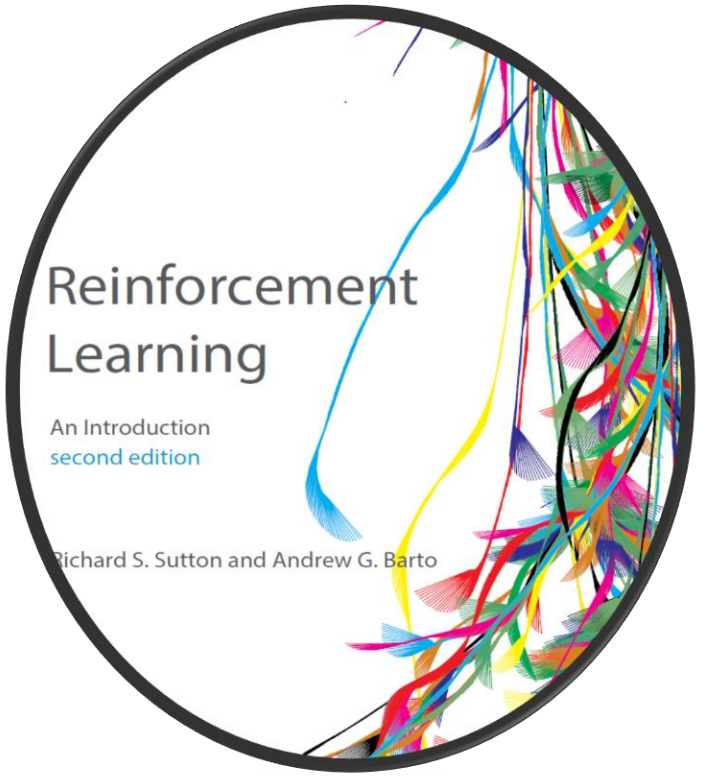
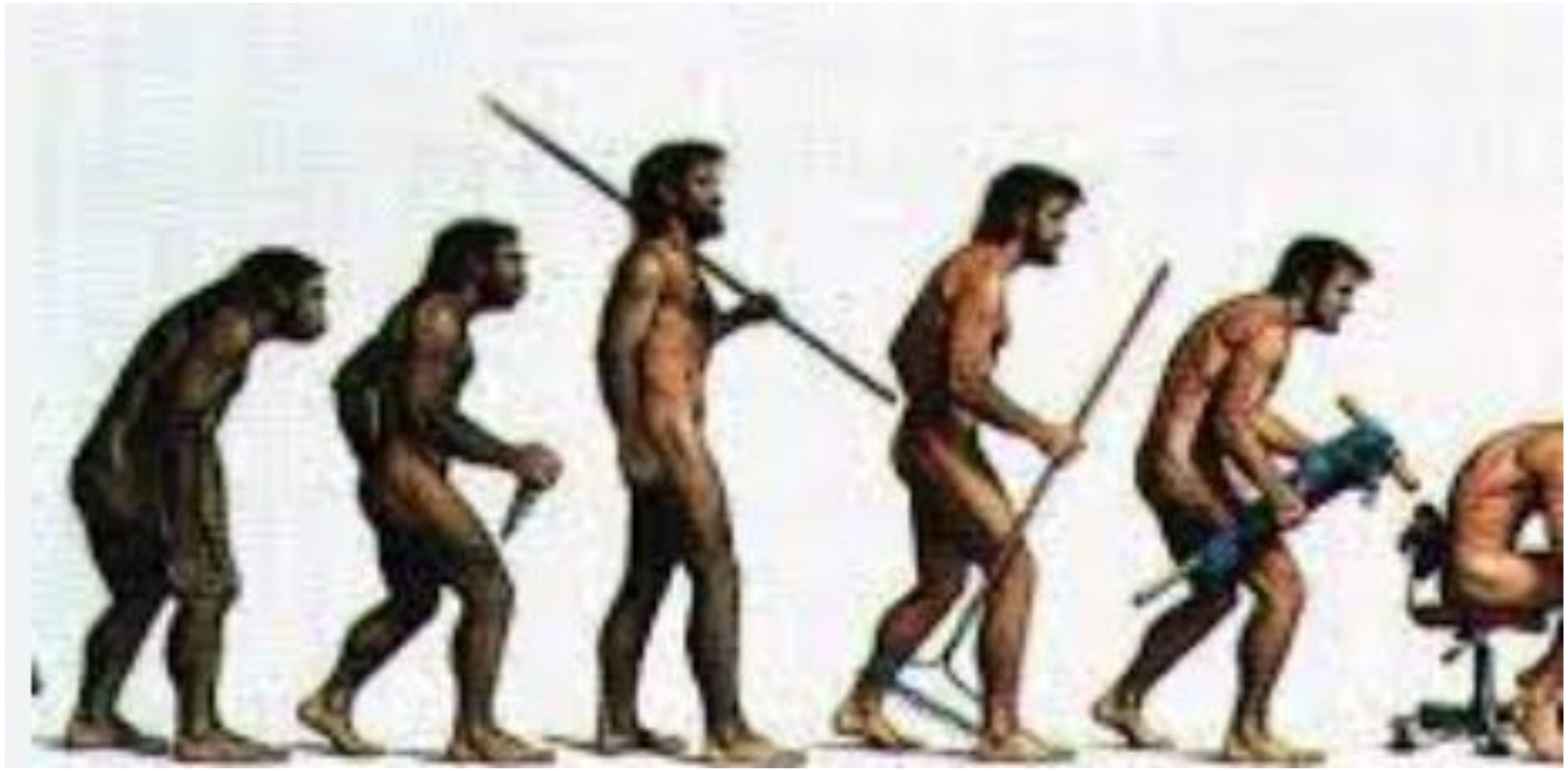


