中国农业大学
China Agricultural University

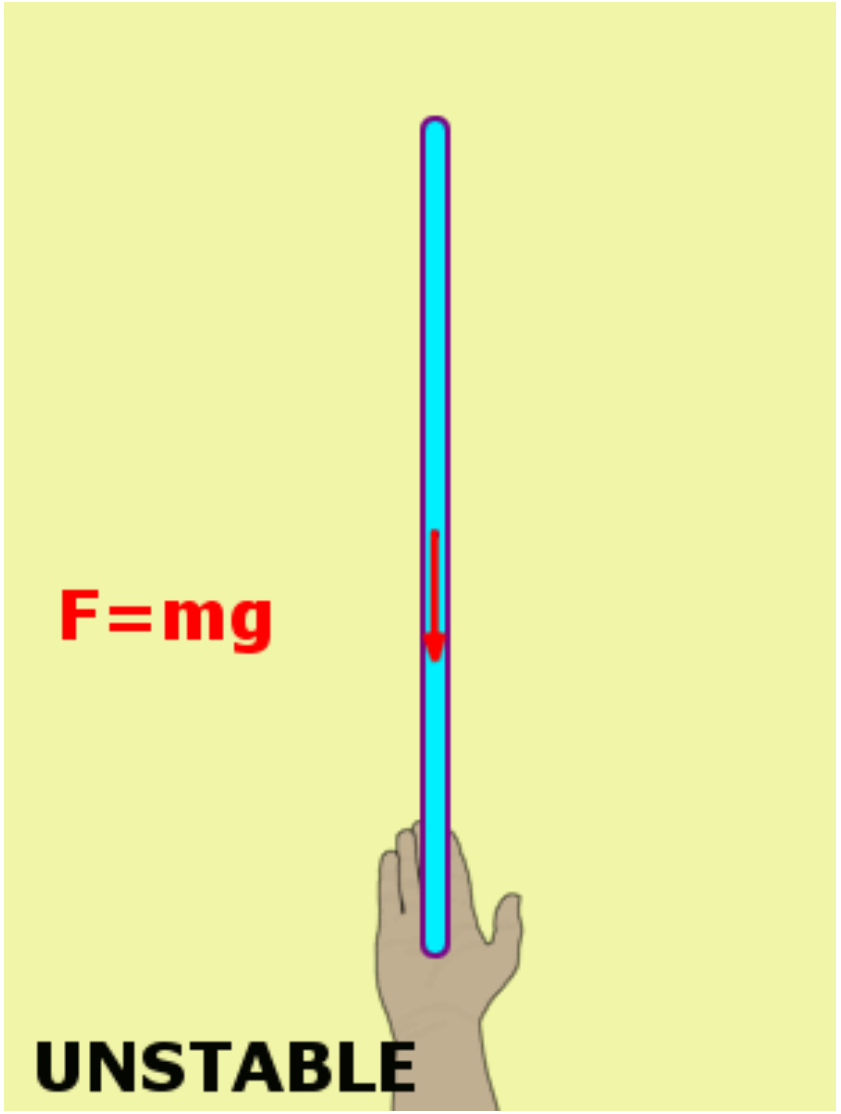
人工智能——深度强化学习

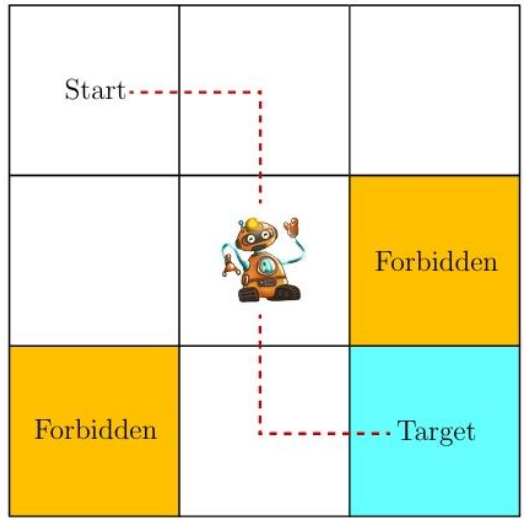
胡标

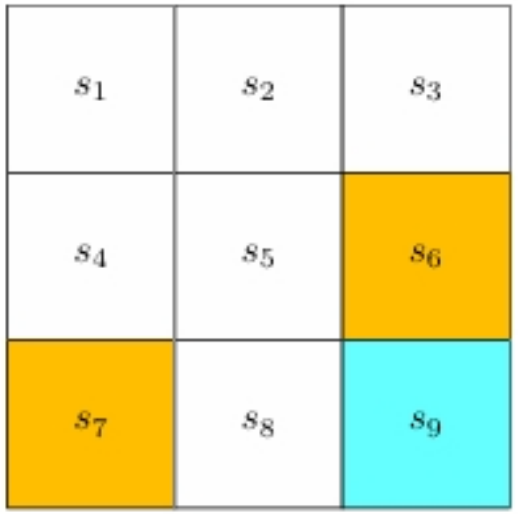
目录
Contents

1. 强化学习基本要素
2. 策略评估
3. 基于模型的强化学习
4. 模型无关的强化学习
5. 深度强化学习
6. 深度强化学习应用

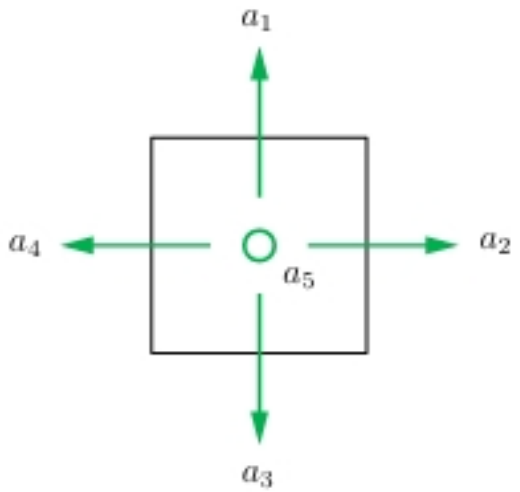
一个例子







(a) States



(b) Actions

交互

- 智能体（Agent）
  - 感知外界环境的状态（State）和奖励反馈（Reward），并进行学习和决策。智能体的决策功能是指根据外界环境的状态来做出不同的动作（Action），而学习功能是指根据外界环境的奖励来调整策略。
- 环境（Environment）
  - 智能体外部的所有事物，并受智能体动作的影响而改变其状态，并反馈给智能体相应的奖励。

强化学习中的基本要素

- 环境的状态集合：S；
- 智能体的动作集合：A；
- 状态转移概率：$p(s'|s,a)$，即智能体根据当前状态s做出一个动作a之后，下一个时刻环境处于不同状态s'的概率；
- 即时奖励：$R: S \times A \times S' \rightarrow R$，即智能体根据当前状态做出一个动作之后，环境会反馈给智能体一个奖励，这个奖励和动作之后下一个时刻的状态有关。

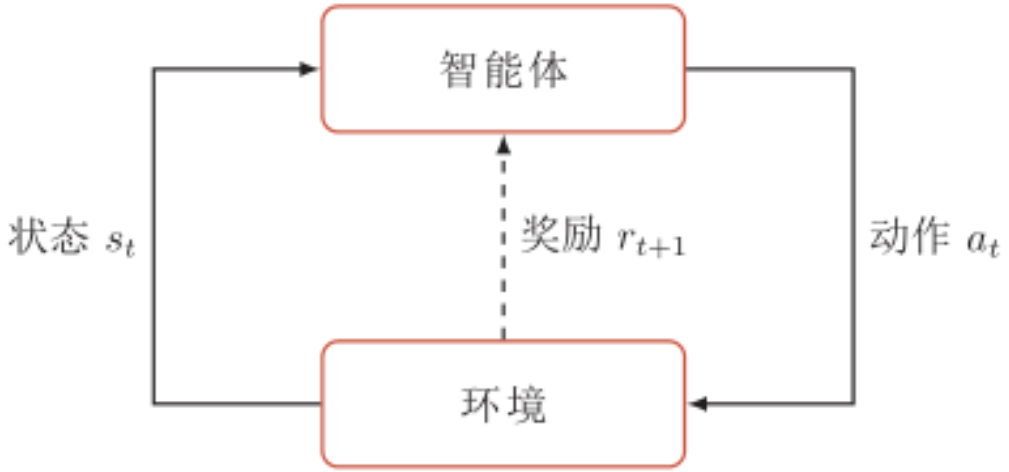
	a_1 (upward)	a_2 (rightward)	a_3 (downward)	a_4 (leftward)	a_5 (still)
s_1	s_1	s_2	s_4	s_1	s_1
s_2	s_2	s_3	s_5	s_1	s_2
s_3	s_3	s_3	s_6	s_2	s_3
s_4	s_1	s_5	s_7	s_4	s_4
s_5	s_2	s_6	s_8	s_4	s_5
s_6	s_3	s_6	s_9	s_5	s_6
s_7	s_4	s_8	s_7	s_7	s_7
s_8	s_5	s_9	s_8	s_7	s_8
s_9	s_6	s_9	s_9	s_8	s_9

比如：

$$p(s_1|s_1,a_2)=0$$
$$p(s_2|s_1,a_2)=1$$
$$p(s_3|s_1,a_2)=0$$
$$p(s_4|s_1,a_2)=0$$
$$p(s_5|s_1,a_2)=0$$

强化学习

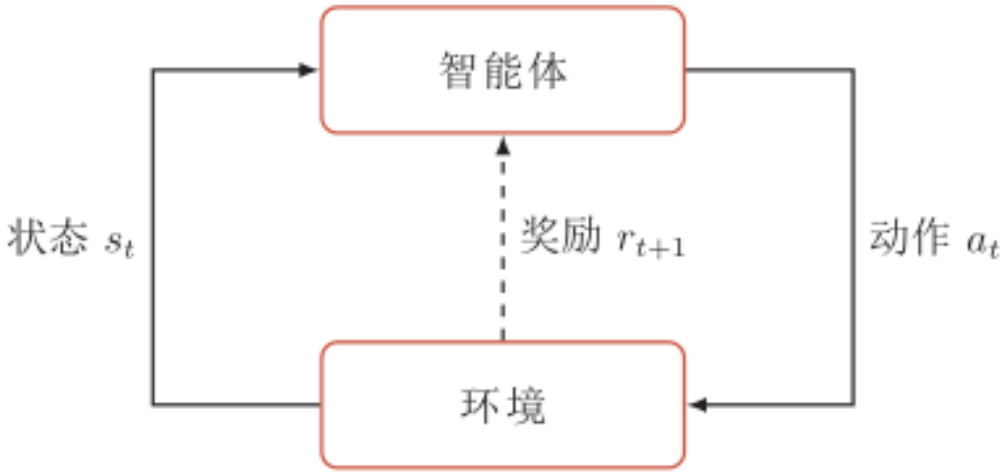
- 强化学习问题可以描述为一个智能体从与环境的交互中不断学习以完成特定目标（比如取得最大奖励值）。
- 强化学习就是智能体不断与环境进行交互，并根据经验调整其策略来最大化其长远的所有奖励的累积值。



