强化学习

第三讲：动态规划

教师：赵冬斌 朱圆恒 张启超

中国科学院大学
中国科学院自动化研究所

March 20, 2020

上一节课回顾

- 马尔可夫性
- 马尔可夫过程
- 马尔可夫奖励过程
- 马尔可夫决策过程
- 策略与价值
- 最优化原理
- MDPs 扩展

$$V_*(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)$$

$$\pi_*(s) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)$$

$$Q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \max_{a'} Q_*(s',a')$$

$$\pi_*(s) = \arg\max_a Q_*(s,a)$$

- 求解贝尔曼最优方程需要：
  1. 求解非线性算子 max
  2. 模型已知
  3. 足够的计算空间

动态规划

动态规划

通过把原问题分解为相对简单的子问题来求解复杂问题的方法

- 将复杂的问题分解为相对简单的子问题
- 求解子问题
- 根据子问题的解得出原问题解

DP 的两个适用条件

1. 最优子结构性质：问题最优解包含子问题的解也是子问题的最优解
   - 也就是满足最优化原理，将问题划分成子问题
2. 子问题重叠性质：使用递归算法自顶向下对问题进行求解，每次产生的子问题并不总是新问题
   - 子问题重复出现多次
   - 保存子问题的首次计算结果，再次需要时直接使用

DP 的两个适用条件

1. 最优子结构性质: 问题最优解包含子问题的解也是子问题的最优解
   - 也就是满足最优化原理，将问题划分成子问题
2. 子问题重叠性质: 使用递归算法自顶向下对问题进行求解，每次产生的子问题并不总是新问题
   - 子问题重复出现多次
   - 保存子问题的首次计算结果，再次需要时直接使用

- 马尔可夫决策过程是满足以上两个属性
  - 贝尔曼方程具有递归形式
  - 价值函数可以保存和重复利用

价值迭代

价值迭代 Value Iteration, VI

$$V_*(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)$$

- 贝尔曼最优方程的难点在于求解的 V_* 同时存在于非线性等式两边
- 价值迭代的基本思路:

$$V_*(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)$$

$V(s')$

- 贝尔曼最优方程的难点在于求解的 V_* 同时存在于非线性等式两边
- 价值迭代的基本思路:
  1. 对 V_* 定义一个估计函数 V

价值迭代 Value Iteration, VI

$$V_*(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)$$

$$V'(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V(s') \right)$$

- 贝尔曼最优方程的难点在于求解的 V_* 同时存在于非线性等式两边
- 价值迭代的基本思路:
  1. 对 V_* 定义一个估计函数 V
  2. 将估计函数代入方程右边, 等式左边得到一个新函数 V'
  3. V' 是对 V_* 更为准确的估计

价值迭代 Value Iteration, VI

$$V_*(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)$$

$$V'(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V(s') \right)$$

- 贝尔曼最优方程的难点在于求解的 V_* 同时存在于非线性等式两边
- 价值迭代的基本思路:
  1. 对 V_* 定义一个估计函数 V
  2. 将估计函数代入方程右边，等式左边得到一个新函数 V'
  3. V' 是对 V_* 更为准确的估计
  4. 将 V' 代入右式继续上述过程

价值迭代流程

1: 初始化一个函数 V_1 (e.g. $V_1(s) = 0, \forall s \in \mathcal{S}$)
2: **loop**
3: 根据已知的 V_k 计算一个新的函数

$$V_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V_k(s') \right), \forall s \in \mathcal{S}$$

4: $k \leftarrow k + 1$
5: **end loop**

价值迭代流程

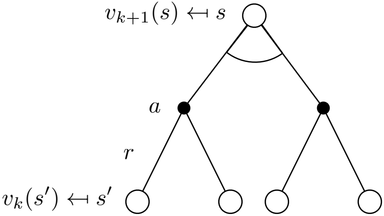
初始化一个函数 V_1 (e.g. $V_1(s) = 0, \forall s \in \mathcal{S}$)
loop
  根据已知的 V_k 计算一个新的函数

$$V_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V_k(s') \right), \forall s \in \mathcal{S}$$

  $k \leftarrow k + 1$
end loop

收敛定理

$$\lim_{k \to \infty} V_k = V_*$$



$$V_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V_k(s') \right)$$

$$\boxed{\text{矩阵形式：} \mathcal{V}_{k+1} = \max_{a \in \mathcal{A}} \left(\mathcal{R}^a + \gamma \mathcal{P}^a \mathcal{V}_k \right)}$$

价值迭代算子

- 价值迭代定义一个以函数作为输入的算子 $\mathcal{T}$, 对给定的函数 V_k 计算新的函数

$$
\begin{aligned}
V_{k+1}(s) &= [\mathcal{T}(V_k)](s) \\
&= \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)
\end{aligned}
$$

- $\mathcal{T}$ 称为价值迭代算子
- 贝尔曼最优方程写成

$$
\begin{aligned}
V_*(s) &= [\mathcal{T}(V_*)](s) \\
&= \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)
\end{aligned}
$$

- **无穷范数** ∞-norm： 用 $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ 表示函数 x 在有限集合上各个元素的取值。函数的无穷范数等于

$$\|x\|_\infty = \max(|x_1|, |x_2|, \ldots, |x_n|)$$

收缩算子

对于任意两个函数 f, g, 如果两个函数的误差经过一个算子 $\mathcal{T}$ 后被缩小, 那么算子称为收缩算子 (contracting operator)

$$\|\mathcal{T}(f - g)\|_\infty < \|f - g\|_\infty$$

- 定理: $\gamma < 1$ 时, 价值迭代算子是一个收缩算子

证明

$$
\begin{aligned}
&\Big|\left[\mathcal{T}(u)\right](s)-\left[\mathcal{T}(v)\right](s)\Big|, \quad \forall s\in\mathcal{S}\\
&=\left|\max_{a_1}\left(\mathcal{R}^{a_1}+\gamma\sum_{s'}\mathcal{P}^{a_1}_{ss'}u(s')\right)-\max_{a_2}\left(\mathcal{R}^{a_2}+\gamma\sum_{s'}\mathcal{P}^{a_2}_{ss'}v(s')\right)\right|\\
&\le\max_{a}\left|\left(\mathcal{R}^{a}+\gamma\sum_{s'}\mathcal{P}^{a}_{ss'}u(s')\right)-\left(\mathcal{R}^{a}+\gamma\sum_{s'}\mathcal{P}^{a}_{ss'}v(s')\right)\right|\\
&=\gamma\max_{a}\left|\sum_{s'}\mathcal{P}^{a}_{ss'}\left(u(s')-v(s')\right)\right|\\
&\le\gamma\|u-v\|_\infty \qquad (\gamma<1)
\end{aligned}
$$

