南京大学
人工智能学院
School of Artificial Intelligence, Nanjing University

第四部分：完全可观察环境中的概率规划系统

章宗长

2020年4月28日

内容安排

- 规划
- **马尔科夫决策过程**
- 精确动态规划
- 近似动态规划
- 在线规划
- 直接策略搜索

马尔科夫决策过程

- 定义
- 例子
- 策略和值函数
- **最优策略和最优值函数**

最优策略

- 策略的偏序关系：对于两个策略π和π'，如果对任意$s \in \mathcal{S}$，都有$U^{\pi}(s) \leq U^{\pi'}(s)$，则称策略$\pi$小于等于$\pi'$，记作$\pi \leq \pi'$
- 最优策略π^*：对于一个动力函数而言，总是存在着一个策略π^*，使得所有的策略都小于等于这个策略
- 最优策略$\pi^*(a \mid s)$满足：

$$\pi^*(a \mid s) = \begin{cases} 1, & a \in \arg\max_{a' \in \mathcal{A}} Q^*(s, a') \\ 0, & \text{其他} \end{cases}$$

- 对于一个动力函数而言，可能存在多个最优策略
- 最优的确定性策略
  - 对所有$s \in \mathcal{S}$，$\pi^*(s) \in \arg\max_{a' \in \mathcal{A}} Q^*(s, a')$
  - 如果有多个行动a使得$Q^*(s, a)$取最大值，则任选一个行动即可

最优状态值函数

- 最优状态值函数$U^*(s)$：从状态s起，执行最优策略π^*的期望回报

$$U^*(s) \doteq \max_{\pi} U^{\pi}(s), \quad \text{for all } s \in \mathcal{S}$$

- Bellman最优方程

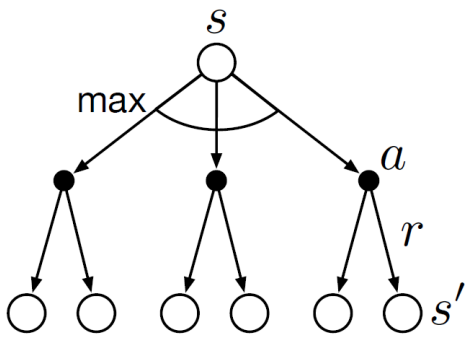
$$\begin{aligned} U^*(s) &= \max_{a \in \mathcal{A}(s)} Q^*(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}_{\pi_*}[R_t + \gamma G_{t+1} \mid S_t = s, A_t = a] \\ &= \max_a \mathbb{E}[R_t + \gamma\, U^*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma\, U^*(s')\right] \end{aligned}$$

$$R(s,a) + \gamma \sum_{s'} T(s' \mid s, a) U^*(s')$$

最优状态值函数的备份图

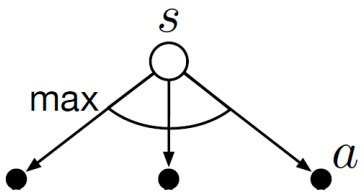
$$U^*(s) = \max_a \sum_{s',r} p(s',r|s,a)\left[r + \gamma\, U^*(s')\right]$$

$Q^*(s,a)$

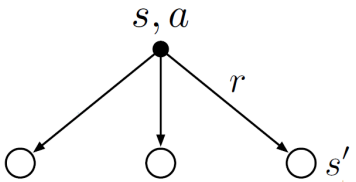


最优状态值函数$U^*(s)$的备份图

$$U^*(s) = \max_{a\in\mathcal{A}(s)} Q^*(s,a)$$



$$Q^*(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma\, U^*(s')\right]$$



最优行动值函数

- 最优行动值函数$Q^*(s,a)$：在状态s采取行动a后，执行最优策略π^*的期望回报

$$Q^*(s,a) \doteq \max_{\pi} Q^{\pi}(s,a), \quad \text{for all } s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s)$$

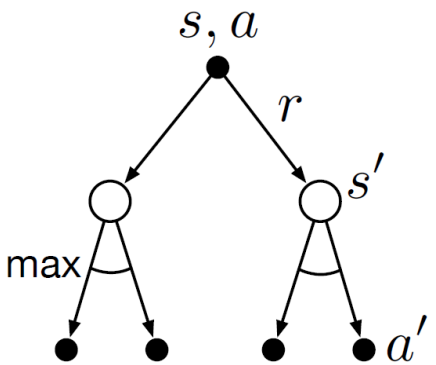
- Bellman最优方程

$$\begin{aligned} Q^*(s,a) &= \mathbb{E}[R_t + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \mathbb{E}\left[R_t + \gamma \max_{a'} Q^*(S_{t+1}, a') \,\middle|\, S_t = s, A_t = a\right] \\ &= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} Q^*(s', a')\right] \end{aligned}$$

最优行动值函数的备份图

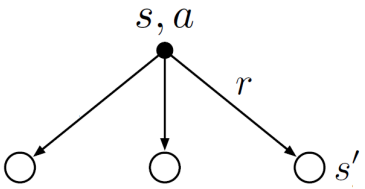
$$Q^*(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma \max_{a'} Q^*(s',a')\right]$$

