

强化学习

主讲：刘夏雷、郭春乐、王亚星

南开大学计算机学院

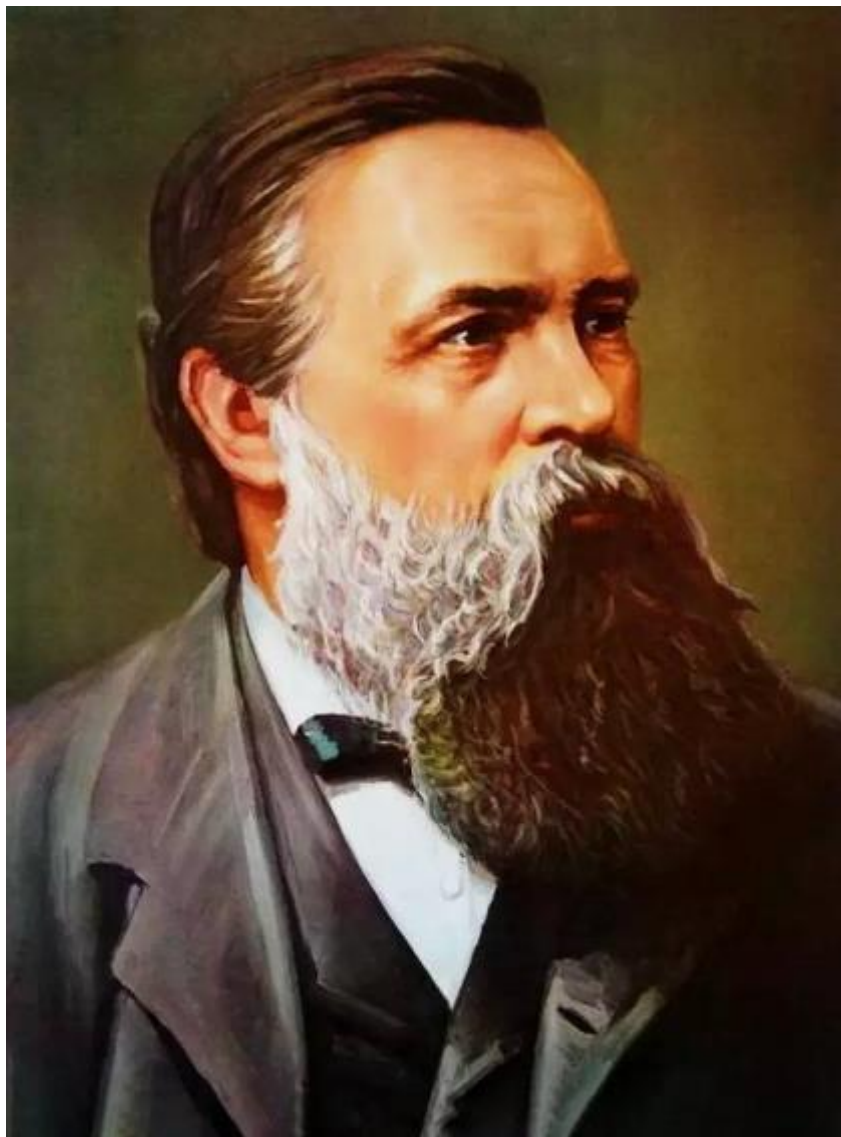
<https://mmcheng.net/xliu/>

致谢：本课件主要内容来自浙江大学吴飞教授、
南开大学程明明教授

提纲

- 一、强化学习问题定义
- 二、基于价值的强化学习
- 三、基于策略的强化学习（了解）
- 四、深度强化学习的应用（了解）

劳动创造了人----恩格斯



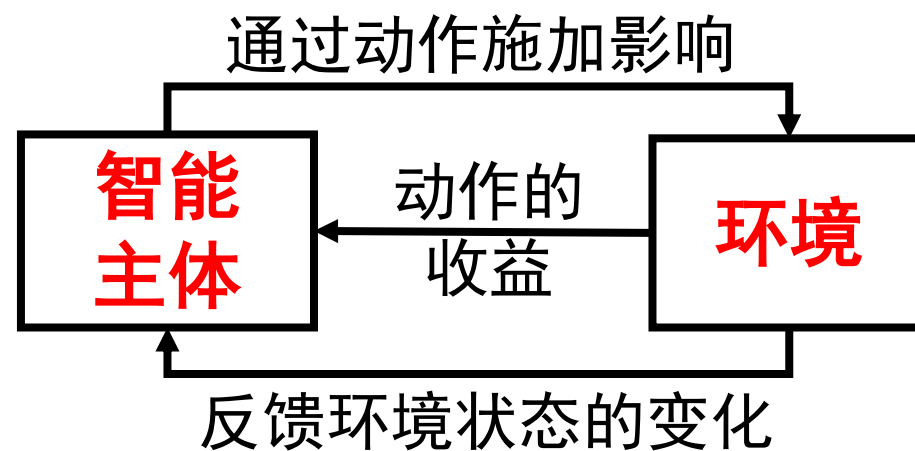
解读：劳动这样一种人与所处环境之间相互作用的实践活动促进了人的智能发育

强化学习：在与环境交互之中进行学习

生活中常见的学习过程

- 人通过动作对环境产生影响 (向前走一步)
- 环境向人反馈状态的变化 (撞到了树上)
- 人估计动作得到的收益 (疼痛)
- 更新做出动作的策略 (下次避免向有树这一障碍的方向前进)

强化学习模仿了这个过程，
在智能主体与环境的交互中，
学习能最大化收益的模式



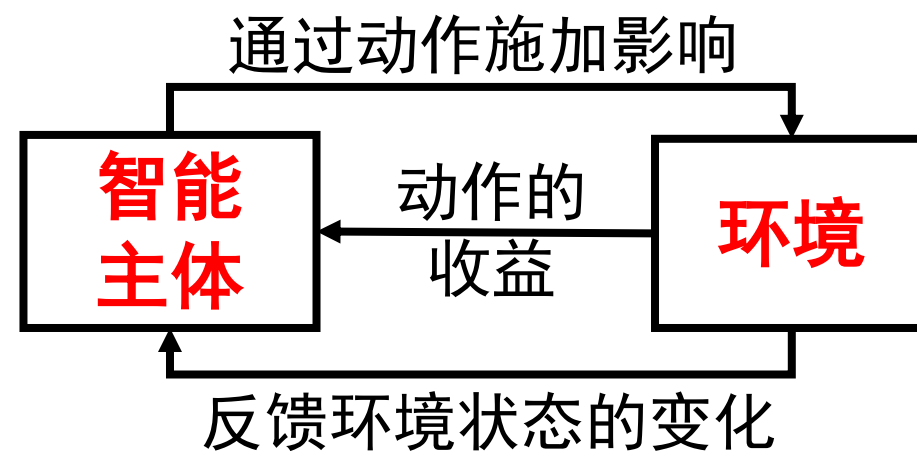
强化学习中的概念

- 智能主体(agent)

- 按照某种策略(policy), 根据当前的状态(state)选择合适的动作(action)
- 状态指的是智能主体对环境的一种解释
- 动作反映了智能主体对环境主观能动的影响, 动作带来的收益称为奖励(reward)
- 智能主体可能知道也可能不知道环境变化的规律