$$\Rightarrow\|\mathcal{T}(u-v)\|_\infty<\|u-v\|_\infty$$

- 所以 $\mathcal{T}$ 是一个收缩算子

- 从任意初始价值函数 V_1 出发，经过价值迭代计算的新函数 V_{k+1} 与最优价值函数之间的误差

$$\| V_{k+1} - V_* \|_\infty = \| \mathcal{T} V_k - \mathcal{T} V_* \|_\infty$$

- 从任意初始价值函数 V_1 出发，经过价值迭代计算的新函数 V_{k+1} 与最优价值函数之间的误差

$$
\begin{aligned}
\|V_{k+1} - V_*\|_\infty =& \|\mathcal{T}V_k - \mathcal{T}V_*\|_\infty \\
\leq& \gamma\|V_k - V_*\|_\infty
\end{aligned}
$$

- 从任意初始价值函数 V_1 出发，经过价值迭代计算的新函数 V_{k+1} 与最优价值函数之间的误差

$$
\begin{aligned}
\| V_{k+1} - V_* \|_\infty =& \| \mathcal{T} V_k - \mathcal{T} V_* \|_\infty \\
\leq & \gamma \| V_k - V_* \|_\infty \\
& \vdots \\
\leq & \gamma^k \| V_1 - V_* \|_\infty
\end{aligned}
$$

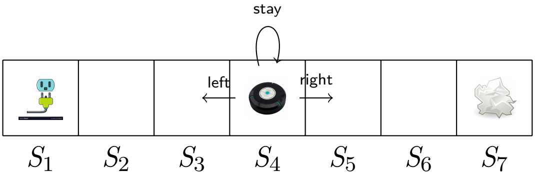
- 当 $k \to \infty$，$V_k \to V_*$

价值迭代收敛性, $\gamma = 1$

- $\gamma = 1$ 时 $\mathcal{T}$ 不再是收缩的, 需要其它的证明方法. 大体思路:
  1. 当初始函数 $V_1 \le V_*$ 时, 价值迭代结果是单调递增的 $V_1 \le V_2 \le \cdots \le V_k \le V_*$
  2. 当初始函数 $V_1 \ge V_*$ 时, 价值迭代结果是单调递减的 $V_1 \ge V_2 \ge \cdots \ge V_k \ge V_*$
  3. 对于任意形状的初始函数 V_1, 额外构造两个函数 $\underline{V}_1 = \min\{V_*, V_1\}$, $\overline{V}_1 = \max\{V_*, V_1\}$, 分别对三个初始函数进行价值迭代, 每代结果都有 $\underline{V}_k \le V_k \le \overline{V}_k$, 所以 $V_k \to V_*$
- 有兴趣的同学参考

1. Tamimi, A.A., F.L. Lewis, and M.A. Khalaf, Discrete-Time Nonlinear HJB Solution Using Approximate Dynamic Programming: Convergence Proof. IEEE Transactions on Systems, Man, and Cybernetics, Part B, 2008. 38(4): p. 943–949.
2. Bertsekas, D.P., Value and Policy Iterations in Optimal Control and Adaptive Dynamic Programming. IEEE Transactions on Neural Networks and Learning Systems, 2017. 28(3): p. 500-509.

举例：扫地机器人



- $\mathcal{S}$: $\{S_1, S_2, \ldots, S_7\}$
- $\mathcal{A}$: $\{A_{left}, A_{right}\}$
- $\mathcal{R}$: $\mathcal{R}(S_1) = +1, \mathcal{R}(S_7) = +10, \mathcal{R}(S_i) = 0$
- G_0: $r_1 + \gamma r_2 + \gamma^2 r_3 + \ldots$
- $\gamma = 0.7$

状态转移

- $s = S_2, \ldots, S_6$

$$\mathcal{P}_{ss'}^{a} = \begin{cases} 0.8 & (a = A_{left}, s' = s - 1) || (a = A_{right}, s' = s + 1) \\ 0.1 & s' = s \\ 0.1 & (a = A_{left}, s' = s + 1) || (a = A_{right}, s' = s - 1) \end{cases}$$

- $s = S_1$

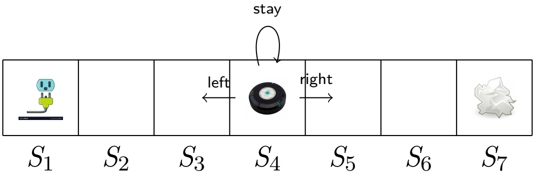
$$\mathcal{P}_{S_1,S_1}^{A_{left}} = 0.9,\ \mathcal{P}_{S_1,S_2}^{A_{left}} = 0.1$$

$$\mathcal{P}_{S_1,S_1}^{A_{right}} = 0.2,\ \mathcal{P}_{S_1,S_2}^{A_{right}} = 0.8$$

- $s = S_7$

$$\mathcal{P}_{S_7,S_6}^{A_{left}} = 0.8,\ \mathcal{P}_{S_7,S_7}^{A_{left}} = 0.2$$

$$\mathcal{P}_{S_7,S_6}^{A_{right}} = 0.1,\ \mathcal{P}_{S_7,S_7}^{A_{right}} = 0.9$$



$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 初始化 $V_1(s) = 0$

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k = 1$ 次迭代
  1. $S_j \in \{S_2, \ldots, S_6\}$,

$$V_2(S_j) = \max \begin{cases} \mathcal{R}(S_j) + \gamma \begin{pmatrix} \mathcal{P}_{S_j,S_{j-1}}^{left} V_1(S_{j-1}) + \\ \mathcal{P}_{S_j,S_j}^{left} V_1(S_j) + \\ \mathcal{P}_{S_j,S_{j+1}}^{left} V_1(S_{j+1}) \end{pmatrix} & \text{a=left} \\ \mathcal{R}(S_j) + \gamma \begin{pmatrix} \mathcal{P}_{S_j,S_{j-1}}^{right} V_1(S_{j-1}) + \\ \mathcal{P}_{S_j,S_j}^{right} V_1(S_j) + \\ \mathcal{P}_{S_j,S_{j+1}}^{right} V_1(S_{j+1}) \end{pmatrix} & \text{a = right} \end{cases}$$