$$U^*(s)$$

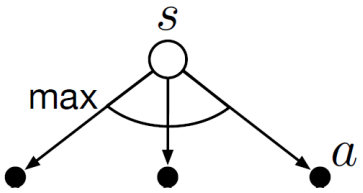


最优行动值函数$Q^*(s,a)$的备份图

$$Q^*(s,a) = \sum_{s',r} p(s',r|s,a)\left[r + \gamma\, U^*(s')\right]$$

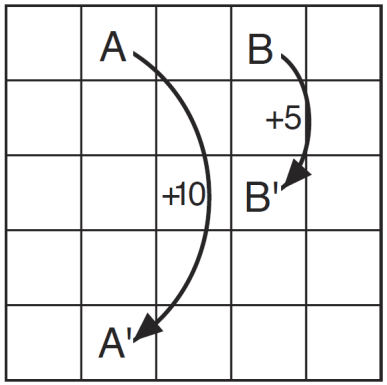


$$U^*(s) = \max_{a \in \mathcal{A}(s)} Q^*(s,a)$$



例子：5 × 5栅格世界

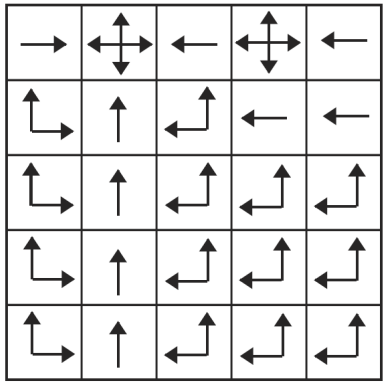
- 最优状态值函数U^*和最优策略π^*



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

U^*



π^*

- 如何计算状态A处的最优值？
  - 最优策略：在状态A执行任意行动到达状态A′，连续执行4次向上的行动回到状态A，如此循环
  - $U^*(s_{\mathrm{A}}) = 10 + 10 \times 0.9^5 + 10 \times 0.9^{5\times 2} + \cdots = \frac{10}{1-0.9^5} \approx 24.4$

小结：马尔科夫决策过程

- 定义
  - 马尔科夫假设、动力函数、稳态MDPs、决策网络表示
- 例子
  - 吸尘机器人、栅格世界、2048、小车上山、飞机避碰
- 策略和值函数
  - 随机性策略、确定性策略
  - 状态值函数、行动值函数、Bellman期望方程、备份图
- 最优策略和最优值函数
  - 最优策略、最优的确定性策略
  - 最优状态值函数、最优行动值函数、Bellman最优方程、备份图

内容安排

- 规划
- 马尔科夫决策过程
- **精确动态规划**
- 近似动态规划
- 在线规划
- 直接策略搜索

精确动态规划

- 策略迭代
- 值迭代
- 结构化表示
- 线性表示

动态规划与Bellman方程

- **动态规划的要素**
  - 最优子结构：把原问题分解成多个子问题
  - 重叠子问题：重复利用子问题的解

- Bellman期望方程 $$U^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s' \mid s, \pi(s)) U^{\pi}(s')$$

- Bellman最优方程 $$U^{*}(s) = \max_{a} \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s, a) U^{*}(s') \right)$$

- MDPs：（最优子结构）Bellman方程提供了递归地分解问题的方法，（重叠子问题）值函数存储的数据可以被重复使用
  - **策略迭代**：用迭代的方式求解**Bellman期望方程**
  - **值迭代**：用迭代的方式求解**Bellman最优方程**

策略评价

- 策略评价：计算一个策略的期望回报
- 可以用动态规划计算一个策略π的t步回报：
  - 如果不执行策略π，则$U_0^\pi(s) = 0$
  - 如果执行策略π一步，则$U_1^\pi(s) = R(s, \pi(s))$
  - 假设已知策略π的$t-1$步回报，则可以计算策略π的t步回报：

$$U_t^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s' \mid s, \pi(s)) U_{t-1}^\pi(s')$$

逐次逼近（successive approximation）

Algorithm 4.1 Iterative policy evaluation

```
function IterativePolicyEvaluation(π, n)
    U_0^π(s) ← 0 for all s
    for t ← 1 to n
        U_t^π(s) ← R(s, π(s)) + γ Σ_{s'} T(s' | s, π(s)) U_{t-1}^π(s') for all s
    return U_n^π
```

迭代地计算一个策略的n步回报

策略评价（续）

- 对带折扣奖赏的无限步数决策问题：

$$U^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s' \mid s, \pi(s)) U^{\pi}(s')$$

- 解法一：用算法4.1，当$\gamma < 1$且n足够大时，用U_n^{π}近似U^{π}

$$U_t^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s' \mid s, \pi(s)) U_{t-1}^{\pi}(s')$$

证明：$\|U_t^{\pi} - U^{\pi}\|_{\infty} = \max_s \gamma \sum_{s'} T(s' \mid s, \pi(s)) |U_{t-1}^{\pi}(s') - U^{\pi}(s')|$

$$\leq \gamma \max_s \sum_{s'} T(s' \mid s, \pi(s)) \max_{s'} |U_{t-1}^{\pi}(s') - U^{\pi}(s')|$$

$$= \gamma \max_{s'} |U_{t-1}^{\pi}(s') - U^{\pi}(s')| = \gamma \|U_{t-1}^{\pi} - U^{\pi}\|_{\infty}$$

从而有：$\|U_n^{\pi} - U^{\pi}\|_{\infty} \leq \gamma^n \|U_0^{\pi} - U^{\pi}\|_{\infty}$

策略评价（续）

- 对带折扣奖赏的无限步数决策问题：

$$U^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s'} T(s' \mid s, \pi(s)) U^{\pi}(s')$$

- 解法二：求解有n个线性方程的系统（n为状态数）

$$\mathbf{U}^{\pi} = \mathbf{R}^{\pi} + \gamma \mathbf{T}^{\pi} \mathbf{U}^{\pi}$$

n维向量，分别表示状态值函数和奖赏函数

$n \times n$矩阵，表示状态转移概率

$$\mathbf{U}^{\pi} - \gamma \mathbf{T}^{\pi} \mathbf{U}^{\pi} = \mathbf{R}^{\pi} \Rightarrow (\mathbf{I} - \gamma \mathbf{T}^{\pi}) \mathbf{U}^{\pi} = \mathbf{R}^{\pi} \Rightarrow \mathbf{U}^{\pi} = (\mathbf{I} - \gamma \mathbf{T}^{\pi})^{-1} \mathbf{R}^{\pi}$$

计算复杂度$O(n^3)$

策略改进

- 策略改进定理：对于两个确定性的策略π和π'，如果

  对于任意$s \in \mathcal{S}$，有$U^{\pi}(s) \leq Q^{\pi}(s, \pi'(s))$

  则$\pi \leq \pi'$，即对于任意$s \in \mathcal{S}$，有$U^{\pi}(s) \leq U^{\pi'}(s)$

- 在此基础上，如果存在状态$s \in \mathcal{S}$，有$U^{\pi}(s) < Q^{\pi}(s, \pi'(s))$，则存在状态$s \in \mathcal{S}$，有$U^{\pi}(s) < U^{\pi'}(s)$