动态规划（Dynamic Programming）

何新卫

华中农业大学信息学院



单选题 50分

设置

在强化学习中，什么是策略迭代?

A 交替进行策略评估和策略改进

B 直接更新值函数以获得最优策略

C 一次性完成策略评估和改进

D 根据固定策略选择动作

提交

上讲回顾

- **环境：有限MDP**
- **价值函数**用来组织和结构化搜索最优策略
- 一旦计算出最优价值函数($v_*(s)$或者$q_*(s,a)$)，就很容易获得最优策略
- **只有**最优价值函数满足**贝尔曼最优方程**（Bellman optimality equations）

$$v_*(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a]$$
$$= \max_a \sum_{s',r} p(s', r | s, a)\,[r + \gamma v_*(s')]$$

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \middle| S_t = s, A_t = a\right]$$
$$= \sum_{s',r} p(s', r | s, a)\,[r + \gamma \max_{a'} q_*(s', a')]$$

for all $s \in \mathcal{S}$, $r \in \mathcal{R}$, $a \in \mathcal{A}(s)$, $s' \in \mathcal{S}^+$

策略评估（Policy Evaluation）

- 策略评估：给定任意策略π，如何计算π对应的状态价值函数$v_\pi(\cdot)$
- 回顾第三章，对于所有$s \in \mathcal{S}$:

$$
\begin{aligned}
v_\pi(s) &\triangleq \mathbb{E}_\pi\left[G_t | S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} | S_t = s\right] \\
&= \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right] \\
&= \sum_a \pi(a|s) \sum_{s',r} p(s', r|\, s, a)[r + \gamma v_\pi(s')]
\end{aligned}
$$

- 其中，$\pi(a|s)$代表状态为s，遵循策略π执行动作a 的概率
- 上述$v_\pi(s)$表示为期望，其存在性和唯一性前提条件是：要么$\gamma < 1$

  或者策略π对于所有状态，都将回到终止态。

策略评估（Policy Evaluation）：解析法求$v_\pi(s)$

- 前提条件：环境动态（$p(s', r|\, s, a)$）完全已知
- 解析法：利用上述方程可以构建$|\mathcal{S}|$个线性方程组，$|\mathcal{S}|$个未知量（即$v_\pi(s)$，$s \in \mathcal{S}$），则很容易求解出来每个状态的价值函数。