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k = 1$ 次迭代
  1. $S_j \in \{S_2, \ldots, S_6\}$, $V_2(S_j) = 0$

$$V_2(S_j) = \max \begin{cases} 0 + 0.7 \begin{pmatrix} 0.8 * 0 + \\ 0.1 * 0 + \\ 0.1 * 0 \end{pmatrix} & \text{a=left} \\ 0 + 0.7 \begin{pmatrix} 0.1 * 0 + \\ 0.1 * 0 + \\ 0.8 * 0 \end{pmatrix} & \text{a = right} \end{cases}$$

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k = 1$ 次迭代
  3. S_1,

$$V_2(S_1) = \max \begin{cases} \mathcal{R}(S_1) + \gamma \begin{pmatrix} \mathcal{P}_{S_1,S_1}^{left} V_1(S_1) + \\ \mathcal{P}_{S_1,S_2}^{left} V_1(S_2) \end{pmatrix} & \text{a=left} \\ \mathcal{R}(S_1) + \gamma \begin{pmatrix} \mathcal{P}_{S_1,S_1}^{right} V_1(S_1) + \\ \mathcal{P}_{S_1,S_2}^{right} V_1(S_2) \end{pmatrix} & \text{a = right} \end{cases}$$

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k = 1$ 次迭代
  3. S_1，$V_2(S_1) = 1$

$$V_2(S_1) = \max \begin{cases} 1 + 0.7 \begin{pmatrix} 0.9 * 0 + \\ 0.1 * 0 \end{pmatrix} & \text{a=left} \\ 1 + 0.7 \begin{pmatrix} 0.2 * 0 + \\ 0.8 * 0 \end{pmatrix} & \text{a = right} \end{cases}$$

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k = 1$ 次迭代
  4. S_7,

$$V_2(S_7) = \max \begin{cases} \mathcal{R}(S_7) + \gamma \begin{pmatrix} \mathcal{P}_{S_7,S_6}^{left} V_1(S_6) + \\ \mathcal{P}_{S_7,S_7}^{left} V_1(S_7) \end{pmatrix} & \text{a=left} \\ \mathcal{R}(S_7) + \gamma \begin{pmatrix} \mathcal{P}_{S_7,S_6}^{right} V_1(S_6) + \\ \mathcal{P}_{S_7,S_7}^{right} V_1(S_7) \end{pmatrix} & \text{a = right} \end{cases}$$

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k=1$ 次迭代
  4. S_7，$V_2(S_7) = 10$

$$V_2(S_7) = \max \begin{cases} 10 + 0.7 \begin{pmatrix} 0.8 * 0 + \\ 0.2 * 0 \end{pmatrix} & \text{a=left} \\ 10 + 0.7 \begin{pmatrix} 0.1 * 0 + \\ 0.9 * 0 \end{pmatrix} & \text{a = right} \end{cases}$$

价值迭代结果

V_k	S_1	S_2	S_3	S_4	S_5	S_6	S_7
$k=1$	0	0	0	0	0	0	0
$k=2$	1	0	0	0	0	0	10

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k=2$ 次迭代
  1. $S_j \in \{S_2, \ldots, S_6\}$,

$$V_3(S_j) = \max \begin{cases} \mathcal{R}(S_j) + \gamma \begin{pmatrix} \mathcal{P}^{left}_{S_j,S_{j-1}} V_2(S_{j-1}) + \\ \mathcal{P}^{left}_{S_j,S_j} V_2(S_j) + \\ \mathcal{P}^{left}_{S_j,S_{j+1}} V_2(S_{j+1}) \end{pmatrix} & \text{a=left} \\ \mathcal{R}(S_j) + \gamma \begin{pmatrix} \mathcal{P}^{right}_{S_j,S_{j-1}} V_2(S_{j-1}) + \\ \mathcal{P}^{right}_{S_j,S_j} V_2(S_j) + \\ \mathcal{P}^{right}_{S_j,S_{j+1}} V_2(S_{j+1}) \end{pmatrix} & \text{a = right} \end{cases}$$

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k = 2$ 次迭代
  2. S_1,

$$V_3(S_1) = \max \begin{cases} \mathcal{R}(S_1) + \gamma \begin{pmatrix} \mathcal{P}_{S_1,S_1}^{left} V_2(S_1) + \\ \mathcal{P}_{S_1,S_2}^{left} V_2(S_2) \end{pmatrix} & \text{a=left} \\ \mathcal{R}(S_1) + \gamma \begin{pmatrix} \mathcal{P}_{S_1,S_1}^{right} V_2(S_1) + \\ \mathcal{P}_{S_1,S_2}^{right} V_2(S_2) \end{pmatrix} & \text{a = right} \end{cases}$$

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 第 $k = 2$ 次迭代
  3. S_7,

$$V_3(S_7) = \max \begin{cases} \mathcal{R}(S_7) + \gamma \begin{pmatrix} \mathcal{P}_{S_7,S_6}^{left} V_2(S_6) + \\ \mathcal{P}_{S_7,S_7}^{left} V_2(S_7) \end{pmatrix} & \text{a=left} \\ \mathcal{R}(S_7) + \gamma \begin{pmatrix} \mathcal{P}_{S_7,S_6}^{right} V_2(S_6) + \\ \mathcal{P}_{S_7,S_7}^{right} V_2(S_7) \end{pmatrix} & \text{a = right} \end{cases}$$

$$V_2(S_1) = \max\begin{cases} 1 + 0.7\,(0.9 * 1 + 0.1 * 0) & a = \mathit{left} \\ 1 + 0.7\,(0.2 * 1 + 0.8 * 0) & a = \mathit{right} \end{cases} = 1.63$$

$$V_2(S_2) = \max\begin{cases} 0 + 0.7\,(0.8 * 1 + 0.1 * 0 + 0.1 * 0) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 1 + 0.1 * 0 + 0.8 * 0) & a = \mathit{right} \end{cases} = 0.56$$

$$V_2(S_3, \ldots, S_5) = \max\begin{cases} 0 + 0.7\,(0.8 * 0 + 0.1 * 0 + 0.1 * 0) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 0 + 0.1 * 0 + 0.8 * 0) & a = \mathit{right} \end{cases} = 0$$

$$V_2(S_6) = \max\begin{cases} 0 + 0.7\,(0.8 * 0 + 0.1 * 0 + 0.1 * 10) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 0 + 0.1 * 0 + 0.8 * 10) & a = \mathit{right} \end{cases} = 5.6$$

$$V_2(S_7) = \max\begin{cases} 10 + 0.7\,(0.8 * 0 + 0.2 * 10) & a = \mathit{left} \\ 10 + 0.7\,(0.1 * 0 + 0.9 * 10) & a = \mathit{right} \end{cases} = 16.3$$