- 令$\pi'(s) = \pi_{k+1}(s) = \arg\max_a Q^{\pi_k}(s, a)$，有$U^{\pi_k}(s) \leq U^{\pi_{k+1}}(s)$

- 当$\pi_{k+1} = \pi_k$时，$\pi_{k+1}(s) = \pi_k(s) = \arg\max_a Q^{\pi_k}(s, a)$，进而$U^{\pi_k}(s) = \max_a Q^{\pi_k}(s, a)$，满足Bellman最优方程，$\pi_k$是最优策略

策略迭代

- 从任一策略π_0开始，重复以下两个步骤，直到策略不再有改进：
  - **策略评价**：给定当前策略π_k，计算U^{π_k}
  - **策略改进**：使用U^{π_k}，用算法4.2的**第5行**计算一个新策略

Algorithm 4.2 Policy iteration

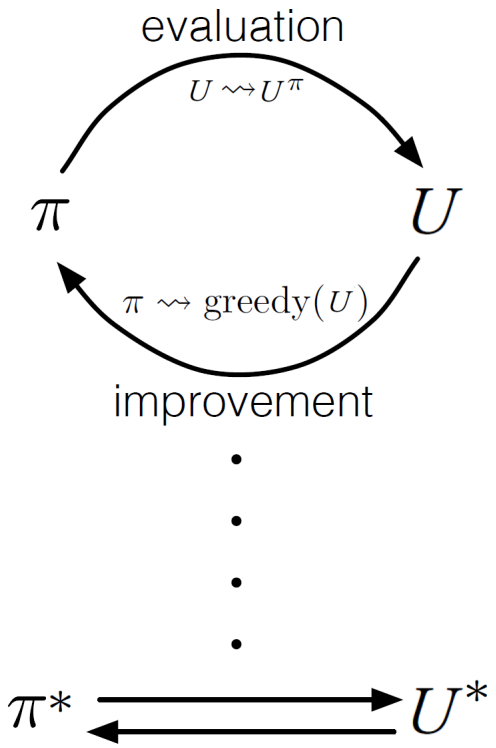
```
1: function PolicyIteration(π_0)
2:     k ← 0
3:     repeat
4:         Compute U^{π_k}
5:         π_{k+1}(s) = arg max_a (R(s,a) + γ Σ_{s'} T(s' | s,a) U^{π_k}(s')) for all states s
6:         k ← k + 1
7:     until π_k = π_{k-1}
8:     return π_k
```

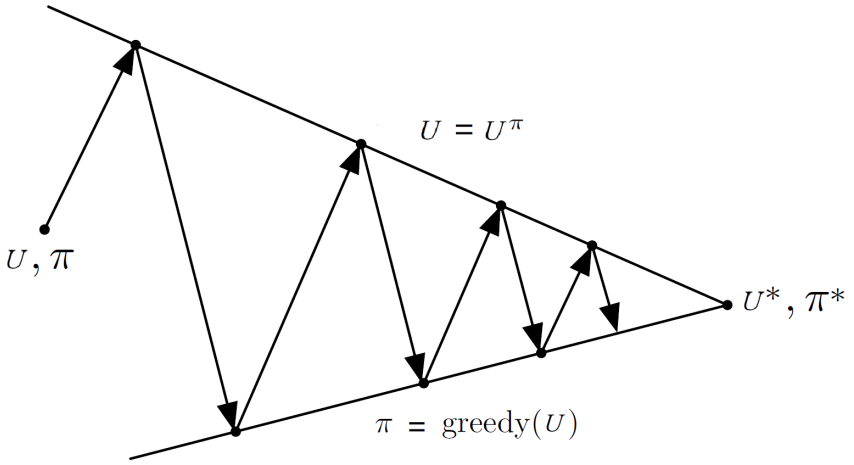
- 策略迭代总能找到最优解
  - 终止前，每一轮迭代都有策略改进，且策略的总数有限

图说策略迭代

E：策略评价
I：策略改进

$$\pi_0 \xrightarrow{\text{E}} U^{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} U^{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi^* \xrightarrow{\text{E}} U^*$$



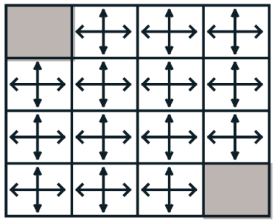
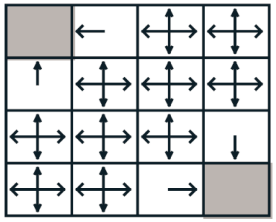
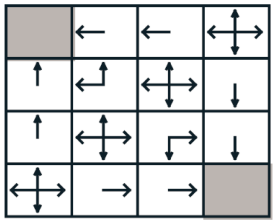