➢ 存在问题：状态空间大小$|\mathcal{S}|$较大，运算量比较大，且复杂费时

策略评估：迭代策略评估求 $v_\pi(s)$

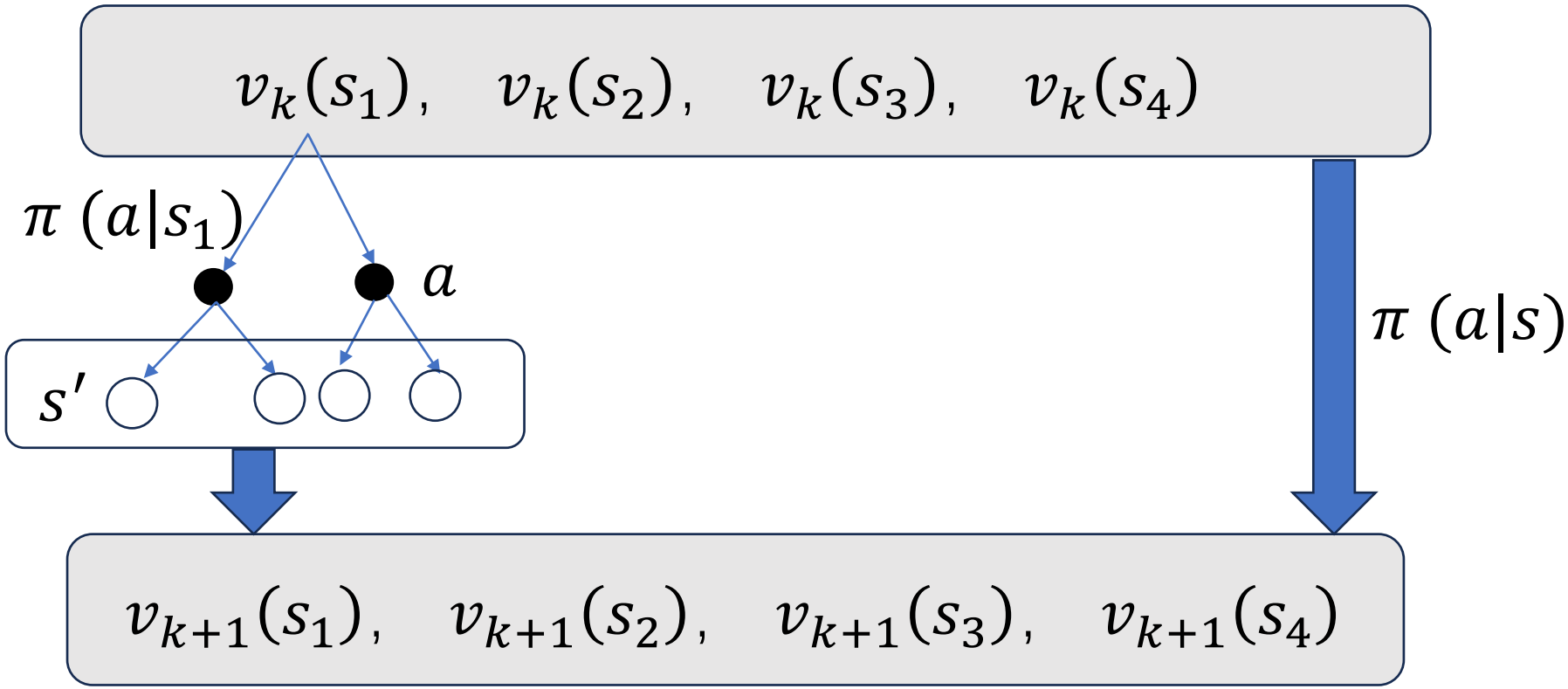
- 前提条件：环境动态（ $p(s', r|\, s, a)$ ）完全已知
- 迭代策略评估：给定近似价值函数序列$v_0(\cdot)$, $v_1(\cdot)$, $v_2(\cdot)$, ..., v_k: $\boldsymbol{S}^+ \to \mathbb{R}$
- 初始近似函数$v_0(\cdot)$，可以随意初始化（注意：终止状态必须初始化为0）。
- 基于贝尔曼方程, 对于连续相邻两次逼近函数的关系如下:

$$v_{k+1}(s) \triangleq \mathbb{E}_\pi\left[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s\right]$$

$$= \sum_a \pi(a|s) \sum_{s',r} p(s', r|\, s, a)[r + \gamma v_k(s')] \text{ for all } s \in \boldsymbol{S}$$

- 容易知道，当$v_k(s) = v_\pi(s)$, for all $s \in \boldsymbol{S}$ 是上述更新规则的不动点。
- $k \to \infty$, $v_k(s)$收敛于$v_\pi(s)$， for all $s \in \boldsymbol{S}$
- 上述算法被称为迭代策略评估

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r| s, a)[r + \gamma v_k(s')] \text{ for all } s \in \mathcal{S}$$



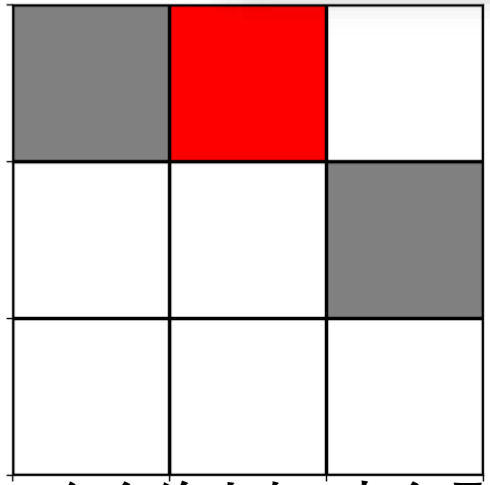
- 期望更新: 当前给定状态s，利用所有下一个可能状态s'的期望价值来更新下次迭代该状态s的价值，从而获得更加准确的价值估计
- **两个数组**，一个存储策略新的近似价值估值，一个存储策略旧的近似价值估值，新的价值近似值均来自旧版本的期望平均

