)$	$0+\gamma v(s_1)$
s_2	$-1+\gamma v(s_2)$	$-1+\gamma v(s_2)$	$1+\gamma v(s_4)$	$0+\gamma v(s_1)$	$-1+\gamma v(s_2)$
s_3	$0+\gamma v(s_1)$	$1+\gamma v(s_4)$	$-1+\gamma v(s_3)$	$-1+\gamma v(s_3)$	$0+\gamma v(s_3)$
s_4	$-1+\gamma v(s_2)$	$-1+\gamma v(s_4)$	$-1+\gamma v(s_4)$	$0+\gamma v(s_3)$	$1+\gamma v(s_4)$

价值迭代 (Value Iteration)

K=0：初始值v_0可任意选择，不失一般性，令

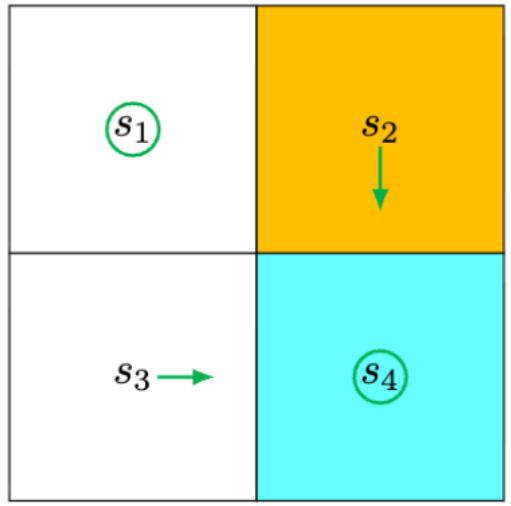
$$v_0(s_1) = v_0(s_2) = v_0(s_3) = v_0(s_4) = 0$$

q-table	a_1	a_2	a_3	a_4	a_5
s_1	-1	-1	0	-1	0
s_2	-1	-1	1	0	-1
s_3	0	1	-1	-1	0
s_4	-1	-1	-1	0	1



价值更新：

$$V(s) = \max_a \sum_{s' \in S} P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right) = \max_a Q^{\pi}(s,a)$$

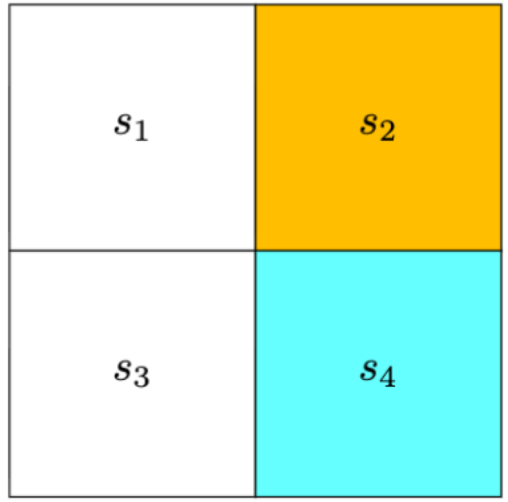


$$v_1(s_1) = 0, v_1(s_2) = 1, v_1(s_3) = 1, v_1(s_4) = 1$$

价值迭代 (Value Iteration)

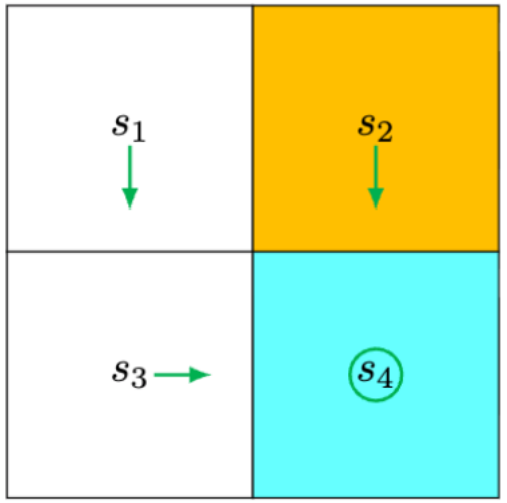
K=1： $v_1(s_1) = 0, v_1(s_2) = 1, v_1(s_3) = 1, v_1(s_4) = 1$

q-table	a_1	a_2	a_3	a_4	a_5
s_1	$-1+\gamma 0$	$-1+\gamma 1$	$0+\gamma 1$	$-1+\gamma 0$	$0+\gamma 0$
s_2	$-1+\gamma 1$	$-1+\gamma 1$	$1+\gamma 1$	$0+\gamma 0$	$-1+\gamma 1$
s_3	$0+\gamma 0$	$1+\gamma 1$	$-1+\gamma 1$	$-1+\gamma 1$	$0+\gamma 1$
s_4	$-1+\gamma 1$	$-1+\gamma 1$	$-1+\gamma 1$	$0+\gamma 1$	$1+\gamma 1$



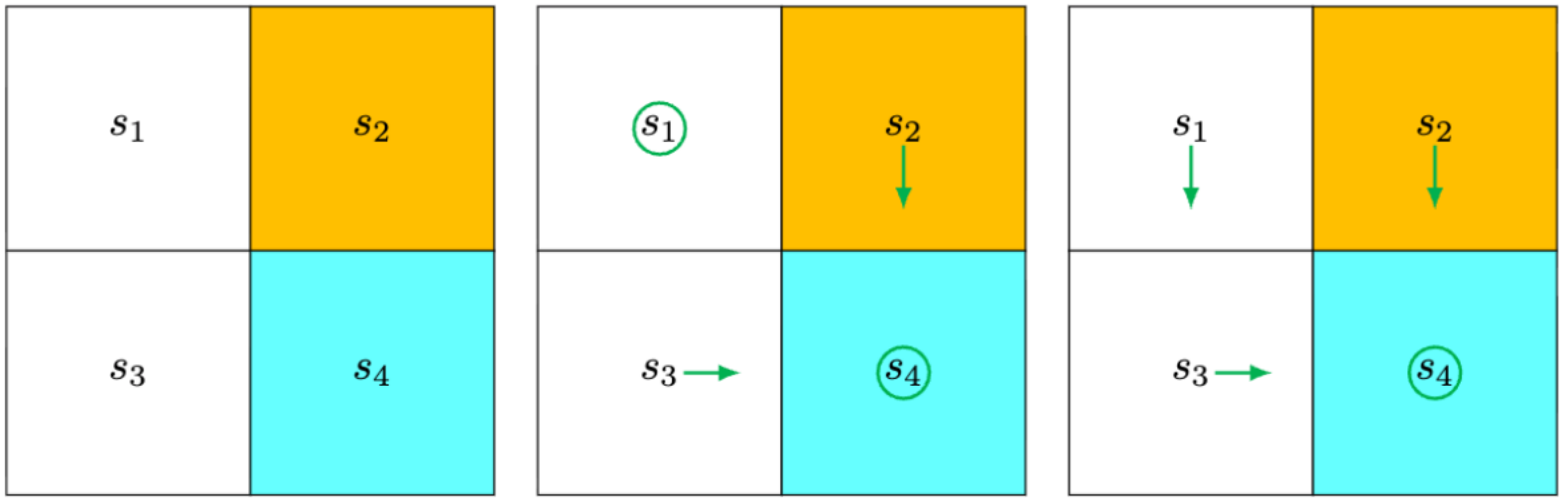
价值更新：

$$V(s) = \max_a \sum_{s' \in S} P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right) = \max_a Q^{\pi}(s,a)$$



$v_2(s_1) = \gamma 1, v_2(s_2) = 1 + \gamma 1, v_2(s_3) = 1 + \gamma 1, v_2(s_4) = 1 + \gamma 1$

价值迭代 (Value Iteration)

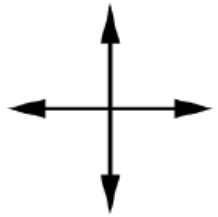


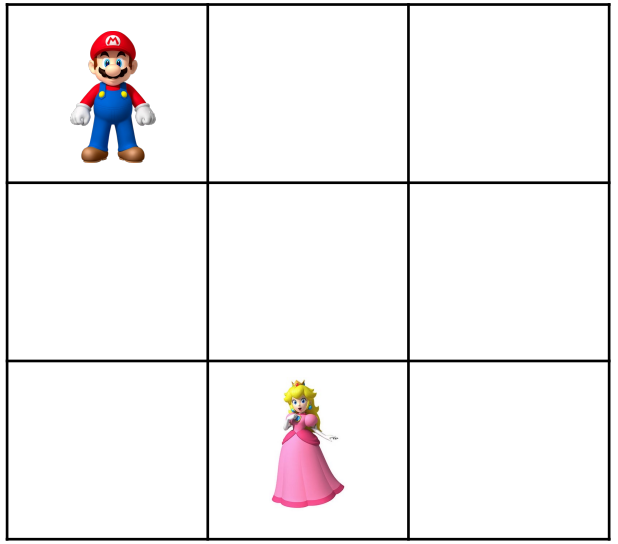
$$v_0(s_1) = v_0(s_2) = v_0(s_3) = v_0(s_4) = 0$$

$$v_1(s_1) = 0, v_1(s_2) = 1, v_1(s_3) = 1, v_1(s_4) = 1$$

$$v_2(s_1) = \gamma 1, v_2(s_2) = 1 + \gamma 1, v_2(s_3) = 1 + \gamma 1, v_2(s_4) = 1 + \gamma 1$$

Exercise: 价值迭代之公主营救





- 非折扣MDP（$\gamma = 1$）
- 每走一步，奖励为-1，直到到达终止状态
- 找到公主游戏结束
- 最小体力消耗找到公主
- 智能体的策略为均匀随机策略

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Exercise: 价值迭代之公主营救

 0	0	0
0	0	0
0	0 	0

Exercise: 价值迭代之公主营救

0	0	0
0	0	0
0	0	0

-1	-1	-1
-1	-1	-1
-1	0	-1

-2	-2	-2
-2	-1	-2
-1	0	-1

-3	-2	-3
-2	-1	-2
-1	0	-1

-3	-2	-3
-2	-1	-2
-1	0	-1

策略迭代 (Policy Iteration)