例子：4 × 4栅格世界

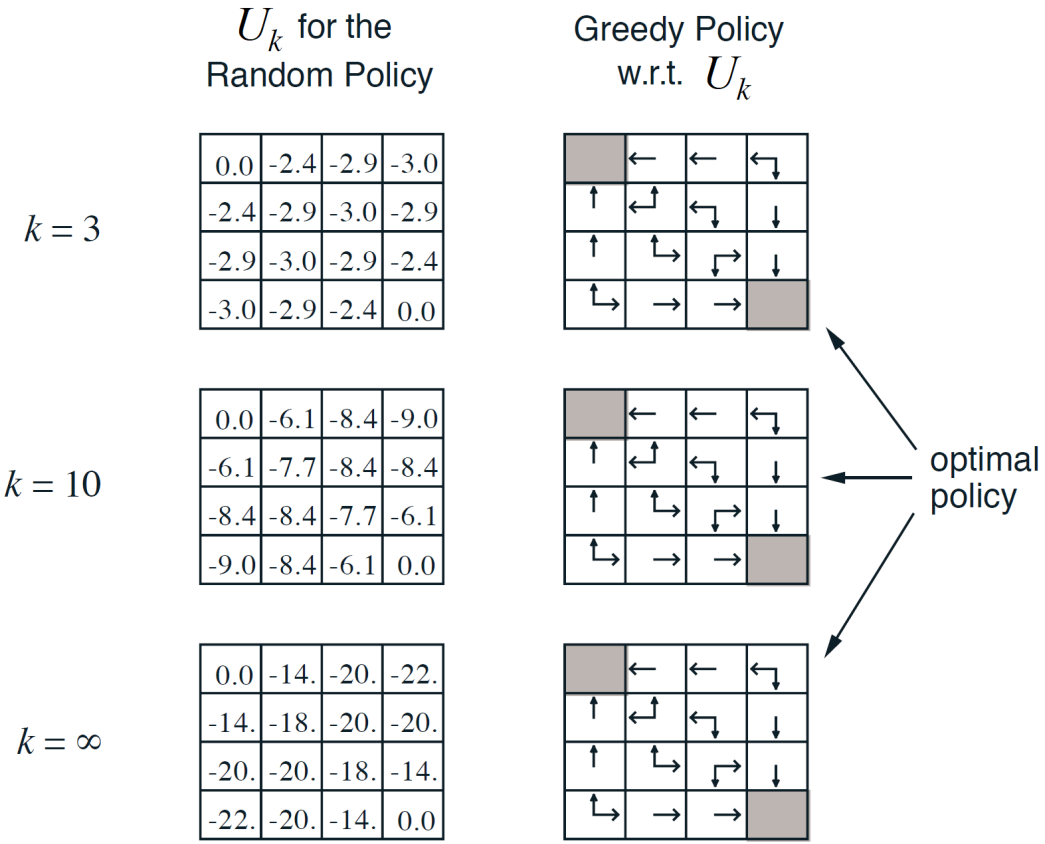
actions

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$R_t = -1$
on all transitions

	U_k for the Random Policy	Greedy Policy w.r.t. U_k
$k = 0$	0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0	 ← random policy
$k = 1$	0.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 -1.0 0.0	
$k = 2$	0.0 -1.7 -2.0 -2.0 -1.7 -2.0 -2.0 -2.0 -2.0 -2.0 -2.0 -1.7 -2.0 -2.0 -1.7 0.0	

例子：4 × 4栅格世界（续）


U_k for the Random Policy
Greedy Policy w.r.t. U_k
$k = 3$
0.0 -2.4 -2.9 -3.0
-2.4 -2.9 -3.0 -2.9
-2.9 -3.0 -2.9 -2.4
-3.0 -2.9 -2.4 0.0
$k = 10$
0.0 -6.1 -8.4 -9.0
-6.1 -7.7 -8.4 -8.4
-8.4 -8.4 -7.7 -6.1
-9.0 -8.4 -6.1 0.0
$k = \infty$
0.0 -14. -20. -22.
-14. -18. -20. -20.
-20. -20. -18. -14.
-22. -20. -14. 0.0
optimal policy