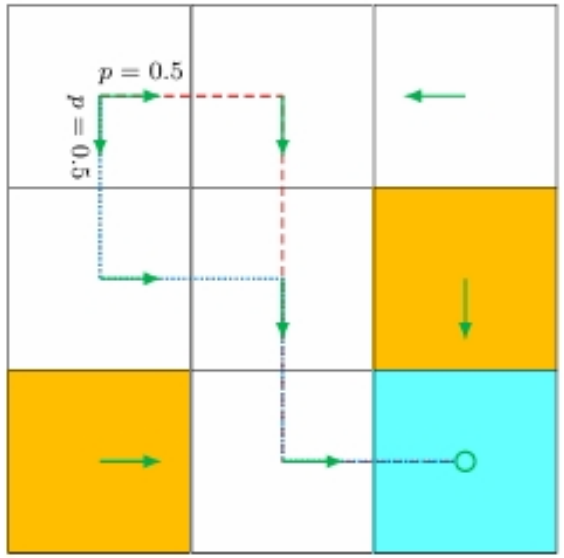
马尔可夫过程

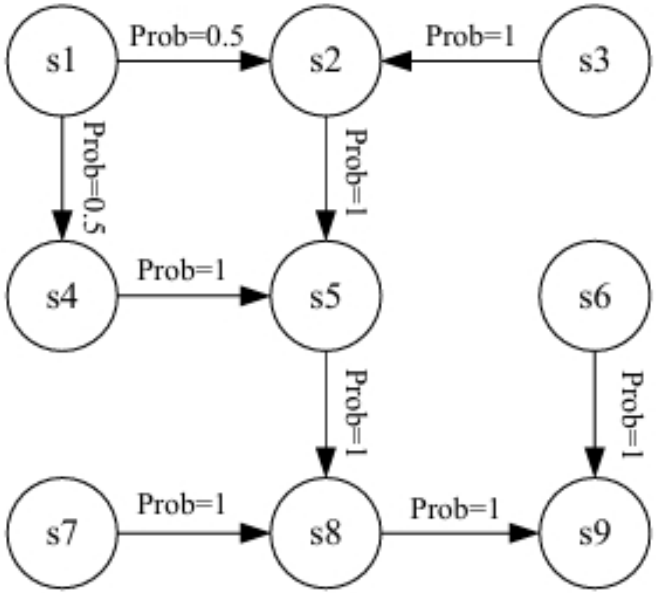
$$P(s_{t+1}|s_t, a_t, \cdots, s_0, a_0) = p(s_{t+1}|s_t, a_t)$$

$$P(r_{t+1}|s_t, a_t, \cdots, s_0, a_0) = p(r_{t+1}|s_t, a_t)$$



$$s_0, a_0, s_1, r_1, a_1, \cdots, s_{t-1}, a_{t-1}, s_t, r_t, \cdots$$





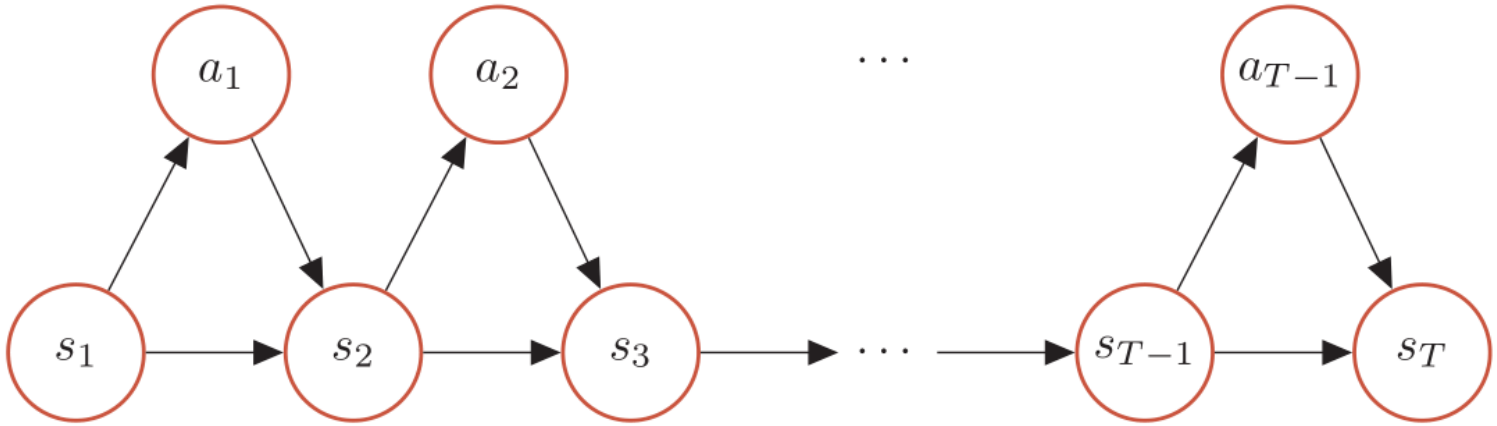
策略$\pi(a|s)$

- 马尔可夫决策过程的一个轨迹（trajectory）

$$\tau = s_0, a_0, s_1, r_1, a_1, \cdots, s_{T-1}, a_{T-1}, s_T$$

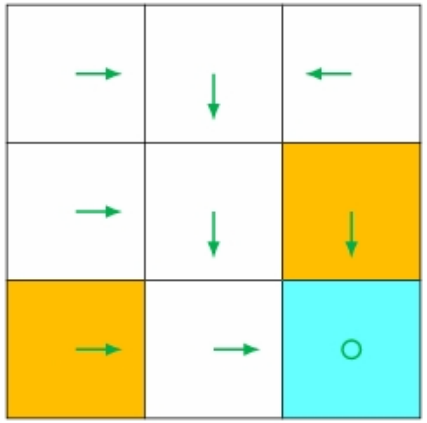
- τ的概率

$$p(\tau) = p(s_0, a_0, s_1, r_1, a_1, \cdots, s_{T-1}, a_{T-1}, s_T) = p(s_0) \prod_{t=0}^{T-1} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t)$$

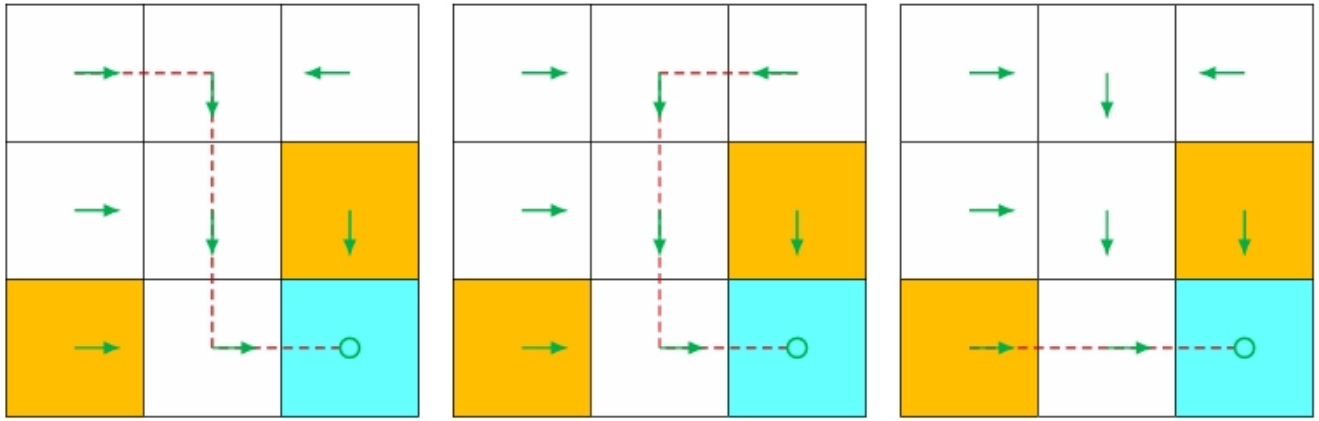


策略$\pi(a|s)$

$\pi(a_2|s_1) = 1$ $\pi(a_4|s_3) = 1$



(a) A deterministic policy

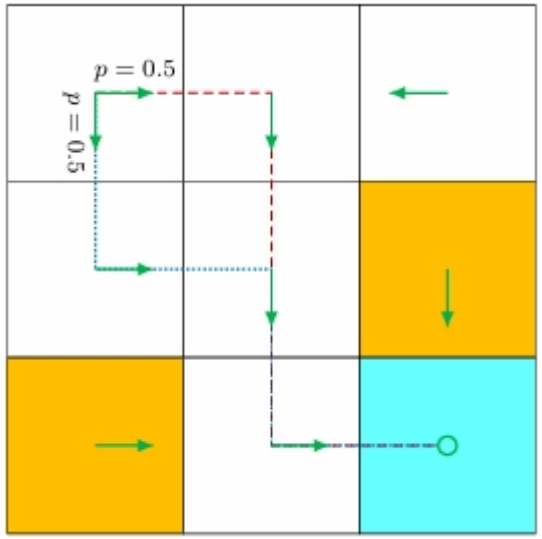


(b) Trajectories obtained from the policy

确定性策略

$\pi(a_2|s_1) = 0.5$

$\pi(a_3|s_1) = 0.5$



随机策略

总回报

- 给定策略$\pi(a|s)$，智能体和环境一次交互过程的轨迹τ 所收到的累积奖励为总回报（return）

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$$

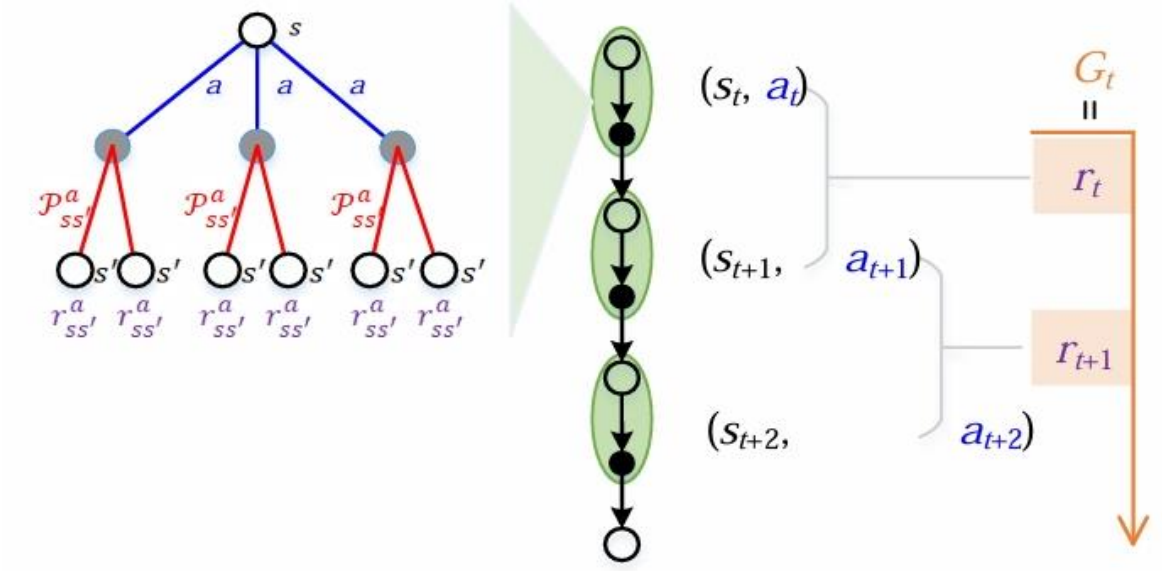
- $\gamma \in [0,1]$是折扣率。当γ接近于0时，智能体更在意短期回报；而当γ接近于1时，长期回报变得更重要。
- 环境中有一个或多个特殊的终止状态（terminal state）

强化学习目标函数

- 强化学习的目标是学习到一个策略$\pi_\theta(a|s)$来最大化期望回报（expected return）

$$J(\theta) = \mathrm{E}_{\tau \sim p_\theta(\tau)}[G(\tau)] = \mathrm{E}_{\tau \sim p_\theta(\tau)}\left[\sum_{t=0}^{T-1} \gamma^t r_{t+1}\right]$$

