# 第 4 章 动态规划法

在强化学习中，动态规划法（dynamic programming，DP）主要用于求解有模型的 MDP 问题。尽管在现实任务中难以获得完备的环境模型，且动态规划需要消耗大量的计算资源，但是作为强化学习的基础，动态规划法仍然具有非常重要的理论意义。事实上，所有其他强化学习方法，如蒙特卡罗法（Monte Carlo，MC）、时序差分法（Temporal Difference，TD）等，都是对动态规划法的一种近似，只是在学习过程中不再需要完整的环境模型，而且在计算资源的消耗方面也可以大大减少。

动态规划算法主要包括基于模型的策略迭代（policy iteration）和基于模型的值迭代（value iteration）两种。这两种算法都是利用值函数来评价策略的，一旦计算出满足贝尔曼最优方程的最优状态值函数 $v_*$ 或最优动作值函数 $q_*$，就能得到最优策略。即利用 $v_*$ 并借助环境模型，或直接利用 $q_*$ 都可以得最优策略。

## 4.1 策略迭代

策略迭代通过构建策略的值函数（状态值函数 $v_\pi$ 或动作值函数 $q_\pi$）来评估当前策略，并利用这些值函数给出改进的新策略。策略迭代由策略评估（Policy Evaluation，PE）和策略改进（Policy Improvement，PI）两部分组成。

策略评估：每一次策略评估都是一个迭代过程，对于一个给定的策略 $\pi$，评估在该策略下，所有状态 $s$（或状态-动作对 $(s,a)$）的值函数 $v_\pi(s)$ (或 $q_\pi(s,a)$)。

策略改进：在策略评估的基础上，直接利用（或状态值函数通过一步搜索计算得出的）策略 $\pi$ 的动作值函数，然后通过贪心策略（或 $\varepsilon$-贪心策略）对策略进行改进。

在采用某一策略 $\pi$ 时，需要考虑该策略是否是最优策略。首先根据策略 $\pi$ 的值函数 $v_\pi$（或 $q_\pi$）产生一个更优策略 $\pi'$，再根据策略 $\pi'$ 的值函数 $v_{\pi'}$（或 $q_{\pi'}$）得到一个更优策略 $\pi''$，以此类推，通过这样的链式方法可以得到一个关于策略和值函数的更新序列，并且能够保证每一个新策略都比前一个策略更优（除非前一个策略已是最优策略）。在有限 MDP 中，策略有限，所以在多次迭代后，一定能收敛到最优策略和最优值函数。其链式关系为：

$$\pi_0 \xrightarrow{\text{PE}} v_{\pi_0} \xrightarrow{\text{PI}} \pi_1 \xrightarrow{\text{PE}} v_{\pi_1} \xrightarrow{\text{PI}} \dots \xrightarrow{\text{PI}} \pi_* \xrightarrow{\text{PE}} v_*$$

其中，策略 $\pi$ 的下标 $0,1,2,\dots$ 表示迭代更新的次序。

### 4.1.1 策略评估

#### 4.1.1.1 基于状态值函数的策略评估

由第 3 章可知，对策略 $\pi$ 进行评价就是估计在该策略下的状态值函数 $v_\pi$，这在动态规划法中称为策略评估。在环境完备的情况下，状态值函数的贝尔曼方程（3.17）就是一个包含有 $|\mathcal{S}|$ 个未知数的 $|\mathcal{S}|$ 元方程组。理论上该方程组可以直接求解，但是当状态较多、甚至无

穷多时，很难使用解方程组的方法来解决强化学习问题，因此引入迭代方法来简化求解过程。

根据迭代思想，可以将状态值函数的贝尔曼方程（3.17）转化为迭代式（4.1），即基于状态值函数的策略评估迭代式：

$$\begin{aligned} v_\tau(s) &= \mathbb{E}_\pi\left(R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right) \\ &= \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_{\tau-1}(s')\right] \end{aligned} \tag{4.1}$$

初始值 $v_0$ 可以为任意值（通常设为 0）。在使用 $v_\tau(s)$ 这样的迭代式时，默认采用同一策略 $\pi$，为了突出迭代关系，省略 $v_\pi(s)$ 的下标 $\pi$。