价值迭代结果

V_k	S_1	S_2	S_3	S_4	S_5	S_6	S_7
$k=1$	0	0	0	0	0	0	0
$k=2$	1	0	0	0	0	0	10
$k=3$	1.63	0.56	0	0	0	5.6	16.3

价值迭代结果

V_k	S_1	S_2	S_3	S_4	S_5	S_6	S_7
$k=1$	0	0	0	0	0	0	0
$k=2$	1	0	0	0	0	0	10
$k=3$	1.63	0.56	0	0	0	5.6	16.3
$k=4$	2.0661	0.952	0.3136	0	3.136	9.52	20.661
$k=5$	2.3683	1.2456	0.5550	1.7781	5.5507	12.4561	23.6828
$k=6$	2.5792	1.4523	1.1218	3.2717	7.4884	14.5229	25.7921
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$k=26$	3.3063	3.2040	4.9096	7.7550	12.2673	19.4052	30.6951
$k=27$	3.3073	3.2051	4.9108	7.7562	12.2684	19.4063	30.6963

- V 函数一直在变化
- 变化误差小于一定阈值认为已经收敛

$$v_{27}(S_1) = \max \begin{cases} 1 + 0.7\,(0.9 * 3.3073 + 0.1 * 3.2051) & a = \mathit{left} \\ 1 + 0.7\,(0.2 * 3.3073 + 0.8 * 3.2051) & a = \mathit{right} \end{cases}$$

$$v_{27}(S_2) = \max \begin{cases} 0 + 0.7\,(0.8 * 3.3073 + 0.1 * 3.2051 + 0.1 * 4.9108) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 3.3073 + 0.1 * 3.2051 + 0.8 * 4.9108) & a = \mathit{right} \end{cases}$$

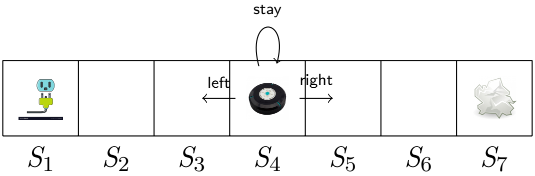
$$v_{27}(S_3) = \max \begin{cases} 0 + 0.7\,(0.8 * 3.2051 + 0.1 * 4.9108 + 0.1 * 7.7562) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 3.2051 + 0.1 * 4.9108 + 0.8 * 7.7562) & a = \mathit{right} \end{cases}$$

$$v_{27}(S_4) = \max \begin{cases} 0 + 0.7\,(0.8 * 4.9108 + 0.1 * 7.7562 + 0.1 * 12.2684) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 4.9108 + 0.1 * 7.7562 + 0.8 * 12.2684) & a = \mathit{right} \end{cases}$$

$$v_{27}(S_5) = \max \begin{cases} 0 + 0.7\,(0.8 * 7.7562 + 0.1 * 12.2684 + 0.1 * 19.4063) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 7.7562 + 0.1 * 12.2684 + 0.8 * 19.4063) & a = \mathit{right} \end{cases}$$

$$v_{27}(S_6) = \max \begin{cases} 0 + 0.7\,(0.8 * 12.2684 + 0.1 * 19.4063 + 0.1 * 30.6963) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 12.2684 + 0.1 * 19.4063 + 0.8 * 30.6963) & a = \mathit{right} \end{cases}$$

$$v_{27}(S_7) = \max \begin{cases} 10 + 0.7\,(0.8 * 19.4063 + 0.2 * 30.6963) & a = \mathit{left} \\ 10 + 0.7\,(0.1 * 19.4063 + 0.9 * 30.6963) & a = \mathit{right} \end{cases}$$



$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s')\right) \\ 0 & \text{otherwise} \end{cases}$$

- 价值迭代找到的最优策略

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
π^*	left	right	right	right	right	right	right

策略迭代

- 价值迭代：迭代价值函数，最优策略从收敛的价值函数提取

$$V_1 \rightarrow V_2 \rightarrow \cdots \rightarrow V_\infty = V_* \rightarrow \pi_*$$

- 价值迭代：迭代价值函数，最优策略从收敛的价值函数提取

$$V_1 \to V_2 \to \cdots \to V_\infty = V_* \to \pi_*$$

- 我们也可以迭代策略，策略收敛到最优策略 (策略迭代)

$$\pi_1 \to \pi_2 \to \cdots \to \pi_\infty = \pi_*$$

- 价值迭代：迭代价值函数，最优策略从收敛的价值函数提取

$$V_1 \to V_2 \to \cdots \to V_\infty = V_* \to \pi_*$$

- 我们也可以迭代策略，策略收敛到最优策略（策略迭代）

$$\pi_1 \to \pi_2 \to \cdots \to \pi_\infty = \pi_*$$

- 中间过程会对策略计算它的价值

$$\pi_1 \to (V_{\pi_1}) \to \pi_2 \to (V_{\pi_2}) \to \cdots \to \pi_*$$

回顾

- 策略 π 是状态到动作的一种分布

$$\pi(a|s) = \mathbb{P}[a_t = a | s_t = s]$$

- 策略的价值 $V_\pi(s)$ 定义为从状态 s 出发，在策略 π 作用下的期望回报

$$V_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$$

- 贝尔曼期望方程

$$V_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

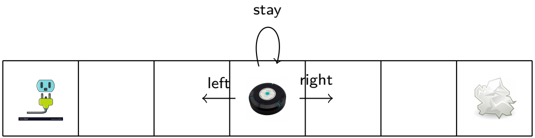
- 矩阵形式

$$\mathcal{V}_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi (\mathcal{V}_\pi)$$

- 方程的解

$$\mathcal{V}_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

举例：扫地机器人



$S_1 \quad S_2 \quad S_3 \quad S_4 \quad S_5 \quad S_6 \quad S_7$

- 随机策略：机器人在所有位置上以相同的概率向左或向右移动

$$\pi(left|s) = 0.5, \quad \pi(right|s) = 0.5$$

- $S_1 \rightarrow S_1$

$$\begin{aligned}\mathcal{P}^{\pi}_{S_1,S_1} &= \pi(left|S_1)\mathcal{P}^{left}_{S_1,S_1} + \pi(right|S_1)\mathcal{P}^{right}_{S_1,S_1} \\ &= 0.5 * 0.9 + 0.5 * 0.2 = 0.55\end{aligned}$$