进一步改进：迭代策略评估原位（inplace）更新

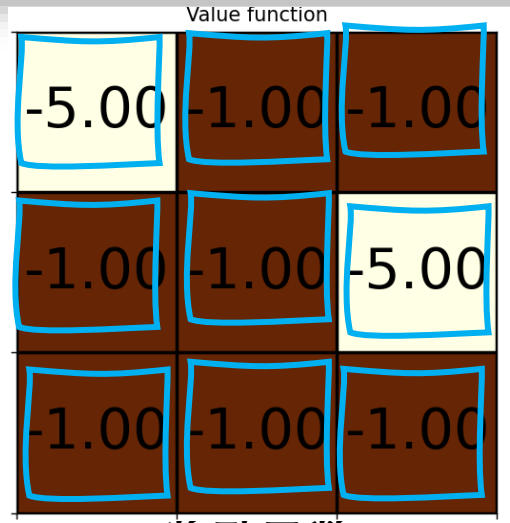
- 单数组原位（inplace）更新
  - ✓ 优点：收敛更快，因为它总能够时刻用到最新数据更新价值近似值
  - ✓ 实践中，我们常使用原位更新算法
- 对状态空间所有状态更新一次，我们称为一次遍历（**sweep**）
- 注意：为了避免迭代策略评估一直迭代，可以加入迭代终止条件。
  - ✓ 如连续两次sweep价值近似值没有太多变化，即可终止继续迭代：

$$\max_{s \in \boldsymbol{S}} |v_{k+1}(s) - v_k(s)| < \theta$$

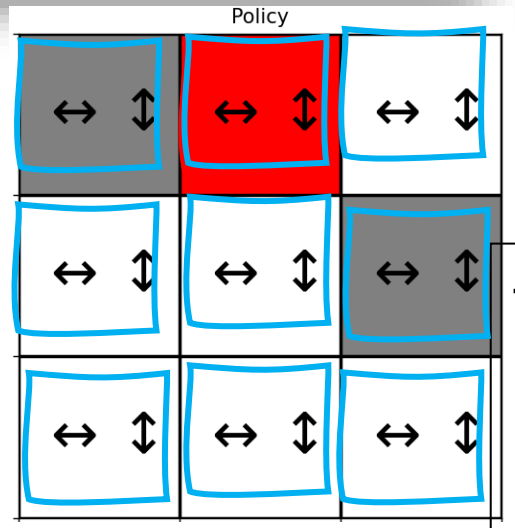
$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|\, s, a)[r + \gamma v_k(s')] \text{ for all } s \in \mathcal{S}$$