强化学习中的概念

- 智能主体(agent)
- 环境(environment)
 - 系统中智能主体以外的部分
 - 向智能主体反馈状态和奖励
 - 按照一定的规律发生变化



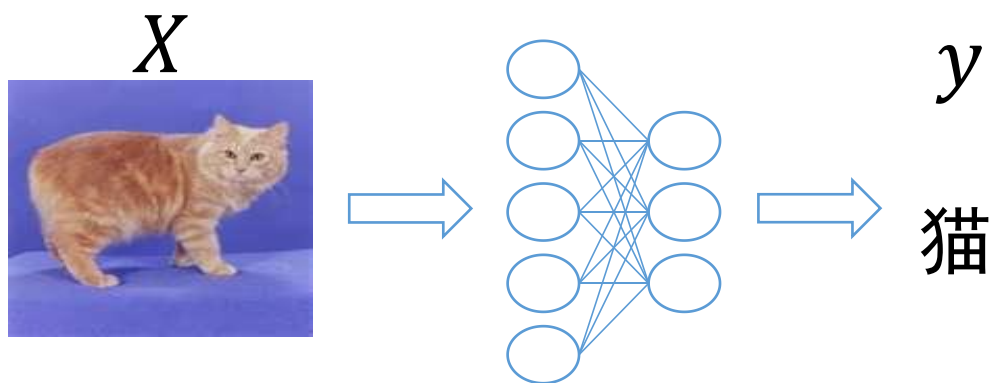
机器学习的不同类型

有监督学习

从数据 X 和标签 y 中学习映射 $f: X \mapsto y$

X : 图像、文本、音频、视频等

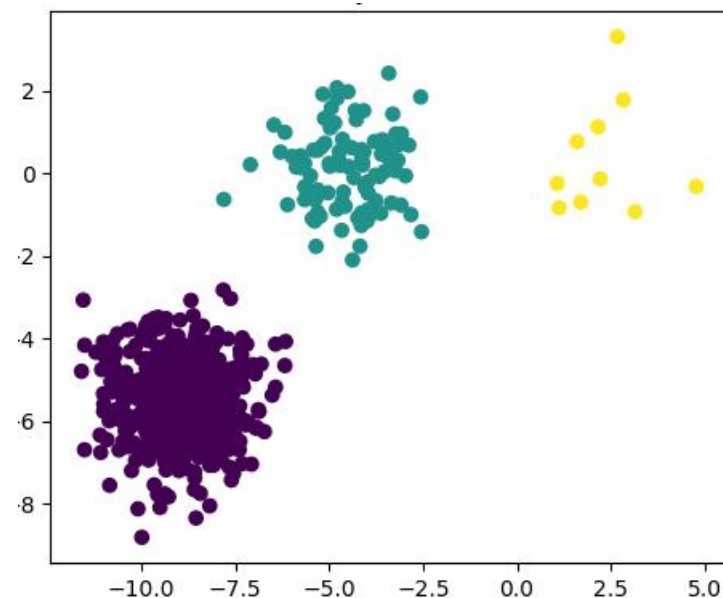
y : 连续或离散的标量、结构数据



分类问题

无监督学习

寻找数据 X 中存在的结构和模式



聚类问题

强化学习的特点

- 基于评估：

- 强化学习利用环境评估当前策略，以此为依据进行优化

- 交互性：

- 强化学习的数据在与环境的交互中产生

- 序列决策过程：

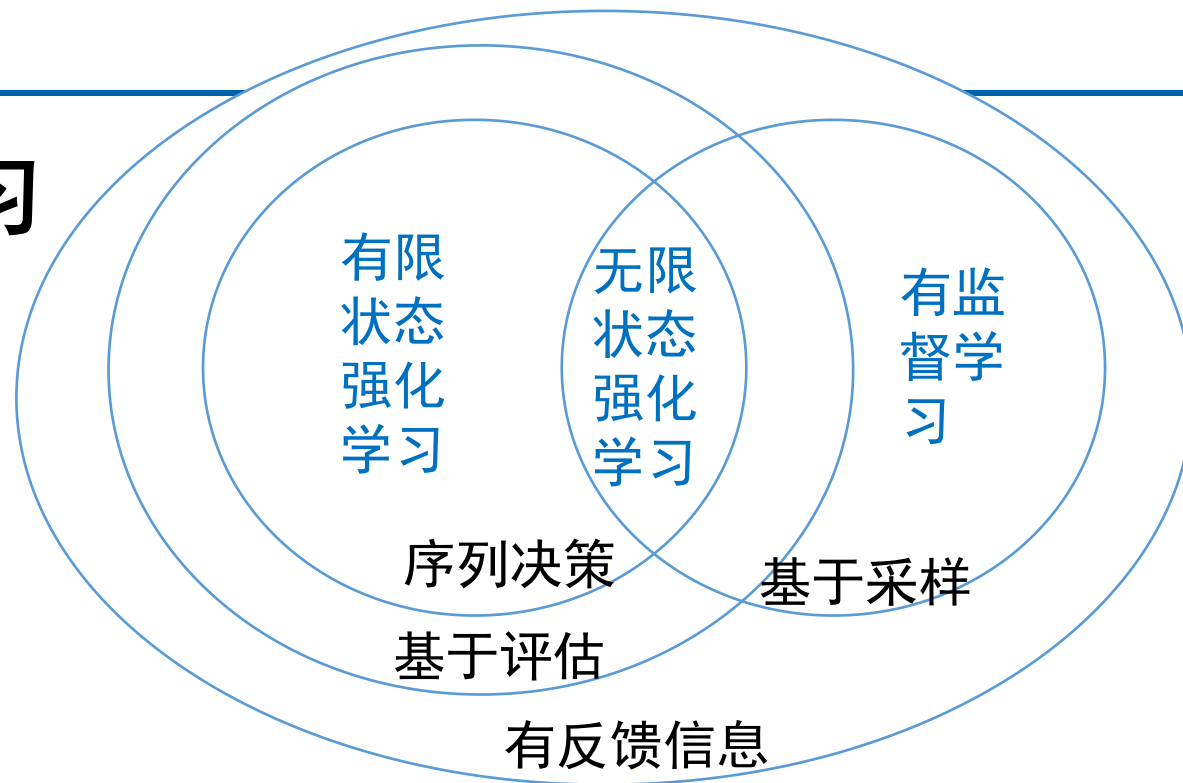
- 智能体在与环境的交互中需要作出一系列的决策，这些决策往往是前后关联的

注：现实中常见的强化学习问题往往还具有奖励滞后，
基于采样的评估等特点

强化学习的特点

• 根据以下特点直观定位强化学习

- 有/无可靠的反馈信息
- 基于评估/基于监督信息
- 序列决策/单步决策
- 基于采样/基于穷举



强化学习的特点

	有监督学习	无监督学习	强化学习
学习依据	基于监督信息	基于对数据结构的假设	基于评价(evaluative)
数据来源	一次性给定	一次性给定	在交互中产生
决策过程	单步(one-shot)	无	序列(sequential)
学习目标	样本到语义标签的映射	同一类数据的分布模式	选择能够获取最大收益的状态到动作的映射

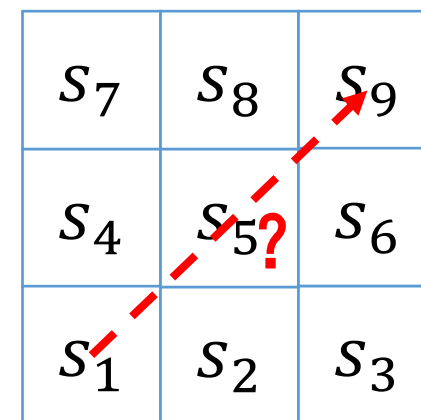
强化学习示例

• 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



离散马尔可夫过程 (Discrete Markov Process)

- 一个随机过程实际上是一列随时间变化的随机变量
 - 其中当时间是离散量时，一个随机过程可以表示为 $\{X_t\}_{t=0,1,2,\dots}$ ，其中每个 X_t 都是一个随机变量，这被称为离散随机过程
- 马尔可夫链(Markov Chain)
 - 满足马尔可夫性(Markov Property)的离散随机过程，也被称为离散马尔可夫过程。

$$Pr(X_{t+1} = x_{t+1} | X_0 = x_0, X_1 = x_1, \dots, X_t = x_t) =$$

$$Pr(X_{t+1} = x_{t+1} | X_t = x_t)$$

$t + 1$ 时刻状态仅与 t 时刻状态相关

• 生活中的马尔科夫链



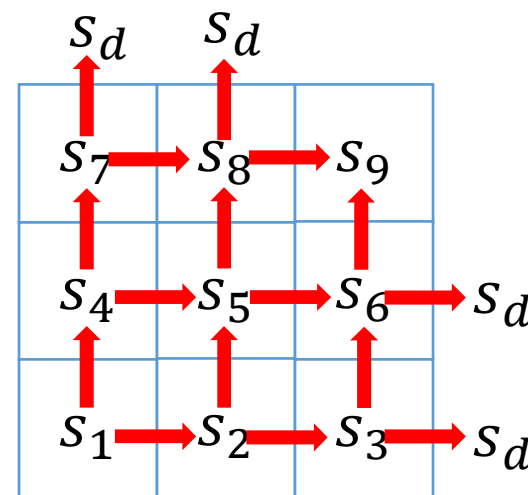
自然语言理解中n元文法(n为2表示前后相邻单词相互依赖) (n-gram grammar)

摩尔定律

集成电路元器件数目(今天|1年半前)

离散马尔可夫过程：机器人移动问题

- $MP = \{S, Pr\}$ 可用来刻画该问题
 - 随机变量序列 $\{S_t\}_{t=0,1,2,\dots}$ ，其中 S_t 表示机器人第 t 步的位置，每个随机变量 S_t 的取值范围为 $S = \{s_1, s_2, \dots, s_9, s_d\}$
 - 状态转移概率 $\Pr(S_{t+1}|S_t)$ 满足马尔可夫性。它的一种取值如图中箭头所示，每个箭头对应 0.5 的转移概率



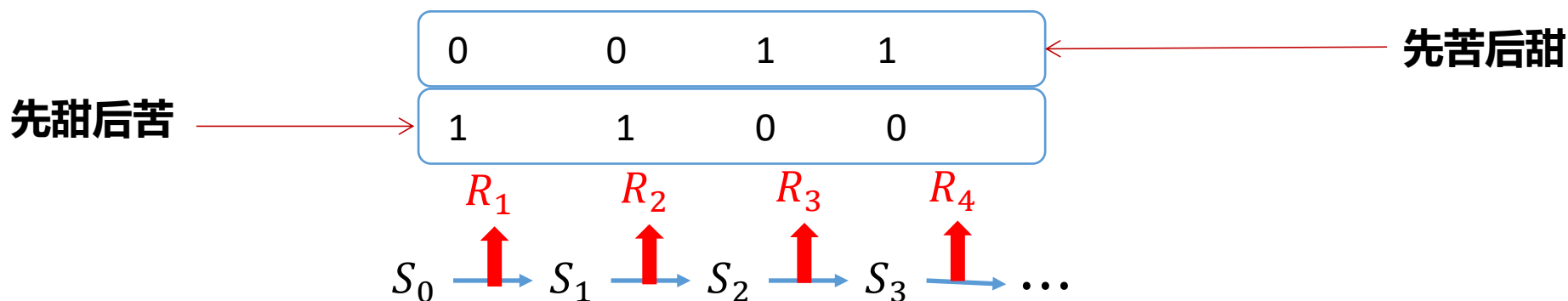
状态空间 S 集合可为无限的，如用经纬度表示现实中机器人的位置。

**这个模型不能体现机器人能动性，
缺乏与环境进行交互的手段
(如目标优化方式等)！**

马尔可夫随机过程：引入奖励

- 为了在序列决策中对目标进行优化，加入奖励机制：
 - 奖励函数 $R: S \times S \mapsto \mathbb{R}$ ，其中 $R(S_t, S_{t+1})$ 描述了从第 t 步状态转移到第 $t + 1$ 步状态所获得奖励，简记为 R_{t+1}
 - 不同状态之间的转移产生了一系列的奖励 (R_1, R_2, \dots)
 - 引入奖励机制可以衡量任意序列的优劣，即对决策进行评价

问题：给定两个因状态转移产生的奖励序列 $(1, 1, 0, 0)$ 和 $(0, 0, 1, 1)$ ，哪个序列决策更好？



马尔可夫奖励过程(Markov Reward Process)

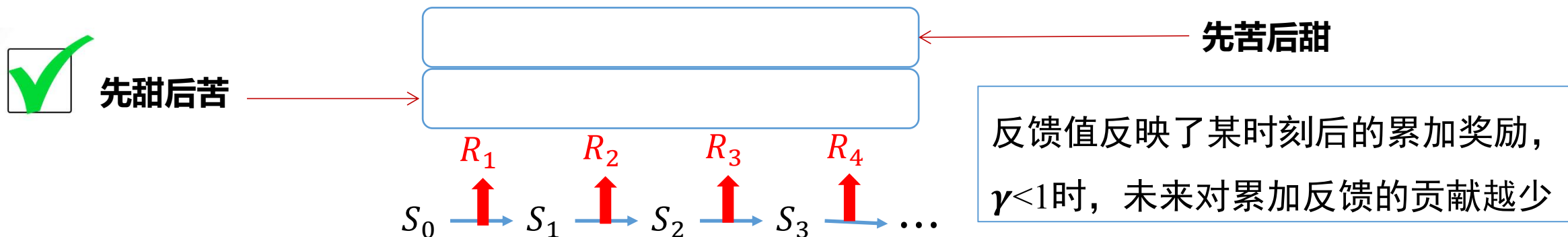
- 为了比较不同的奖励序列，**定义反馈，用来反映累加奖励：**

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

- 其中折扣系数(discount factor) $\gamma \in [0, 1]$ ，假设 $\gamma = 0.99$

$$(1,1,0,0): G_0 = 1 + 0.99 \times 1 + 0.99^2 \times 0 + 0.99^3 \times 0 = 1.99$$

$$(0,0,1,1): G_0 = 0 + 0.99 \times 0 + 0.99^2 \times 1 + 0.99^3 \times 1 = 1.9504$$



马尔可夫奖励过程(Markov Reward 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 步的位置，每个随机变量 S_t 的取值范围为 $S = \{s_1, s_2, \dots, s_9, s_d\}$
- 状态转移概率： $Pr(S_{t+1}|S_t)$ 满足马尔可夫性
- 定义奖励函数 $R(S_t, S_{t+1})$ ：从 S_t 到 S_{t+1} 所获得奖励
- 定义衰退系数： $\gamma \in [0, 1]$

马尔科夫奖励过程刻画：

$$MRP = \{S, Pr, R, \gamma\}$$

	-1	
0	0	1
0	0	0
0	0	0

马尔可夫过程

-1

该模型不能体现机器人能动性，仍缺乏与环境交互的手段

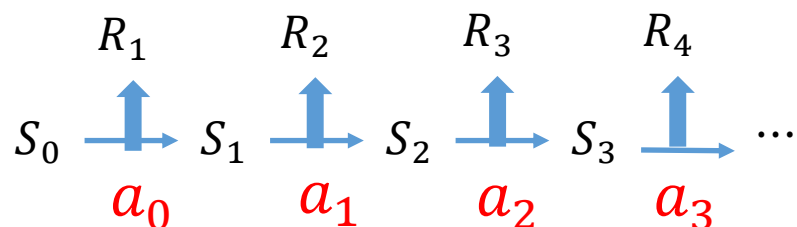
马尔可夫决策过程：引入动作

在强化学习问题中，智能主体与环境交互过程中可自主决定所采取的动作，不同动作会对环境产生不同影响，为此：

- 定义智能主体能够采取的动作集合为 A
- 由于不同的动作对环境造成的影响不同，因此状态转移概率定义为 $Pr(S_{t+1}|S_t, a_t)$ ，其中 $a_t \in A$ 为第 t 步采取的动作
- 奖励可能受动作的影响，因此修改奖励函数为 $R(S_t, a_t, S_{t+1})$

马尔可夫决策过程：引入动作

- 动作集合 A 可以是有限的，也可以是无限制的
- 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

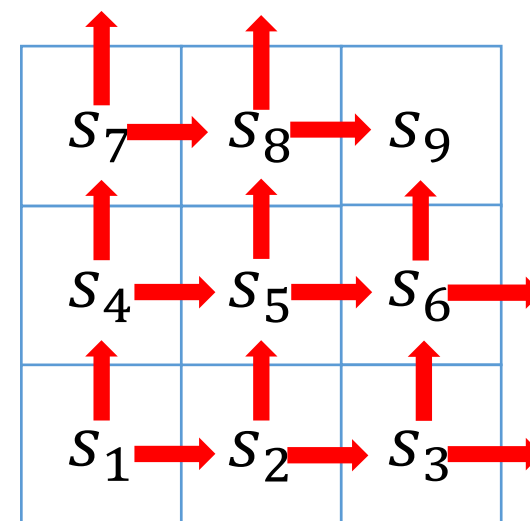


马尔可夫决策过程(Markov Decision Process)