例子：汽车租赁问题

停车场1

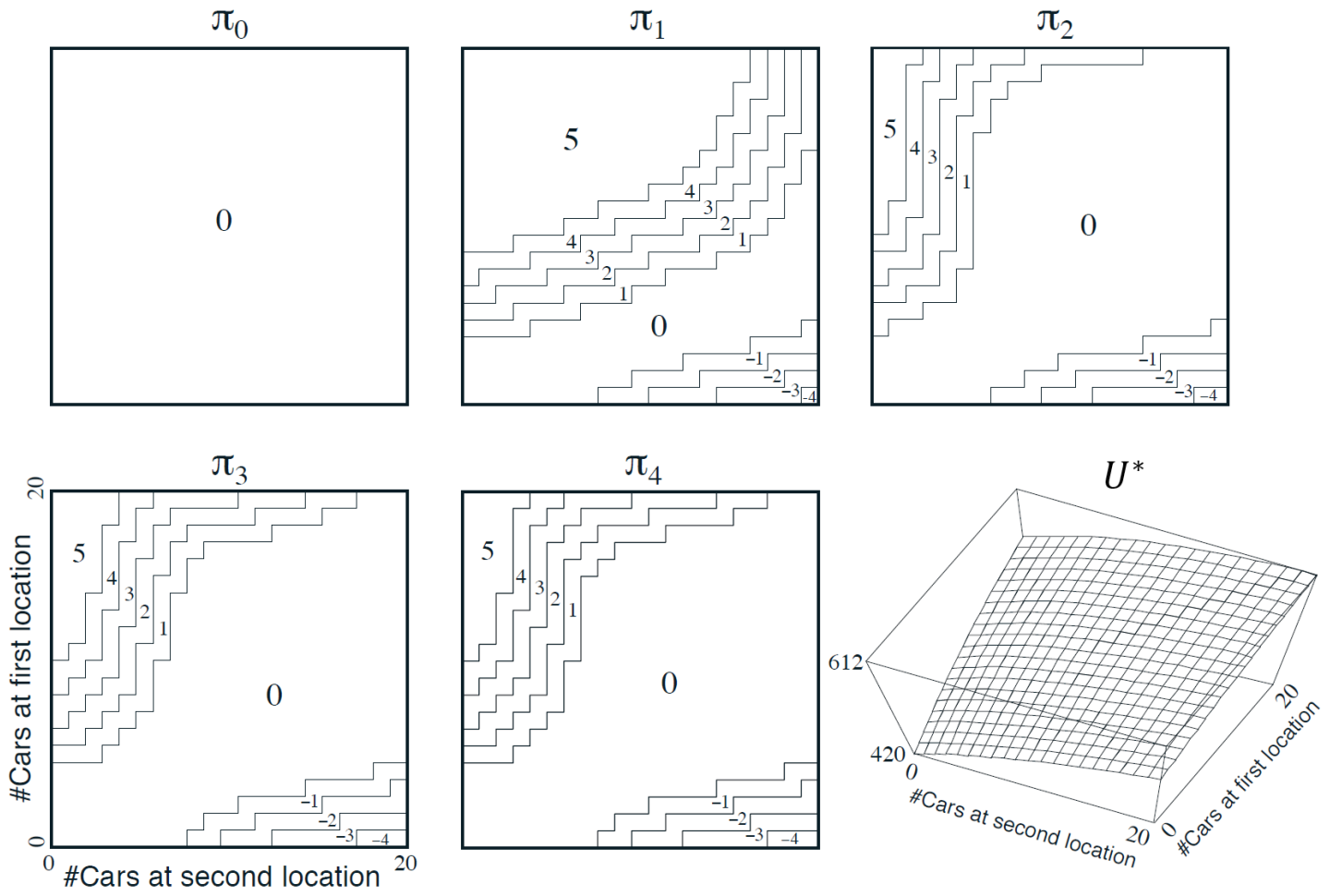
如何在两个停车场之间调度车辆，使得回报最大？

停车场2

- 状态：两个停车场的车子数，每个地方最多20辆车
- 行动：车辆调度，一次最多调度5辆车
- 状态转移函数
  - 两个停车场每天被租用和被还回的车辆数n服从均值分别为λ_{rent}和λ_{return}的泊松分布$p(n) = \frac{e^{\lambda}}{n!}e^{-\lambda}$
- 期望奖赏函数
  - 每出租一辆车，有\$10的奖赏
  - 每调度一辆车，有\$2的成本
- 折扣因子$\gamma = 0.9$

停车场1：$\lambda_{\text{rent}} = 3$和$\lambda_{\text{return}} = 3$
停车场2：$\lambda_{\text{rent}} = 4$和$\lambda_{\text{return}} = 2$

例子：汽车租赁问题（续）


π_0
0
π_1
5
4
3
2
1
0
−1
−2
−3
−4
π_2
5
4
3
2
1
0
−1
−2
−3
−4
π_3
#Cars at first location
#Cars at second location
0
20
5
4
3
2
1
0
−1
−2
−3
−4
π_4
5
4
3
2
1
0
−1
−2
−3
−4
U^*
612
420
0
20
#Cars at second location
#Cars at first location

精确动态规划

- 策略迭代
- **值迭代**
- 结构化表示
- 线性表示

值迭代

- 对**无限步数**的问题，一个最优策略的值满足Bellman最优方程：

$$U^*(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) U^*(s') \right)$$

- 计算带折扣的n步最优值函数U_n：
  - 如果$n = 0$，则对所有s：$U_0(s) \leftarrow 0$
  - 迭代地计算U_n：

$$U_n(s) \leftarrow \max_a \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) U_{n-1}(s') \right)$$

Algorithm 4.3 Value iteration

```
1: function ValueIteration
2:     k ← 0
3:     U_0(s) ← 0 for all states s
4:     repeat
5:         U_{k+1}(s) ← max_a [R(s,a) + γ Σ_{s'} T(s' | s,a) U_k(s')] for all states s
6:         k ← k + 1
7:     until convergence
8:     return U_k
```

第5行：通过迭代地更新U^*的估计来逼近U^*

值迭代（续）

- 一旦知道了U^*，就可以提取一个最优策略：

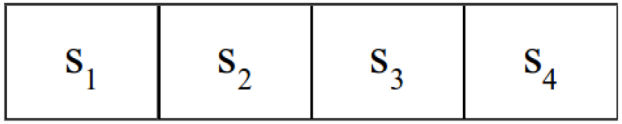
$$\pi(s) \leftarrow \arg\max_a \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) U^*(s') \right)$$

- 初始化
  - 可以是任一有界的初始值
  - 好的初始值能加速收敛
- 收敛条件
  - $\|U_k - U_{k-1}\|_\infty < \delta$

Bellman残差

- 当$\|U_k - U_{k-1}\|_\infty < \delta$，$\delta = \frac{\epsilon(1-\gamma)}{\gamma}$时，$\|U^* - U_k\|_\infty < \epsilon$
  - 当$\gamma \longrightarrow 1$，δ越小，意味着收敛更慢

例子：1×4栅格世界



- 考虑右上角的栅格世界。Agent能向左或向右移动，向左移动到左边相邻的格子中，向右移动到右边相邻的格子中。在s_1向左移动将得到奖赏100 ，游戏结束；在s_4向右移动将得到奖赏0，游戏结束。假设折扣因子为0.9，初始化各状态的折扣回报为0。用值迭代算法计算最优值函数U^*。
- 状态空间$\mathcal{S} = \{s_1, s_2, s_3, s_4\}$
- 行动空间$\mathcal{A} = \{\text{left}, \text{right}\}$
- 状态转移函数
  - $T(s_{终止} \mid s_1, \text{left}) = 1, T(s_{终止} \mid s_4, \text{right}) = 1$
  - $T(s_i \mid s_{i+1}, \text{left}) = 1, i = \{2, 3, 4\}$
  - $T(s_{i+1} \mid s_i, \text{right}) = 1, i = \{1, 2, 3\}$
  - 其余状态转移概率为0