- 对于一个动作空间和状态空间有限的MDP

$$|S| < \infty, |A| < \infty$$

- 策略迭代过程

1. 随机初始化策略 π
2. 重复以下过程直到收敛 {

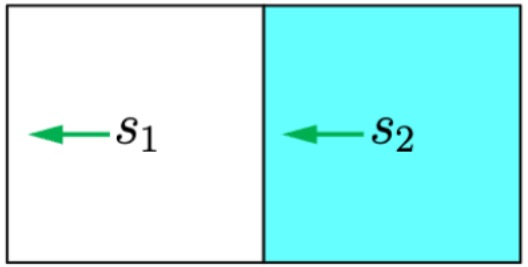
a) 策略评估：$\boldsymbol{V := V^{\pi}} = \sum_{a} \boldsymbol{\pi(s,a)} \sum_{s' \in S} \boldsymbol{P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right)}$

b) 策略改进：对每个状态，更新

$$\boldsymbol{\pi(s) = arg\max_{a \in A} Q^{\pi}(s,a)}$$

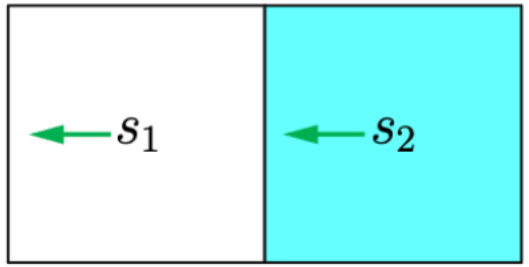
}

策略迭代：网格世界



- 动作：向左、向右、不动
- $r_{boundary} = -1, r_{target} = 1$
- $\gamma = 0.9$

策略迭代：网格世界



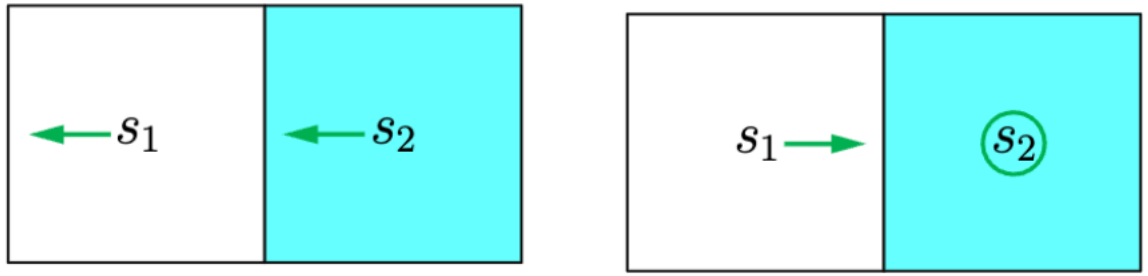
策略评估

$$V^{\pi_0}(s_1) = -1 + \gamma V^{\pi_0}(s_1)$$

$$V^{\pi_0}(s_2) = 0 + \gamma V^{\pi_0}(s_2)$$

$$V^{\pi_0}(s_1) = -10, \qquad V^{\pi_0}(s_2) = -9$$

策略迭代：网格世界



策略改进

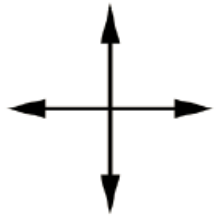
$q_{\pi_k}(s,a)$	a_ℓ	a_0	a_r
s_1	$-1+\gamma v_{\pi_k}(s_1)$	$0+\gamma v_{\pi_k}(s_1)$	$1+\gamma v_{\pi_k}(s_2)$
s_2	$0+\gamma v_{\pi_k}(s_1)$	$1+\gamma v_{\pi_k}(s_2)$	$-1+\gamma v_{\pi_k}(s_2)$

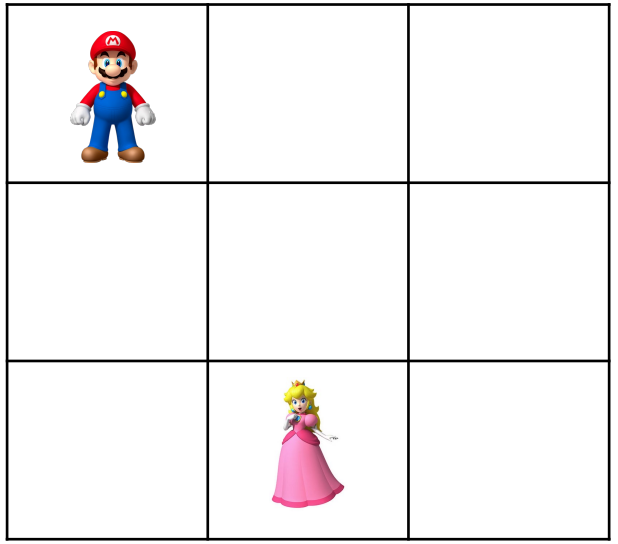
代入$V^{\pi_0}(s_1)=-10, V^{\pi_0}(s_2)=-9$，可得

$q_{\pi_0}(s,a)$	a_ℓ	a_0	a_r
s_1	-10	-9	-7.1
s_2	-9	-7.1	-9.1

在状态s_1应该选择向右，在状态s_2应该选择不动

Exercise：策略迭代之公主营救

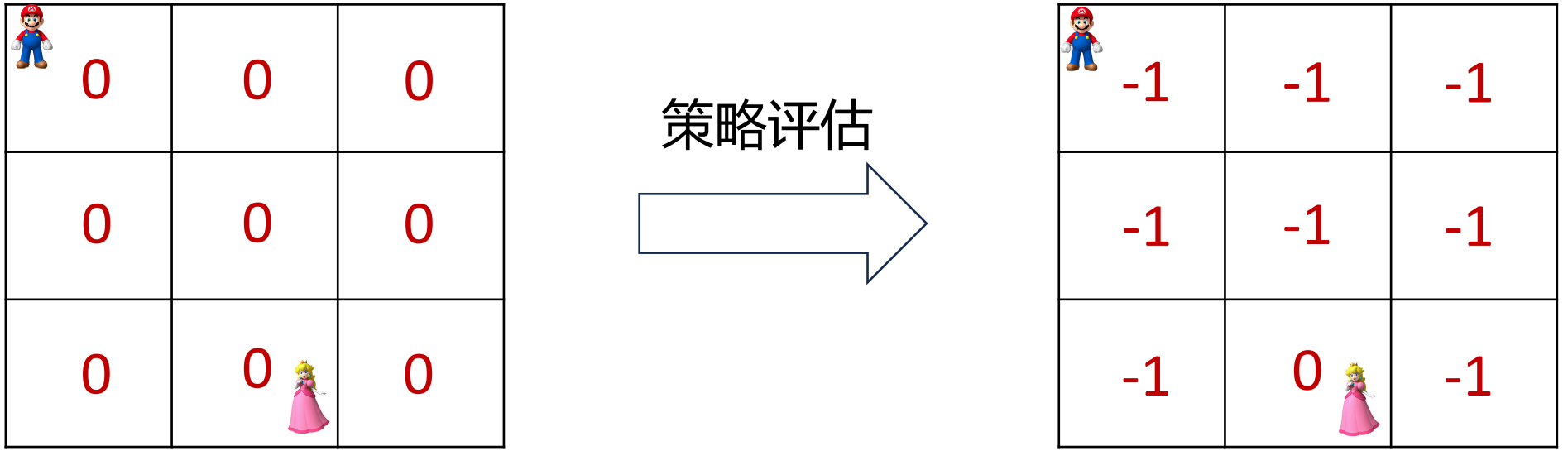




- 非折扣MDP（$\gamma = 1$）
- 每走一步，奖励为-1，直到到达终止状态
- 找到公主游戏结束
- 最小体力消耗找到公主
- 智能体的策略为均匀随机策略

$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

Exercise：策略迭代之公主营救



Exercise：策略迭代之公主营救

-1	-1	-1
-1	-1	-1
-1	0	-1

-2	-2	-2
-2	-1.75	-2
-2	0	-1.75

Exercise：策略迭代之公主营救

-2	-2	-2
-2	-1.75	-2
-2	0	-1.75

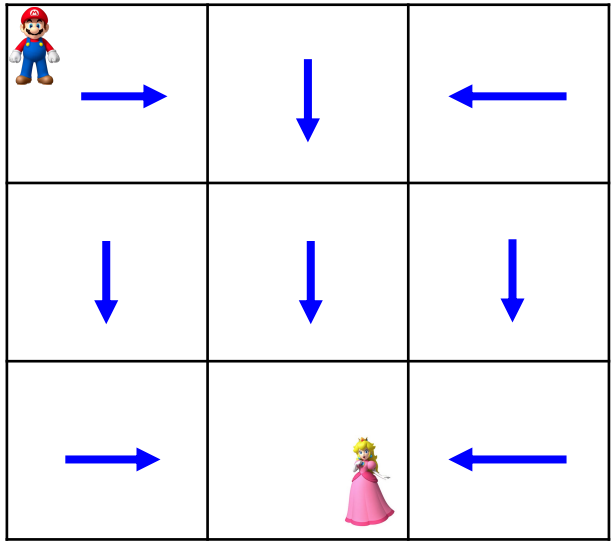
迭代至收敛

-3	-2.6	-3
-2.9	-2.5	-2.9
-2.4	0	-2.4

Exercise：策略迭代之公主营救

-3	-2.6	-3
-2.9	-2.5	-2.9
-2.4	0	-2.4

策略提升



价值迭代 vs. 策略迭代

- 价值迭代过程

  1. 对每个状态s，初始化 $V(s)$=0

  2. 重复以下过程直到收敛 {

     对每个状态，更新

  $$V(s) = \max_a \sum_{s' \in S} P_{sa}(s')\left(r_{sa}(s') + \gamma V^{\pi}(s')\right)$$

  }

- 策略迭代过程

  1. 随机初始化策略 π

  2. 重复以下过程直到收敛 {

     a) 策略评估：$\boldsymbol{V} := \boldsymbol{V^{\pi}}$

     b) 策略改进：对每个状态，更新

  $$\pi(s) = arg\max_{a \in A} Q^{\pi}(s, a)$$

  }

- ✓ 均可归结为基于动态规划的寻优算法
- ✓ 价值迭代是贪心更新法
- ✓ 策略迭代中，用Bellman等式更新价值函数代价很大
- ✓ 对于空间较小的MDP，策略迭代通常很快收敛
- ✓ 对于空间较大的MDP，价值迭代更实用（效率更高）

如果模型未知呢?