- 使用离散马尔可夫过程描述机器人移动问题

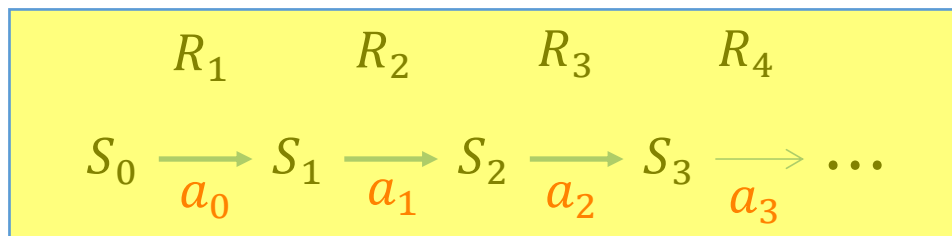
- 随机变量序列 $\{S_t\}_{t=0,1,2,\dots}$ ： S_t 表示机器人第 t 步所在位置(状态)，每个随机变量 S_t 的取值范围为 $S = \{s_1, s_2, \dots, s_9, s_d\}$
- 动作集合： $A = \{\text{上}, \text{右}\}$
- 状态转移概率 $Pr(S_{t+1}|S_t, a_t)$ ：满足马尔可夫性，其中 $a_t \in A$ 。
- 奖励函数： $R(S_t, a_t, S_{t+1})$
- 衰退系数： $\gamma \in [0, 1]$

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)

- 马尔可夫决策过程 $MDP = \{S, A, P, r, R, \gamma\}$ 是刻画强化学习中环境的标准形式
- 马尔可夫决策过程可用如下序列来表示：



马尔科夫过程中产生的状态序列称为轨迹(trajectory)，可如下表示

$$(S_0, a_0, R_1, S_1, a_1, R_2, \dots, S_T)$$

- 轨迹长度可以是无限的，也可以有终止状态 S_T 。有终止状态的问题叫做分段的(即存在回合的)(episodic)，否则叫做持续的(continuing)

分段问题中，一个从初始状态到终止状态的完整轨迹称为一个片段或回合(episode)。如围棋对弈中一个胜败对局为一个回合。

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、反馈、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: $S_T \in \{s_9, s_d\}$

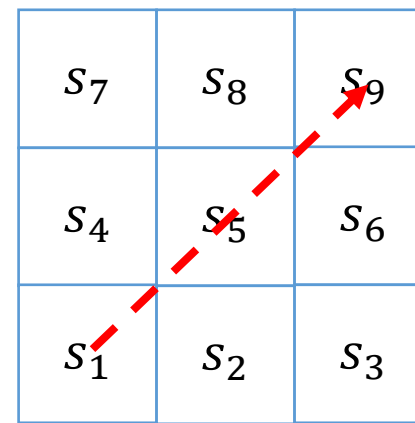
$$R(S_t, a_t, S_{t+1}) = \begin{cases} 1, & \text{如果 } S_{t+1} = s_9 \\ -1, & \text{如果 } S_{t+1} = s_d \\ 0, & \text{其他情况} \end{cases}$$

$$Pr(S_{t+1}|S_t, a_t = \text{右})$$

$S_{t+1} \backslash S_t$	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?

马尔可夫决策过程中的策略学习

- 马尔可夫决策过程 $M D P = \{S, A, P, r, R, \gamma\}$ 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**

对环境各种因素的说明

已知的: S, A, R, γ

不一定已知的: P, r

观察到的: $(S_0, a_0, R_1, S_1, a_1, R_2, \dots, S_T)$

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 $a = \pi(s)$ 。

强化学习问题定义

- 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反馈：

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

为了对策略函数 π 进行评估，定义

- **价值函数(Value Function)** $V: S \mapsto \mathbb{R}$ ，其中 $V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$ ，即在第 t 步状态为 s 时，按照策略 π 行动后在未来所获得反馈值的期望
- **动作-价值函数(Action-Value Function)** $q: S \times A \mapsto \mathbb{R}$ ，其中 $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$ 表示在第 t 步状态为 s 时，按照策略 π 采取动作 a 后，在未来所获得反馈值的期望

强化学习问题定义

- 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反馈：

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

为了对策略函数 π 进行评估，定义

- 价值函数(Value Function)** $V: S \mapsto \mathbb{R}$ ，其中 $V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$ ，即在第 t 步状态为 s 时，按照策略 π 行动后在未来所获得反馈值的期望
- 动作-价值函数(Action-Value Function)** $q: S \times A \mapsto \mathbb{R}$ ，其中 $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$ 表示在第 t 步状态为 s 时，按照策略 π 采取动作 a 后，在未来所获得反馈值的期望

由马尔可夫性，未来的状态和奖励 **只与当前状态相关**，与 t 无关。因此 t 取任意值该等式均成立，如“逢山开路，遇水搭桥”。

这样，策略学习转换为如下优化问题：

寻找一个最优策略 π^* ，对任意 $s \in S$ 使得 $V_{\pi^*}(s)$ 值最大

强化学习问题定义

智能体选择动作的模型:

策略函数 $\pi: S \times A \mapsto [0, 1]$, $\pi(s, a)$ 表示智能体在状态 s 下采取动作 a 的概率。

最大化每一时刻的回报值:

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

- **价值函数(value function):** $V: S \mapsto \mathbb{R}$, 其中 $V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$, 表示智能体在时刻 t 处于状态 s 时, 按照策略 π 采取行动时所获得回报的期望。价值函数衡量了某个状态的好坏程度, 反映了智能体从当前状态转移到该状态时能够为目标完成带来多大“好处”。
- **动作-价值函数(action-value function):** $q: S \times A \mapsto \mathbb{R}$, 其中 $q_\pi(s, a) = \mathbb{E}_\pi[G_t | S_t = s, A_t = a]$, 表示智能体在时刻 t 处于状态 s 时, 选择了动作 a 后, 在 t 时刻后根据策略 π 采取行动所获得回报的期望。

一个好的策略函数应该能够使得智能体在采取了一系列行动后可得到最佳奖励

强化学习问题定义

- 定义：给定一个马尔可夫决策过程 $M D P = (S, A, P, R, \gamma)$ ，学习一个最优策略 π^* ，对任意 $s \in S$ 使得 $V_{\pi^*}(s)$ 值最大。
- 价值函数和动作-价值函数反映了智能体在某一策略下所对应状态序列获得回报的期望，它比回报本身更加准确地刻画了智能体的目标。
- 价值函数和动作-价值函数的定义之所以能够成立，离不开决策过程所具有的马尔可夫性，即当位于当前状态 s 时，无论当前时刻 t 的取值是多少，一个策略回报值的期望是一定的(当前状态只与前一状态有关，与时间无关)。

贝尔曼方程 (Bellman Equation)

贝尔曼方程 (Bellman Equation) 也被称作动态规划方程 (Dynamic Programming Equation)，由理查德·贝尔曼 (Richard Bellman) 提出。

- **价值函数 (Value Function)** $V_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$
- **动作-价值函数 (Action-Value Function)**

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a]$$

$$\begin{aligned} V_{\pi}(s) &= \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s] \\ &= \mathbb{E}_{a \sim \pi(s, \cdot)} [\mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a]] \\ &= \sum_{a \in A} \underbrace{\pi(s, a)}_{\text{采取动作 } a \text{ 的概率}} \times \underbrace{q_{\pi}(s, a)}_{\text{采取动作 } a \text{ 后带来的回报期望}} \\ &= \sum_{a \in A} \pi(s, a) q_{\pi}(s, a) \end{aligned}$$

贝尔曼方程 (Bellman Equation)

贝尔曼方程 (Bellman Equation) 也被称作动态规划方程 (Dynamic Programming Equation)，由理查德·贝尔曼 (Richard Bellman) 提出。

- **价值函数 (Value Function)** $V_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$
- **动作-价值函数 (Action-Value Function)**

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a]$$

$$\begin{aligned} q_{\pi}(s, a) &= \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a] \\ &= \mathbb{E}_{s' \sim P(\cdot | s, a)} [R(s, a, s') + \gamma \mathbb{E}_{\pi}[R_{t+2} + \gamma R_{t+3} + \dots | S_{t+1} = s']] \\ &= \sum_{s' \in \mathcal{S}} \underbrace{P(s' | s, a)}_{\text{在状态 } s \text{ 采取行动 } a \text{ 进入状态 } s' \text{ 的概率}} \times \left[\underbrace{R(s, a, s')}_{\text{在 } s \text{ 采取 } a \text{ 进入 } s' \text{ 得到的回报}} + \gamma \times \underbrace{V_{\pi}(s')}_{\text{在 } s' \text{ 获得的回报期望}} \right] = \sum_{s' \in \mathcal{S}} P(s' | s, a) [R(s, a, s') + \gamma V_{\pi}(s')] \end{aligned}$$

贝尔曼方程 (Bellman Equation)

贝尔曼方程 (Bellman Equation) 也被称作动态规划方程 (Dynamic Programming Equation)，由理查德·贝尔曼 (Richard Bellman) 提出。

- **价值函数 (Value Function)** $V_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$
- **动作-价值函数 (Action-Value Function)**

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a]$$

$$\begin{aligned} V_{\pi}(s) &= \sum_{a \in A} \pi(s, a) \sum_{s' \in \mathcal{S}} P(s' | s, a) [R(s, a, s') + \gamma V_{\pi}(s')] \\ &= \mathbb{E}_{a \sim \pi(s, \cdot)} \mathbb{E}_{s' \sim P(\cdot | s, a)} [R(s, a, s') + \gamma V_{\pi}(s')] \end{aligned}$$

价值函数取值与时间没有关系，只与策略 π 、在策略 π 下从某个状态转移到其后继状态所取得的回报以及在后续所得回报有关。

贝尔曼方程 (Bellman Equation)

贝尔曼方程 (Bellman Equation) 也被称作动态规划方程 (Dynamic Programming Equation)，由理查德·贝尔曼 (Richard Bellman) 提出。

- **价值函数 (Value Function)** $V_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$
- **动作-价值函数 (Action-Value Function)**

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s, A_t = a]$$

$$\begin{aligned} q_{\pi}(s, a) &= \sum_{s' \in \mathcal{S}} P(s' | s, a) \left[R(s, a, s') + \gamma \sum_{a' \in \mathcal{A}} \pi(s', a') q_{\pi}(s', a') \right] \\ &= \mathbb{E}_{s' \sim P(\cdot | s, a)} [R(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi(s', \cdot)} [q_{\pi}(s', a')]] \end{aligned}$$

动作-价值函数取值同样与时间没有关系，而是与瞬时奖励和下一步的状态和动作有关。

价值函数与动作-价值函数的关系：动作-价值函数的贝尔曼方程

价值函数的贝尔曼方程

$$V_{\pi}(s) = \mathbb{E}_{a \sim \pi(s, \cdot)} \mathbb{E}_{s' \sim P(\cdot | s, a)} [R(s, a, s') + \gamma V_{\pi}(s')]$$

动作-价值函数的贝尔曼方程

$$q_{\pi}(s, a) = \mathbb{E}_{s' \sim P(\cdot | s, a)} [R(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi(s', \cdot)} [q_{\pi}(s', a')]]$$

贝尔曼方程描述了价值函数或动作-价值函数的递推关系，是研究强化学习问题的重要手段。其中价值函数的贝尔曼方程描述了当前状态价值函数和其后续状态价值函数之间的关系，即当前状态价值函数等于**瞬时奖励的期望**加上后续状态的(折扣)价值函数的期望。而动作-价值函数的贝尔曼方程描述了当前动作-价值函数和其后续动作-价值函数之间的关系，即当前状态下的动作-价值函数等于**瞬时奖励的期望**加上后续状态的(折扣)动作-价值函数的期望。