- θ为策略函数的参数

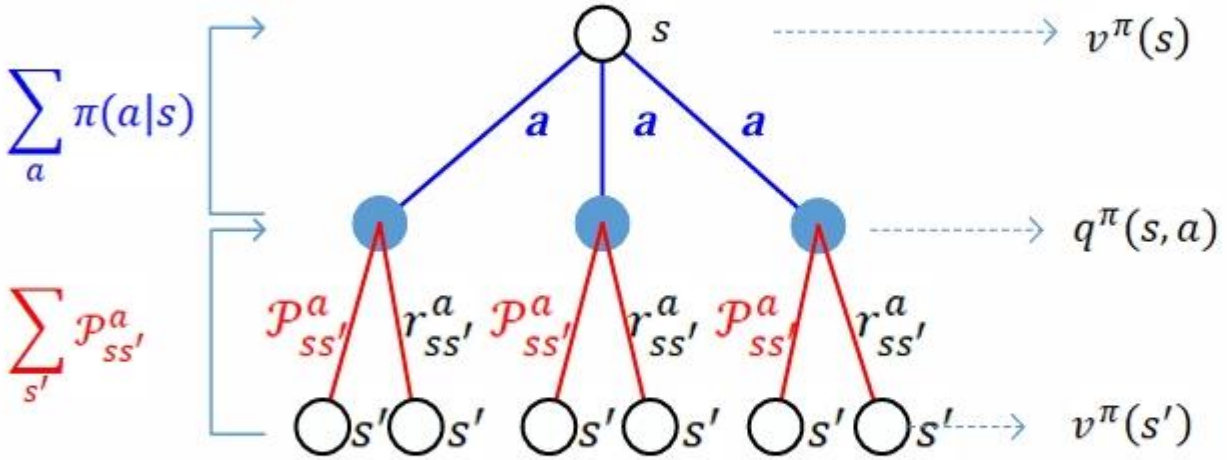


目录
Contents

1. 强化学习基本要素
2. 策略评估
3. 基于模型的强化学习
4. 模型无关的强化学习
5. 深度强化学习
6. 深度强化学习应用

如何评估策略$\pi_\theta(a|s)$

$$v^\pi(s) = \sum_{a\in\mathcal{A}} \pi(a|s) q^\pi(s,a)$$



$$q^\pi(s,a) = \sum_{s'\in\mathcal{S}} \mathcal{P}^a_{ss'} \left(r^a_{ss'} + \gamma v^\pi(s') \right)$$

如何评估策略$\pi_\theta(a|s)$

- 状态值函数：从状态s开始，执行策略π得到的期望总回报

$$V^{\pi}(s)=\mathrm{E}_{\tau\sim p_\theta(\tau)}\left[\sum_{t=0}^{T-1}\gamma^{t}r_{t+1}\mid\tau_{s_0}=s\right]=\mathrm{E}_{\tau\sim p_\theta(\tau)}\left[r_1+\sum_{t=1}^{T-1}\gamma^{t-1}r_{t+1}\mid\tau_{s_0}=s\right]$$

$$=\mathrm{E}_{\tau\sim p_\theta(\tau)}\mathrm{E}_{s'\sim p\,(s'|s,a)}\left[r(s,a,s')+\gamma\sum_{t=1}^{T-1}\gamma^{t-1}r_{t+1}\mid\tau_{s_1}=s'\right]$$

$$=\mathrm{E}_{\tau\sim p_\theta(\tau)}\mathrm{E}_{s'\sim p\,(s'|s,a)}[r(s,a,s')+\gamma V^{\pi}(s')]\leftarrow \text{贝尔曼方程}$$

如何评估策略$\pi_\theta(a|s)$

状态-动作值函数（Q函数）

$$Q^\pi(s,a) = \mathrm{E}_{s'\sim p_\theta(s'|s,a)}[r(s,a,s') + \gamma V^\pi(s')]$$

$$= \mathrm{E}_{s'\sim p_\theta(s'|s,a)}\left[r(s,a,s') + \gamma \mathrm{E}_{a'\sim\pi(a'|s')}[Q^\pi(s',a')]\right] \leftarrow \text{贝尔曼方程}$$

最优策略

- 最优策略：存在一个最优的策略π^*，其在所有状态上的期望回报最大。

$$\forall s, \pi^* = \arg\max V^{\pi}(s)$$

- 策略改进：
  - 值函数可以看作是对策略π的评估。
  - 如果在状态s，有一个动作a使得$Q^{\pi}(s,a) > V^{\pi}(s)$，说明执行动作$a$比当前的策略$\pi(a|s)$要好，我们就可以调整参数使得策略$\pi(a|s)$的概率增加。

如何计算值函数?

- 基于模型的强化学习算法
  - 基于MDP过程：状态转移概率$p(s'|s,a)$和奖励函数$R(s,a,s')$
  - 策略迭代
  - 值迭代

- 模型无关的强化学习
  - 无MDP过程
  - 蒙特卡罗采样方法
  - 时序差分学习

目录
Contents

1. 强化学习基本要素
2. 策略评估
3. **基于模型的强化学习**
4. 模型无关的强化学习
5. 深度强化学习
6. 深度强化学习应用

策略迭代

算法 15.1: 策略迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s, \forall a, \pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  // 策略评估

  repeat

    根据贝尔曼方程（公式(15.18)），计算 $V^{\pi}(s)$, $\forall s$;

  until $\forall s$，$V^{\pi}(s)$ 收敛;

  // 策略改进

  根据公式(15.19)，计算 $Q(s,a)$;

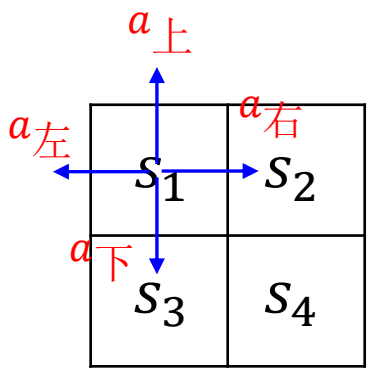
  $\forall s, \pi(s) = \arg\max_a Q(s,a)$;

until $\forall s$，$\pi(s)$ 收敛;

输出: 策略 π

示例场景

假设我们有一个2x2的网格世界(Grid World)，有 4个状态 $S = \{s_1, s_2, s_3, s_4\}$，并且每个状态都有 4个动作(上、下、左、右)可供选择，即动作集合 $A = \{a_{上}, a_{下}, a_{左}, a_{右}\}$。如果代理人尝试移动到网格外，则会保持在原位。目标是找到让累积奖励最大的策略。



- s_4是目标状态
- 移动一步扣1分
- 到达目标状态不扣分
- 折扣因子$\gamma = 0.9$

策略迭代

算法 15.1: 策略迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s, \forall a, \pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  // 策略评估

  repeat

    根据贝尔曼方程（公式(15.18)），计算$V^{\pi}(s)$, $\forall s$;

  until $\forall s$，$V^{\pi}(s)$ 收敛;

  // 策略改进

  根据公式(15.19)，计算$Q(s,a)$;

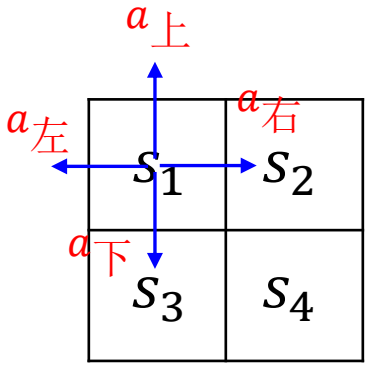
  $\forall s, \pi(s) = \arg\max_a Q(s,a)$;

until $\forall s$，$\pi(s)$ 收敛;

输出: 策略π

迭代1：初始状态价值和策略

- **初始化状态价值：**所有状态的初始值设为0，即$V^{\pi}(s) = 0$
- **初始化策略：**每个状态下四个动作的概率均等，$\pi(a|s) = \frac{1}{4}$



- s_4是目标状态
- 移动一步扣1分
- 到达目标状态不扣分
- 折扣因子$\gamma = 0.9$

策略迭代

算法 15.1: 策略迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s, \forall a, \pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  // 策略评估

  repeat

    根据贝尔曼方程（公式(15.18)），计算$V^{\pi}(s)$, $\forall s$;

  until $\forall s$，$V^{\pi}(s)$ 收敛;

  // 策略改进

  根据公式(15.19)，计算$Q(s,a)$;

  $\forall s, \pi(s) = \arg\max_a Q(s,a)$;

until $\forall s$，$\pi(s)$ 收敛;

输出: 策略π

迭代2：策略评估

- 根据当前策略计算状态价值$V^{\pi}(s)$。由于每个动作的概率均等，我们可以使用贝尔曼方程进行状态价值更新

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} p(s'|s,a)\left[r(s,a,s') + \gamma V^{\pi}(s')\right]$$

因为转移概率$p(s'|s,a) = 1$，第一次更新($\boldsymbol{V^{\pi}(s_1) = V^{\pi}(s_2) = V^{\pi}(s_3) = V^{\pi}(s_4) = 0}$)

$$\boldsymbol{V^{\pi}(s_1) = \frac{1}{4}\left(\left(-1 + 0.9 \cdot V^{\pi}(s_1)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_2)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_3)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_1)\right)\right) = -1}$$

$$\boldsymbol{V^{\pi}(s_2) = \frac{1}{4}\left(\left(-1 + 0.9 \cdot V^{\pi}(s_2)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_2)\right) + \left(0.9 \cdot V^{\pi}(s_4)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_1)\right)\right) = -\frac{3}{4}}$$

$$\boldsymbol{V^{\pi}(s_3) = \frac{1}{4}\left(\left(-1 + 0.9 \cdot V^{\pi}(s_1)\right) + \left(0.9 \cdot V^{\pi}(s_4)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_3)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_3)\right)\right) = -\frac{3}{4}}$$

$$\boldsymbol{V^{\pi}(s_4) = \frac{1}{4}\left(\left(-1 + 0.9 \cdot V^{\pi}(s_2)\right) + \left(0.9 \cdot V^{\pi}(s_4)\right) + \left(0.9 \cdot V^{\pi}(s_4)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_3)\right)\right) = -\frac{1}{2}}$$

策略迭代

算法 15.1: 策略迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s, \forall a, \pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  // 策略评估

  repeat

    根据贝尔曼方程（公式(15.18)），计算$V^{\pi}(s)$, $\forall s$;

  until $\forall s$，$V^{\pi}(s)$ 收敛;

  // 策略改进

  根据公式(15.19)，计算$Q(s,a)$;

  $\forall s, \pi(s) = \arg\max_a Q(s,a)$;

until $\forall s$，$\pi(s)$ 收敛;

输出: 策略π

迭代2：策略评估

- 根据当前策略计算状态价值$V^{\pi}(s)$。由于每个动作的概率均等，我们可以使用贝尔曼方程进行状态价值更新

$$V^{\pi}(s) = \sum_{a \in A} \pi(a|s) \sum_{s' \in S} p(s'|s,a)\left[r(s,a,s') + \gamma V^{\pi}(s')\right]$$