- 在随机策略下，机器人的状态转移矩阵

$$
\mathcal{P}^{\pi} = \begin{bmatrix} \mathcal{P}^{\pi}_{S_1,S_1} & \cdots & \mathcal{P}^{\pi}_{S_1,S_n} \\ \vdots & \cdots & \vdots \\ \mathcal{P}^{\pi}_{S_n,S_1} & \cdots & \mathcal{P}^{\pi}_{S_n,S_n} \end{bmatrix}
= \begin{bmatrix} 0.55 & 0.45 & 0 & 0 & 0 & 0 & 0 \\ 0.45 & 0.1 & 0.45 & 0 & 0 & 0 & 0 \\ 0 & 0.45 & 0.1 & 0.45 & 0 & 0 & 0 \\ 0 & 0 & 0.45 & 0.1 & 0.45 & 0 & 0 \\ 0 & 0 & 0 & 0.45 & 0.1 & 0.45 & 0 \\ 0 & 0 & 0 & 0 & 0.45 & 0.1 & 0.45 \\ 0 & 0 & 0 & 0 & 0 & 0.45 & 0.55 \end{bmatrix}
$$

- 奖励: $\mathcal{R}(S_1) = +1, \mathcal{R}(S_7) = +10, \mathcal{R}(S_2) = \cdots = \mathcal{R}(S_6) = 0$
- $\mathcal{R}^\pi = [1, 0, 0, 0, 0, 0, 10]^T$
- $\gamma = 0.7$

- 随机策略的价值函数

$$\mathcal{V}_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi = \begin{bmatrix} 2.1322 \\ 0.9883 \\ 0.7856 \\ 1.3311 \\ 3.1443 \\ 7.9520 \\ 20.3332 \end{bmatrix}$$

提升策略

- 最优策略是由 V_* 根据如下公式提取

$$\pi_*(s) = \mathcal{G}(V_*) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)$$

提升策略

- 最优策略是由 V_* 根据如下公式提取

$$\pi_*(s) = \mathcal{G}(V_*) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s')\right)$$

- 同样的思路，从已知策略的价值 V_π 提取出贪心策略

$$\pi'(s) = \mathcal{G}(V_\pi) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s')\right)$$

策略迭代

给定一个初始策略 π_1，$k = 1$

loop

策略评估 policy evaluation：对当前策略 π_k 计算它的价值函数 V_{π_k}

$$V_{\pi_k}(s) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \ldots | s_t = s, a_t \sim \pi_k(s_t)]$$

$$= \sum_a \pi_k(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_{\pi_k}(s') \right)$$

策略提升 policy improvement：根据 V_{π_k} 提取贪心策略

$$\pi_{k+1}(s) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_{\pi_k}(s') \right)$$

$k \leftarrow k + 1$

end loop

策略迭代收敛性

- 考虑确定性的策略 $a = \pi(s)$
- 从已知的策略价值 $V_\pi(s)$ 提取贪心策略

$$\begin{aligned}\pi'(s) &= \arg\max_{a\in\mathcal{A}}(\mathcal{R}_s^a + \gamma\sum_{s'}\mathcal{P}_{ss'}^a V_\pi(s'))\\ &= \arg\max_{a\in\mathcal{A}} Q_\pi(s,a)\end{aligned}$$

- 上述过程提升了 s 的一步期望回报

$$Q_\pi(s,\pi'(s)) = \max_{a\in\mathcal{A}} Q_\pi(s,a) \geq Q_\pi(s,\pi(s)) = V_\pi(s)$$

- 展开得到

$$\begin{aligned}V_\pi(s_t) &\leq Q_\pi(s_t,\pi'(s_t))\\ &= \mathbb{E}[r_{t+1} + \gamma V_\pi(s_{t+1})|a_t = \pi'(s_t), a_k = \pi(s_k), k > t]\\ &= \mathcal{R}(s_t,\pi'(s_t)) + \gamma\sum_{s_{t+1}}\mathcal{P}_{s_t s_{t+1}}^{\pi'} V_\pi(s_{t+1})\end{aligned}$$

其中 $\mathcal{P}_{s_t,s_{t+1}}^{\pi'} = \mathbb{P}[s_{t+1}|s_t, a_t = \pi'(s_t)]$

- 继续对 $V_\pi(s_{t+1})$ 重复策略提升

$$
\begin{aligned}
V_\pi(s_t) &\le \mathcal{R}(s_t, \pi'(s_t)) + \gamma \sum_{s_{t+1}} \mathcal{P}^{\pi'}_{s_t s_{t+1}} V_\pi(s_{t+1}) \\
&\le \mathcal{R}(s_t, \pi'(s_t)) + \gamma \sum_{s_{t+1}} \mathcal{P}^{\pi'}_{s_t s_{t+1}} \Big[\mathcal{R}(s_{t+1}, \pi'(s_{t+1})) \\
&\qquad\qquad\qquad\qquad\qquad + \gamma \sum_{s_{t+2}} \mathcal{P}^{\pi'}_{s_{t+1} s_{t+2}} V_\pi(s_{t+2})\Big] \\
&\ \ \vdots \\
&\le \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \cdots | a_k = \pi'(s_k), k \ge t] \\
&= V_{\pi'}(s_t)
\end{aligned}
$$

- 新的策略优于旧的策略
- 上述结论同样适用于随机策略

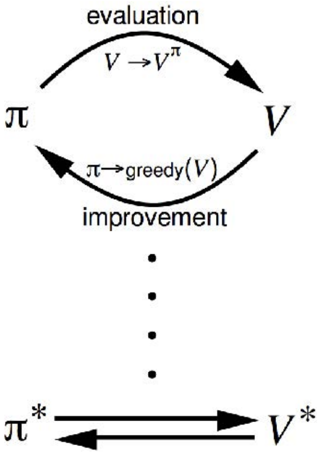
- $V_{\pi_1} \le V_{\pi_2} \le \cdots \le V_{\pi_k} \le V_*$, 单调递增有上界，所以收敛
- 当策略不再变化时

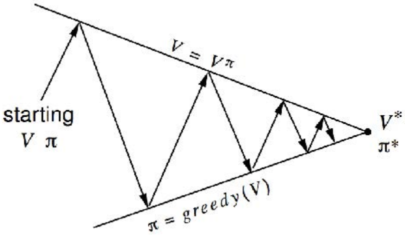
$$\pi_\infty(s) = \arg\max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a V_{\pi_\infty}(s')\right)$$

- 满足贝尔曼最优方程

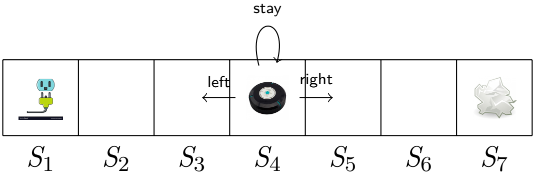
$$\begin{aligned} V_{\pi_\infty}(s) &= \sum_a \pi_\infty(a|s)\left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_{\pi_\infty}(s')\right) \\ &= \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_{\pi_\infty}(s')\right) \end{aligned}$$