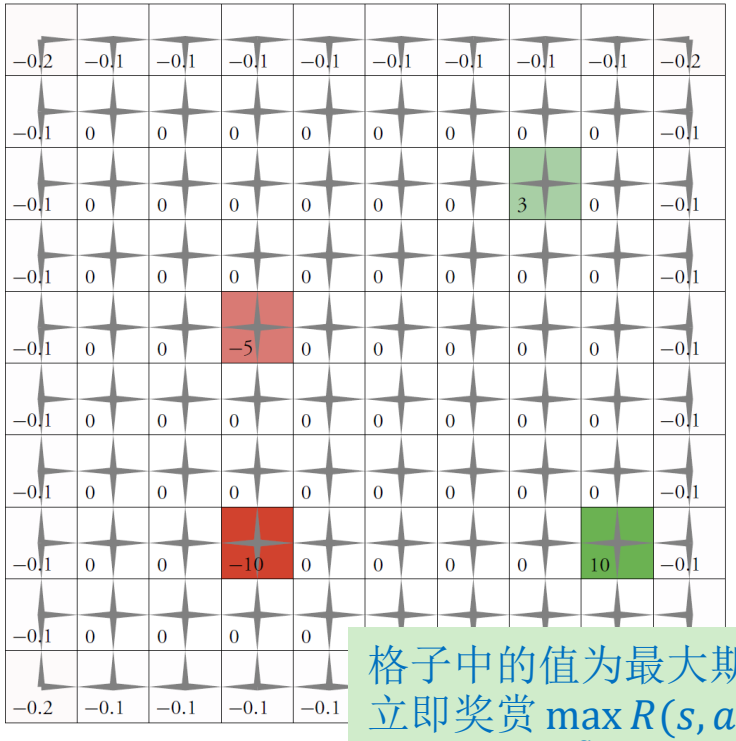
例子：1×4栅格世界（续）

- 期望奖赏函数
  - $R(s_1, \text{left}) = 100$，其余为0
- $U_0(s_i) = 0,\ i = \{1, 2, 3, 4\}$
- 由值迭代公式 $$U_n(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s, a) U_{n-1}(s') \right)$$
- $U_1(s_1) = \max\{R(s_1, \text{left}) + \gamma U_0(s_{\text{终止}}), R(s_1, \text{right}) + \gamma U_0(s_2)\}$
  $= \max\{100, 0\} = 100$
- 类似地，$U_1(s_i) = 0,\ i = \{2, 3, 4\}$
- $U_2 = [100, 90, 0, 0],\ U_3 = [100, 90, 81, 0]$
- $U_4 = [100, 90, 81, 72.9],\ U_5 = U_4$ $U^* = [100, 90, 81, 72.9]$

例子：10 × 10栅格世界

- 10 × 10的栅格世界
  - 每个格子表示一个状态
  - 行动：上、下、左、右
  - 行动的效果是随机的
    - 朝指定方向移动的概率为0.7，朝其余3个方向移动的概率各为0.1
  - 若与墙碰撞，则原地不动
  - 与墙碰撞的惩罚为1
  - 进入格子$(8,9)$和$(3,8)$后执行任一行动的奖赏分别为$+10$和$+3$，随后转移到终止状态，情节结束
  - 进入格子$(5,4)$和$(8,4)$后执行任一行动的奖赏分别为-5和-10