- 在模型未知的情形下，从目前我们关注在给出一个已知MDP模型后:

  (也就是说，**状态转移**明确给定)

  - 计算最优价值函数
  - 学习最优策略

- 现实问题中，通常难以明确地给出状态转移和奖励函数,导致策略无法评估

  (模型未知导致无法做全概率展开)

学习一个MDP模型

Episode1：$s_0^{(1)} \xrightarrow{a_0^{(1)},R(s_0)^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)},R(s_1)^{(1)}} s_2^{(1)} \xrightarrow{a_2^{(1)},R(s_2)^{(1)}} s_3^{(1)} \cdots s_T^{(1)}$

Episode2：$s_0^{(2)} \xrightarrow{a_0^{(2)},R(s_0)^{(2)}} s_1^{(2)} \xrightarrow{a_1^{(2)},R(s_1)^{(2)}} s_2^{(2)} \xrightarrow{a_2^{(2)},R(s_2)^{(2)}} s_3^{(2)} \cdots s_T^{(2)}$

⋮ ⋮

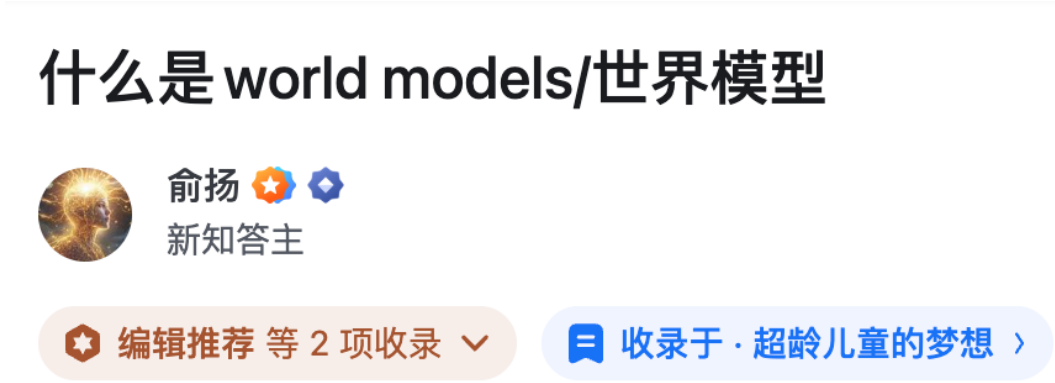
- 从“经验”中学习一个MDP模型：
  - 学习**状态转移概率**

$$P_{sa}(s') = \frac{\textbf{在}s\textbf{下采取动作}a\textbf{并转移到}s'\textbf{的次数}}{\textbf{在}s\textbf{下采取动作}a\textbf{的次数}}$$

世界模型(World Model)

“所谓世界模型，就是世界在人脑或AI的神经网络中的内部表征（心理模型）。在大脑里对外部世界形成抽象表示，形成内部模拟，这可以用于理解、预测和规划外部环境的交互。

大脑构建了一个关于世界如何运作的、经过压缩的、可预测的内部模型。人知道苹果熟了会掉下来而不是飞上去，sora等视频生成AI知道人走在水面会沉下去而不是如履平地，这都是“世界模型”的作用“”



https://zhuanlan.zhihu.com/p/661768957

有没有办法不学习MDP?

能不能从经验数据中直接学习价值函数和策略函数?

模型无关的强化学习
(Model-Free Reinforcement Learning)

大纲

- 强化学习简介
- 有模型学习
  - 马尔科夫决策过程
  - 贝尔曼方程
  - 值迭代
  - 策略迭代
- **无模型学习**
  - 蒙特卡洛强化学习
  - 时序差分学习

价值学习与策略学习

- 价值学习（Value-based learning）通常是指学习最优价值函数$Q^*(s,a)$，假设我们有了Q^*，智能体就可以根据Q^*来做决策，选出最好的动作

- 例如，以超级马里奥为例，每次观测到一个状态s_t，让Q^*对所有动作做评价：

$$Q_\star(s_t, \text{左}) = 273, \qquad Q_\star(s_t, \text{右}) = -139, \qquad Q_\star(s_t, \text{上}) = 195$$

- 策略学习（Policy-based learning）通常是指学习最优策略函数$\pi^*(a|s)$，假设我们有了π^*，我们可以直接算出所有动作的概率值，然后去采样执行

$$\pi(\text{左} \mid s_t) = 0.6, \qquad \pi(\text{右} \mid s_t) = 0.1, \qquad \pi(\text{上} \mid s_t) = 0.3$$

蒙特卡洛强化学习

(Monte Carlo Reinforcement Learning)

蒙特卡洛强化学习

- 在模型未知的情形下，从起始状态出发，使用某个策略进行采样，获得轨迹
  - Episode 1：$s_1, a_1, r_1,\ s_2, a_2, r_2,\ \cdots\ ,\ s_T, a_T, r_T$
  - …
- **对于轨迹中出现的每一对状态-动作，记录其奖赏之和，作为该状态-动作对的一次累计奖赏采样值**
- **多次采样得到多条轨迹之后，将每个状态-动作对的累计奖赏采样值进行平均，得到状态-动作值函数Q的估计**

蒙特卡洛强化学习

- **从某个状态-动作出发，执行策略得到一个回合的轨迹，基于该回合数据估计状态-动作的价值、**

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots$$

- **一个长的回合可以看做很多个从不同状态-动作出发的子回合，可以用于估计多个不同状态-动作的值；样本效率更高**

$$\begin{aligned}
s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad & [\text{original episode}] \\
s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad & [\text{subepisode starting from } (s_2, a_4)] \\
s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad & [\text{subepisode starting from } (s_1, a_2)] \\
s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots \quad & [\text{subepisode starting from } (s_2, a_3)] \\
s_5 \xrightarrow{a_1} \dots \quad & [\text{subepisode starting from } (s_5, a_1)]
\end{aligned}$$

蒙特卡洛强化学习

$$s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_4} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_5 \xrightarrow{a_1} \dots$$

- 访问在一个回合中的每个时间步长t的状态s
  - 增量计数器 $N(s) \leftarrow N(s) + 1$
  - 增量总累计奖励 $S(s) \leftarrow S(s) + G_t$
  - 价值被估计为累计奖励的均值 $V(s) = S(s)/N(s)$

- 同理，可计算出$Q(s, a)$

蒙特卡洛强化学习

- 在模型未知的情形下，从起始状态出发，使用某个策略进行采样，获得轨迹
  - Episode 1：$s_1, a_1, r_1,\ s_2, a_2, r_2,\ \cdots\ ,\ s_T, a_T, r_T$
  - …

如果策略是确定性的，对于某个状态只会输出一个动作，使用这样的策略进行采样，只能得到多条相同的轨迹

蒙特卡洛强化学习

引入ϵ-贪心算法

$$\pi^{\epsilon}(s) = \begin{cases} \pi(s), & \text{以概率} 1-\epsilon \\ \text{以均匀概率从} A \text{中随机选取动作} & \text{以概率} \epsilon \end{cases}$$

1. **轨迹采样**：按照当前策略的ϵ-贪心版本与环境交互，得到 episode
2. **策略评估**：根据采样的轨迹更新 $\mathrm{Q}(s,a)$的估计
3. **策略改进**：根据新的Q，更新策略

蒙特卡洛强化学习

- 同策略蒙特卡洛强化学习：同策略(on-policy)，被评估和改进的是同一个策略

循环{

- 根据策略π的ϵ贪心版本采样一条轨迹　　**行为策略**
- 更新$Q(s,a)$
- $$\pi(s)=\begin{cases}\arg\max_a Q(s,a) & \text{以概率}1-\epsilon \\ \text{以均匀概率从}A\text{中随机选取动作} & \text{以概率}\epsilon\end{cases}$$

}

目标策略

异策略(off-policy)蒙特卡洛强化学习

- 能不能仅在评估时引入ϵ贪心策略，策略改进时改进原始策略

- 估计一个不同分布的期望

$$\begin{aligned}\mathbb{E}_{x\sim p}[f(x)] &= \int_x p(x)f(x)dx \\ &= \int_x q(x)\frac{p(x)}{q(x)}f(x)dx \\ &= \mathbb{E}_{x\sim q}\left[\frac{p(x)}{q(x)}f(x)\right]\end{aligned}$$

重要性采样

- 将每个实例的权重重新分配为 $\beta(x)=\frac{p(x)}{q(x)}$

异策略(off-policy)蒙特卡洛强化学习

- 能不能仅在评估时引入ϵ贪心策略，策略改进时改进原始策略
- 使用策略π的采样轨迹来评估策略π，实际上就是对累计奖赏估计期望

$$Q(s,a)=\frac{1}{m}\sum_{i=1}^{m}r_i$$

- 使用策略π'的采样轨迹来评估策略π，仅需对累计奖赏加权

$$Q(s,a)=\frac{1}{m}\sum_{i=1}^{m}\frac{p_i^{\pi}}{p_i^{\pi'}}r_i$$

p_i^{π}和$p_i^{\pi'}$分别表示两个策略产生第i条轨迹的概率

增量蒙特卡洛更新

- 对于状态-动作对(s,a)，假定基于t个采样已经估计出 $Q_t^\pi(s,a)=\frac{1}{t}\sum_{i=1}^{t}G_i$
- 则在得到第$t+1$个采样时，有

$$Q_{t+1}^\pi(s,a)=Q_t^\pi(s,a)+\frac{1}{t+1}(G_{t+1}-Q_t^\pi(s,a))$$

- 更一般地，可以把$\frac{1}{t+1}$更换为系数a

$$Q_{t+1}^\pi(s,a)=Q_t^\pi(s,a)+\alpha\,(G_{t+1}-\ Q_t^\pi(s,a))$$

蒙特卡罗强化学习

- 蒙特卡罗强化学习通过考虑采样轨迹，克服了模型未知给策略估计造成的困难

  - 蒙特卡罗强化学习的缺点：低效
    - 求平均时以“批处理式”进行
    - 在一个完整的采样轨迹完成后才对状态-动作值函数进行更新

如何提升蒙特卡洛强化学习的效率呢?