策略评估对每个状态 $s$ 都采用相同的操作：根据给定的策略（动作分布），采取可能的动作，得到单步转移后的所有状态 $s'$ 和奖励 $r$，并利用下一状态 $s'$ 的值函数 $v_{\tau-1}(s')$，通过分布的期望值，更新状态 $s$ 的值函数 $v_\tau(s)$。该方法称为期望更新法（expected update）。另外，从式（4.1）可以看出，对状态 $s$ 值函数的估算是建立在所有可能下一状态 $s'$ 的值函数估计 $v_{\tau-1}(s')$ 之上的，这种更新方式也称为自举方法（bootstrapping）。

在保证 $v_\pi$ 存在的前提下，当 $\tau \to \infty$ 时，序列 $\{v_\tau\}$ 将会收敛到 $v_\pi$。在实际情况下，只有更新后的状态值基本不变，才可以停止迭代。通常可以采取以下两种方式提前结束迭代：

（1）直接设置迭代次数。只要达到预期的迭代次数，即可停止迭代；

（2）设定较小的阈值（次优界限）$\theta$。当 $|v_\tau(s) - v_{\tau-1}(s)|_\infty < \theta$ 时，停止迭代。比较两次迭代的状态值函数差的绝对值 $\Delta$（或差的平方），当 $\Delta$ 最大值小于阈值时，终止迭代。

算法 4.1 给出了在有穷状态空间 MDP 中，基于状态值函数的策略评估算法。

**算法 4.1** 基于状态值函数的策略评估算法

| | |
|---|---|
| **输　入** | 初始策略 $\pi(a \mid s)$，动态性 $p$，奖赏函数 $r$，折扣因子 $\gamma$ |
| **初始化** | 1. 对任意 $s \in \mathcal{S}$，初始化状态值函数，如 $V(s) = 0$<br>2. 阈值 $\theta$ 设置为一个较小的实数值，如 $\theta = 0.01$ |
| **策　略<br>评　估** | 4. **repeat** 对每一轮策略评估 $\tau = 1, 2, \cdots$<br>5. $delta \leftarrow 0$<br>6. **for** 每个状态 $s$ **do**<br>7. $v \leftarrow V(s)$<br>8. $V(s) \leftarrow \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma V(s')\right]$<br>9. $delta \leftarrow \max(delta, \mid v - V(s) \mid)$<br>10. **end for**<br>11. **until** $delta < \theta$ |
| **输　出** | $v_\pi = V$ |

算法 4.1 使用的迭代方式是异步计算方式，即在相邻的两个迭代轮次 $\tau$ 和 $\tau - 1$，保存同一组状态值函数 $V(s)$。在 $V(s)$ 中，存储两轮混合的函数值。因此在每次计算中，如果状

态 $s$ 的值函数已被更新，那么当用到 $V(s)$ 时，就使用已经更新过的数据。评估过程中，中间结果与状态评估的先后次序密切相关。另外一种迭代方式是同步计算方式，即在每一迭代轮次 $\tau$，都保存相邻两轮的状态值函数：$V_\tau(s)$ 和 $V_{\tau-1}(s)$。在计算 $V_\tau(s)$ 过程中，使用的全部是上一轮的 $V_{\tau-1}(s)$ 值。评估过程中，中间结果与状态评估的先后次序无关。在相同情况下，利用两种迭代方式评估，收敛后结果是相同的。但收敛速度方面相比较，通常异步计算方式收敛速度会更快。

另外，算法 4.1 可用于有穷状态空间的确定 MDP 问题和随机 MDP 问题，这具体体现在 MDP 的状态转移动态 $p(s',r\mid s,a)$ 上。对于确定 MDP 问题，在当前状态 $s$ 下采取动作 $a$，到达下一状态 $s'$ 的概率为 1，而到达其他状态的概率均为 0。对于随机 MDP 问题，在当前状态 $s$ 下采取动作 $a$，到达下一状态 $s'$ 是随机的。因此利用算法 4.1 解决确定 MDP 问题和随机 MDP 时，只是状态转移动态的变化，而算法本身不需要改变。这也体现出确定 MDP 是随机 MDP 的特例。