价值函数与动作-价值函数的关系：动作-价值函数的贝尔曼方程

价值函数的贝尔曼方程

$$V_{\pi}(s) = \mathbb{E}_{a \sim \pi(s, \cdot)} \mathbb{E}_{s' \sim P(\cdot | s, a)} [R(s, a, s') + \gamma V_{\pi}(s')]$$

动作-价值函数的贝尔曼方程

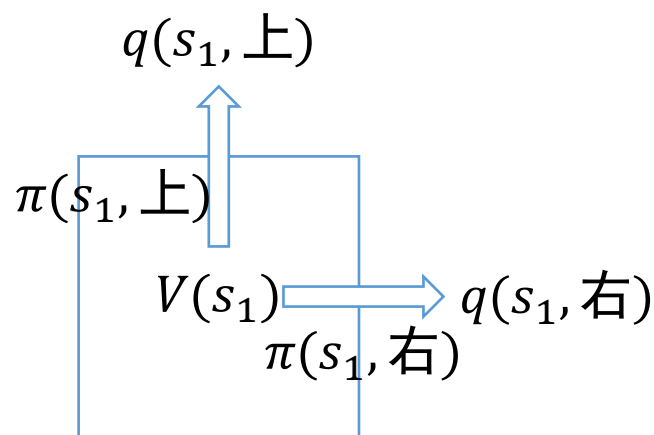
$$q_{\pi}(s, a) = \mathbb{E}_{s' \sim P(\cdot | s, a)} [R(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi(s', \cdot)} [q_{\pi}(s', a')]]$$

在实际中，需要计算得到最优策略以指导智能体在当前状态如何选择一个可获得最大回报的动作。求解最优策略的一种方法就是去求解最优的价值函数或最优的动作-价值函数(即基于价值方法，value-based approach)。一旦找到了最优的价值函数或动作-价值函数，自然而然也就是找到最优策略。当然，在强化学习中还有基于策略(policy-based)和基于模型(model-based)等不同方法。

价值函数与动作-价值函数的关系：以状态 s_1 的计算为例

$$V_{\pi}(s) = \sum_{a \in A} \pi(s, a) q_{\pi}(s, a)$$

$$V_{\pi}(s_1) = \pi(s_1, \text{上}) q_{\pi}(s_1, \text{上}) \\ + \pi(s_1, \text{右}) q_{\pi}(s_1, \text{右})$$

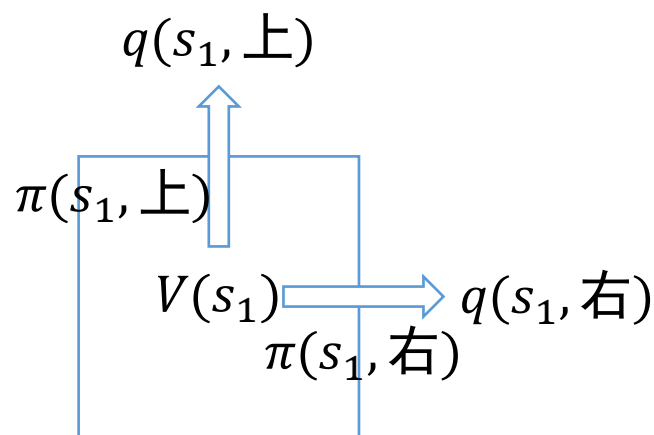


不同动作下的反馈累加

价值函数与动作-价值函数的关系：以状态 s_1 的计算为例

$$V_{\pi}(s) = \sum_{a \in A} \pi(s, a) q_{\pi}(s, a)$$

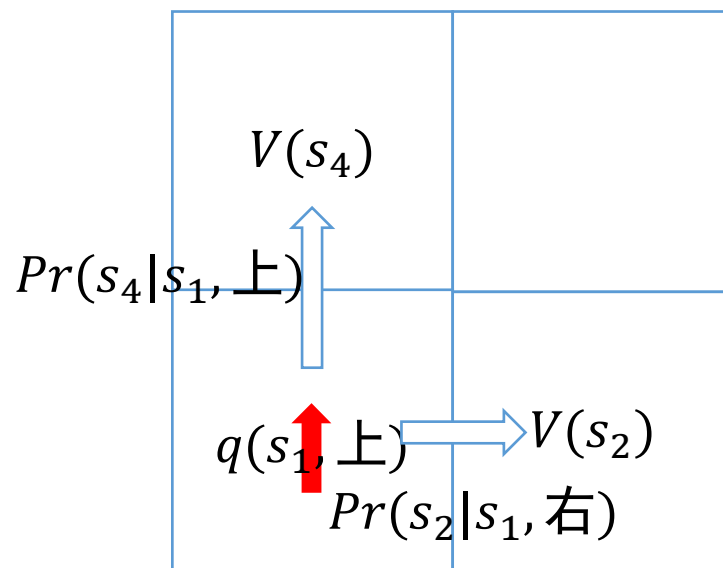
$$V_{\pi}(s_1) = \pi(s_1, \text{上}) q_{\pi}(s_1, \text{上}) + \pi(s_1, \text{右}) q_{\pi}(s_1, \text{右})$$



不同动作下的反馈累加

$$q_{\pi}(s, a) = \sum_{s' \in S} Pr(s'|s, a) [R(s, a, s') + \gamma V_{\pi}(s')]$$

$$q_{\pi}(s_1, \text{上}) = Pr(s_4|s_1, \text{上}) [R(s_1, \text{上}, s_4) + \gamma V_{\pi}(s_4)]$$



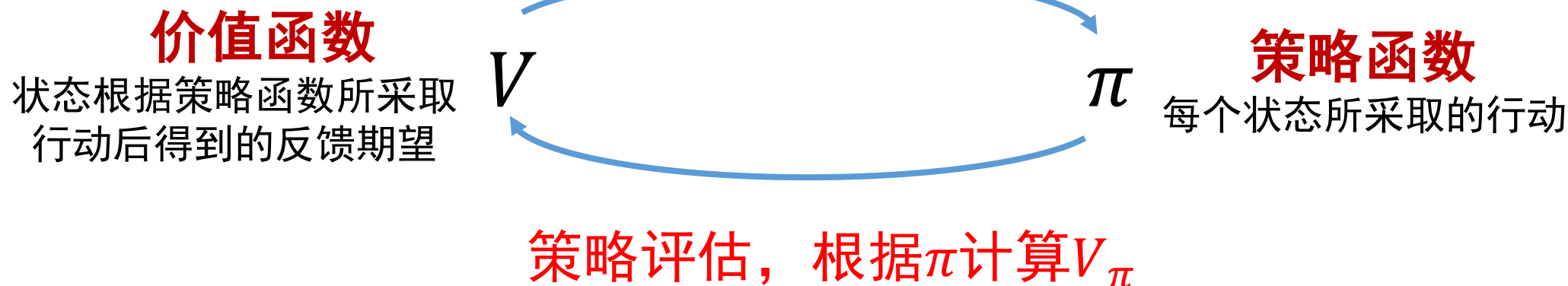
动作确定时状态转移后的反馈结果

提纲

- 一、强化学习问题定义
- 二、基于价值的强化学习
- 三、基于策略的强化学习
- 四、深度强化学习的应用

强化学习的问题与求解

- 强化学习的问题定义：给定马尔可夫决策过程 $M D P = \{S, A, P, r, R, \gamma\}$
- 强化学习会寻找一个最优策略 π^* ，在策略 π^* 作用下使得任意状态 $s \in S$ 对应的价值函数 $V_{\pi^*}(s)$ 取值最大。
- 强化学习求解：在策略优化和策略评估的交替迭代中优化参数



强化学习中的策略优化

- 策略优化定理:

- 对于确定的策略 π 和 π' , 如果对于任意状态 $s \in S$

$$q_{\pi}(s, \pi'(s)) \geq q_{\pi}(s, \pi(s))$$

- 那么对于任意状态 $s \in S$, 有

$$V_{\pi'}(s) \geq V_{\pi}(s)$$

- 即策略 π' 不比 π 差

注意, 不等式左侧的含义是只在当前这一步将动作修改为 $\pi'(s)$, 未来的动作仍然按照 π 的指导进行

在讨论如何优化策略之前, 首先需要明确什么是“更好”的策略。分别给出 π 和 π' 两个策略, 如果对于任意状态 $s \in S$, 有 $V_{\pi}(s) \leq V_{\pi'}(s)$, 那么可以认为策略 π' 不比策略 π 差, 可见“更优”策略是一个偏序关系。

强化学习中的策略优化

- 给定当前策略 π 、价值函数 V_π 和行动-价值函数 q_π 时，可如下构造新的策略 π' ，只要 π' 满足如下条件：

$$\pi'(s) = \operatorname{argmax}_a q_\pi(s, a) \text{ (对于任意 } s \in S \text{)}$$

- π' 便是对 π 的一个改进。于是对于任意 $s \in S$ ，有

$$q_\pi(s, \pi'(s)) = q_\pi(s, \operatorname{argmax}_a q_\pi(s, a)) = \max_a q_\pi(s, a) \geq q_\pi(s, \pi(s))$$

强化学习中的策略优化：机器人寻路问题为例子

假设当前价值函数在右图中给出，策略用箭头表示，对状态 s_1 而言：

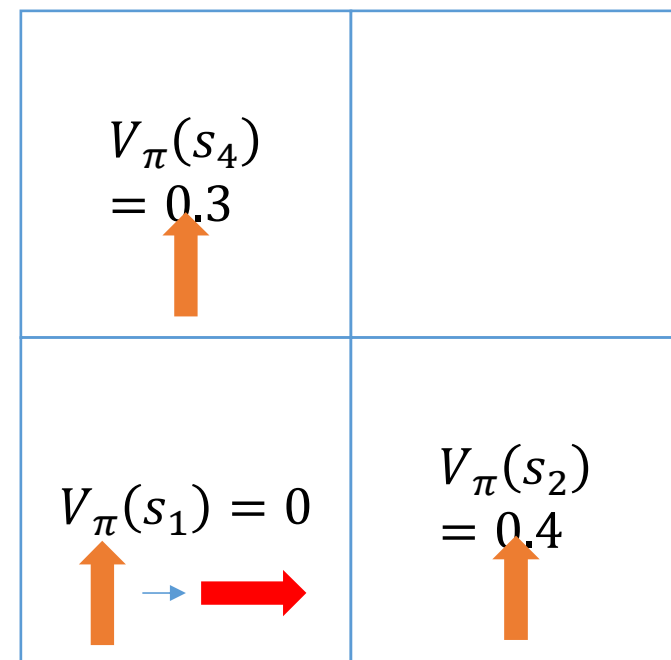
$$q_{\pi}(s_1, \text{上}) = 0 + 0.99 \times 0.3 = 0.297$$

$$q_{\pi}(s_1, \text{右}) = \underbrace{0}_{R(s_1, \text{右}, s_2)} + \underbrace{0.99}_{\gamma} \times \underbrace{0.4}_{V_{\pi}(s_2)} = 0.396$$

注意这个问题里状态转移是确定的

因此根据策略优化原则 $\pi'(s) = \operatorname{argmax}_a q_{\pi}(s, a)$

更新状态 s_1 策略 $\pi'(s_1) = \operatorname{argmax}_a q_{\pi}(s, a) = \text{右}$



强化学习中的策略评估方法

- 动态规划
- 蒙特卡洛采样
- 时序差分(Temporal Difference)

假定当前策略为 π ，策略评估指的是根据策略 π 来计算相应的价值函数 V_π 或动作-价值函数 q_π 。这里将介绍在状态集合有限前提下三种常见的策略评估方法，它们分别是基于动态规划的方法、基于蒙特卡洛采样的方法和时序差分(temporal difference)法。

强化学习中的策略评估：动态规划

- 基于动态规划的价值函数更新：使用迭代的方法求解贝尔曼方程组

初始化 V_π 函数

循环

枚举 $s \in S$

$$V_\pi(s) \leftarrow \sum_{a \in A} \pi(s, a) \left[\sum_{s' \in S} Pr(s'|s, a) [R(s, a, s') + \gamma V_\pi(s')] \right]$$