环境，红色终止态，灰色是障碍单元



奖励函数

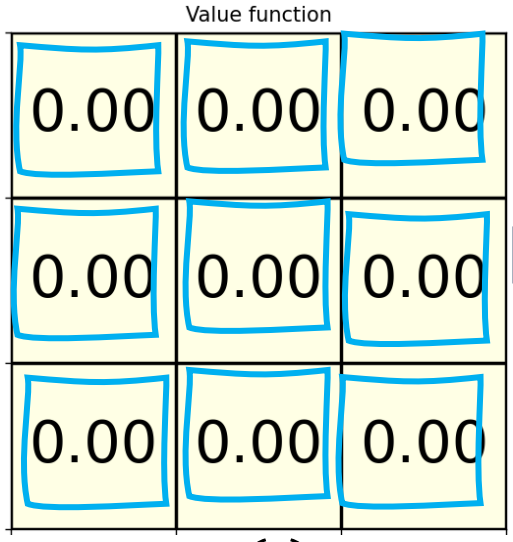


策略 π

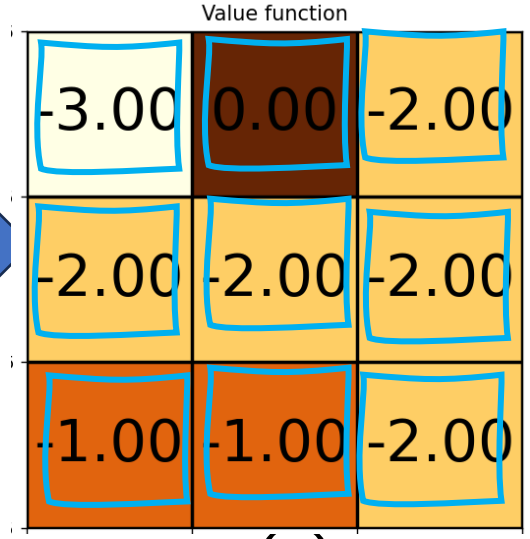
设终止态价值为0

$\gamma = 0.9$

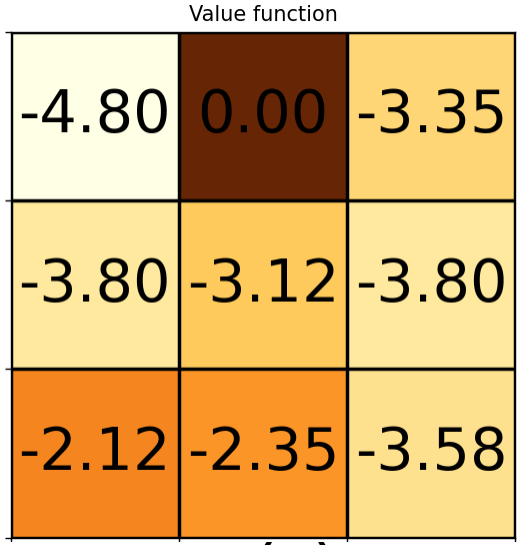
$v_\pi(\cdot) = ?$



$v_0(\cdot)$



$v_1(\cdot)$



$v_2(\cdot)$

策略评估（Policy Evaluation），直至符合收敛条件，退出策略评估

迭代策略评估伪代码

Iterative Policy Evaluation, for estimating $V \approx v_\pi$

Input π, the policy to be evaluated
Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

所有状态扫描一遍，更新价值

单选题 50分

设置

策略评估是什么？

A 寻找最优动作

B 评估当前策略在各状态下的长期回报

C 改进当前策略

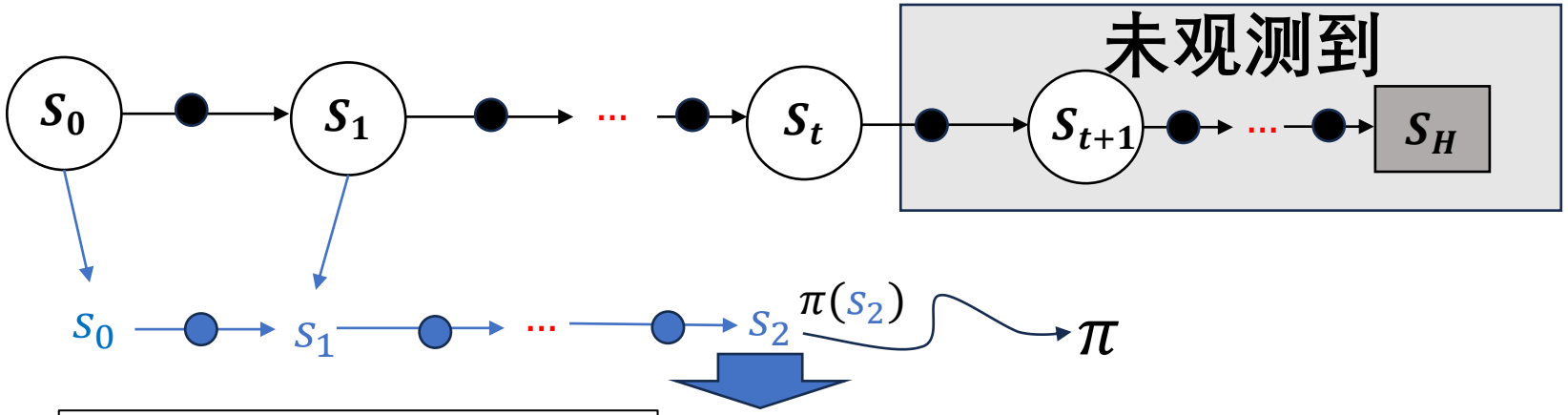
提交

策略改进（Policy Improvement）

- 计算策略对应的状态价值函数的目的是为了帮助进行策略改进
- 假设给定一个确定性策略π, 我们已经计算其价值$v_\pi(\cdot)$。
  - 对于某一状态s, 通过选择一个新动作$a(\neq \pi(s))$, 是否演进出更好的策略
  - 如果遵循当前策略π，选择动作$\pi(s)$能够拿到的价值是$v_\pi(s)$，那么可否考虑选择一个其他动作$a(\neq \pi(s))$，之后在后续状态继续遵循老策略π，看看价值是否进一步提升。
- 上述过程可以表达为如下公式：

$$q_\pi(s,a) \triangleq \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = a]$$
$$= \sum_{s',r} p(s', r | s, a)[r + \gamma v_\pi(s')]$$