因为转移概率$p(s'|s,a) = 1$，第二次更新

$$\left(V^{\pi}(s_1) = -1, V^{\pi}(s_2) = -\frac{3}{4}, V^{\pi}(s_3) = -\frac{3}{4}, V^{\pi}(s_4) = -\frac{1}{2}\right)$$

$$V^{\pi}(s_1) = \frac{1}{4}\left(\left(-1 + 0.9 \cdot V^{\pi}(s_1)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_2)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_3)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_1)\right)\right) = ?$$

$$V^{\pi}(s_2) = \frac{1}{4}\left(\left(-1 + 0.9 \cdot V^{\pi}(s_2)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_2)\right) + \left(0.9 \cdot V^{\pi}(s_4)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_1)\right)\right) = ?$$

$$V^{\pi}(s_3) = \frac{1}{4}\left(\left(-1 + 0.9 \cdot V^{\pi}(s_1)\right) + \left(0.9 \cdot V^{\pi}(s_4)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_3)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_3)\right)\right) = ?$$

$$V^{\pi}(s_4) = \frac{1}{4}\left(\left(-1 + 0.9 \cdot V^{\pi}(s_2)\right) + \left(0.9 \cdot V^{\pi}(s_4)\right) + \left(0.9 \cdot V^{\pi}(s_4)\right) + \left(-1 + 0.9 \cdot V^{\pi}(s_3)\right)\right) = ?$$

一直更新，直到收敛

策略迭代

算法 15.1: 策略迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s, \forall a, \pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  // 策略评估

  repeat

    根据贝尔曼方程（公式(15.18)），计算 $V^{\pi}(s)$, $\forall s$;

  until $\forall s$，$V^{\pi}(s)$ 收敛;

  // 策略改进

  根据公式(15.19)，计算 $Q(s,a)$;

  $\forall s, \pi(s) = \arg\max_a Q(s,a)$;

until $\forall s$，$\pi(s)$ 收敛;

输出: 策略 π

迭代3：策略改进

- 利用上述状态价值来改进策略。在每个状态选择使得 $Q(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^{\pi}(s')$最大的动作。

1. 状态s_1：往右或者往下使Q值最大，更新策略$\pi\left(a_{下} \middle| s_2\right) = 1$
2. 状态s_2：往下使Q值最大，更新策略$\pi\left(a_{下} \middle| s_2\right) = 1$
3. 状态s_3：往右使Q值最大，更新策略$\pi\left(a_{右} \middle| s_2\right) = 1$
4. 状态s_4：往右使Q值最大，更新策略$\pi\left(a_{右} \middle| s_2\right) = 1$

迭代4：再次进行策略评估

迭代5：再次进行策略改进，发现策略不变，收敛，停止

值迭代

- 值迭代方法将策略评估和策略改进两个过程合并，来直接计算出最优策略。

算法 15.2: 值迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s \in \mathcal{S}, V(s) = 0$;

repeat

$$\forall s,\ V(s) \leftarrow \max_a \mathbb{E}_{s' \sim p(s'|s,a)} \left[r(s,a,s') + \gamma V(s') \right] ;$$

until $\forall s$，$V(s)$ 收敛;

根据公式 (15.19) 计算 $Q(s,a)$;

$\forall s, \pi(s) = \arg\max_a Q(s,a)$;

输出: 策略 π

迭代1：初始状态价值和策略

- **初始化状态价值：**所有状态的初始值设为0，即$\boldsymbol{V(s_1) = V(s_2) = V(s_3) = V(s_4) = 0}$
- **第1次迭代**

1. 状态s_1的价值更新
   - 右：$-1 + 0.9 \cdot V(s_2) = -1$
   - 下：$-1 + 0.9 \cdot V(s_3) = -1$
   - 上和左：$-1 + 0.9 \cdot V(s_1) = -1$
   - 取最大值，更新$V(s_1) = -1$
2. 状态s_2的价值更新
   - 左：$-1 + 0.9 \cdot V(s_1) = -1$
   - 下：$0.9 \cdot V(s_4) = 0$
   - 上和右：$-1 + 0.9 \cdot V(s_2) = -1$
   - 取最大值，更新$V(s_2) = 0$

值迭代

- 值迭代方法将策略评估和策略改进两个过程合并，来直接计算出最优策略。

算法 15.2: 值迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s \in \mathcal{S}, V(s) = 0$;

repeat

$$\forall s,\ V(s) \leftarrow \max_a \mathbb{E}_{s' \sim p(s'|s,a)} \left[r(s,a,s') + \gamma V(s') \right] ;$$

until $\forall s$，$V(s)$ 收敛;

根据公式 (15.19) 计算 $Q(s,a)$;

$\forall s, \pi(s) = \arg\max_a Q(s,a)$;

输出: 策略 π

迭代1：初始状态价值和策略

- **初始化状态价值：**所有状态的初始值设为0，即$\boldsymbol{V(s_1) = V(s_2) = V(s_3) = V(s_4) = 0}$
- **第1次迭代**

1. 状态s_3的价值更新
   - 右：$0.9 \cdot V(s_4) = 0$
   - 上：$-1 + 0.9 \cdot V(s_3) = -1$
   - 下和左：$-1 + 0.9 \cdot V(s_3) = -1$
   - 取最大值，更新$V(s_3) = 0$
2. 状态s_4的价值更新
   - 左：$-1 + 0.9 \cdot V(s_3) = -1$
   - 上：$-1 + 0.9 \cdot V(s_2) = -1$
   - 下和右：$0.9 \cdot V(s_4) = 0$
   - 取最大值，更新$V(s_4) = 0$

值迭代

- 值迭代方法将策略评估和策略改进两个过程合并，来直接计算出最优策略。

算法 15.2: 值迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s \in \mathcal{S}, V(s) = 0$;

repeat

$$\forall s,\ V(s) \leftarrow \max_a \mathbb{E}_{s' \sim p(s'|s,a)} \left[r(s,a,s') + \gamma V(s') \right] ;$$

until $\forall s$，$V(s)$ 收敛;

根据公式 (15.19) 计算 $Q(s,a)$;

$\forall s, \pi(s) = \arg\max_a Q(s,a)$;

输出: 策略 π

- **第1次迭代：** $\boldsymbol{V(s_1) = -1, V(s_2) = 0, V(s_3) = 0, V(s_4) = 0}$
- **第2次迭代**

1. 状态s_1的价值更新
   - 右：$-1 + 0.9 \cdot V(s_2) = -1$
   - 下：$-1 + 0.9 \cdot V(s_3) = -1$
   - 上和左：$-1 + 0.9 \cdot V(s_1) = -1.9$
   - 取最大值，更新$V(s_1) = -1$
2. 状态s_2的价值更新
   - 左：$-1 + 0.9 \cdot V(s_1) = -1.9$
   - 下：$0.9 \cdot V(s_4) = 0$
   - 上和右：$-1 + 0.9 \cdot V(s_2) = -1$
   - 取最大值，更新$V(s_2) = 0$

值迭代

- 值迭代方法将策略评估和策略改进两个过程合并，来直接计算出最优策略。

算法 15.2: 值迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s \in \mathcal{S}, V(s) = 0$;

repeat

$$\forall s,\ V(s) \leftarrow \max_a \mathbb{E}_{s' \sim p(s'|s,a)} \left[r(s,a,s') + \gamma V(s') \right] ;$$

until $\forall s$，$V(s)$ 收敛;

根据公式 (15.19) 计算 $Q(s,a)$;

$\forall s, \pi(s) = \arg\max_a Q(s,a)$;

输出: 策略 π

- **第1次迭代：** $\boldsymbol{V(s_1) = -1, V(s_2) = 0, V(s_3) = 0, V(s_4) = 0}$
- **第2次迭代**

1. 状态s_3的价值更新
   - 右：$0.9 \cdot V(s_4) = 0$
   - 上：$-1 + 0.9 \cdot V(1) = -1.9$
   - 下和左：$-1 + 0.9 \cdot V(s_3) = -1$
   - 取最大值，更新$V(s_3) = 0$
2. 状态s_4的价值更新
   - 左：$-1 + 0.9 \cdot V(s_3) = -1$
   - 上：$-1 + 0.9 \cdot V(s_2) = -1$
   - 下和右：$0.9 \cdot V(s_4) = 0$
   - 取最大值，更新$V(s_4) = 0$

已经收敛，停止更新

策略迭代

算法 15.2: 值迭代算法

输入: MDP 五元组：$\mathcal{S}, \mathcal{A}, P, r, \gamma$;

初始化：$\forall s \in \mathcal{S}, V(s) = 0$;

repeat

$$\forall s,\ V(s) \leftarrow \max_a \mathbb{E}_{s' \sim p(s'|s,a)} \left[r(s,a,s') + \gamma V(s') \right] ;$$

until $\forall s$，$V(s)$ 收敛;

根据公式 (15.19) 计算 $Q(s,a)$;

$\forall s, \pi(s) = \arg\max_a Q(s,a)$;

输出: 策略 π

策略改进

- 利用上述状态价值来改进策略。在每个状态选择使得

$$Q(s,a) = r(s,a) + \gamma \sum_{s'} p(s'|s,a) V^{\pi}(s')$$

最大的动作。

1. 状态s_1：往右或者往下使Q值最大，更新策略$\pi\left(a_{下} \middle| s_2\right) = 1$
2. 状态s_2：往下使Q值最大，更新策略$\pi\left(a_{下} \middle| s_2\right) = 1$
3. 状态s_3：往右使Q值最大，更新策略$\pi\left(a_{右} \middle| s_2\right) = 1$
4. 状态s_4：往右使Q值最大，更新策略$\pi\left(a_{右} \middle| s_2\right) = 1$

输出策略π

策略迭代和值迭代对比

特点	策略迭代	值迭代
核心思想	分为策略评估和策略改进的循环	直接在值函数上进行 Bellman 更新
每次迭代计算量	较高，策略评估需要多次迭代	较低，每次迭代都是一步 Bellman 更新
收敛速度	通常收敛稍慢	通常收敛理更快
策略维护	需要显式维护策略	不需要显式维护策略
适用场景	状态空间较小、需要精确策略评估的场景	状态空间较大、需要直接通近最优值函数的场界

目录
Contents

1. 强化学习基本要素
2. 策略评估
3. 基于模型的强化学习
4. **模型无关的强化学习**
5. 深度强化学习
6. 深度强化学习应用

策略学习过程

- 通过采样的方式来计算值函数

$$Q_\pi(s,a) \approx \hat{Q}_\pi(s,a) = \frac{1}{N}\sum_{n=1}^{N} G\left(\tau^{(n)}\right)$$

- 当$N \to \infty$时，$\hat{Q}_\pi(s,a) \to Q_\pi(s,a)$。

- 在估计出$Q_\pi(s,a)$之后，就可以进行策略改进。
- 然后在新的策略下重新通过采样来估计Q函数，并不断重复，直至收敛。

ϵ-贪心法

- 利用和探索
  - 对当前策略的利用（Exploitation），
  - 对环境的探索（Exploration）以找到更好的策略
- 对于一个确定性策略π，其对应的ϵ −贪心法策略为

$$\pi^{\epsilon}(s)=\begin{cases}\pi(s), & \text{按概率}1-\epsilon\\ \text{随机选择}A\text{中的动作}, & \text{按概率}\epsilon\end{cases}$$

Q学习算法

- Q学习算法不通过π^{ϵ}来选下一步的动作a'，而是直接选最优的Q函数，

算法 15.4: Q学习算法

输入: 状态空间$\mathcal{S}$，动作空间$\mathcal{A}$，折扣率γ，学习率α

随机初始化$Q(s,a)$;

$\forall s, \forall a, \pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  初始化起始状态s;

  repeat

    在状态s，选择动作$a = \pi^{\epsilon}(s)$;

    执行动作a，得到即时奖励r和新状态s';

    $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$;

    $s \leftarrow s'$;