时序差分 (temporal difference，TD) 学习：

同时结合了动态规划和蒙特卡罗方法的思想

能做到更高效的免模型学习

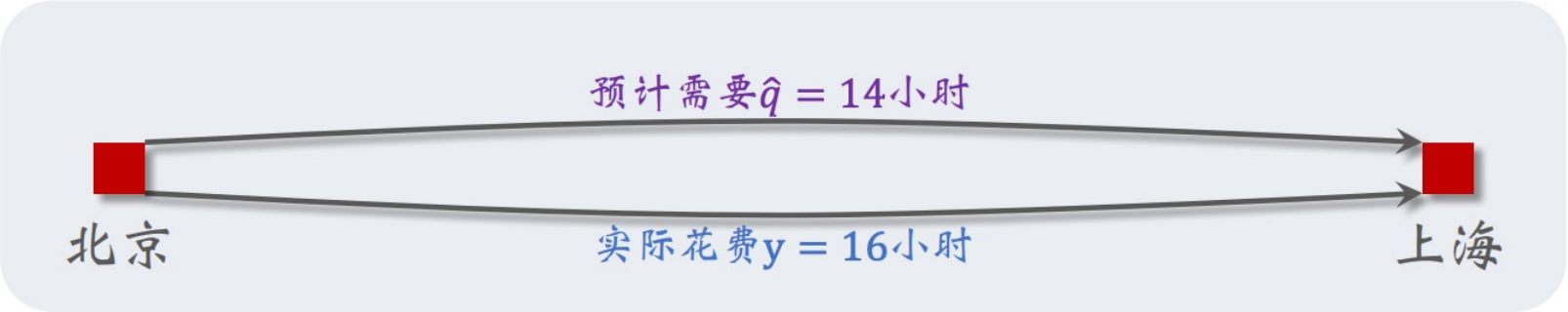
一个例子：驾车时间预测

- 假设我们有一个模型$Q(s,d)$，其中s是起点，d是终点，模型Q可以预测开车出行的时间
  开销，这个模型一开始不准确，但是得到更多数据之后，模型可以训练的越来越准

- 如何训练模型呢？在用户出发前，用户告诉模型起点和终点，模型做一个预测$\hat{q} = Q(s,d)$，
  当用户结束行程时，把实际用时y告诉模型，根据两者之差$y - \hat{q}$修正模型

驾车时间预测的例子

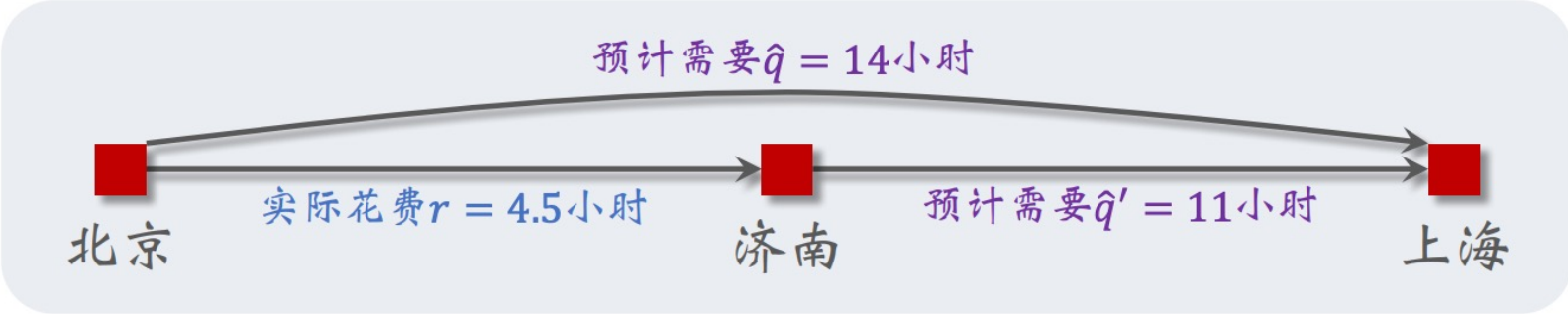
- 假设我要从北京开车去上海，出发前模型预测结果总车程为14小时，即$\hat{q} = 14$，到达上海时发现实际用时是16小时



- 基于两者之差（16-14）去更新模型，让模型预测更准

驾车时间预测的例子

- 出发前模型预测总车程为14小时，假设我过了4.5小时到达济南，让模型再做一次预测，模型告诉我从济南到上海需要11小时



- 根据模型的最新估计，整个旅程的总时间是15.5小时，$\hat{y} = r + \hat{q}' = 4.5 + 11 = 15.5$

- TD算法将$\hat{y} = 15.5$称为TD目标，它比最初的预测$\hat{q} = 14$更可靠

驾车时间预测的例子

- 因此，可以用$\hat{y} = 15.5$去对模型做修正，希望估计值$\hat{q}$尽量接近TD目标$\hat{y}$
- 基于两者之差（15.5-14）去更新模型
- 模型估计从北京到上海全程需要14小时，模型还估计从济南到上海全程需要11小时。相当于模型估计从北京到济南需要的时间为3小时
- 而实际花费4.5小时，模型估计与真实观测之差$\delta = 3 - 4.5 = -1.5$（TD误差）

时序差分学习

- 蒙特卡洛强化学习

$$Q_{t+1}^{\pi}(s,a) = Q_t^{\pi}(s,a) + \alpha\left(\boldsymbol{G_{t+1} - Q_t^{\pi}(s,a)}\right)$$

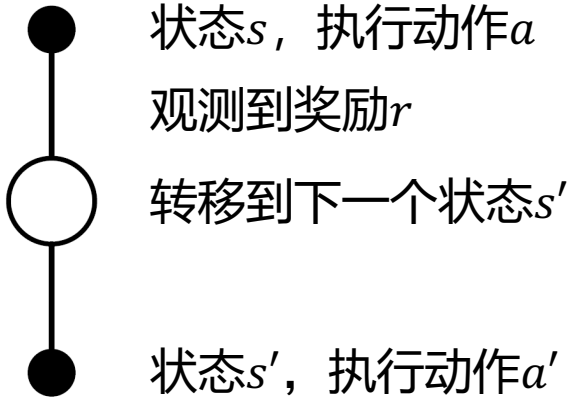
- 时序差分学习

$$Q_{t+1}^{\pi}(s,a) = Q_t^{\pi}(s,a) + \alpha\left(\boldsymbol{r_{sa}(s') + \gamma Q_t^{\pi}(s',a') - Q_t^{\pi}(s,a)}\right)$$

其中s'是前一次在状态s处执行动作a后转移到的状态， a'是策略π在s'选择的动作

SARSA

- 对于当前策略执行的每个（状态-动作-奖励-状态-动作）元组



- SARSA更新状态-动作值函数为

$$\boldsymbol{Q(s,a) \leftarrow Q(s,a) + \alpha(r_{sa}(s') + \gamma Q(s',a') - Q(s,a))}$$

SARSA

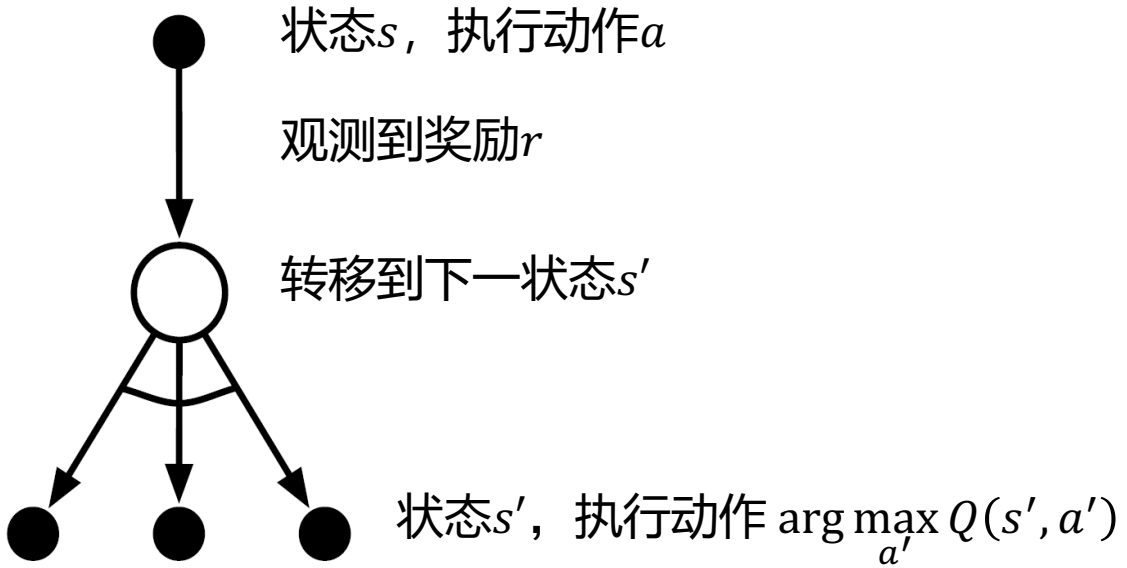
Sarsa: An on-policy TD control algorithm

Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

SARSA算法中的两个“A”都是由**当前策略**选择的

Q-Learning

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r_{sa}(s') + \gamma \max_{a'} Q(s',a') - Q(s,a))$$



- SARSA只能估计给定策略的动作值，必须结合策略改进步骤才能得到最优策略
- Q-Learning可以直接估计最优动作值进而找到最优策略

从表格到函数

$Q_\theta(s,a)$

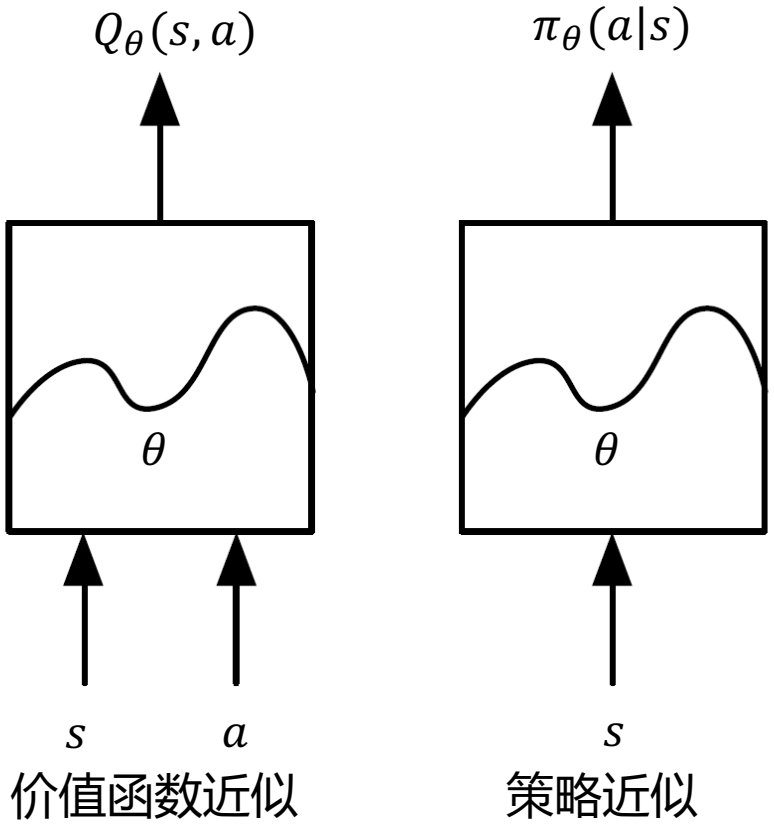
	a_1	a_2
s_1	3.2	4.1
s_2	0.8	1.2
s_3	2.3	-0.3

价值函数表格

$\pi_\theta(s,a)$

	a_1	a_2
s_1	0.2	0.8
s_2	0.4	0.6
s_3	0.9	0.1

策略表格



从表格到函数

$Q_\theta(s,a)$

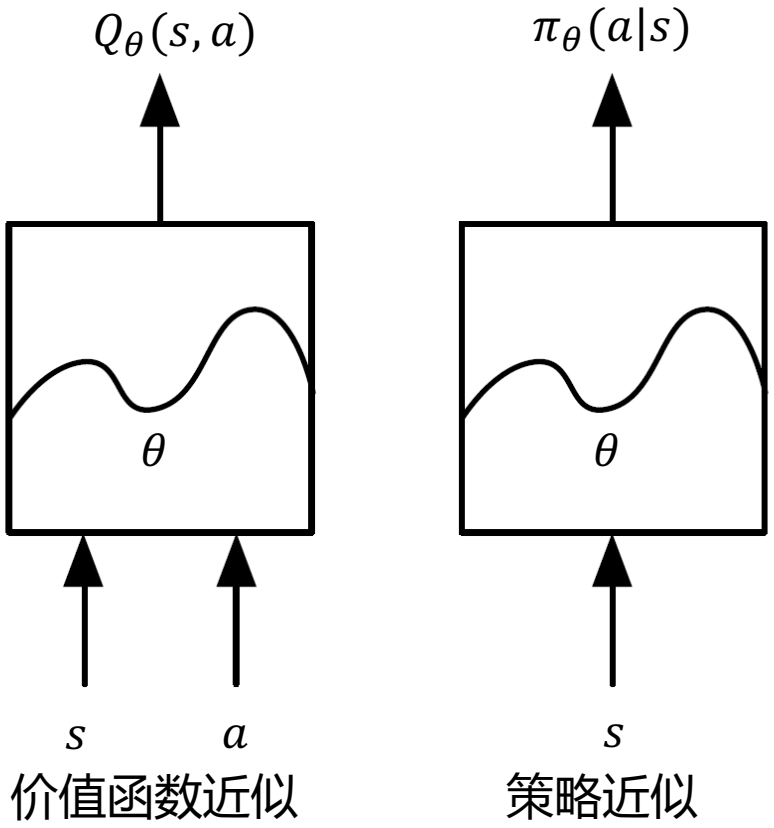
	a_1	a_2
s_1	3.2	4.1
s_2	0.8	1.2
s_3	2.3	-0.3

价值函数表格

$\pi_\theta(s,a)$

	a_1	a_2
s_1	0.2	0.8
s_2	0.4	0.6
s_3	0.9	0.1

策略表格



- 假如我们直接使用深度神经网络建立这些近似函数呢?
- **深度强化学习!**

深度强化学习

2013年12月，第一篇深度强化学习论文出自NIPS 2013

Reinforcement Learning Workshop

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

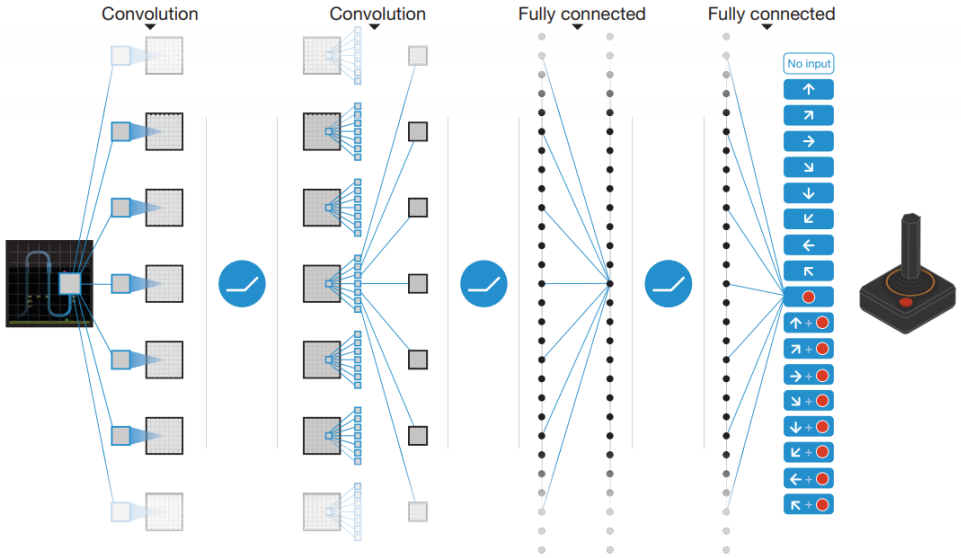
Daan Wierstra **Martin Riedmiller**

DeepMind Technologies

{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.

深度强化学习

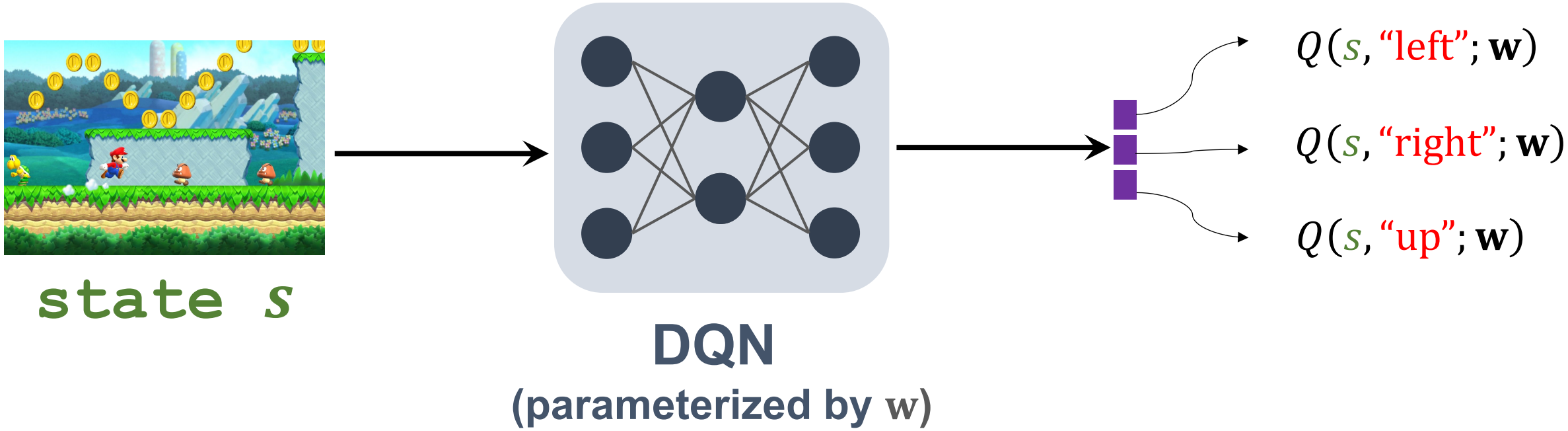


Q函数的参数通过神经网络反向传播学习

Volodymyr Mnih, Koray Kavukcuoglu, David Silver et al. Playing Atari with Deep Reinforcement Learning. NIPS 2013 workshop.