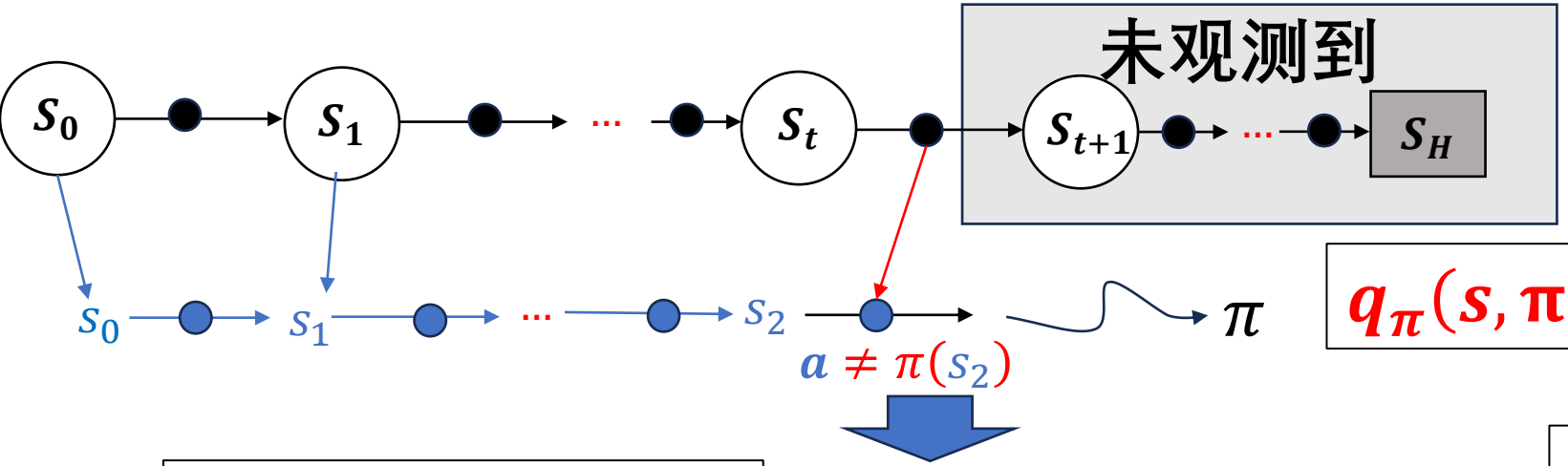
策略提升定理（Policy Improvement Theorem）



价值(或期望汇报): $v_\pi(s_2)$

旧确定性策略π

新确定性策略π'



价值(或期望汇报): $q_\pi(s, \pi'(s))$

$$q_\pi(s, \pi'(s)) \geq v_\pi(s), \text{for all } s \in S$$

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

在s_2时选择不同的动作，其实生成了一个不同的策略π'

定理证明

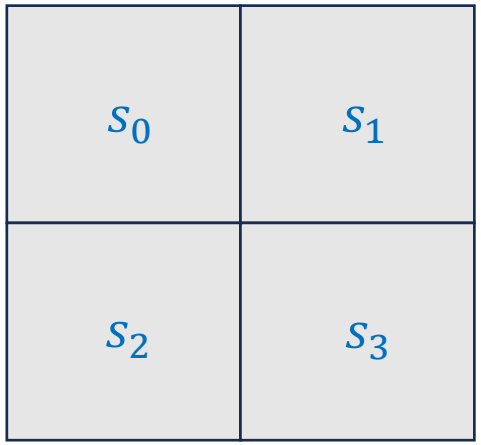
$q_\pi(s,\pi'(s)) \geq v_\pi(s), \text{for all } s \in \mathcal{S}$ ⇒ $v_{\pi'}(s) \geq v_\pi(s)$