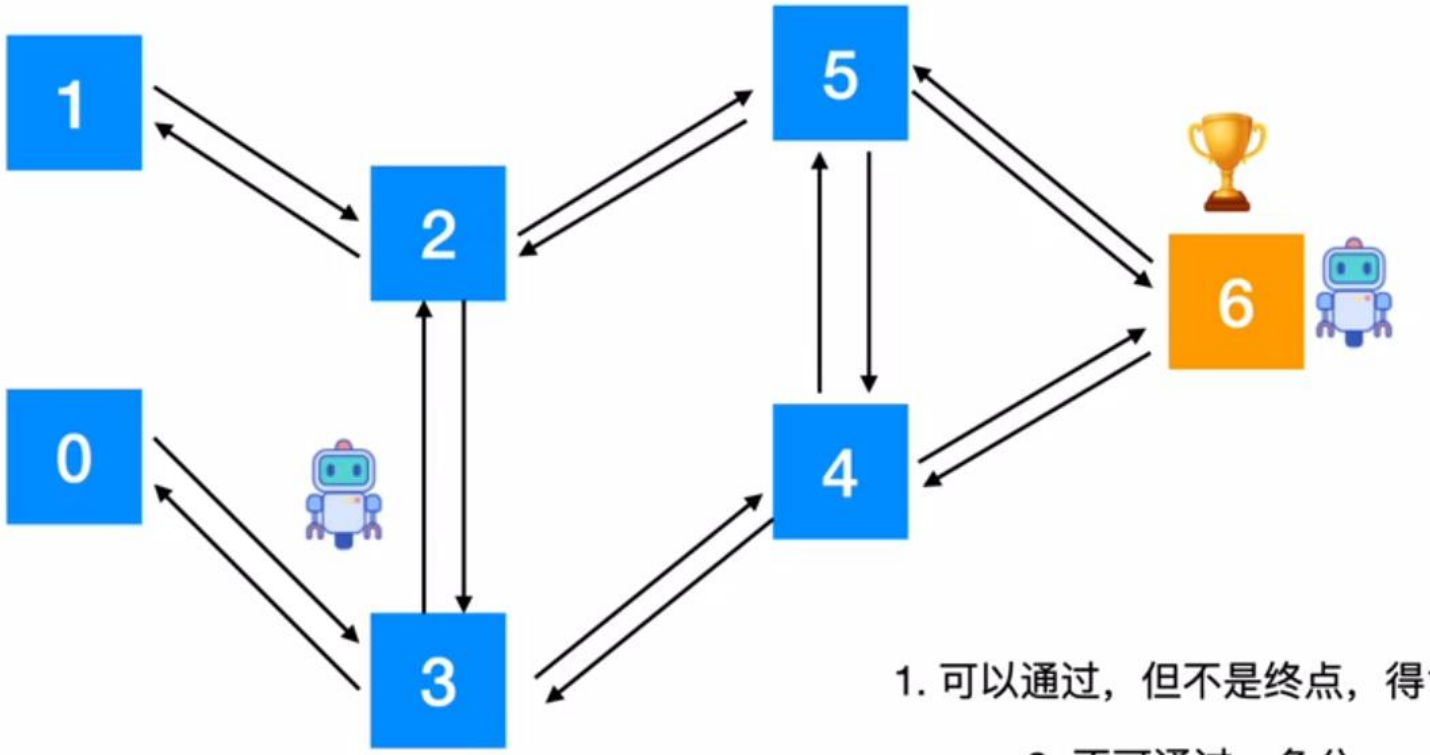
  until s为终止状态;

until $\forall s, a$，$Q(s,a)$收敛;

输出: 策略$\pi(s) = \arg\max_{a \in |\mathcal{A}|} Q(s,a)$

Q学习算法

多步决策过程问题



第一步：分析下小机器人在各个位置上能做的选择

	0	1	2	3	4	5	6
0	-1	-1	-1	1	-1	-1	-1
1	-1	-1	1	-1	-1	-1	-1
2	-1	1	-1	1	-1	1	-1
3	1	-1	1	-1	1	-1	-1
4	-1	-1	-1	1	-1	1	100
5	-1	-1	1	-1	1	-1	100
6	-1	-1	-1	-1	1	1	100

1. 可以通过，但不是终点，得1分

2. 不可通过，负分

3. 可以通过，且是终点 得100分

除了终点，机器人在其余位置，则必须选择下一个不同点，不能原地不动

Q学习算法

第一步：分析下小机器人在各个位置上能做的选择

	0	1	2	3	4	5	6
0	-1	-1	-1	1	-1	-1	-1
1	-1	-1	1	-1	-1	-1	-1
2	-1	1	-1	1	-1	1	-1
3	1	-1	1	-1	1	-1	-1
4	-1	-1	-1	1	-1	1	100
5	-1	-1	1	-1	1	-1	100
6	-1	-1	-1	-1	1	1	100

动态规划

大问题由小问题组成

1. 机器人每一步的选择将决定其最后能不能走到终点

小问题的求解可以递归实现

2. 机器人每一步做选择这件事情，是同质的

如何评价机器人在 s 时候选择 某个 a 时，的好坏?🤔

Step1:机器人在s 时，选择 a 的效果　同时机器人会到新的状态 s'

Step2:机器人在新状态s' 时，选择新的a' 的效果 同时机器人会到新的状态 s''

Step3: 机器人在更新的状态s''时，选择新的a'' 的效果

Step ...

只考虑Step1,Step2 两步

$$Q(s,a) = r(s,a) + \gamma \cdot max(Q(s',a'))$$

Q学习算法

第一步：分析下小机器人在各个位置上能做的选择

	0	1	2	3	4	5	6
0	-1	-1	-1	1	-1	-1	-1
1	-1	-1	1	-1	-1	-1	-1
2	-1	1	-1	1	-1	1	-1
3	1	-1	1	-1	1	-1	-1
4	-1	-1	-1	1	-1	1	100
5	-1	-1	1	-1	1	-1	100
6	-1	-1	-1	-1	1	1	100

大问题由小问题组成

1. 机器人每一步的选择将决定其最后能不能走到终点

小问题的求解可以递归实现

2. 机器人每一步面临的选择，这是同一个问题

$$Q(s,a) = r(s,a) + \gamma \cdot max(Q(s',a'))$$

γ 超参

$Q(2,1)$ 2 1 $r(2,1) = 1$

1 2 $Q(s',a')$

$Q(2,3)$ 2 3 $r(3,2) = 1$

3 $Q(s',a')$

0 2 4

$Q(1,2)$

初始化为0
在不断迭代中更新

$Q(3,0)$ $Q(3,2)$ $Q(3,4)$

Q学习算法

大问题由小问题组成

第一步：分析下小机器人在各个位置上能做的选择

	0	1	2	3	4	5	6
0	-1	-1	-1	1	-1	-1	-1
1	-1	-1	1	-1	-1	-1	-1
2	-1	1	-1	1	-1	1	-1
3	1	-1	1	-1	1	-1	-1
4	-1	-1	-1	1	-1	1	100
5	-1	-1	1	-1	1	-1	100
6	-1	-1	-1	-1	1	1	100

1. 机器人每一步的选择将决定其最后能不能走到终点

2. 机器人每一步面临的选择，这是同一个问题

小问题的求解可以递归实现

$$Q(s,a) = r(s,a) + \gamma \cdot max(Q(s',a'))$$

Q

	0	1	2	3	4	5	6
0	0	0	0	64	0	0	0
1	0	0	64	0	0	0	0
2	0	51.2	0	64	0	80	0
3	51.2	0	64	0	80	0	0
4	0	0	0	64	0	80	100
5	0	0	64	0	80	0	100
6	0	0	0	0	0	0	0

SARSA算法（State Action Reward State Action，SARSA）

算法 15.3: SARSA：一种同策略的时序差分学习算法

输入: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，学习率 α

随机初始化 $Q(s,a)$;

$\forall s, \forall a, \pi(a|s) = \frac{1}{|\mathcal{A}|}$;

repeat

  初始化起始状态 s;

  选择动作 $a = \pi^{\epsilon}(s)$;

  repeat

    执行动作 a，得到即时奖励 r 和新状态 s';

    在状态 s'，选择动作 $a' = \pi^{\epsilon}(s')$;

    $Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s',a') - Q(s,a)\Big)$;

    更新策略：$\pi(s) = \arg\max_{a \in |\mathcal{A}|} Q(s,a)$;

    $s \leftarrow s', a \leftarrow a'$;

  until s 为终止状态;

until $\forall s, a$，$Q(s,a)$ 收敛;

输出: 策略 $\pi(s)$

SARSA算法（State Action Reward State Action，SARSA）

第一步：分析下小机器人在各个位置上能做的选择

	0	1	2	3	4	5	6
0	-1	-1	-1	1	-1	-1	-1
1	-1	-1	1	-1	-1	-1	-1
2	-1	1	-1	1	-1	1	-1
3	1	-1	1	-1	1	-1	-1
4	-1	-1	-1	1	-1	1	100
5	-1	-1	1	-1	1	-1	100
6	-1	-1	-1	-1	1	1	100

如何评价机器人在 s 时候选择 某个 a 时，的好坏?🤔

Step1:机器人在s 时，选择 a 的效果　同时机器人会到新的状态 s'

Step2:机器人在新状态s' 时，选择新的a' 的效果　同时机器人会到新的状态 s''

Step3: 机器人在更新的状态s''时，选择新的a'' 的效果

Step …

Q-Learning

只考虑Step1,Step2 两步　　Step1　　Step2

$$Q(s,a) = r(s,a) + \gamma \cdot max(Q(s',a'))$$

SARSA

只考虑Step1,Step2 两步

$$Q(s,a) = r(s,a) + \gamma \cdot (Q(s',a'))$$

SARSA算法（State Action Reward State Action，SARSA）

如何评价机器人在 s 时候选择 某个 a 时，的好坏?🤔

Step1:机器人在s 时，选择 a 的效果　同时机器人会到新的状态 s'

Step2:机器人在新状态s' 时，选择新的a' 的效果　同时机器人会到新的状态 s''

Step3: 机器人在更新的状态s''时，选择新的a'' 的效果

Step …

Q-Learning

只考虑Step1,Step2 两步　　Step1　　Step2

$$Q(s,a) = r(s,a) + \gamma \cdot max(Q(s',a'))$$

到达新状态

在新状态的基础上， 选择可能动作中 收益最大的 动作

SARSA

只考虑Step1,Step2 两步　　Step1　　Step2

$$Q(s,a) = r(s,a) + \gamma \cdot (Q(s',a'))$$

到达新状态

在新状态的基础上，继续按照之前的策略，选择一个动作

目录
Contents

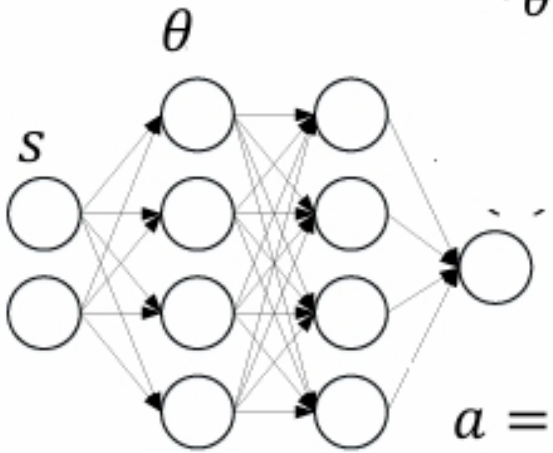
1. 强化学习基本要素
2. 策略评估
3. 基于模型的强化学习
4. 模型无关的强化学习
5. 深度强化学习
6. 深度强化学习应用