- 因此 $V_{\pi_\infty} = V_*, \pi_\infty = \pi_*$


evaluation
$V \to V^\pi$
π
V
π→greedy(V)
improvement
π*
V*


starting
V π
V = Vπ
π = greedy(V)
V*
π*

举例：扫地机器人



1. $\pi_1(left|s) = 0.5, \quad \pi_1(right|s) = 0.5$
2. $V_{\pi_1} = [2.132, 0.988, 0.786, 1.331, 3.144, 7.952, 20.333]^T$
3. $\pi_2(s) = \arg\max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a V_{\pi_1}(s')\right)$

$$\pi_2(S_1) = \arg\max \begin{cases} 1 + 0.7\,(0.9 * 2.1322 + 0.1 * 0.9883) & a = \mathit{left} \\ 1 + 0.7\,(0.2 * 2.1322 + 0.8 * 0.9883) & a = \mathit{right} \end{cases}$$

$$\pi_2(S_2) = \arg\max \begin{cases} 0 + 0.7\,(0.8 * 2.1322 + 0.1 * 0.9883 + 0.1 * 0.7856) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 2.1322 + 0.1 * 0.9883 + 0.8 * 0.7856) & a = \mathit{right} \end{cases}$$

$$\pi_2(S_3) = \arg\max \begin{cases} 0 + 0.7\,(0.8 * 0.9883 + 0.1 * 0.7856 + 0.1 * 1.3311) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 0.9883 + 0.1 * 0.7856 + 0.8 * 1.3311) & a = \mathit{right} \end{cases}$$

$$\pi_2(S_4) = \arg\max \begin{cases} 0 + 0.7\,(0.8 * 0.7856 + 0.1 * 1.3311 + 0.1 * 3.1443) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 0.7856 + 0.1 * 1.3311 + 0.8 * 3.1443) & a = \mathit{right} \end{cases}$$

$$\pi_2(S_5) = \arg\max \begin{cases} 0 + 0.7\,(0.8 * 1.3311 + 0.1 * 3.1443 + 0.1 * 7.9520) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 1.3311 + 0.1 * 3.1443 + 0.8 * 7.9520) & a = \mathit{right} \end{cases}$$

$$\pi_2(S_6) = \arg\max \begin{cases} 0 + 0.7\,(0.8 * 3.1443 + 0.1 * 7.9520 + 0.1 * 20.3332) & a = \mathit{left} \\ 0 + 0.7\,(0.1 * 3.1443 + 0.1 * 7.9520 + 0.8 * 20.3332) & a = \mathit{right} \end{cases}$$

$$\pi_2(S_7) = \arg\max \begin{cases} 10 + 0.7\,(0.8 * 7.9520 + 0.2 * 20.3332) & a = \mathit{left} \\ 10 + 0.7\,(0.1 * 7.9520 + 0.9 * 20.3332) & a = \mathit{right} \end{cases}$$

策略迭代结果

π_k	1	2	3	4
S_1	left(0.5)/right(0.5)	left	left	left
S_2	left(0.5)/right(0.5)	left	right	right
S_3	left(0.5)/right(0.5)	right	right	right
S_4	left(0.5)/right(0.5)	right	right	right
S_5	left(0.5)/right(0.5)	right	right	right
S_6	left(0.5)/right(0.5)	right	right	right
S_7	left(0.5)/right(0.5)	right	right	right
V_{π_k}	1	2	3	
S_1	2.1322	3.1279	3.3096	
S_2	0.9883	2.2476	3.2078	
S_3	0.7856	4.8376	4.9135	
S_4	1.3311	7.7529	7.7589	
S_5	3.1443	12.2707	12.2712	
S_6	7.9520	19.4090	19.4091	
S_7	20.3332	30.6990	30.6990	

价值迭代结果

价值函数和对应的贪心策略

V_k	1	2	3	4	5	6	7	8	9	...	27
S_1	0	1	1.63	2.066	2.368	2.579	2.727	2.831	2.908		3.307
S_2	0	0	0.56	0.952	1.246	1.452	1.624	1.781	1.905		3.205
S_3	0	0	0	0.314	0.555	1.122	2.012	2.775	3.361		4.911
S_4	0	0	0	0	1.778	3.272	4.501	5.432	6.112		7.756
S_5	0	0	0	3.136	5.551	7.488	8.886	9.888	10.598		12.268
S_6	0	0	5.6	9.52	12.456	14.523	15.984	17.010	17.729		19.406
S_7	0	0	16.3	20.661	23.683	25.792	27.266	28.296	29.017		30.696
π_k	1	2	3	4	5	6	7	8	9	...	27
S_1	right	left	left	left	left	left	left	left	left		left
S_2	right	left	left	left	left	left	left	left	right		right
S_3	right	right	left	left	right	right	right	right	right		right
S_4	right	right	right	right	right	right	right	right	right		right
S_5	right	right	right	right	right	right	right	right	right		right
S_6	right	right	right	right	right	right	right	right	right		right
S_7	right	right	right	right	right	right	right	right	right		right

价值迭代 vs 策略迭代

- 价值迭代

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

- 策略迭代

$$V_\pi(s) = \sum_a \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

$$\pi'(s) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

价值迭代 vs 策略迭代

- 价值迭代

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

  - 只在收敛得到 V_* 后计算 π^*，中间过程不产生策略

- 策略迭代

$$V_\pi(s) = \sum_a \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

$$\pi'(s) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

  - 每次迭代开始时给定一个 π，结束时产生一个新 π'

价值迭代 vs 策略迭代

- 价值迭代

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

  - 只在收敛得到 V_* 后计算 π^*，中间过程不产生策略
  - 涉及赋值操作，计算量小，$\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$

- 策略迭代

$$V_\pi(s) = \sum_a \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

$$\pi'(s) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

  - 每次迭代开始时给定一个 π，结束时产生一个新 π'
  - 求解方程，计算量大，矩阵求逆 $\mathcal{O}(|\mathcal{S}|^3)$，策略提升 $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$

价值迭代 vs 策略迭代

- 价值迭代

$$V_{k+1}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_k(s') \right)$$

  - 只在收敛得到 V_* 后计算 π^*，中间过程不产生策略
  - 涉及赋值操作，计算量小，$\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$
  - 通常迭代次数多