深度强化学习： DQN

DQN， $Q(s,a;\mathbf{w})$.是对最优动作价值函数 $Q^{\star}(s,a)$的近似



深度强化学习： DQN

DQN，$Q(s,a;\mathbf{w})$，是对最优动作价值函数 $Q^{\star}(s,a)$的近似

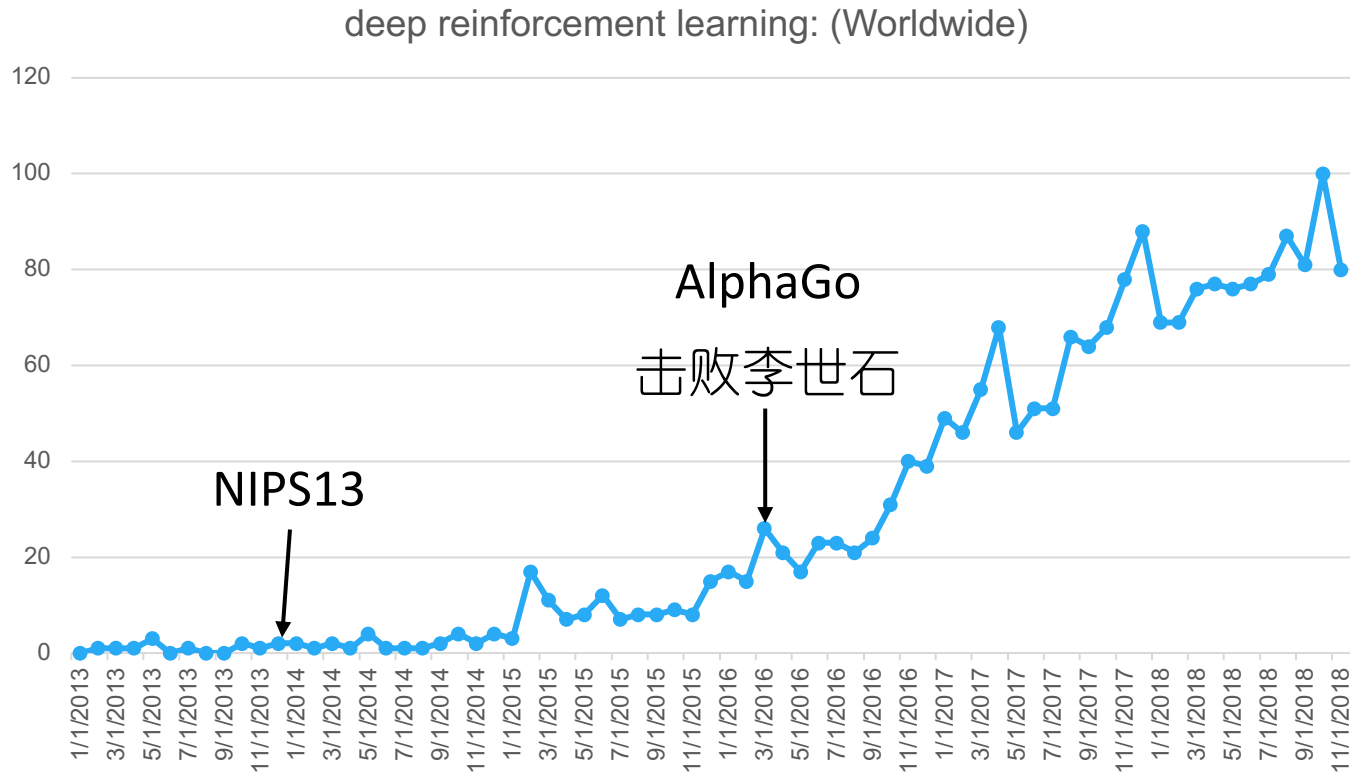
每次观察到一个transition：(s_t, a_t, r_t, s_{t+1})

TD 目标: $y_t = r_t + \gamma \cdot \max_{a} Q(s_{t+1}, a; \mathbf{w})$

TD 误差: $\delta_t = Q(s_t, a_t; \mathbf{w}) - y_t$

参数更新: $\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \frac{\partial\, Q(s_t, a_t; \mathbf{w})}{\partial\, \mathbf{w}}$

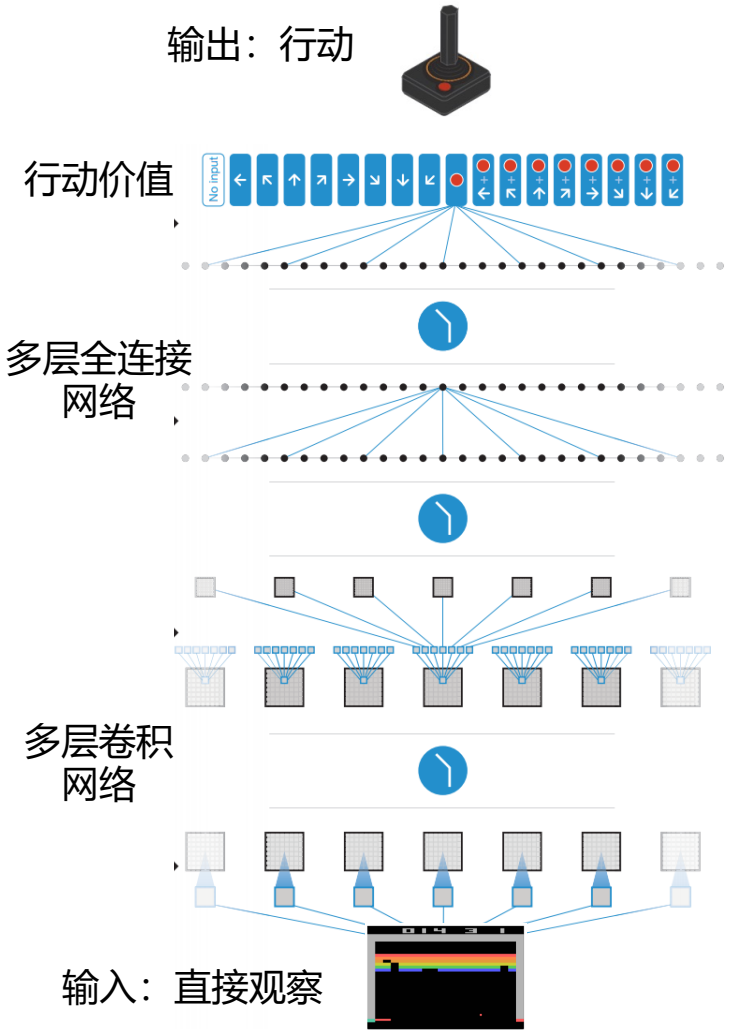
深度强化学习的趋势



Google搜索中词条“深度强化学习（deep reinforcement learning）”的趋势

深度强化学习

- 将深度学习（DL）和强化学习（RL）结合在一起会发生什么?
  - 价值函数和策略变成了深度神经网络
  - 相当高维的参数空间
  - 难以稳定地训练
  - 容易过拟合
  - 需要大量的数据
  - 需要高性能计算
  - CPU（用于收集经验数据）和GPU（用于训练神经网络）之间的平衡
  - …
- 这些新的问题促进着深度强化学习算法的创新



强化学习的研究前沿

基于模拟模型的强化学习

- 模拟器的无比重要性

目标策动的层次化强化学习

- 长程任务的中间目标是桥梁的基石

模仿学习

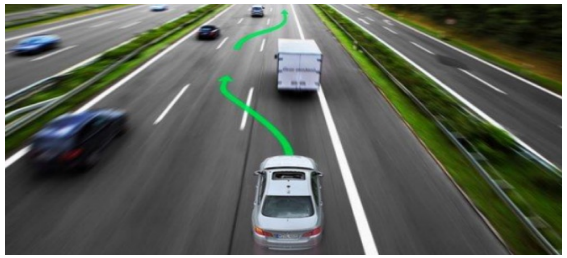
- 无奖励信号下跟随专家做策略学习

多智能体强化学习

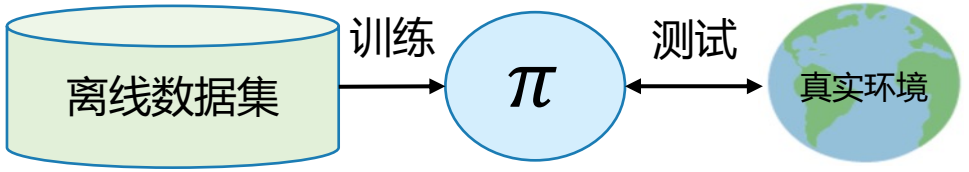
- 分散式、去中心化的人工智能

离线强化学习 (offline RL)

- 动机：在真实环境中从零开始训练一个强化学习智能体往往不可取
  - 风险较高，例如无人驾驶归控、智能医疗等
  - 十分昂贵，例如机器人控制、推荐系统等



- 离线强化学习：在一个给定的离线数据集上直接训练出智能体策略，训练的过程中，智能体不得和环境做交互



模仿学习(Imitation Learning)

- 模仿学习：直接模仿人类专家的状态-动作对来学习策略
- 目的：都是学习策略，从而控制智能体
- 原理：
  - 强化学习利用环境反馈的奖励改进策略，目标是让累计奖励最大化
  - 模仿学习向人类专家学习，目标是让策略函数做出的决策与人类专家相同
- 不是真正的强化学习，而是强化学习的一种替代品

模仿学习算法

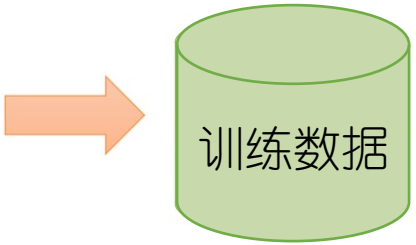
- 行为克隆(behavior cloning)或直接模仿学习
  - 无需奖励函数，模仿人类专家的动作来学习策略
- 逆强化学习(inverse reinforcement learning)
  - 找到能够使专家策略比任何其它策略更优的奖励函数
- 生成对抗模仿学习(Generative adversarial imitation learning)
  - 通过生成对抗网络的方式自动学习一个好的奖励函数