下面利用算法 4.1，分别来解决例 3.1 的确定环境扫地机器人任务和例 3.2 的随机环境扫地机器人任务。

**例 4.1** 对例 3.1 的确定环境扫地机器人任务进行策略评估。如图 4.1 所示，设定策略评估的起始位置为左下角，即充电桩位置，按序号顺序进行评估。机器人在非终止状态（除位置 0、12、19）均采取等概率策略 $\pi(a\mid s)=1/|\mathcal{A}(s)|$，$a\in\mathcal{A}(s)$，$|\mathcal{A}(s)|$ 为当前状态 $s$ 可采取的动作个数。扫地机器人最多可以采取 {*Up*, *Down*, *Left*, *Right*} 4 个动作。到达充电桩位置时，$r=+1$；到达垃圾位置时，$r=+3$；撞到障碍物时，$r=-10$；其他情况，获得的奖赏 $r$ 均为 0。折扣系数 $\gamma=0.8$，利用算法 4.1 计算，在策略 $\pi$ 下，确定环境扫地机器人任务的状态值函数。

| | | | | |
|---|---|---|---|---|
| 20 | 21 | 22 | 23 | 24 |
| 15 | 16 | 17 | 18 | 19 |
| 10 | 11 | **12** | 13 | 14 |
| 5 | 6 | 7 | 8 | 9 |
| 0 | 1 | 2 | 3 | 4 |

图 4.1 扫地机器人环境

根据算法 4.1，在随机策略 $\pi(a\mid s)$ 下，确定环境扫地机器人任务的评估过程如下：

（1）当 $\tau=0$ 时（$\tau$ 为迭代的次数），对于所有的 $s$ 初始化为：$V(s)=0$；

（2）当 $\tau=1$ 时，以状态 $S_{24}$ 为例。在策略 $\pi$ 下，只能采取向下和向左 2 个动作，概率

各为 0.5。采取向下的动作时，到达状态 $S_{19}\left(V(S_{19})=0\right)$，并可以捡到垃圾，获得 $r_1=+3$ 的奖赏；采取向左的动作时，到达状态 $S_{23}\left(V(S_{23})=-0.27\right)$，获得 $r_2=0$ 的奖赏。根据算法 4.1 计算得：

$$\begin{aligned}V(S_{24}) &= 0.5*1*\left(r_1+\gamma V(S_{19})\right)+0.5*1*\left(r_2+\gamma V(S_{23})\right)\\ &= 0.5*1*(3+0.8*0)+0.5*1*(0+0.8*(-0.27));\\ &\approx 1.39\end{aligned}$$

同理可以计算状态 $S_5$、$S_{17}$ 的状态值函数：

$$\begin{aligned}V(S_5) &= \tfrac{1}{3}*1*\left(r_1+\gamma V(S_{10})\right)+\tfrac{1}{3}*1*\left(r_2+\gamma V(S_0)\right)+\tfrac{1}{3}*1*\left(r_3+\gamma V(S_6)\right)\\ &= \tfrac{1}{3}*1*(0+0)+\tfrac{1}{3}*1*(1+0)+\tfrac{1}{3}*1*(0+0) \quad ;\\ &\approx 0.33\end{aligned}$$

$$\begin{aligned}V(S_{17}) &= \tfrac{1}{4}*1*\left(r_1+\gamma V(S_{22})\right)+\tfrac{1}{4}*1*\left(r_2+\gamma V(S_{17})\right)\\ &\quad+\tfrac{1}{4}*1*\left(r_3+\gamma V(S_{16})\right)+\tfrac{1}{4}*1*\left(r_4+\gamma V(S_{18})\right)\\ &= 0.25*(0+0)+0.25*(-10+0)+0.25*(0+(-0.49))+0.25*(0+0) \quad ;\\ &\approx -2.60\end{aligned}$$

按顺序计算完一轮后，得到值函数 $V(s)$，如图 4.2（$\tau=1$）所示。

（3）当 $\tau=2$ 时，仍然以状态 $S_5$、$S_{17}$、$S_{24}$ 为例。采用异步计算方式，状态值函数的计算，通常与迭代计算顺序有关。这里策略评估的起始位置为左下角，即充电桩位置，按序号顺序进行评估。那么每一轮次中，这 3 个状态的计算顺序为 $S_5$、$S_{17}$、$S_{24}$。3 个状态值函数分别计算为：