- 策略迭代

$$V_\pi(s) = \sum_a \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

$$\pi'(s) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s') \right)$$

  - 每次迭代开始时给定一个 π，结束时产生一个新 π'
  - 求解方程，计算量大，矩阵求逆 $\mathcal{O}(|\mathcal{S}|^3)$，策略提升 $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$
  - 通常迭代次数少

价值迭代次数上限

$$\| V_k - V_* \|_\infty \le \gamma^{k-1} \| V_1 - V_* \|_\infty$$

- 假设初始 $V_1(s) = 0, \forall s \in \mathcal{S}$
- 收敛阈值 ϵ，即 $\| V_k - V_* \|_\infty \le \epsilon$ 时认为收敛到 V_*

$$\gamma^{k-1} \| V_1 - V_* \|_\infty \le \epsilon$$

$$\Rightarrow \boxed{k \ge 1 + \log_\gamma \frac{\epsilon}{\| V_* \|_\infty}}$$

- e.g. $\max V_*(s) = 10, \gamma = 0.95, \epsilon = 0.001,\ k \ge 180.56$

策略迭代次数上限

- 策略迭代收敛时策略不再变化
- 迭代的次数不超过 MDP 的最大 (确定性) 策略数量

$$\pi_1 \to \pi_2 \to \cdots \to \pi_k = \pi_*$$

$$k \leq |\mathcal{A}|^{|\mathcal{S}|}$$

  - e.g. $|\mathcal{A}| = 2, |\mathcal{S}| = 10,\ k \leq 2^{10} = 1024$

策略迭代次数上限

- 策略迭代收敛时策略不再变化
- 迭代的次数不超过 MDP 的最大 (确定性) 策略数量

$$\pi_1 \to \pi_2 \to \cdots \to \pi_k = \pi_*$$

$$k \leq |\mathcal{A}|^{|\mathcal{S}|}$$

  - e.g. $|\mathcal{A}| = 2, |\mathcal{S}| = 10,\ k \leq 2^{10} = 1024$

- 思考：策略迭代过程会出现死循环吗？即 $\pi_i \to \pi_{i+1} \to \cdots \to \pi_j = \pi_i$

动态规划总结

$$V_*(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_*(s') \right)$$

- 动态规划：价值迭代/策略迭代
  - 通过迭代有效求解原始问题中非线性 max 算子
  - 依赖系统模型 $\mathcal{R}$, $\mathcal{P}$
  - 足够的计算空间记录每个状态的函数值
    - 价值迭代 $V(s)$
    - 策略迭代 $V_\pi(s)$, $\pi(s)$

- 同样可以将动态规划用于基于动作-价值函数的贝尔曼最优方程

$$Q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \max_{a'} Q_*(s',a')$$

- 基于 Q 函数的价值迭代
  - 初始化 Q_1, (e.g. $Q_1(s,a) = 0, \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$)
  - 迭代计算新的 Q 函数

$$Q_{k+1}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \max_{a'} Q_k(s',a')$$

- 同样可以将动态规划用于基于动作 -价值函数的贝尔曼最优方程

$$Q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a \max_{a'} Q_*(s',a')$$

- 基于 Q 函数的策略迭代
  - 给定一个策略 π_1
  - **策略评估**: 计算 π_k 的动作 -价值函数

$$Q_{\pi_k}(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi_k(a'|s') Q_{\pi_k}(s',a')$$

  - **策略提升**: 根据 Q_{π_k} 提取出新的策略

$$\pi_{k+1}(s) = \arg\max_{a \in \mathcal{A}} Q_{\pi_k}(s,a)$$

迭代策略评估

策略评估

- 计算给定策略的价值，求解贝尔曼期望方程

$$V_\pi(s) = \sum_a \pi(a|s)\left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s')\right)$$

- 矩阵形式求解

$$\mathcal{V}_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

- 不足：矩阵求逆，$\mathcal{O}(|\mathcal{S}|^3)$
  1. 矩阵很大：计算量大
  2. 矩阵稀疏：结果不一定准确

策略评估

- 计算给定策略的价值，求解贝尔曼期望方程

$$V_\pi(s) = \sum_a \pi(a|s)\left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a V_\pi(s')\right)$$

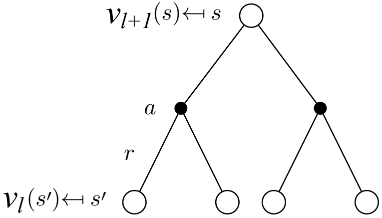
- 矩阵形式求解

$$\mathcal{V}_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi$$

- 不足：矩阵求逆，$\mathcal{O}(|\mathcal{S}|^3)$
  1. 矩阵很大：计算量大
  2. 矩阵稀疏：结果不一定准确
- 满足动态规划条件
  - 原问题包含子问题
  - 子问题重复出现
- 可以使用迭代方法求解

迭代策略评估

- 初始化一个价值函数 V_1



$$V_{l+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V_l(s') \right)$$

- 收敛到真实价值函数 V_π

- 定义贝尔曼期望算子 $\mathcal{T}^\pi$

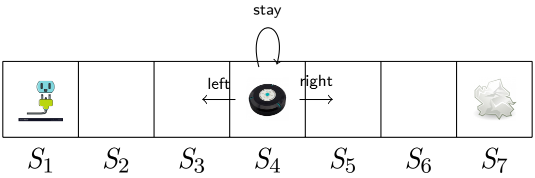
$$\mathcal{T}^\pi(v) = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v$$

- 该算子是 γ-收缩的，经过该算子两个函数的距离变为原来的 γ 倍

$$\begin{aligned}
\|\mathcal{T}^\pi(u) - \mathcal{T}^\pi(v)\|_\infty &= \|(\mathcal{R}^\pi + \gamma \mathcal{P}^\pi u) - (\mathcal{R}^\pi + \gamma \mathcal{P}^\pi v)\|_\infty \\
&= \|\gamma \mathcal{P}^\pi (u - v)\|_\infty \\
&\leq \|\gamma \mathcal{P}^\pi \| u - v\|_\infty\|_\infty \\
&\leq \gamma \|u - v\|_\infty
\end{aligned}$$