直到 V_π 收敛

更新 $V_\pi(s_1)$ 的值：

$$q_\pi(s_1, \text{上}) = 1 \times (0 + 0.99 \times 0.3) + 0 \times (0 + 0.99 \times 0.4) + \dots = 0.297$$

$$V_\pi(s_1) = 1 \times q_\pi(s_1, \text{上}) + 0 \times q_\pi(s_1, \text{右}) = 0.297$$

- 动态规划法的缺点：

- 1) 智能主体需要事先知道状态转移概率；
- 2) 无法处理状态集合大小无限的情况

$V_\pi(s_4) = 0.3$ ↑	
$V_\pi(s_1) = 0.297$ ↑	$V_\pi(s_2) = 0.4$ ↑

强化学习中的策略评估：动态规划

- 基于蒙特卡洛采样的价值函数更新

选择不同的起始状态，按照当前策略 π 采样若干轨迹，记它们的集合为 D
枚举 $s \in S$

计算 D 中 s 每次出现时对应的反馈 G_1, G_2, \dots, G_k

$$V_{\pi}(s) \leftarrow \frac{1}{k} \sum_{i=1}^k G_i$$

假设按照当前策略可样得到以下两条轨迹

$(s_1, s_4, s_7, s_8, s_9)$

(s_1, s_2, s_3, s_d)

s_1 对应的反馈值分别为

$$0 + \gamma \times 0 + \dots + \gamma^3 \times 1 = 0.970$$

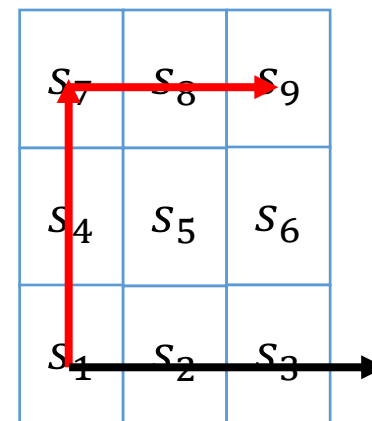
$$0 + \gamma \times 0 + \gamma^2 \times (-1) = -0.980$$

因此估计

$$V(s_1) = \frac{1}{2} (0.970 - 0.980) = -0.005$$

根据数理统计的知识，期望可以通过样本均值来估计的，这正是蒙特卡洛方法(Monte-Carlo method)的核心思想。即大数定理指出：对于独立同分布(i.i.d)的样本数据，当样本足够大的时候，样本平均值向期望值收敛。

如果是确定的策略，每个起点只会产生一种轨迹



强化学习中的策略评估：动态规划

- 基于蒙特卡洛采样的价值函数更新
- 按照这样的思路可使用蒙特卡洛方法来进行策略评估：给定状态 s ，从该状态出发不断采样后续状态，得到不同的采样序列。通过这些采样序列来分别计算状态 s 的回报值，对这些回报值取均值，作为对状态 s 价值函数的估计，从而避免对状态转移概率的依赖。

蒙特卡洛采样法的优点

- 智能主体不必知道状态转移概率
- 容易扩展到无限状态集合的问题中

蒙特卡洛采样法的缺点

- 状态集合比较大时，一个状态在轨迹可能非常稀疏，不利于估计期望
- 在实际问题中，最终反馈需要在终止状态才能知晓，导致反馈周期较长

强化学习中的策略评估：动态规划

- 基于时序差分(Temporal Difference)的价值函数更新
- 时序差分法可以看作蒙特卡罗方法和动态规划方法的有机结合。时序差分算法与蒙特卡罗方法相似之处在于，时序差分方法从实际经验中获取信息，无需提前获知环境模型的全部信息。时序差分算法与动态规划方法的相似之处在于，时序差分方法能够利用前序已知信息来进行在线实时学习，无需等到整个片段结束(终止状态抵达)再进行价值函数的更新。

强化学习中的策略评估：动态规划

- 动态规划法根据贝尔曼方程迭代更新价值函数，要求算法事先知道状态之间的转移概率，这往往是不现实的。为了解决这个问题，时序差分法借鉴蒙特卡洛法思想，通过采样 a 和 s' 来估计计算 $V_{\pi}(s)$
- 由于通过采样进行计算，所得结果可能不准确，因此时序差分法并没有将这个估计值照单全收，而是以 α 作为权重来接受新的估计值，即把价值函数更新为 $(1 - \alpha)V_{\pi}(s) + \alpha[R + \gamma V_{\pi}(s')]$ ，对这个式子稍加整理就能得到算法7.2.3中第7行形式： $V_{\pi}(s) \leftarrow V_{\pi}(s) + \alpha[R + \gamma V_{\pi}(s') - V_{\pi}(s)]$ 。这里 $R + \gamma V_{\pi}(s')$ 为时序差分目标， $R + \gamma V_{\pi}(s') - V_{\pi}(s)$ 为时序差分偏差。

- 更新 $V_{\pi}(s)$ 的值： $V_{\pi}(s) \leftarrow (1 - \alpha)V_{\pi}(s) + \alpha[R(s, a, s') + \gamma V_{\pi}(s')]$

过去的
价值函数值

学习得到的
价值函数值

强化学习中的策略评估：动态规划

- 基于时序差分(Temporal Difference)的价值函数更新

初始化 V_π 函数

循环

初始化 s 为初始状态

循环

$a \sim \pi(s, \cdot)$

执行动作 a ，观察奖励 R 和下一个状态 s'

更新 $V_\pi(s) \leftarrow V_\pi(s) + \alpha[R(s, a, s') + \gamma V_\pi(s') - V_\pi(s)]$

$s \leftarrow s'$

直到 s 是终止状态

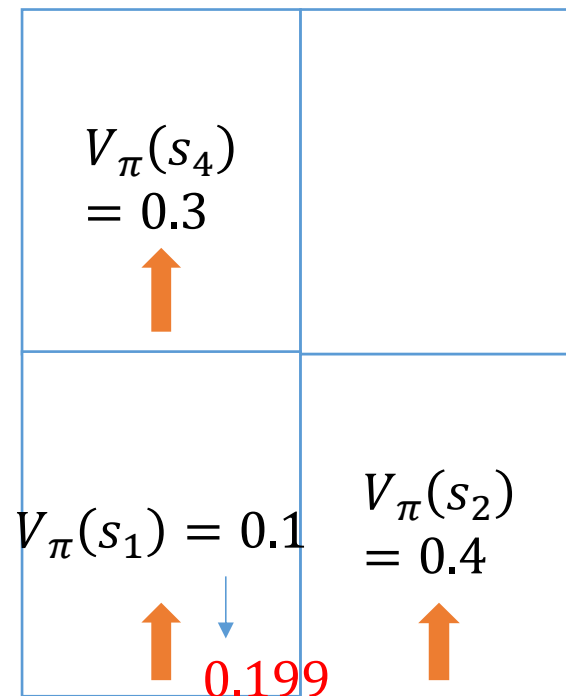
直到 V_π 收敛

假设 $\alpha = 0.5$ ，更新 $V_\pi(s_1)$ 的值：

从 $\pi(s_1, \cdot)$ 中采样得到动作 $a = \text{上}$

从 $Pr(\cdot | s_1, \text{上})$ 中采样得到下一步状态 $s' = s_4$

$$\begin{aligned} V_\pi(s_1) &\leftarrow V_\pi(s_1) + \alpha[R(s_1, \text{上}, s_4) + \gamma V_\pi(s_4) - V_\pi(s_1)] \\ &= 0.1 + 0.5 \times [0 + 0.99 \times 0.3 - 0.1] = 0.199 \end{aligned}$$



在对片段进行采样的同时，不断以上述方法更新当前状态的价值函数，不断迭代直到价值函数收敛为止。

强化学习中的策略评估：Q-learning

- 基于时序差分的方法 – Q学习(Q-Learning)[Q: quality]

初始化 q_π 函数

循环

初始化 s 为初始状态

循环

$a \sim \pi(s, \cdot) \rightarrow a = \operatorname{argmax}_{a'} q_\pi(s, a')$

执行动作 a ，观察奖励 R 和下一个状态 s'

更新 $V_\pi(s) \leftarrow V_\pi(s) + \alpha[R + \gamma V_\pi(s') - V_\pi(s)]$

\rightarrow 更新 $q_\pi(s, a) \leftarrow q_\pi(s, a) + \alpha[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s, a)]$

$s \leftarrow s'$

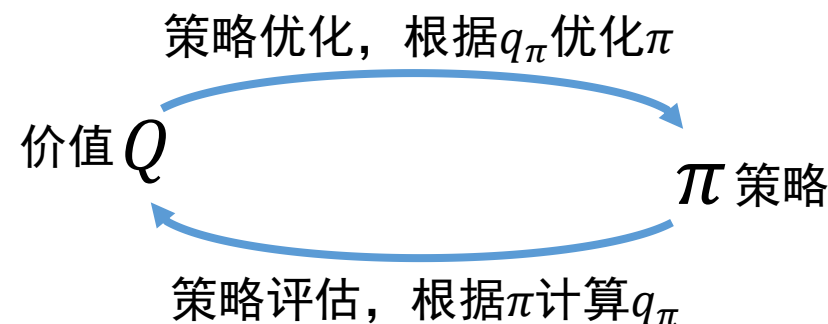
直到 s 是终止状态

直到 q_π 收敛

策略优化: $\pi'(s) = \operatorname{argmax}_a q_\pi(s, a)$

$q_\pi(s, a) \leftarrow (1 - \alpha)q_\pi(s, a) + \alpha[R + \gamma \max_{a'} q_\pi(s', a')]$

- 基于价值的方法不直接对策略建模，因此策略优化在采样和更新两步中之 \max 操作上得以间接体现
- 在同一次循环中策略评估和策略优化交替进行
- 由于策略优化要求计算动作-价值函数 q ，因此Q学习直接利用 q 函数的贝尔曼方程进行更新



强化学习中的策略评估：Q-learning

s_d	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

初始化 q_π 函数

循环

初始化 s 为初始状态

循环

$$a = \operatorname{argmax}_{a'} q_\pi(s, a')$$

执行动作 a ，观察奖励 R 和下一个状态 s'

$$\text{更新 } q_\pi(s, a) \leftarrow q_\pi(s, a) + \alpha \left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s, a) \right]$$

$$s \leftarrow s'$$

直到 s 是终止状态

直到 q_π 收敛

初始化 q_π 函数

在右图中， a/b 表示 $q_\pi(s, \text{上}) = a$, $q_\pi(s, \text{右}) = b$

所有终止状态的 q 函数值设为 $0/0$ ，其余状态可随机初始化，

此处设 $0.2/0$
初始化 s ， s 的值在右图中用黑框框出

$0/0$	$0.2/0$	$0.2/0$	$0/0$
	$0.2/0$	$0.2/0$	$0.2/0$
	<div>$0.2/0$</div>	$0.2/0$	$0.2/0$

强化学习中的策略评估：Q-learning

s_d	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

初始化 q_π 函数

循环

初始化 s 为初始状态

循环

$$a = \operatorname{argmax}_{a'} q_\pi(s, a')$$

执行动作 a ，观察奖励 R 和下一个状态 s'

$$\text{更新 } q_\pi(s, a) \leftarrow q_\pi(s, a) + \alpha \left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s, a) \right]$$

$$s \leftarrow s'$$

直到 s 是终止状态

直到 q_π 收敛

$$a = \operatorname{argmax}_{a'} q_\pi(s_1, a') = \text{上}$$

$$R = 0, s' = s_4$$

$$q_\pi(s_1, \text{上}) \leftarrow 0.2 + 0.5 \times [0 + 0.99 \times \max\{0, 0.2\} - 0.2] = 0.199$$

$$s \leftarrow s_4$$

$0/0$	$0.2/0$	$0.2/0$	$0/0$
	<div>$0.2/0$</div>	$0.2/0$	$0.2/0$
	<div>$0.199/0$</div>	$0.2/0$	$0.2/0$