$$\begin{aligned}V(S_5) &= \tfrac{1}{3}*1*\left(r_1+\gamma V(S_{10})\right)+\tfrac{1}{3}*1*\left(r_2+\gamma V(S_0)\right)+\tfrac{1}{3}*1*\left(r_3+\gamma V(S_6)\right)\\ &= \tfrac{1}{3}*(0+0.8*0.09)+\tfrac{1}{3}*1*(0+0.8*0)+\tfrac{1}{3}*(0+0.8*0.13) \quad ;\\ &\approx 0.39\end{aligned}$$

$$\begin{aligned}V(S_{17}) &= \tfrac{1}{4}*1*\left(r_1+\gamma V(S_{22})\right)+\tfrac{1}{4}*1*\left(r_2+\gamma V(S_{17})\right)\\ &\quad+\tfrac{1}{4}*1*\left(r_3+\gamma V(S_{16})\right)+\tfrac{1}{4}*1*\left(r_4+\gamma V(S_{18})\right)\\ &= 0.25*(0+0.8*(-0.73))+0.25*(-10+0.8*(-2.60));\\ &\quad+0.25*(0+0.8*(-1.27))+0.25*(0+0.8*(-2.29))\\ &\approx -3.48\end{aligned}$$

$$\begin{aligned}V(S_{24}) &= \tfrac{1}{2}*1*\left(r_1+\gamma V(S_{19})\right)+\tfrac{1}{2}*1*\left(r_2+\gamma V(S_{23})\right)\\ &= 0.5*(3+0.8*0)+0.5*(0+0.8*(-0.11));\\ &= 1.46\end{aligned}$$

按顺序计算完一轮后，得到状态值函数 $V(s)$，如图 4.2（$\tau=2$）所示。

（4） 当 $\tau=30$ 时，$|V_\tau(s)-V_{\tau-1}(s)|_\infty<\theta$，认为 $V_\tau(s)$ 已经收敛于 $v_\pi(s)$，计算得到的 $v_\pi(s)$ 就是在策略 $\pi$ 下的有效评估。