- 为了在连续的状态和动作空间中计算值函数$Q_\pi(s,a)$，我们可以用一个函数$Q_\theta(s,a)$来表示近似计算，称为值函数近似（Value Function Approximation）

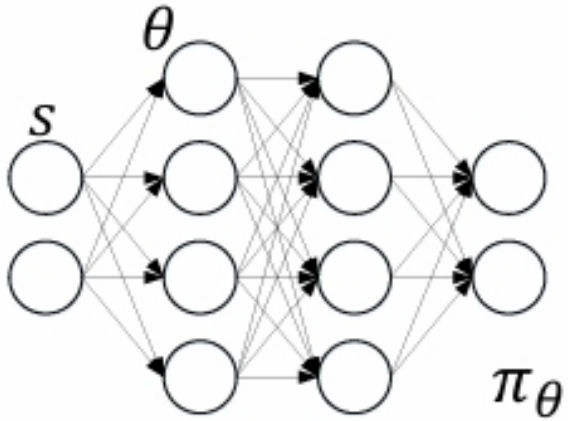
$$Q_\theta(s,a) \approx Q^\pi(s,a)$$

$$\pi_\theta(s) \overset{\text{def}}{=} \pi(s;\theta)$$

$$\pi_\theta(a|s) \overset{\text{def}}{=} \pi(a|s;\theta)$$



$a = \pi_\theta(s)$



$\pi_\theta(a|s) \sim N(\mu, \sigma^2)$

目标函数

$$\mathcal{L}(s,a,s';\theta)=\left(r+\gamma\max_{a'}\left(Q_\theta(s',a')-Q_\theta(s,a)\right)\right)^2$$

- 存在两个问题：
  - 目标不稳定，参数学习的目标依赖于参数本身；
  - 样本之间有很强的相关性。
- 深度Q网络
  - 一是目标网络冻结（freezing target networks），即在一个时间段内固定目标中的参数，来稳定学习目标；
  - 二是经验回放（experience replay），构建一个经验池来去除数据相关性。

算法 15.5: 带经验回放的深度 Q 网络

输入: 状态空间 $\mathcal{S}$，动作空间 $\mathcal{A}$，折扣率 γ，学习率 α

初始化经验池 $\mathcal{D}$，容量为 N;

随机初始化 Q 网络的参数 ϕ;

随机初始化目标 Q 网络的参数 $\hat{\phi} = \phi$;

repeat

  初始化起始状态 s;

  repeat

    在状态 s，选择动作 $a = \pi^{\epsilon}$;

    执行动作 a，观测环境，得到即时奖励 r 和新的状态 s';

    将 s, a, r, s' 放入 $\mathcal{D}$ 中;

    从 $\mathcal{D}$ 中采样 ss, aa, rr, ss';

$$y = \begin{cases} rr, & ss' \text{为终止状态}, \\ rr + \gamma \max_{a'} Q_{\hat{\phi}}(\mathbf{ss}', \mathbf{a}'), & \text{否则} \end{cases};$$

    以 $\left(y - Q_{\phi}(\mathbf{s}, \mathbf{a})\right)^2$ 为损失函数来训练 Q 网络;

    $s \leftarrow s'$;

    每隔 C 步，$\hat{\phi} \leftarrow \phi$;

  until s 为终止状态;

until $\forall s, a$，$Q_{\phi}(\mathbf{s}, \mathbf{a})$ 收敛;

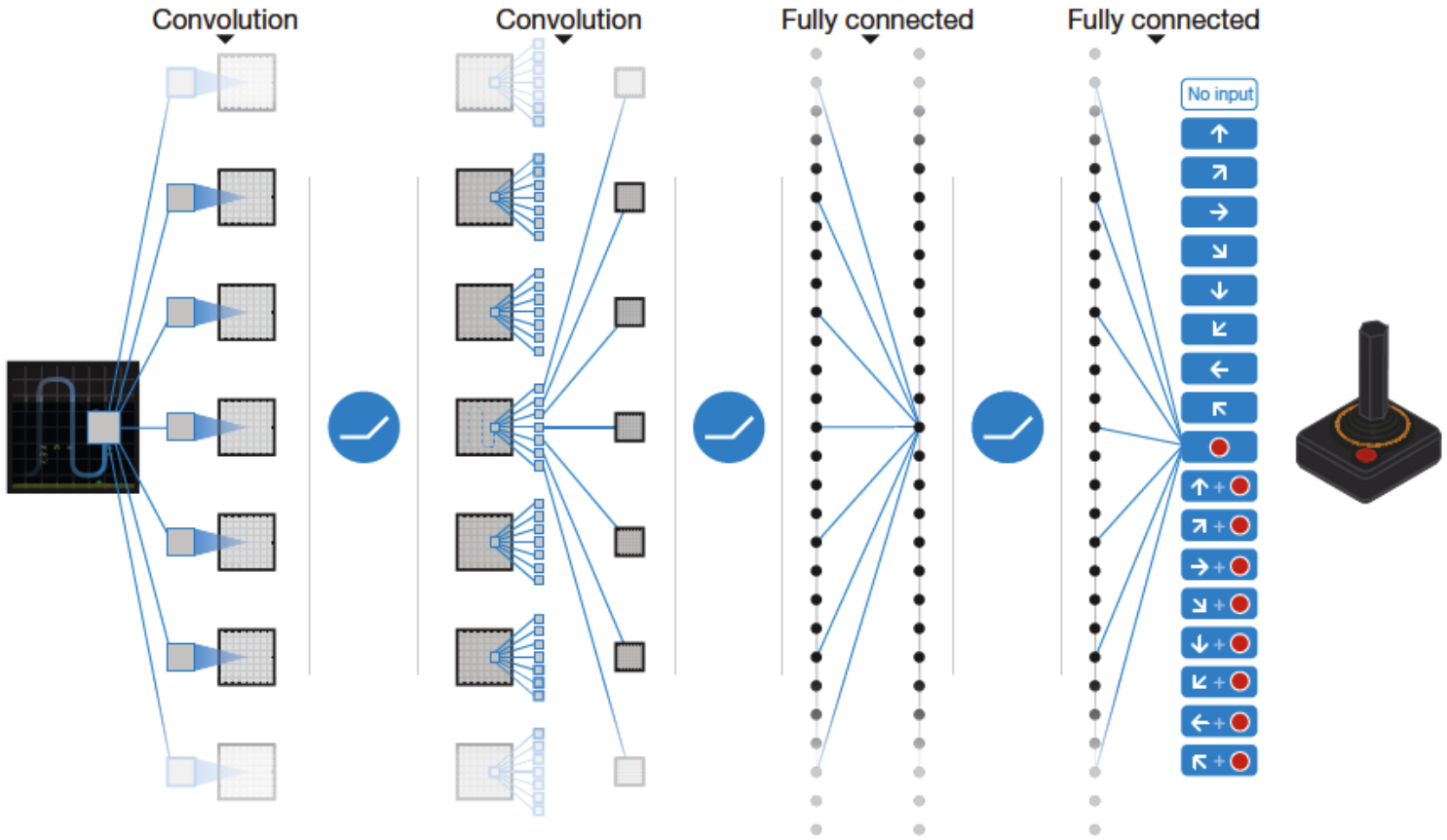
输出: Q 网络 $Q_{\phi}(\mathbf{s}, \mathbf{a})$

- 经验回放机制：算法使用经验池 DD 来存储智能体与环境交互的经验样本 (s, a, r, s')。这些样本被随机抽样，用于训练深度 Q 网络 Q_{ϕ}。这种方法减少了样本间的相关性，提升了训练的稳定性和效率。
- 目标网络与主网络的分离：算法引入了目标 Q 网络 $Q_{\hat{\phi}}$，用于计算目标值 y。目标网络的参数 $\hat{\phi}$ 定期从主网络的参数 ϕ 同步而来，但在训练过程中保持不变。这种分离有助于避免目标值频繁更新引起的不稳定。
- ε-贪婪策略：在每一步中，算法通过 ε-贪婪策略选择动作 a。具体来说，以概率 ϵ 随机选择动作（探索），以概率 $1-\epsilon$ 选择当前 Q 网络输出的最佳动作（利用）。这种策略平衡了探索新状态与利用当前策略的权衡。

目录
Contents

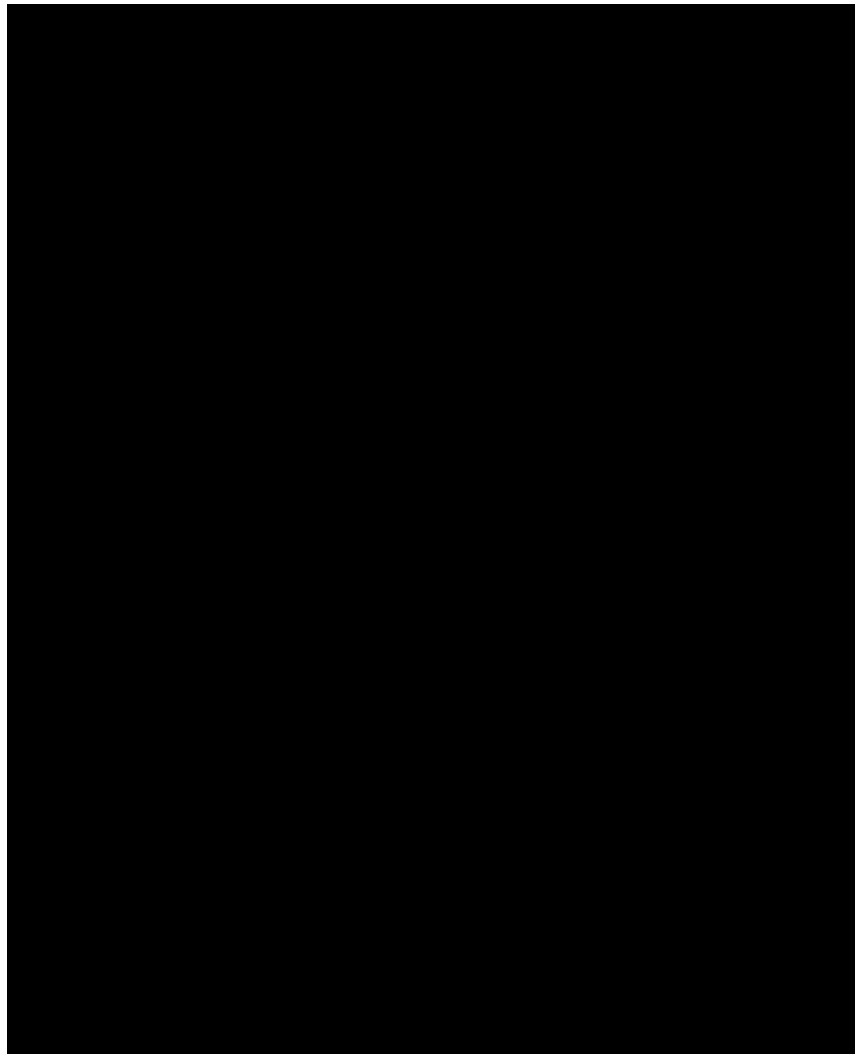
1. 强化学习基本要素
2. 策略评估
3. 基于模型的强化学习
4. 模型无关的强化学习
5. 深度强化学习
6. **深度强化学习应用**