强化学习中的策略评估：Q-learning

s_d	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

初始化 q_π 函数

循环

初始化 s 为初始状态

循环

$$a = \operatorname{argmax}_{a'} q_\pi(s, a')$$

执行动作 a ，观察奖励 R 和下一个状态 s'

$$\text{更新 } q_\pi(s, a) \leftarrow q_\pi(s, a) + \alpha \left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s, a) \right]$$

$$s \leftarrow s'$$

直到 s 是终止状态

直到 q_π 收敛

$$a = \operatorname{argmax}_{a'} q_\pi(s_4, a') = \text{上}$$

$$R = 0, s' = s_7$$

$$q_\pi(s_4, \text{上}) \leftarrow 0.2 + 0.5 \times [0 + 0.99 \times \max\{0, 0.2\} - 0.2] = 0.199$$

$$s \leftarrow s_7$$

0/0	<div>0.2/0</div>	0.2/0	0/0
	<div>0.199/0</div>	0.2/0	0.2/0
	0.199/0	0.2/0	0.2/0

强化学习中的策略评估：Q-learning

s_d	s_7	s_8	s_9
	s_4	s_5	s_6
	s_1	s_2	s_3

初始化 q_π 函数

循环

初始化 s 为初始状态

循环

$a = \operatorname{argmax}_{a'} q_\pi(s, a')$

执行动作 a ，观察奖励 R 和下一个状态 s'

更新 $q_\pi(s, a) \leftarrow q_\pi(s, a) + \alpha [R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s, a)]$

$s \leftarrow s'$

直到 s 是终止状态

直到 q_π 收敛

$a = \operatorname{argmax}_{a'} q_\pi(s_7, a') = \text{上}$

$R = -1, s' = s_d$

$q_\pi(s_7, \text{上}) \leftarrow 0.2 + 0.5 \times [-1 + 0.99 \times \max\{0, 0\} - 0.2] = -0.4$

$s \leftarrow s_d$

因为 s_d 是终止状态，因此一个片段(episode)结束

$0/0$	$-0.4/0$	$0.2/0$	$0/0$
$0.199/0$	$0.2/0$	$0.2/0$	
$0.199/0$	$0.2/0$	$0.2/0$	

强化学习中的策略评估：Q-learning

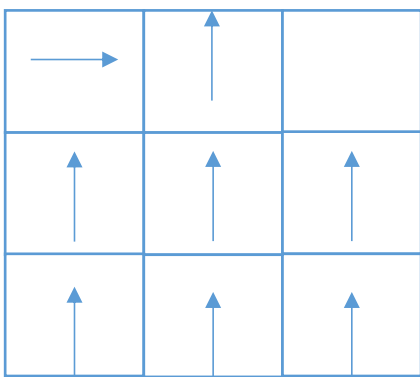
q函数

$-0.4/0$	$0.2/0$	$0/0$
$0.199/0$	$0.2/0$	$0.2/0$
$0.199/0$	$0.2/0$	$0.2/0$

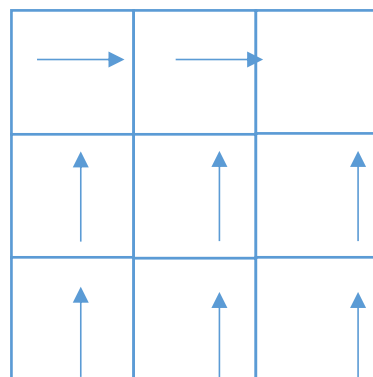
$-0.4/0.099$	$-0.4/0$	$0/0$
$0.100/0$	$0.2/0$	$0.2/0$
$0.198/0$	$0.2/0$	$0.2/0$

$-0.4/0.050$	$-0.4/0.5$	$0/0$
$0.099/0$	$0.2/0$	$0.2/0$
$0.148/0$	$0.2/0$	$0.2/0$

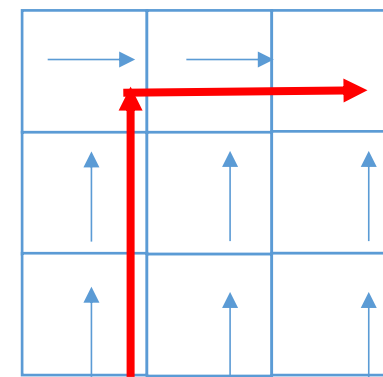
策略



第一个片段后



第二个片段后



第三个片段后

强化学习中的策略评估：Q-learning

行百里者半于九十

q 函数

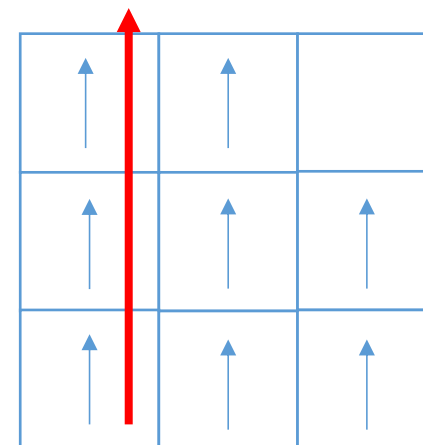
$1/-2$	$1/-2$	$0/0$
$1/-2$	$1/-2$	$1/-2$
$1/-2$	$1/-2$	$1/-2$

.....

$-1/-2$	$1/-2$	$0/0$
$-0.990/-$	$1/-2$	$1/-2$
$-0.980/-$	$1/-2$	$1/-2$

- 如果 q 函数的初始化为 $1/-2$ ，在模型收敛后，策略仍无法使得智能主体找到目标状态
- 这种情况并非个例，例如状态 s_7 往上走的期望反馈保持0.2、往右走的期望反馈保持0的条件下，将 s_7 往上走的惩罚值适当减少（在先前例子中，该惩罚值为-1），会得到类似的效果
问题：这种情况出现的原因为何？

策略



第 n 个片段后

策略学习中探索(exploration)与利用(exploitation)的平衡

• 问题：为何Q学习收敛到非最优策略？

观察每个片段的轨迹

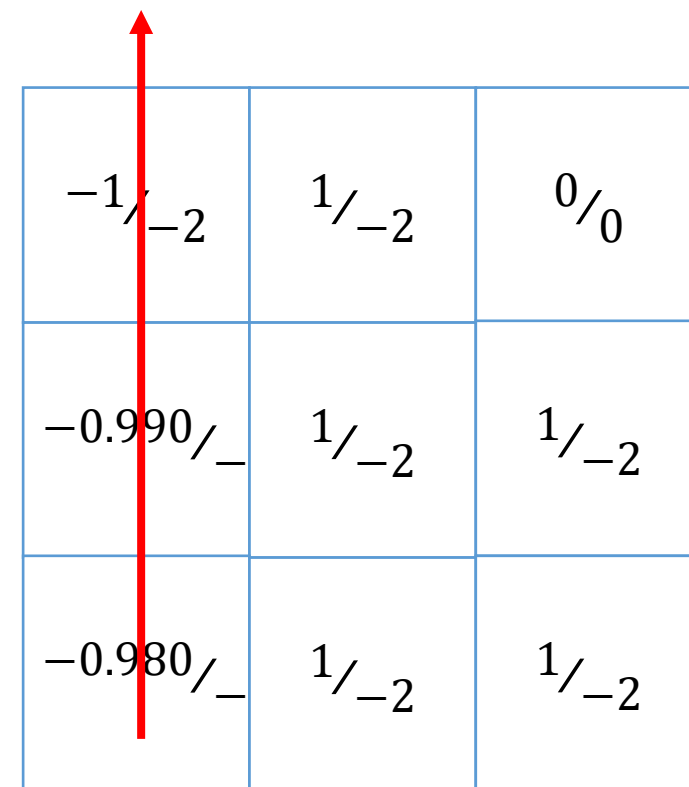
- (s_1, s_4, s_7, s_d)
- (s_1, s_4, s_7, s_d)
-
- (s_1, s_4, s_7, s_d)

智能主体的策略(即按照动作-价值函数选择反馈最大的行为)始终不变，因此与环境交互的轨迹是固定的，过程中没有得到任何有关目标 s_9 的信息

- 外力：缺乏推动智能主体改变策略的外在因素
- 内因：智能主体缺乏从内部改变策略的动力

智能主体的“创新精神”：

- 根据目前已知的最优策略来选择动作，被称为**利用 (exploitation)**
- 不根据当前策略而去尝试未知的动作被称为**探索 (exploration)**



$-1/-2$	$1/-2$	$0/0$
$-0.990/-$	$1/-2$	$1/-2$
$-0.980/-$	$1/-2$	$1/-2$

策略学习中探索(exploration)与利用(exploitation)的平衡

- 问题：为何Q学习收敛到非最优策略？
- 回答：算法中只有利用没有探索

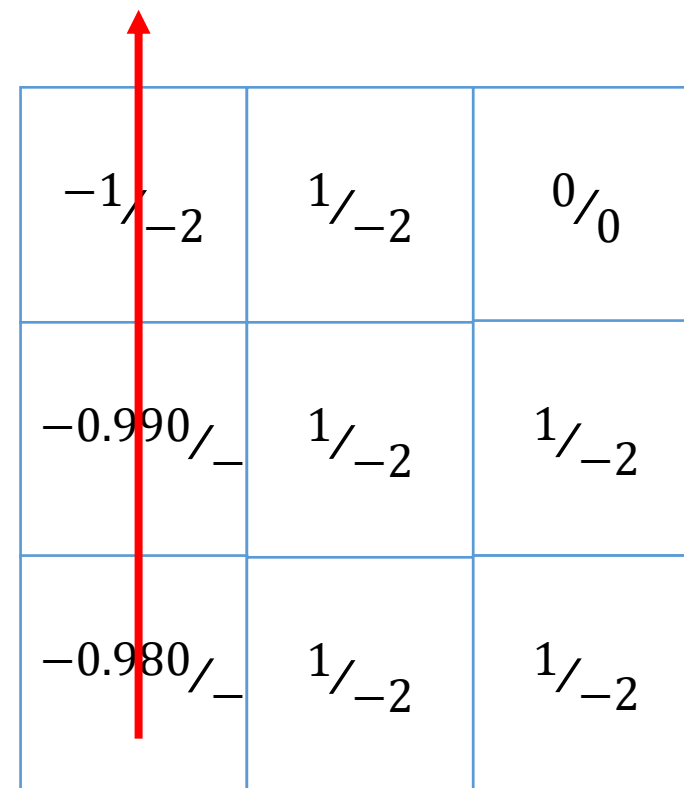
探索与利用之间如何取得平衡

- 只利用而不探索 ❌
- 只探索而不利用(则训练过程完全没有意义) ❌
- 大体上利用，偶尔探索 ✅

ϵ 贪心(ϵ -greedy)策略

$$\epsilon\text{-greedy}_{\pi}(s) = \begin{cases} \operatorname{argmax}_a q_{\pi}(s, a), & \text{以 } 1 - \epsilon \text{ 的概率} \\ \text{随机的 } a \in A, & \text{以 } \epsilon \text{ 的概率} \end{cases}$$