第1轮值迭代的结果
（折扣因子$\gamma = 0.9$）

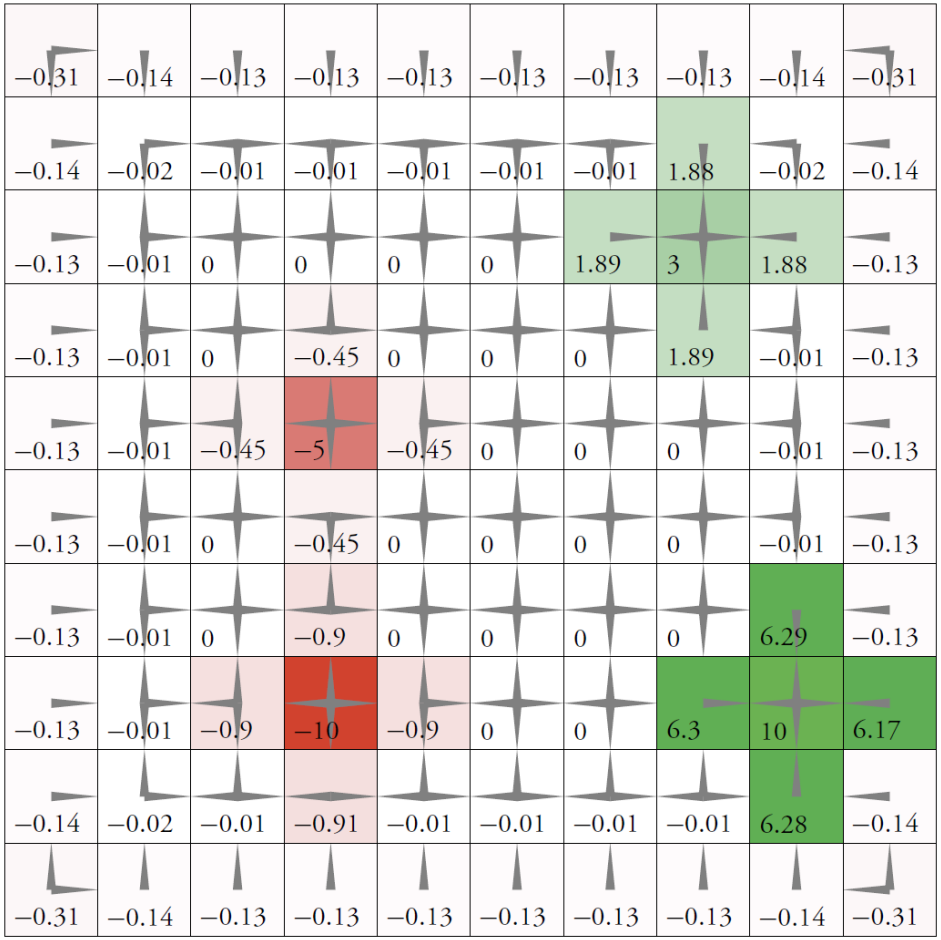


格子中的值为最大期望立即奖赏 $\max_a R(s,a)$

- 与墙不相邻的格子
  - 所有行动都是最优行动
- 与墙相邻的格子
  - 最优行动是远离墙

例子：10 × 10栅格世界（续）

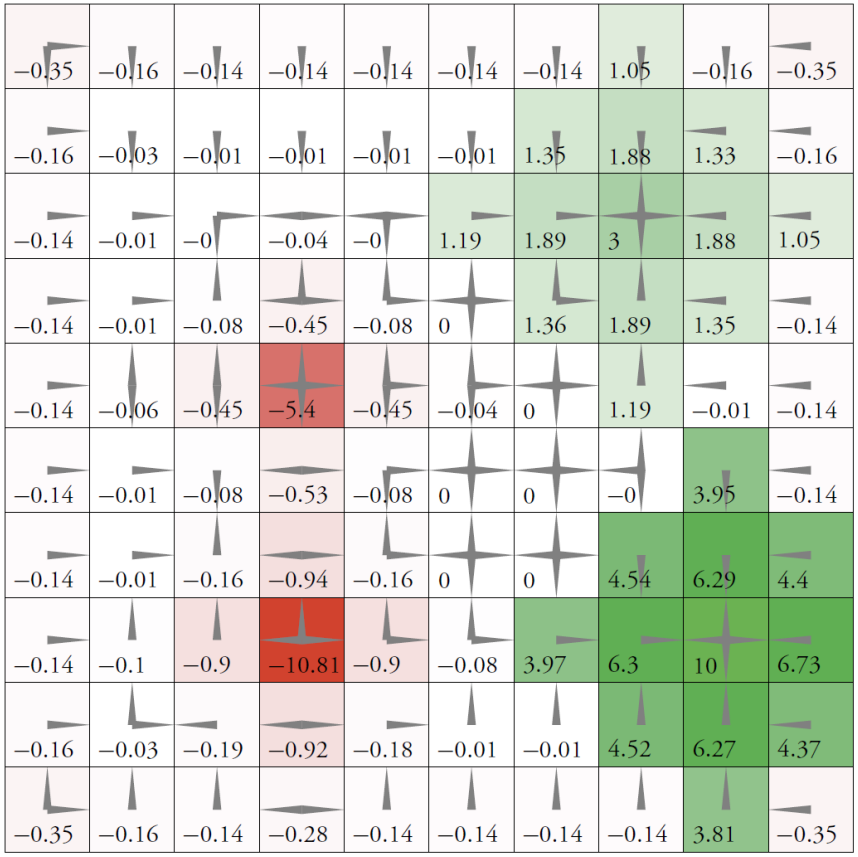
- 第2轮值迭代的结果（折扣因子$\gamma = 0.9$）

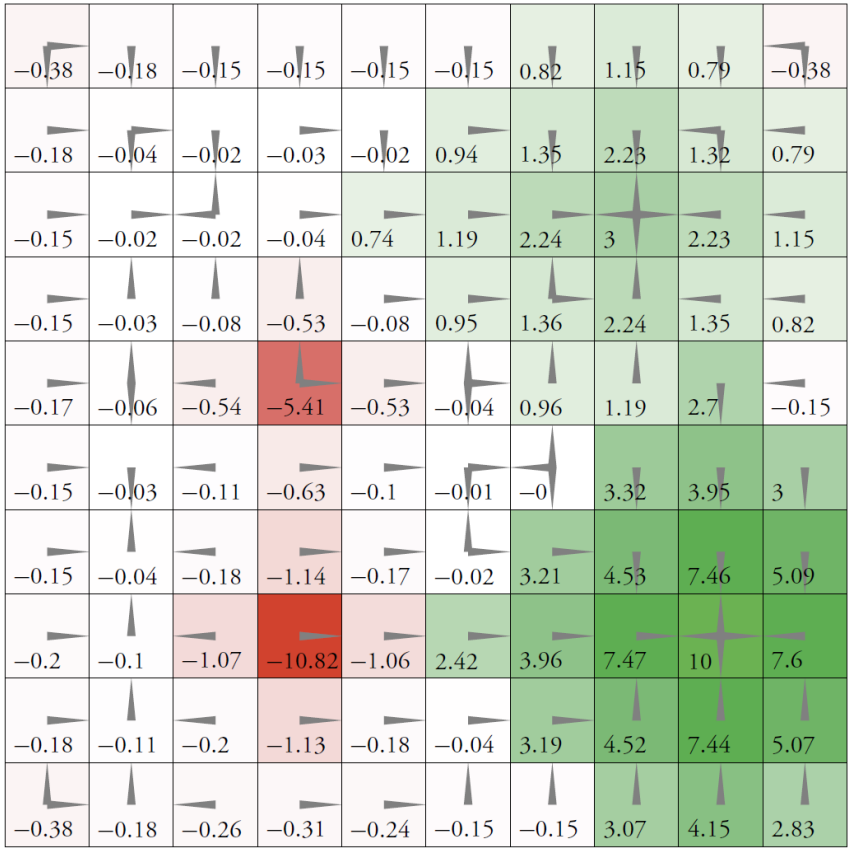


- 有非0奖赏的吸收格子的值保持不变，但值扩散到了相邻格子
- 格子的值：两步的期望折扣回报
- 离吸收格子或边界格子不止一步的格子，值为0
- 离吸收格子或边界格子最多一步的格子，最优行动集合会更新
  - 向正奖赏格子靠近
  - 远离负奖赏格子

例子：10 × 10栅格世界（续）

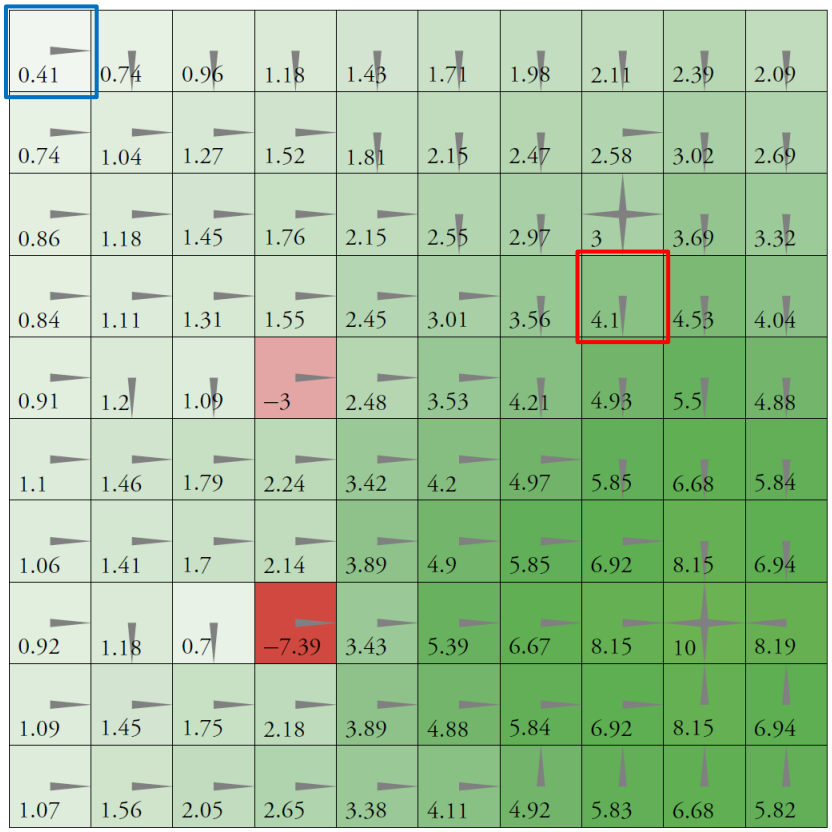
- 第3轮（左）、第4轮（右）值迭代的结果（折扣因子$\gamma = 0.9$）