行为克隆

- 利用专家的决策轨迹，构造训练数据：状态作为特征，动作作为标记
- 利用数据集，使用标准的机器学习算法即可学得策略

$\mathbf{o}_t$

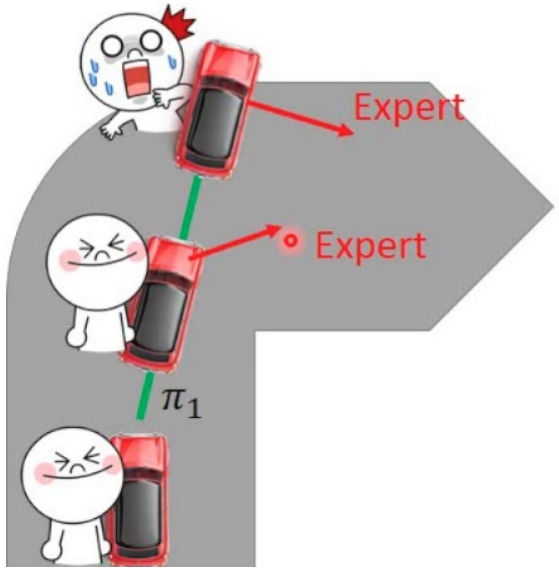
$\mathbf{a}_t$



$\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$

行为克隆

- 人类不会探索奇怪的状态和动作，因此数据集上的状态和动作缺乏多样性，
  智能体面对真实的环境，可能会遇到陌生的状态，做出的决策可能会很糟糕



行为克隆的优势在于离线训练，可以避免与真实环境交互，不会对环境造成影响（例如，行为克隆训练手术机器人）

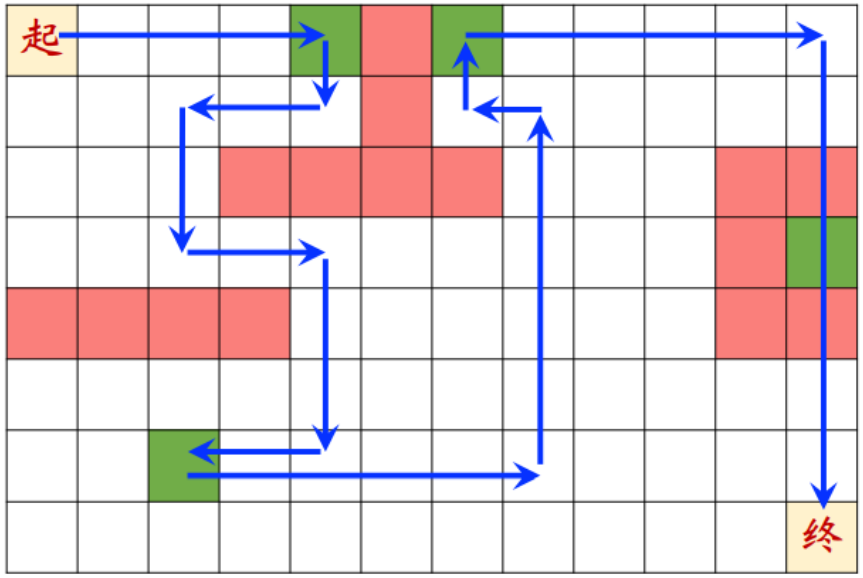
实践中，可以先通过行为克隆初始化策略函数，然后再进行强化学习

逆强化学习

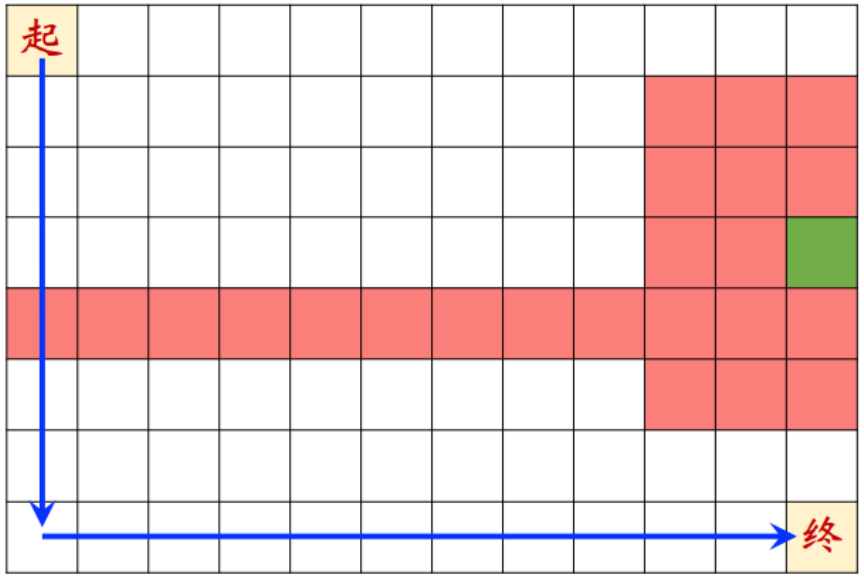
- 强化学习任务中，设计合理的符合应用场景的奖赏函数往往相当困难
- 缓解方法：从人类专家提供的范例数据中反推出奖赏函数
- 逆强化学习 (inverse reinforcement learning)
  - 基本思想：寻找某种奖赏函数使得范例数据是最优的，然后即可使用这个奖赏函数来训练策略

逆强化学习

- 蓝色箭头代表专家策略，如何反推奖励函数?



(a)

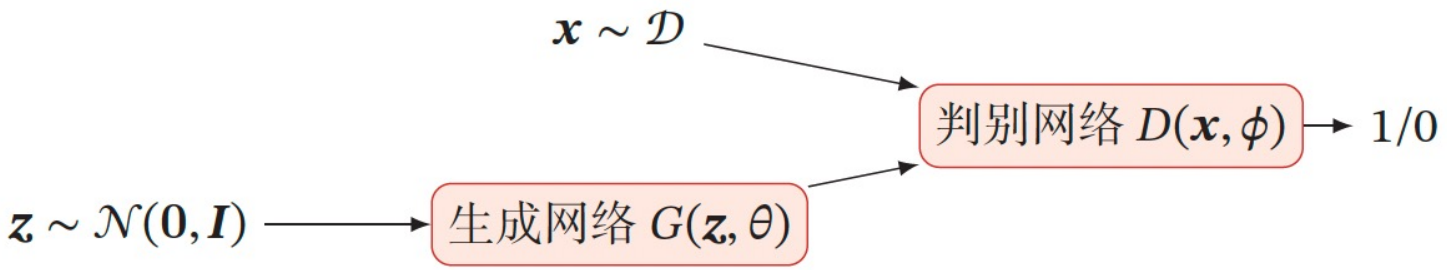


(b)

生成对抗模仿学习(Generative Adversarial imitation learning)

- 生成对抗网络[Goodfellow et al., 2014]通过对抗训练的方式使得生成网络产生的样本服从真实数据分布

- 判别网络(Discriminator)：目标是尽量准确地判断一个样本是来自于真实数据还是由生成网络产生
- 生成网络(Geneartor)：目标是尽量生成判别网络无法区分来源的样本



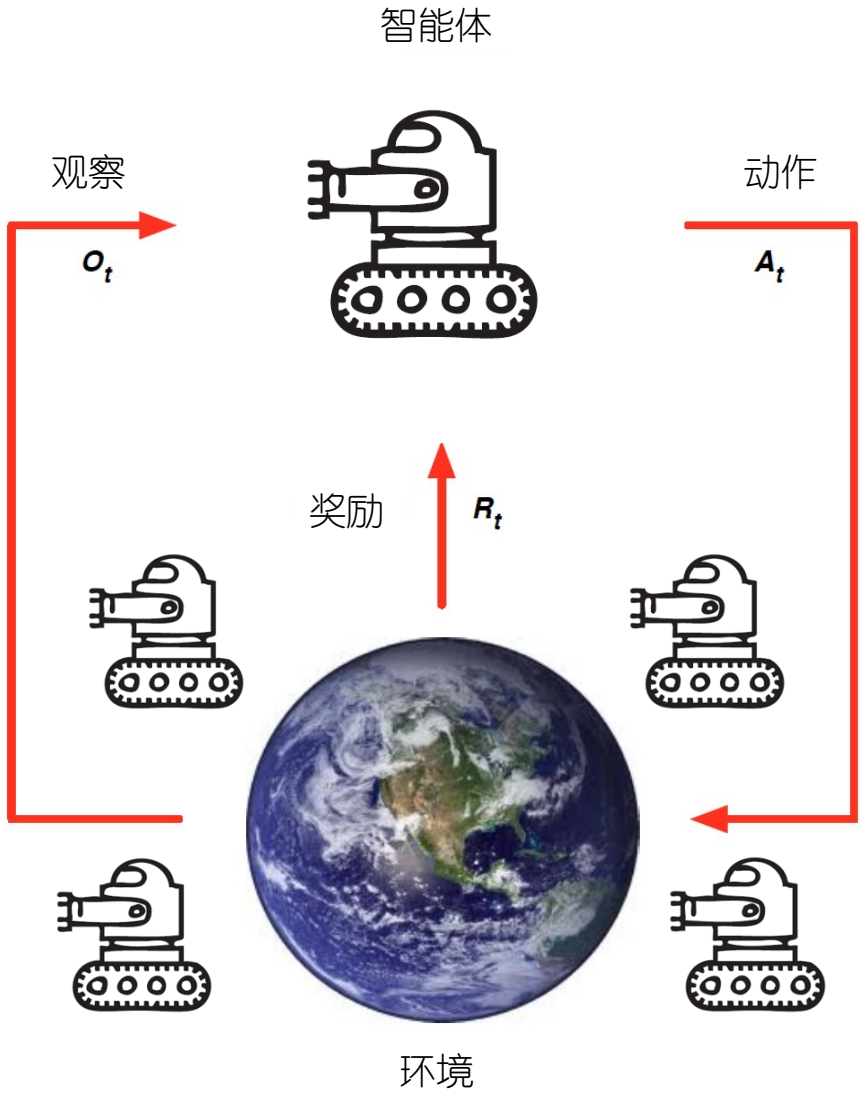
生成对抗模仿学习(Generative Adversarial imitation learning)