DQN in Atari



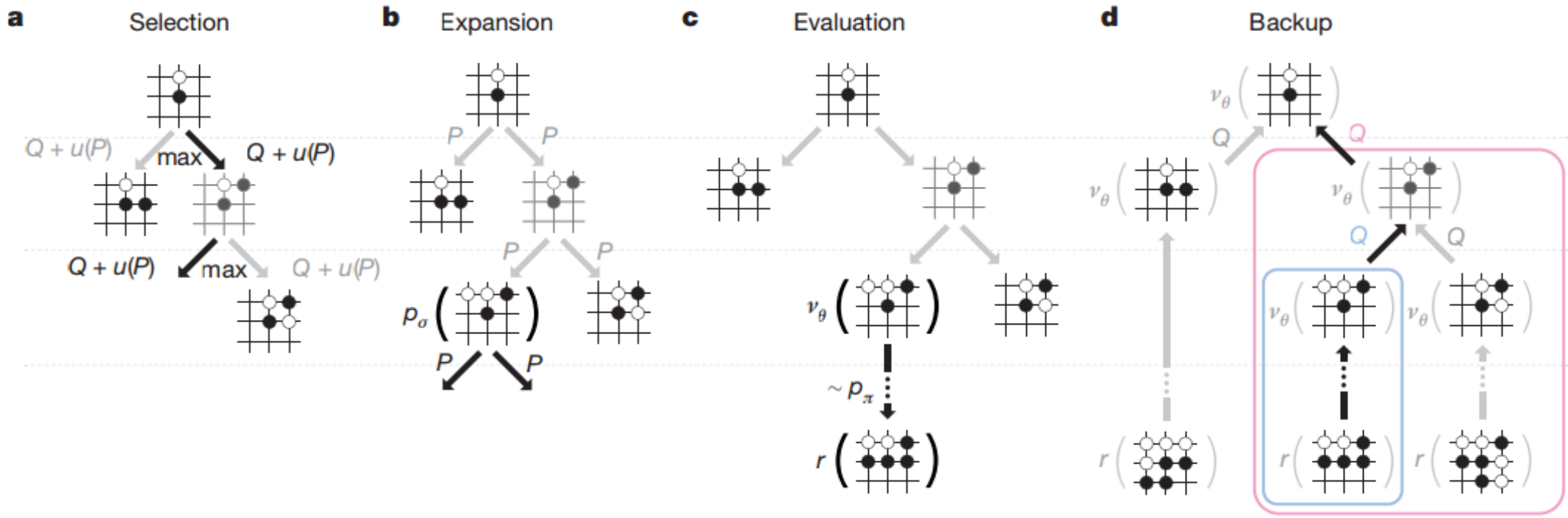
Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.

DQN in Atari



AlphaGO: 蒙特卡洛树搜索

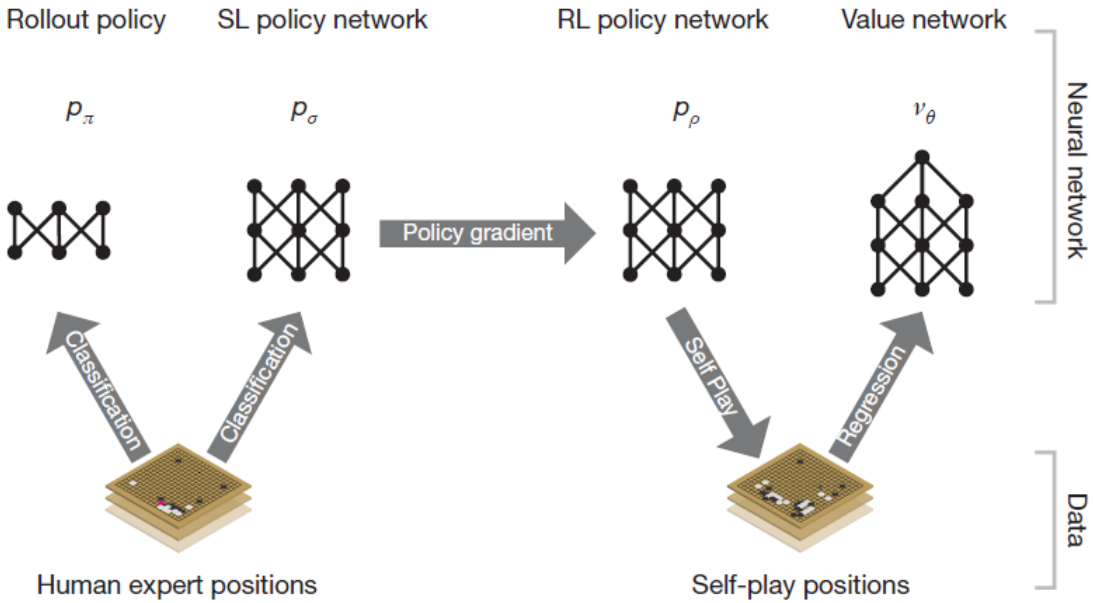
- MCTS：通过预测对手的动作进行模型前瞻以减少搜索空间



Silver, David, et al. "Mastering the game of Go with deep neural networks and tree search." Nature 529.7587 (2016): 484-489.

AlphaGo：学习流程

- 结合监督学习（SL）和强化学习（RL）来学习蒙特卡罗树搜索（MCTS）中的搜索方向
- 监督学习策略网络（SL Policy Network）：
  - 提供先验搜索概率或潜在价值
- Rollout（快速模拟）：
  - 与MCTS结合，在叶子节点上进行快速模拟
- 价值网络（Value Network）：
  - 构建对叶子节点局势的全局评估



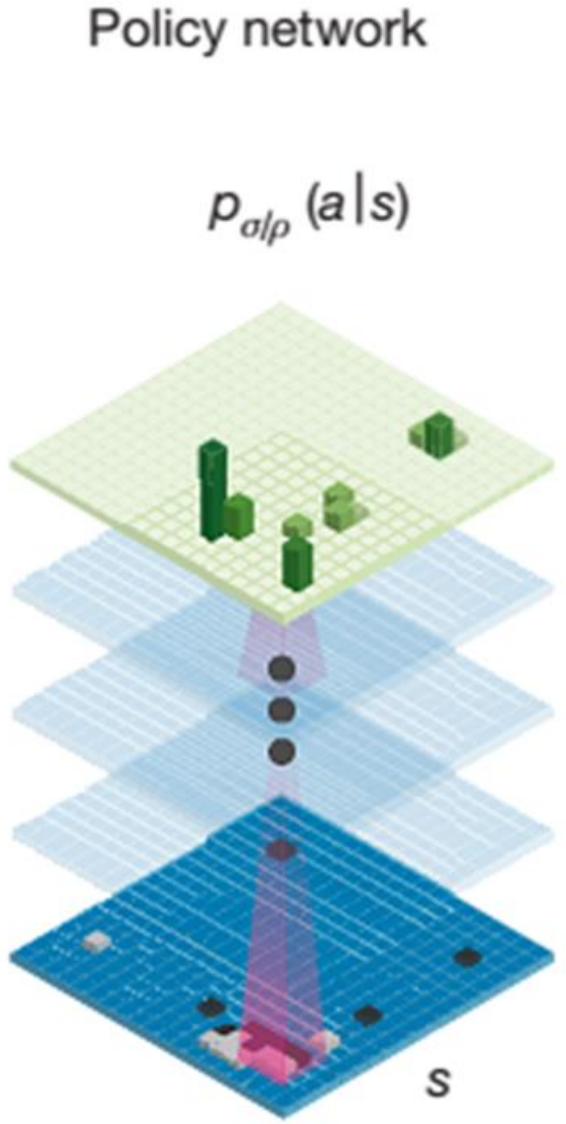
学习剪枝：监督学习策略网络（SL Policy Network）

- 3 层卷积神经网络（CNN）输入：
- 棋盘位置 s
- 输出： $p_\theta(a \mid s)$，其中 a 是下一步行动

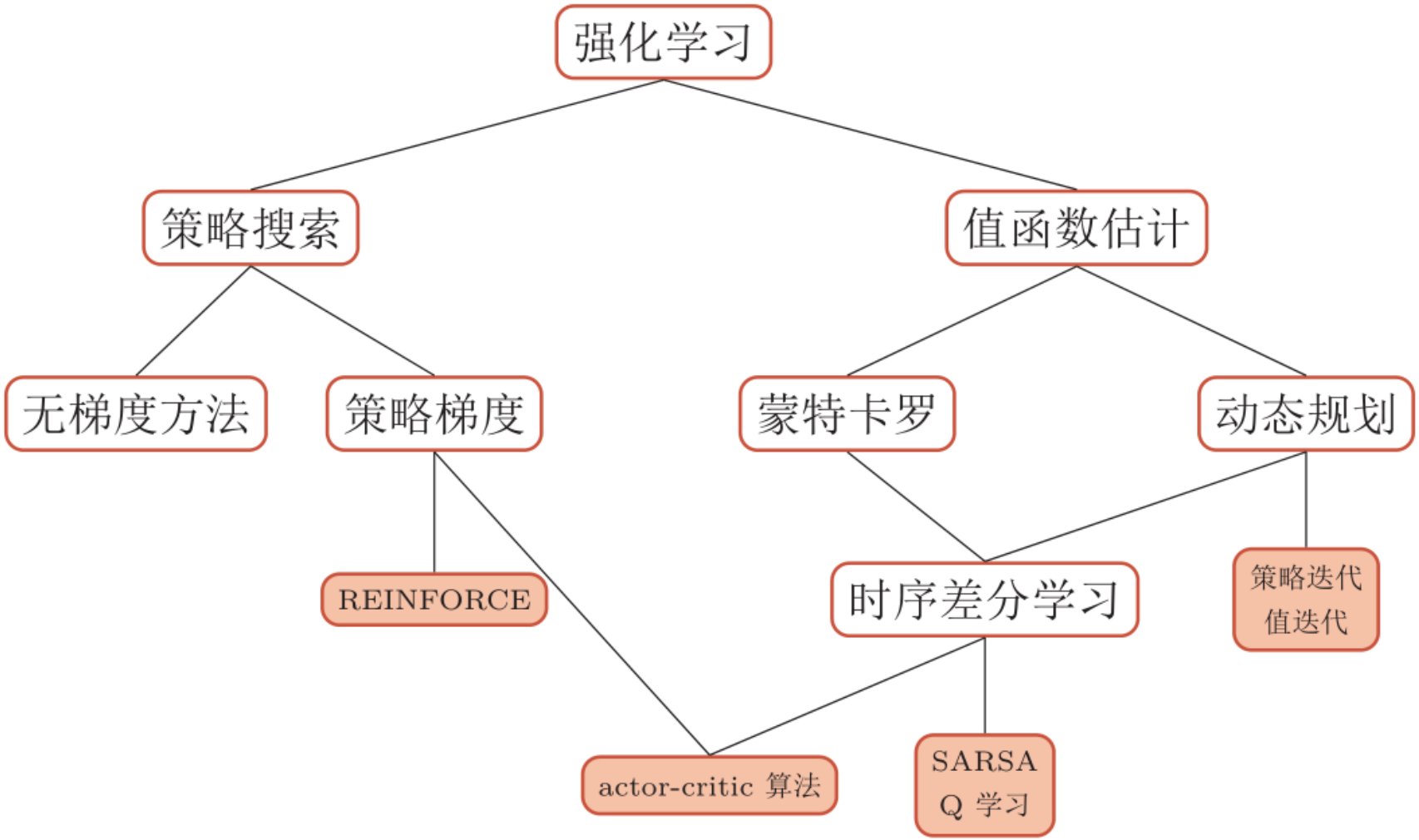
Extended Data Table 2 | Input features for neural networks

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

Feature planes used by the policy network (all but last feature) and value network (all features).