证明：对于具体状态s，我们选择执行更优动作$\pi'(s)$，其它仍然执行$\pi(\cdot)$

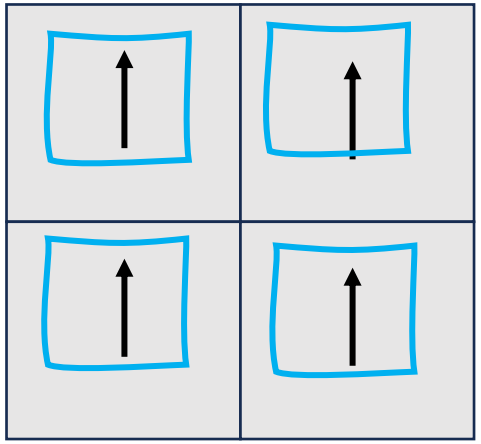
$$
\begin{aligned}
v_\pi(s) &\leq q_\pi(s,\pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma \mathbb{E}[R_{t+2} + \gamma v_\pi(S_{t+2}) | S_{t+1}, A_{t+1} = \pi'(S_{t+1})] | S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) | S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) + \gamma^3 v_\pi(S_{t+3}) | S_t = s] \\
&\cdots \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots | S_t = s] \\
&= v_{\pi'}(s)
\end{aligned}
$$

- 事实上，上述可以对所有状态进行提升

策略提升定理（Policy Improvement Theorem）

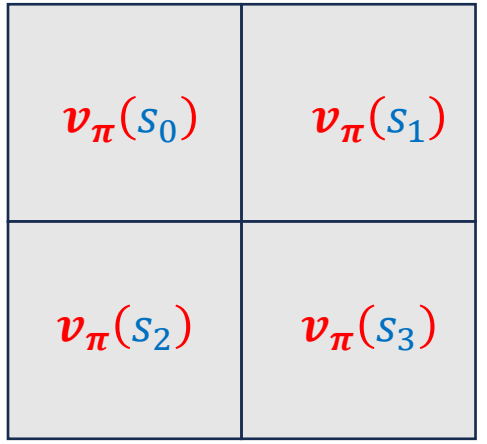


MDP环境 $< p, \mathcal{S}, \mathcal{R}, \mathcal{A}, \gamma >$



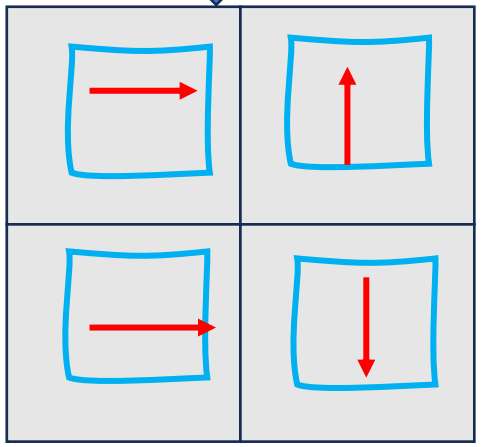
一个确定性策略π

策略
评估



策略价值$v_\pi(\cdot)$

策略
提升



提升后确定性策略π'

$$q_\pi(s_0, \pi'(s_0)) \geq v_\pi(s_0)$$

$$q_\pi(s_1, \pi'(s_1)) \geq v_\pi(s_1)$$

$$q_\pi(s_2, \pi'(s_2)) \geq v_\pi(s_2)$$

$$q_\pi(s_3, \pi'(s_3)) \geq v_\pi(s_3)$$

策略提升定理（Policy Improvement Theorem）

- 基于上述定理，我们可一直贪心地在每个状态选择动作价值最大的动作：

$$\pi'(s) = \arg\max_a q_\pi(s,a)$$

$$= \arg\max_a \sum_{s',r} p(s',r|\, s,a)[r + \gamma v_\pi(s')]$$

$$= \arg\max_a \; r(s,a) + \gamma \sum_{s'} p(s'|\, s,a) v_\pi(s')$$

- 上述构造的贪心策略满足策略提升定理的条件，所以π'能比π更好或者至少一样好。
- 上述公式的反映到价值函数，形式如下：

$$v_{\pi'}(s) = \max_a \; r(s,a) + \gamma \sum_{s'} p(s'|\, s,a) v_\pi(s')$$

最优策略的保证： 收敛

- 假如从旧策略π, 通过贪心方法获得新策略π'， 但是这个新策略和老策略π一样好，并没有更好。即$v_\pi = v_{\pi'}$, 则for all $s \in \mathcal{S}$：

$$v_{\pi'}(s) = \max_a \mathbb{E}[R_{t+1} + \gamma v_{\pi'}(S_{t+1}) | S_t = s, A_t = a]$$
$$= \max_a \sum_{s',r} p(s', r | s, a) [r + \gamma v_{\pi'}(s')]$$