$\epsilon\text{-greedy}_{\pi}$ 策略是非确定的策略，严格来说应该写成概率形式，此处用了其简化表达

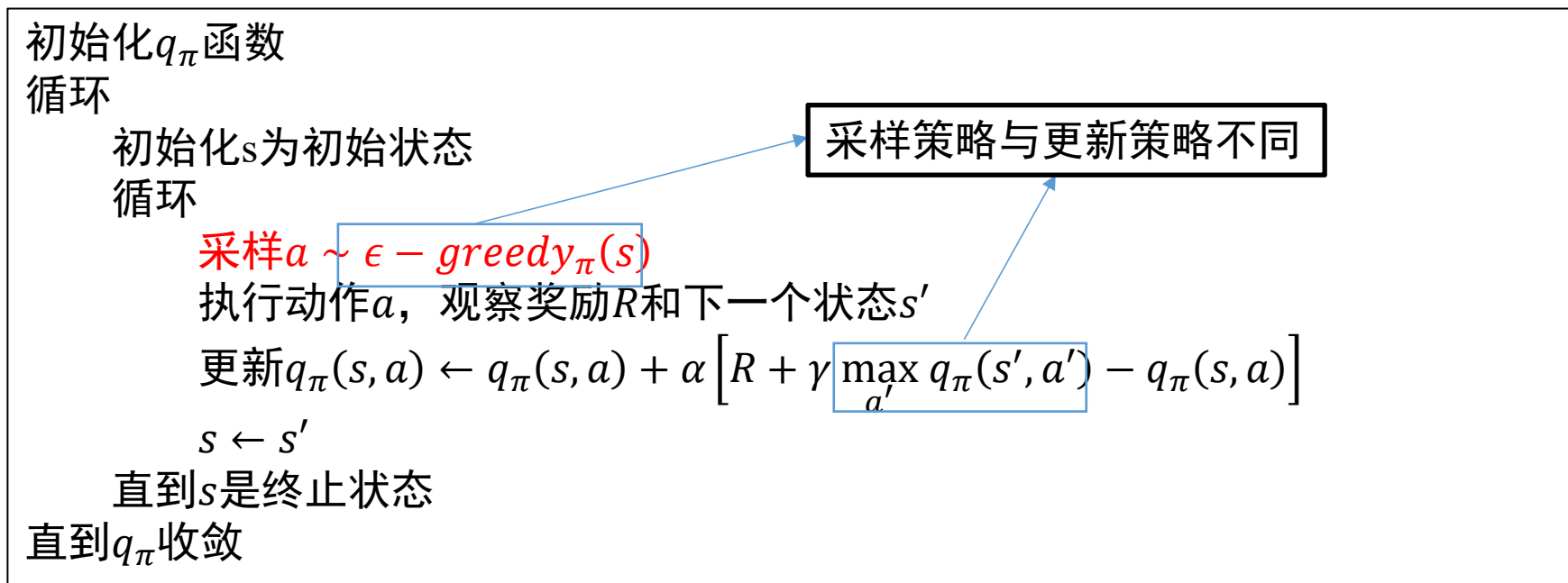


-1/-2	1/-2	0/0
-0.990/-	1/-2	1/-2
-0.980/-	1/-2	1/-2

ϵ 贪心策略的解释：大体上遵循最优策略的决定，偶尔(以 ϵ 的小概率)进行探索

策略学习中探索(exploration)与利用(exploitation)的平衡

• 使用 ϵ 贪心策略的Q学习



- 将动作采样从“确定地选取最优动作”改为“按照 ϵ 贪心策略选取动作”
- 更新时仍保持用 \max 操作选取最佳策略。像这样更新时的目标策略与采样策略不同的方法，叫做离策略(off-policy)方法

强化学习中的策略评估：使用 ϵ 贪心策略的Q学习

q 函数

$1/-2$	$1/-2$	$0/0$
$1/-2$	$1/-2$	$1/-2$
$1/-2$	$1/-2$	$1/-2$

.....

$-0.99/0.7$	$-1.00/1$	$0/0$
$0.12/0.98$	$0.99/0.80$	$1/-2$
$0.97/0.24$	$0.98/-2$	$1/-2$

令 $\epsilon = 0.1$ ，采用探索策略的Q学习，执行了100个片段后

- 学习得到的策略能够将智能主体导向目标 s_9
- 仍有部分状态是没有被探索过的(s_3)
- 增大 ϵ 值有助于算法去探索这些未知状态

探索与利用
相互平衡策略

→	→	
→	↑	↑
↑	↑	↑

第100个片段后

用神经网络拟合(行动)价值函数：Deep Q-learning

使用 ϵ 贪心策略的Q学习

- 状态数量太多时，有些状态可能始终无法采样到，因此对这些状态的 q 函数进行估计是很困难的
- 状态数量无限时，不可能用一张表(数组)来记录 q 函数的值

初始化 q_π 函数

循环

初始化 s 为初始状态

循环

采样 $a \sim \epsilon\text{-greedy}_\pi(s)$

执行动作 a ，观察奖励 R 和下一个状态 s'

更新 $q_\pi(s, a) \leftarrow q_\pi(s, a) + \alpha \left[R + \gamma \max_{a'} q_\pi(s', a') - q_\pi(s, a) \right]$

$s \leftarrow s'$

直到 s 是终止状态

直到 q_π 收敛

思路：将 q 函数参数化(parametrize)，用一个非线性回归模型来拟合 q 函数，例如(深度)神经网络

- 能够用有限的参数刻画无限的状态
- 由于回归函数的连续性，没有探索过的状态也可通过周围的状态来估计

用神经网络拟合(行动)价值函数： Deep Q-learning

• 用深度神经网络拟合 q 函数

初始化 q_π 函数的参数 θ

循环

初始化 s 为初始状态

循环

采样 $a \sim \epsilon\text{-greedy}_\pi(s; \theta)$

执行动作 a ，观察奖励 R 和下一个状态 s'

损失函数 $L(\theta) = \frac{1}{2} \left[R + \gamma \max_{a'} q_\pi(s', a'; \theta) - q_\pi(s, a; \theta) \right]^2$

根据梯度 $\partial L(\theta) / \partial \theta$ 更新参数 θ

$s \leftarrow s'$

直到 s 是终止状态

直到 q_π 收敛

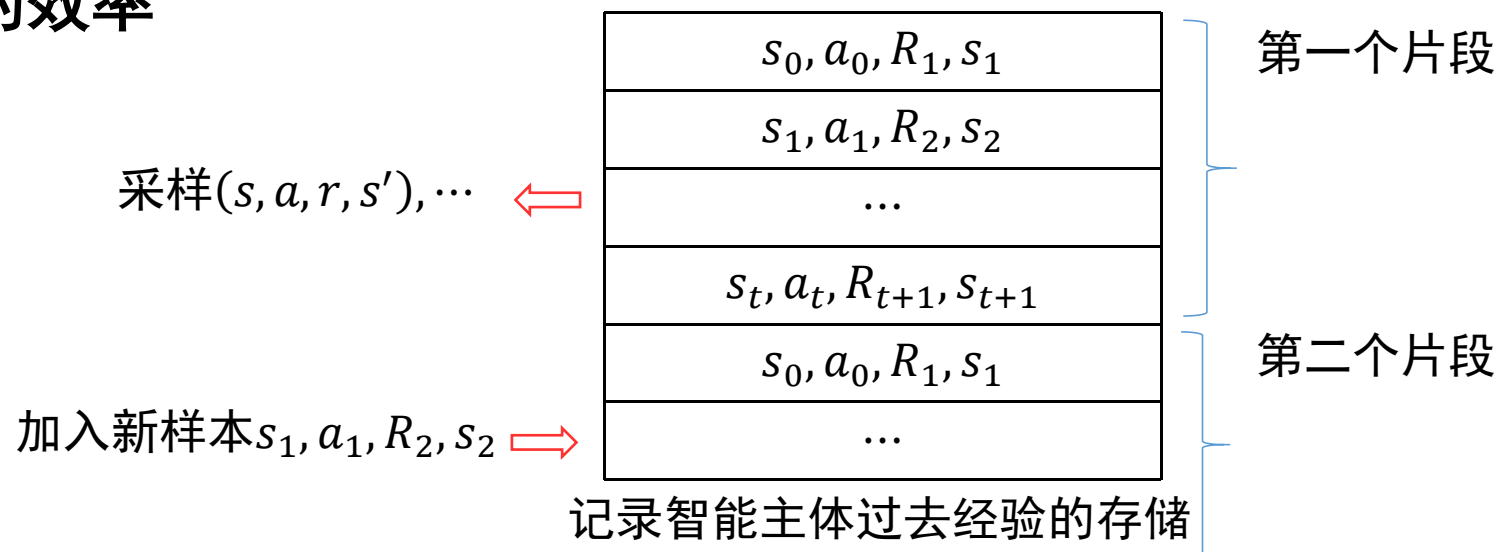
- 损失函数刻画了 q 的估计值 $R + \gamma \max_{a'} q_\pi(s', a'; \theta)$ 与当前值的平方误差
- 利用梯度下降法优化参数 θ
- 如果用深度神经网络来拟合 q 函数，则算法称为深度Q学习或者深度强化学习

经验重现(Experience Replay)

- 相邻的样本来自同一条轨迹，样本之间相关性太强，集中优化相关性强的样本可能导致神经网络在其他样本上效果下降。

将过去的经验存储下来，每次将新的样本加入到存储中去，并从存储中采样一批样本进行优化

- 解决了样本相关性强的问题
- 重用经验，提高了信息利用的效率



目标网络(Target Network)

- 在损失函数中， q 函数的值既用来估计目标值，又用来计算当前值。现在这两处的 q 函数通过 θ 有所关联，可能导致优化时不稳定

$$\frac{1}{2} \left[R + \gamma \max_{a'} \overset{\text{目标网络}}{q_{\pi}(s', a'; \theta^-)} - q_{\pi}(s, a; \theta) \right]^2$$

损失函数的两个 q 函数使用不同的参数计算

- 用于计算估计值的 q 使用参数 θ^- 计算，这个网络叫做目标网络
- 用于计算当前值的 q 使用参数 θ 计算
- 保持 θ^- 的值相对稳定，例如 θ 每更新多次后才同步两者的值

$$\theta^- \leftarrow \theta$$

提纲

- 一、强化学习问题定义
- 二、基于价值的强化学习
- 三、基于策略的强化学习
- 四、深度强化学习的应用

基于策略的强化学习

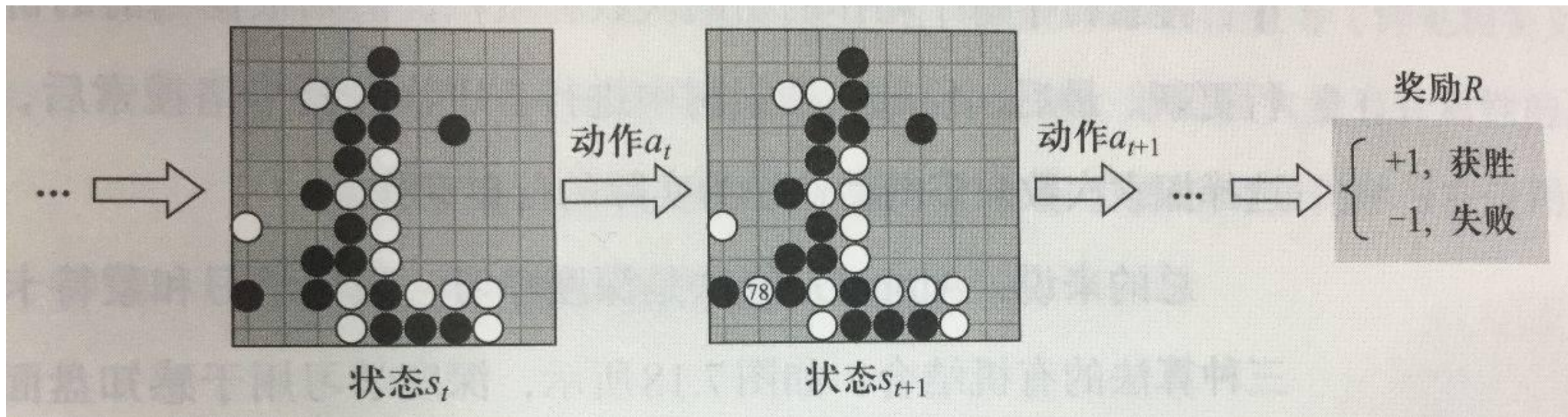
- 基于价值的强化学习:以对价值函数或动作-价值函数的建模为核心。
- 基于策略的强化学习:直接参数化策略函数，求解参数化的策略函数的梯度。
- 策略函数的参数化可以表示为 $\pi_{\theta}(s, a)$ ，其中 θ 为一组参数，函数取值表示在状态 s 下选择动作 a 的概率。和Q学习的 ϵ 贪心策略相比，这种参数化的一个显著好处是：选择一个动作的概率是随着参数的改变而光滑变化的，实际上这种光滑性对算法收敛有更好的保证。