不同强化学习算法之间的关系



算法汇总

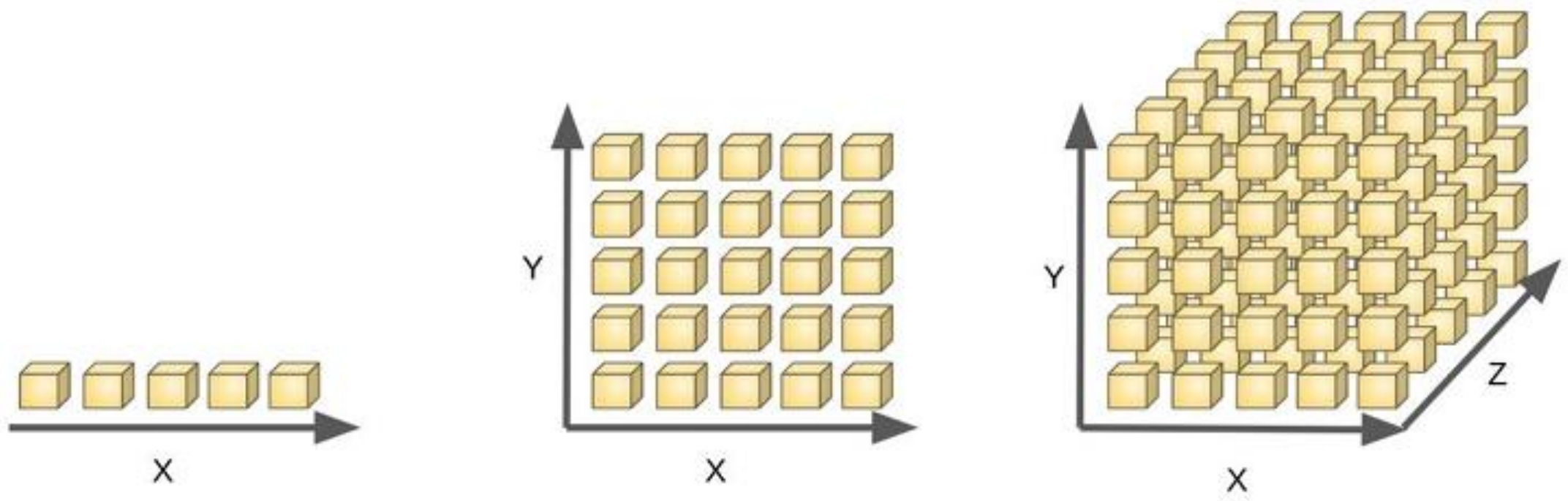
算法	步骤
SARSA	（1）执行策略，生成样本：s, a, r, s', a' （2）估计回报：$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma Q(s',a') - Q(s,a)\Big)$ （3）更新策略：$\pi(s) = \arg\max_{a \in \lvert\mathcal{A}\rvert} Q(s,a)$
Q学习	（1）执行策略，生成样本：s, a, r, s' （2）估计回报：$Q(s,a) \leftarrow Q(s,a) + \alpha\Big(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\Big)$ （3）更新策略：$\pi(s) = \arg\max_{a \in \lvert\mathcal{A}\rvert} Q(s,a)$
REINFORCE	（1）执行策略，生成样本：$\tau = s_0, a_0, s_1, a_1, \cdots$ （2）估计回报：$G(\tau) = \sum_{t=0}^{T-1} r_{t+1}$ （3）更新策略：$\theta \leftarrow \theta + \sum_{t=0}^{T-1} \Big(\frac{\partial}{\partial\theta} \log \pi_\theta(a_t \lvert s_t) \gamma^t G(\tau_{t:T})\Big)$
Actor-Critic	（1）执行策略，生成样本：s, a, s', r （2）估计回报：$G(s) = r + \gamma V_\phi(s')$ $\phi \leftarrow \phi + \beta\Big(G(s) - V_\phi(s)\Big)\frac{\partial}{\partial\phi} V_\phi(s)$ （3）更新策略：$\lambda \leftarrow \gamma\lambda$ $\theta \leftarrow \theta + \alpha\lambda\Big(G(s) - V_\phi(s)\Big)\frac{\partial}{\partial\theta} \log \pi_\theta(a \lvert s)$

强化学习的挑战

- 强化学习相较于规划和模型预测控制的优势：
  - 大规模：支持超过100个状态/动作。
  - 复杂性：适用于非线性、混合整数、非平滑问题。
  - 随机性：能处理人类行为或股票市场等随机性场景。
- 当今强化学习的挑战
  - 维度灾难
  - 探索与利用的困境
  - 部分可观测性
  - 时间延迟的奖励
  - 非平稳环境

维度灾难

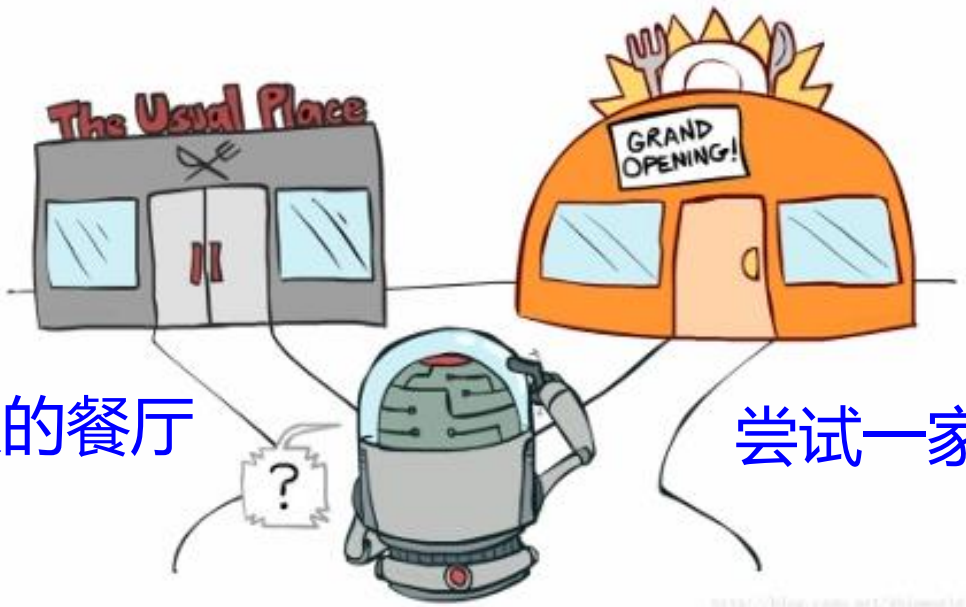
- 由R. Bellman在动态规划（1957年）中提出：
  - 计算需求随着状态变量数量呈指数增长。



- 一般来说，具有n个维度且每个维度允许m种状态时，将会有m^n种可能的组合。

探索与利用的困境

- 探索
  - 挖掘更多关于环境的信息
- 利用
  - 利用已知信息以最大化奖励



去你最喜欢的餐厅　　尝试一家新餐厅

部分可观测性

- 完全可观测性 vs 部分可观测性
  - 环境的整个状态对于外部观察者并不是完全可见的。
  - 完全可观测性具有一些优势，如理论上的完整性和易于实现。

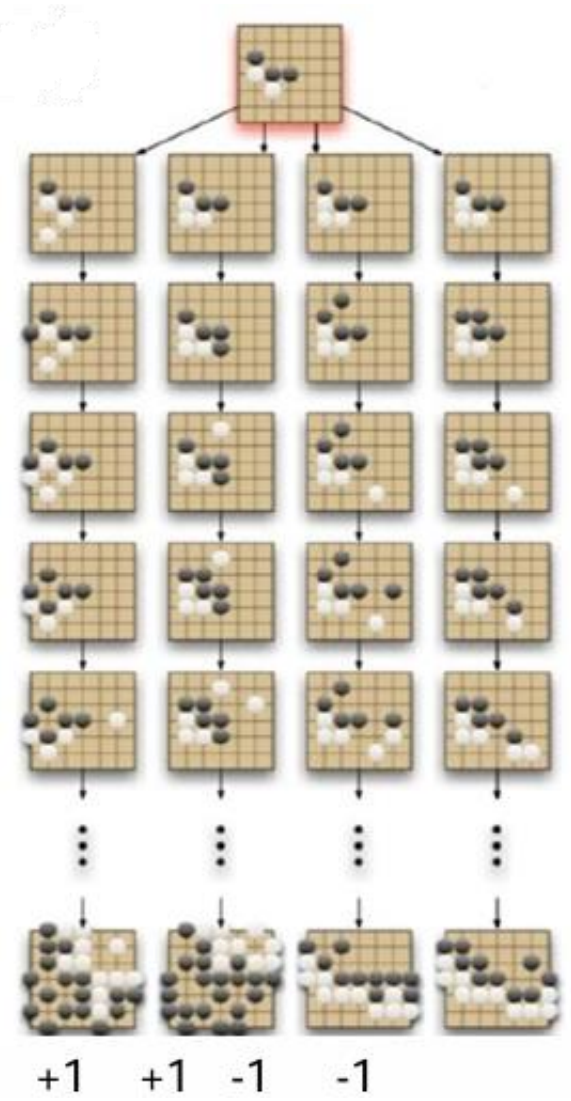
全可观测

部分可观测

时间延迟的奖励

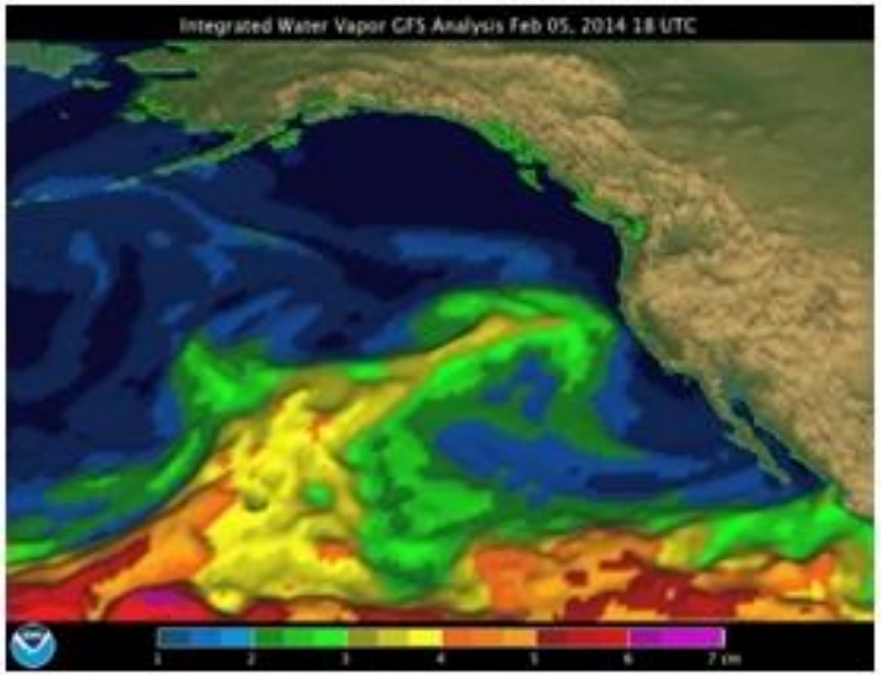
- 延迟奖励导致不良的收敛性
  - 例如，在最后才知道输赢
  - 累积的奖励信号
  - 延迟奖励对时间上远离的先前状态的影响非常弱
  - 没有足够的信息来支持一个好的策略

AlphaGo zero
(1: win -1: lose)



非平稳环境

- 环境可能非常不确定
- 如果环境变化太快，强化学习可能会失败
- 设计不当的强化学习可能在变化缓慢的环境中失败



强化学习展新篇，智能交互探自然。
策略评估深度算，奖励回馈引智前。