- 训练数据：人类专家操作智能体得到的轨迹数据
- 训练目标：让生成器生成的轨迹与数据集中的轨迹一样好，在训练结束的时候，判别器无法区分生成的轨迹与数据集里的轨迹

- 生成网络：输入状态s，输出选择各个动作的概率
- 判别网络：输入状态s，输出$p(s,a)$，越接近1表示动作a是人类专家做的，越接近于0表示动作a是生成网络生成的

多智能体强化学习

- 在与环境的交互过程中学习
- 环境包含有不断进行学习和更新的其他智能体
- 在任何一个智能体的视角下，环境是非稳态的（non-stationary）
  - 状态转移的概率分布会发生改变

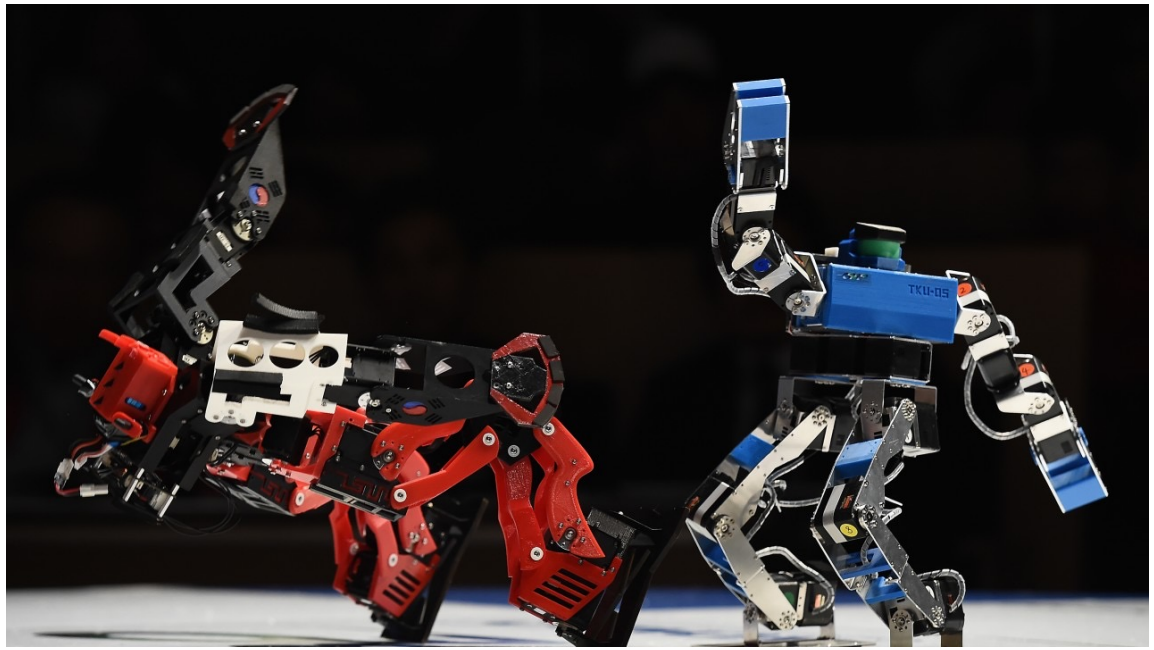


多智能体系统的常见设定

完全合作(fully cooperative)

多智能体系统的常见设定

完全竞争(fully competitive)

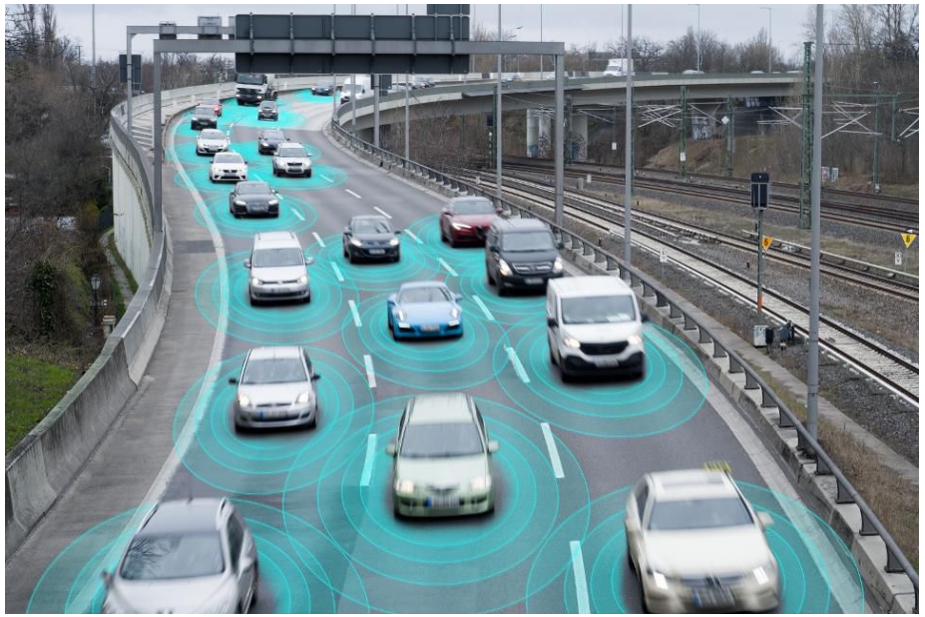


多智能体系统的常见设定

合作与竞争混合
(mixed cooperative & competitive)

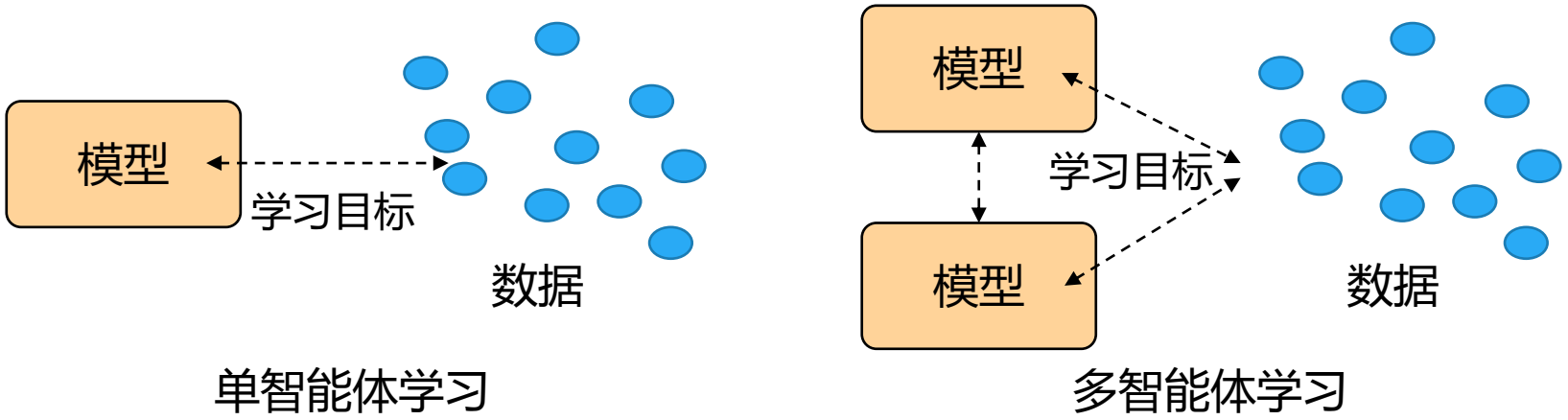
多智能体系统的常见设定

利己主义(self-interested)



多智能体强化学习的难点

- 多智能体学习从原理上来讲更加困难
  - 智能体不仅要与环境进行交互，还要相互之间进行交互



多智能体强化学习的分类

- 完全中心化方法（fully centralized）
  - 将多个智能体进行决策当作一个超级智能体在做决策，也即是把所有智能体的状态聚合在一起当作一个全局的超级状态，把所有智能体的动作连起来作为一个联合动作
  - 优点：环境是稳态；缺点：复杂度高
- 完全去中心化方法（fully decentralized）
  - 假设每个智能体都在自身的环境中独立地进行学习，不考虑其他智能体的改变。完全去中心化方法直接对每个智能体用一个单智能体强化学习算法来学习
  - 优点：简单好实现；缺点：环境非稳态，很可能不收敛
- 中心化训练去中心化执行 （centralized training with decentralized execution）
  - 在训练的时候使用一些单个智能体看不到的全局信息而达到更好的训练效果，而在执行时不使用这些信息，每个智能体完全根据自己的策略直接行动，以达到去中心化执行的效果
  - 特点：介于完全中心化方法和完全去中心化方法之间

强化学习的落地场景

- 无人驾驶
- 游戏AI
- 交通灯调度
- 网约车派单
- 组合优化
- 推荐搜索系统
- 数据中心节能优化
- 对话系统
- 机器人控制
- 路由选路
- 工业互联网场景
- ...

小结

- 强化学习：序列决策任务，延迟反馈、agent的动作会影响环境
- 有模型学习：MDP模型已知，动态规划问题
- 无模型学习：利用采样评估价值函数和价值动作函数，了解时序差分学习的基本思想
- 深度强化学习：利用神经网络建模值函数
- 了解模仿学习、多智能体强化学习等前沿方向