提纲

- 一、强化学习问题定义
- 二、基于价值的强化学习
- 三、基于策略的强化学习
- 四、深度强化学习的应用

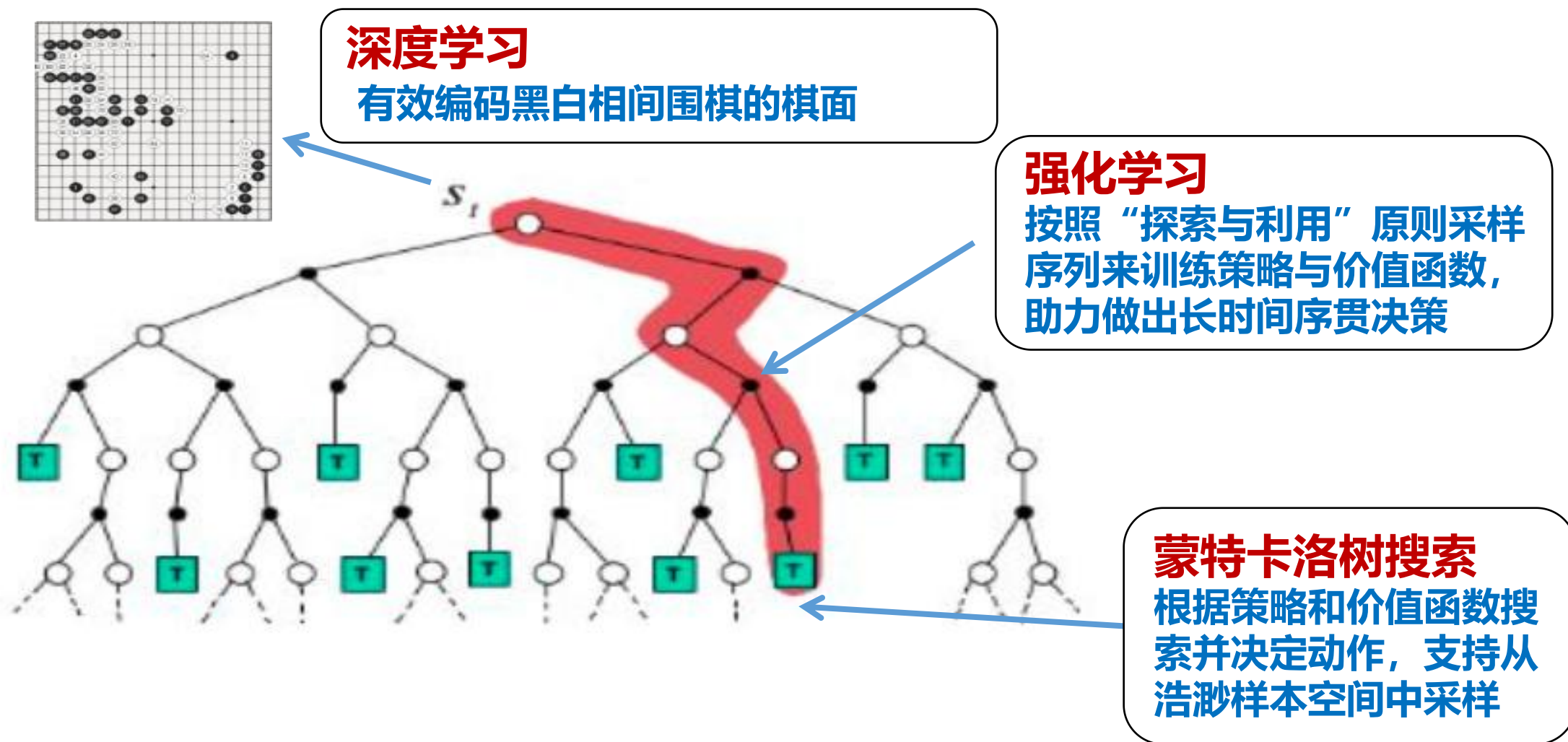
深度Q学习的应用实例：围棋博弈

- 围棋游戏一个片段的轨迹



深度Q学习的应用实例：围棋博弈

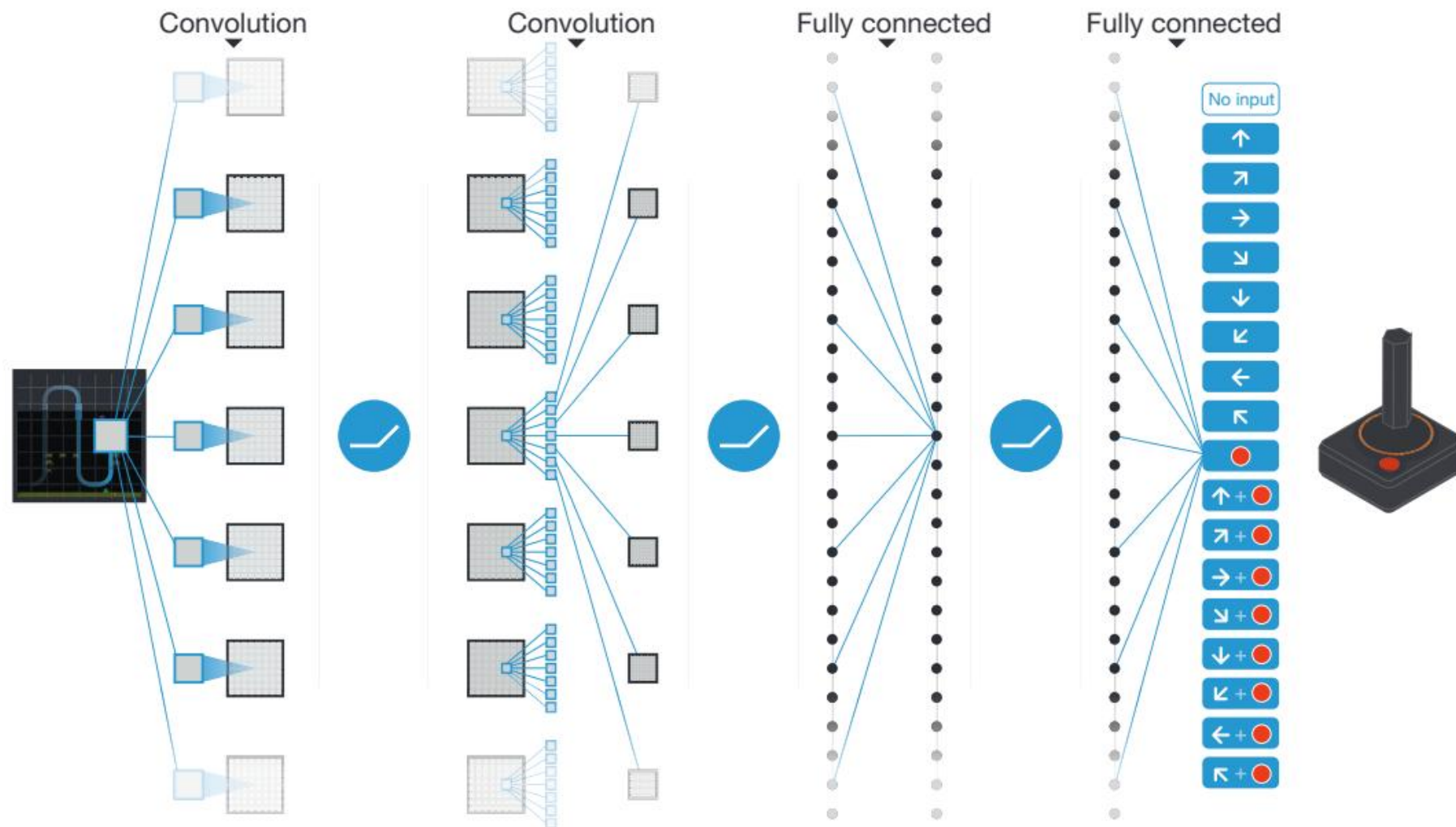
- AlphaGo算法的三个重要组成部分



深度Q学习的应用实例: 雅达利游戏

• 用于游戏的DQN动作-价值函数模型

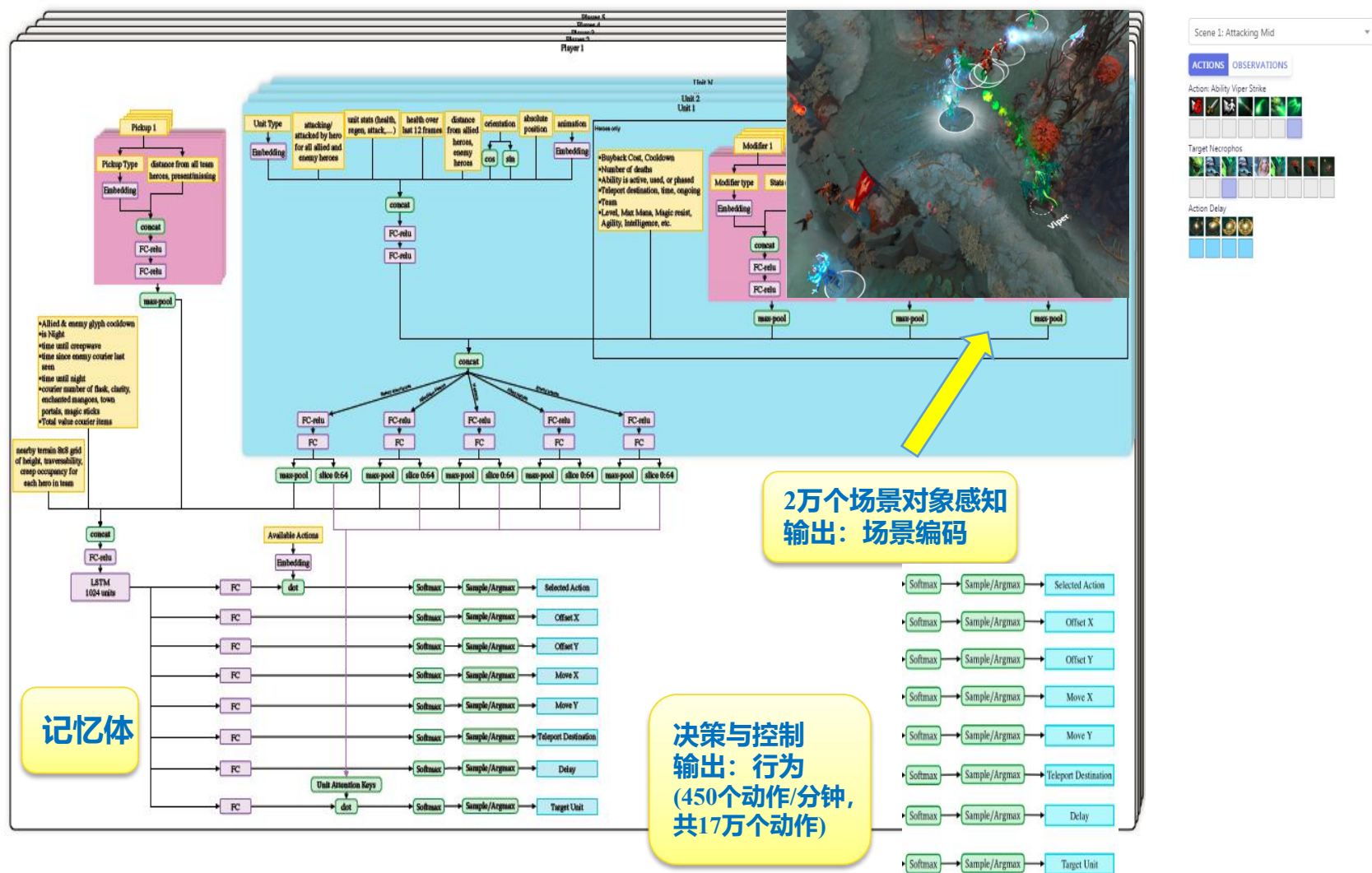
q 函数的学习模型



Mnih, Volodymyr, et al, Human-level control through deep reinforcement learning, Nature 518.7540 (2015)

博弈对抗的算法例子

• DOTA2 (状态空间极其庞大下完全信息下博弈)



谢谢!