- 上述公式和贝尔曼最优方程相同，那么$v_{\pi'}$ 必然是v_*, 即π 和π' 均为最优策略。

- 策略改进提供了一个严格更好的策略，除非老策略已经是最优的了

。

策略迭代（Policy Iteration）

- 一旦将策略π通过v_π改进为一个更好的策略π'，我们就可以计算$v_{\pi'}$并再次改进它，以获得一个更好的策略π''。因此，我们可以得到一系列递增的策略和值函数。

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \cdots \xrightarrow{I} \pi_* \xrightarrow{E} v_{\pi_*}$$

- 其中$\xrightarrow{E}$代表策略评估，$\xrightarrow{I}$代表策略改进$v_{\pi_{k+1}}(s) \geq v_{\pi_k}(s)$, for all $s \in \mathcal{S}$
- 根据策略提升理论，更新后的策略的价值函数满足单调性，因此，只要策略个数有限，则策略迭代就能收敛到最优策略
- 有限MDP仅包含有限个策略数目($|\mathcal{A}|^{|\mathcal{S}|}$)，上述过程通过有限次的迭代**必然收敛到最优策略和最优价值函数**。
- 上述寻找最优策略的方法叫做**策略迭代**

策略迭代（Policy Iteration）伪代码

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
      $\Delta \leftarrow 0$
      Loop for each $s \in \mathcal{S}$:
         $v \leftarrow V(s)$
         $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s))\left[r + \gamma V(s')\right]$
         $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
      old-action $\leftarrow \pi(s)$
      $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$
      If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

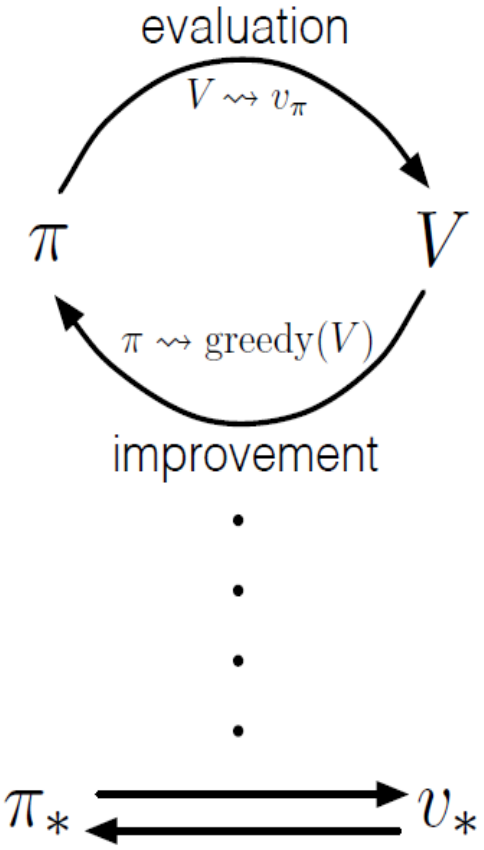
完成一次扫描

直至收敛

从所有动作里面，挑选回报最大的动作

策略提升

策略迭代终止条件



策略迭代（Policy Iteration）

- 缺点：每次迭代都涉及到策略评估，这本身可能是一个长时间的迭代计算，需要**多次循环遍历状态集合**。
- 如果策略评估是通过迭代进行的，那么只有在极限情况下才会完全收敛到v_π。
- 我们必须等待完全收敛？还是可以提前停止呢？

价值迭代（Value Iteration）

- 实际上，策略迭代的策略评估步骤可以通过多种方式截断，而不会丧失策略迭代的收敛保证。
- 一种重要的特殊情况是仅进行一次扫描（对每个状态进行一次更新）就停止后续策略评估。这个算法被称为价值迭代。
- 它可以被写成一个特别简单的结合了策略改进和截断的策略评估步骤的更新操作。

$$v_{k+1}(s) \triangleq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s, A_t = a]$$
$$= \max_a \sum_{s',r} p(s', r | s, a)[r + \gamma v_k(s')] \text{ for all } s \in \mathcal{S}$$

- 对于任意v_0，只要保证v_*存在的条件满足，则序列$[v_k]$将收敛到v_*

价值迭代伪代码

Value Iteration, for estimating $\pi \approx \pi_*$

Algorithm parameter: a small threshold $\theta > 0$ determining accuracy of estimation
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop:
  $\Delta \leftarrow 0$
  Loop for each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$

Output a deterministic policy, $\pi \approx \pi_*$, such that
  $\pi(s) = \text{argmax}_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

- 价值迭代有效地将在每次迭代中的一次策略评估和一次策略改进合并起来。
- 收敛速度更快

总结

1) 策略迭代（Value Iteration）

2) 价值迭代（Value Iteration）