例子：10 × 10栅格世界（续）

- 收敛时的值函数和策略（左：$\gamma = 0.9$，右：$\gamma = 0.5$）
  - 不同折扣因子会影响值函数和策略



(a) $\gamma = 0.9$

(b) $\gamma = 0.5$

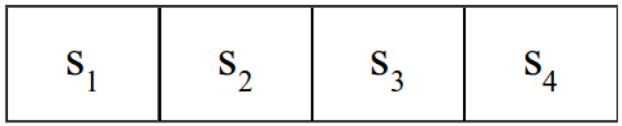
异步值迭代

- 值迭代：在每一轮迭代，基于U_k，对所有状态计算U_{k+1}
- 异步值迭代：在每一轮迭代，仅更新状态空间的一个子集
- 高斯-赛德尔（Gauss-Seidel）值迭代
  - 只保存一份状态值，对值就地（in place）更新：

$$U(s) \leftarrow \max_a \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) U(s') \right)$$

  - 通常能更快收敛（vs. 值迭代）
  - 只要值函数在每个状态上的更新次数无限多，就能收敛到最优值函数

例子：1×4栅格世界（续）

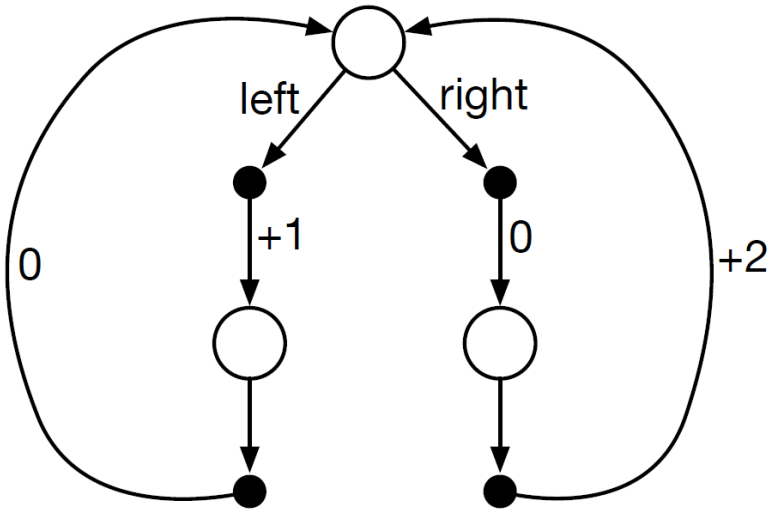


- 应用高斯-赛德尔值迭代求解1×4栅格世界问题，状态的更新顺序为s_1, s_2, s_3, s_4。问：需要多少轮迭代可以收敛到U^*？
  - $U_0(s_i) = 0,\ i = \{1, 2, 3, 4\}$
  - 由高斯-塞德尔值迭代公式 $U(s) \leftarrow \max_a \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) U(s') \right)$
  - $U_1(s_1) = \max\{R(s_1, \text{left}) + \gamma U_0(s_{终止}), R(s_1, \text{right}) + \gamma U_0(s_2)\}$
    $= \max\{100, 0\} = 100$
  - $U_1(s_2) = \max\{R(s_2, \text{left}) + \gamma U_1(s_1), R(s_1, \text{right}) + \gamma U_0(s_2)\}$
    $= \max\{90, 0\} = 90$
  - 类似地，$U_1(s_3) = 81, U_1(s_4) = 72.9$

只需要1轮迭代！

课后练习4.2

- 考虑一个无限步数的MDP（如下图所示）。该MDP仅需在顶部状态做决策，有left和right两个行动可供选择。每次行动后会得到确定性的奖赏。有两个确定性策略：π_{left}和π_{right}。请问，当γ分别为0、0.5和0.9时，哪个策略最优？



课后练习4.3

- 用h和l表示状态high和low，用s、w和re表示search、wait和recharge。写出吸尘机器人MDP的最优状态值函数的Bellman最优方程。

课后练习4.4

- 考虑图（a）中的3×3 世界，每个格子中的数值表示的是$R(s)$，即状态s的立即奖赏，右上角含有+10的格子是终止状态（进入终止状态得到+10的奖赏后，采取任意行动都会导致情节结束）。转移模型如图（b）所示，它表示的含义是：以0.8的概率向选择的方向移动，各以0.1的概率向与它垂直的两个方向移动。假设Agent的可选行动为上（U）、下（D）、左（L）、右（R），使用折扣因子为0.99的折扣奖赏定义效用（即回报）。对于下面的每种情况，计算最优策略。

（1）$r = 100$

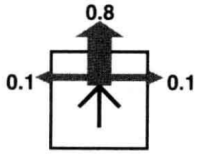
（2）$r = -3$

（3）$r = 0$

（4）$r = 3$

r	−1	+10
−1	−1	−1
−1	−1	−1

(a)



(b)

课后练习4.5

- 已知Bellman最优方程：

$$U^*(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) U^*(s') \right)$$

和值迭代的更新公式：

$$U_{k+1}(s) \leftarrow \max_a \left[R(s,a) + \gamma \sum_{s'} T(s' \mid s,a) U_k(s') \right]$$

试证明：当$\|U_k - U_{k-1}\|_\infty < \delta$，$\delta = \frac{\epsilon(1-\gamma)}{\gamma}$时，$\|U^* - U_k\|_\infty < \epsilon$

编程作业1

（1）实现值迭代算法。复现值迭代算法在10×10栅格世界问题上的实验结果。

（2）实现高斯-赛德尔值迭代算法、策略迭代算法，在10×10栅格世界问题上这3个算法的实验结果。

提交代码（用Python或者C++实现 ）和实验报告。

截止时间为：**2021年5月12日**

本科生班的同学把作业发给徐峰： xufeng@lamda.nju.edu.cn

研究生班的同学把作业发给刘旭辉： liuxh@lamda.nju.edu.cn