- 所以迭代策略评估收敛到 V_π

举例：扫地机器人



- 机器人在所有位置上以相同的概率向左或向右移动

$$\pi(left|s) = 0.5, \quad \pi(right|s) = 0.5$$

- 矩阵求解价值函数

$$V_\pi = [2.1322, 0.9883, 0.7856, 1.3311, 3.1443, 7.9520, 20.3332]^T$$

迭代策略评估计算结果

V_l	S_1	S_2	S_3	S_4	S_5	S_6	S_7
$l=1$	0	0	0	0	0	0	0
$l=2$	1	0	0	0	0	0	10
$l=3$	1.385	0.315	0	0	0	3.15	13.85
$l=4$	1.6324	0.4583	0.0992	0	0.9922	4.5832	16.3245
$l=5$	1.7729	0.5776	0.1513	0.3438	1.5132	5.7756	17.7287
$l=6$	1.8645	0.6465	0.3008	0.5484	2.0335	6.4655	18.6448
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$l=22$	2.1298	0.9858	0.7829	1.3282	3.1411	7.9487	20.3297
$l=23$	2.1305	0.9865	0.7837	1.3290	3.1421	7.9497	20.3308
v_π	2.1322	0.9883	0.7856	1.3311	3.1443	7.9520	20.3332

- $|V_l - V_\pi|$ 不断减小，但误差一直存在
- 当误差小于一定阈值后认为收敛

广义策略迭代

- 考虑使用迭代策略评估的策略迭代过程
  - 策略提升：已知策略价值 V_{π_k}，提取贪心策略 π_{k+1}

$$\pi_{k+1}(s) = \arg\max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a V_{\pi_k}(s')\right)$$

$$\boxed{\pi_{k+1} = \mathcal{G}(V_{\pi_k})}$$

  - 迭代策略评估：以 $V_0 = V_{\pi_k}$ 作为初始价值函数，做无限次迭代策略评估，计算 π_{k+1} 的价值 $V_{\pi_{k+1}}$

$$V_{l+1}(s) = \sum_{a\in\mathcal{A}} \pi_{k+1}(a|s)\left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a V_l(s')\right)$$

$$\boxed{V_{\pi_{k+1}} = \left(\mathcal{T}^{\pi_{k+1}}\right)^{\infty}(V_{\pi_k})}$$

- 价值迭代拆解成两个步骤:

$$V_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V_k(s') \right)$$

  - 根据 V_k 提取贪心策略

$$\boxed{\pi_{k+1} = \mathcal{G}(V_k)}$$

  - 对 V_k 做一次迭代策略评估 ($\mathcal{T}^{\pi_{k+1}}$ 算子)

$$\boxed{V_{k+1} = \mathcal{T}^{\pi_{k+1}}(V_k)}$$

- 价值迭代拆解成两个步骤:

$$V_{k+1}(s) = \max_{a \in \mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a V_k(s') \right)$$

  - 根据 V_k 提取贪心策略

$$\boxed{\pi_{k+1} = \mathcal{G}(V_k)}$$

  - 对 V_k 做一次迭代策略评估 ($\mathcal{T}^{\pi_{k+1}}$ 算子)

$$\boxed{V_{k+1} = \mathcal{T}^{\pi_{k+1}}(V_k)}$$

在无限次和一次之间，是否可以存在只做 n 次的迭代评估?

广义策略迭代 Generalized Policy Iteration

1: 给定一个价值函数 V_1，$k=1$

2: **loop**

3: 策略提升：根据 V_k 提取贪心策略

$$\pi_{k+1}(s) = \arg\max_{a\in\mathcal{A}} \left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a V_k(s')\right)$$

4: 迭代策略评估：令 $V_{k,1} = V_k$，做 n 次 $\mathcal{T}^{\pi_{k+1}}$ 迭代策略评估

for $l = 1, \ldots, n$

$$V_{k,l+1}(s) = \sum_{a\in\mathcal{A}} \pi_{k+1}(a|s)\left(\mathcal{R}_s^a + \gamma \sum_{s'\in\mathcal{S}} \mathcal{P}_{ss'}^a V_{k,l}(s')\right)$$

$$V_{k+1} = V_{k,n+1}$$

5: $k \leftarrow k+1$

6: **end loop**

GPI 收敛性

- $k \to \infty, V_k \to V_*$
- 但是 GPI 即不具有 γ-收缩性，也不满足单调性
- 理论分析比价值迭代和策略迭代更加复杂
- 具体收敛定理参考
    - B. Scherrer, M. Ghavamzadeh, V. Gabillon, et al. Approximate modified policy iteration, in Proceedings of the 29th International Coference on International Conference on Machine Learning. Omnipress, 2012, pp. 1889–1896.

维数灾

动态规划计算复杂度

- 价值迭代：寻找最优价值函数
  - $\mathcal{O}(\mathcal{S}^2\mathcal{A})$ per iteration
- 策略评估：固定策略，计算价值
  - $\mathcal{O}(\mathcal{S}^3)$
- 策略迭代：寻找最优策略 (策略评估 + 策略提升)
  - $\mathcal{O}(\mathcal{S}^3 + \mathcal{S}^2\mathcal{A})$ per iteration

维数灾

- 动态规划的计算复杂度与状态数量呈 多项式关系
- 但是许多问题拥有巨大的状态数量，e.g. 与状态维数呈指数型增长
- Bellman 称之为维数灾 “the curse of dimensionality”
- 通常情况，传统动态规划只能解决几百万个状态以下的问题
- 在未来的课程里我们会介绍如何解决大规划或连续空间的 MDPs 问题

价值迭代 $V_{k+1}(s) = \max_{a} \left(\boxed{\mathcal{R}_s^a} + \gamma \sum_{s'} \boxed{\mathcal{P}_{ss'}^a} V_k(s') \right)$

策略评估 $V_\pi(s) = \sum_{a} \pi(a|s) \left(\boxed{\mathcal{R}_s^a} + \gamma \sum_{s'} \boxed{\mathcal{P}_{ss'}^a} V_\pi(s') \right)$

贪心策略 $\pi'(s) = \arg\max_{a} \left(\boxed{\mathcal{R}_s^a} + \gamma \sum_{s'} \boxed{\mathcal{P}_{ss'}^a} V(s') \right)$

- 动态规划要求
  - 模型已知：奖励函数 $\mathcal{R}$ 和状态转移函数 $\mathcal{P}$
  - 更新繁琐：每次迭代即使是更新一个状态的价值也要<u>遍历整个</u>后继状态和动作空间

- 使用智能体从环境观测的 样本奖励和状态转移 $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$ 更新价值或策略
- 好处包括:
  - 无模型: 不需要预先 MDP 的模型信息
  - 只在样本点上更新, 避免更新整个状态空间, 解决维数灾问题
  - 每次更新的计算量是固定, 与后继的状态空间 $n = |S|$ 无关

总结

动态规划

价值迭代

策略迭代

迭代策略评估

广义策略迭代

维数灾