随着迭代的进行，每轮状态值函数的更新过程，如图 4.2 所示。

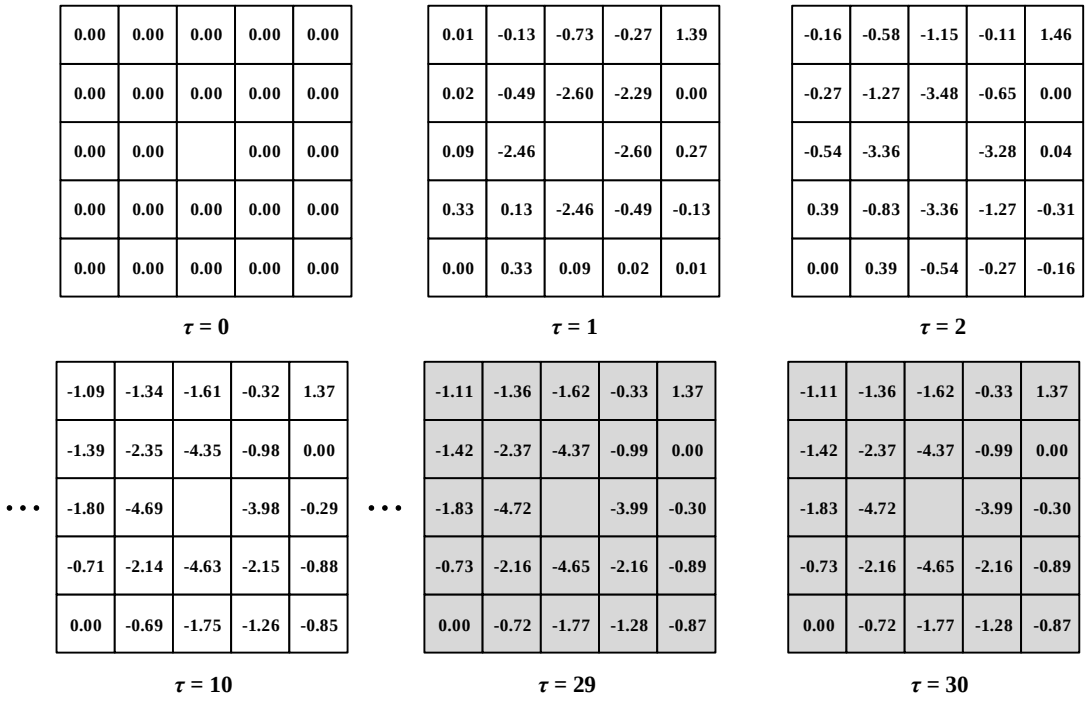



图 4.2 面向确定环境扫地机器人任务的状态值函数 $v_\pi$ 的评估过程（异步计算方式）

对于例 4.1 的确定情况扫地机器人任务，策略评估时通常采用异步计算方式。而同步计算方式评估值函数时，环境状态越多，效率也越低。采用异步计算方式时，每一轮迭代都直接用新产生的值函数来替换旧的值函数，不需要对上一轮迭代的状态值函数进行备份，既减少了迭代次数，又节省了存储空间。对于该扫地机器人任务，采用异步计算方式进行评估，30 轮迭代后既可以收敛到 $v_\pi$，而采用同步计算方式，收敛到 $v_\pi$ 则需要 51 轮迭代。另外，使用异步计算方式时，每次遍历并不需要对所有的状态值函数都做一次更新，而可以任意顺序更新状态值，这样其中的某些状态值可能会在其他状态值更新一次之前已经更新过多次。这种基于异步计算方式，以任意顺序进行规划的方法，称为异步动态规划法（Asynchronous Dynamic Progarmming，ADP）。ADP 的特点归纳如下：

（1）ADP 可以对更新顺序进行调整，通常重要的状态优先更新；

（2）在实际情况中，ADP 必须保证完成所有状态的价值更新；

（3）ADP 并不一定能减少计算量。但该方法的作用在于：算法在改进策略之前不需要陷入无望的长时间扫描。

**例 4.2** 对例 3.2 的随机环境下扫地机器人任务进行策略评估。重新考虑例 3.1 的扫地机器人问题。假设由于地面的问题，采取某一动作后，状态转换不再确定。当采取某一动作试图向某一方向移动时，机器人成功移动的概率为 0.80，保持原地不动的概率为 0.15，移动到相反方向的概率为 0.05，具体如图 4.3 所示，其中 $s'$ 表示下一个状态，$s''$ 表示相反方向的状态。

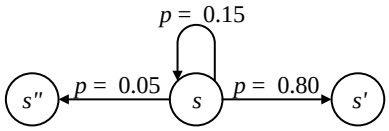



图 4.3 随机环境扫地机器人状态转移示意图

在其他条件与例 4.1 不变的情况下，采用异步计算方式，给出策略 $\pi$ 下的随机环境扫地机器人任务的状态值函数更新过程。

根据算法 4.1，在随机策略 $\pi(a|s)$ 下，随机环境扫地机器人任务的评估过程如下：

（1）当 $\tau=0$ 时（ $\tau$ 是迭代的次数），对于所有的 $s$ 初始化为： $v(s)=0$；

（2）当 $\tau=1$ 时，以状态 $S_{24}$ 为例。在策略 $\pi$ 下，只能采取向下和向左 2 个动作，概率各为 0.5。采取向下的动作时，以 0.8 的概率到达状态 $S_{19}\left(V(S_{19})=0\right)$，并可以捡到垃圾，获得 $r_{11}=+3$ 的奖赏；以 0.15 的概率保持原地 $S_{24}\left(V(S_{24})=0\right)$ 不动，获得 $r_{12}=0$ 的奖赏；以 0.05 的概率采取冲出边界的动作，保持原地 $S_{24}\left(V(S_{24})=0\right)$ 不动，获得 $r_{13}=0$ 的奖赏。采取向左的动作时，以 0.8 的概率到达状态 $S_{23}\left(V(S_{23})=?\right)$，获得 $r_{21}=0$ 的奖赏；以 0.15 的概率保持原地 $S_{24}\left(V(S_{24})=0\right)$ 不动，获得 $r_{22}=0$ 的奖赏；以 0.05 的概率采取冲出边界的动作，保持原地 $S_{24}\left(V(S_{24})=0\right)$ 不动，获得 $r_{23}=0$ 的奖赏。根据算法 4.1 计算得：

$$
\begin{aligned}
V(S_{24}) &= 0.5*\left[0.8*\left(r_{11}+\gamma V(S_{19})\right)+0.15*\left(r_{12}+\gamma V(S_{24})\right)+0.05*\left(r_{13}+\gamma V(S_{24})\right)\right] \\
&\quad +0.5*\left[0.8*\left(r_{21}+\gamma V(S_{23})\right)+0.15*\left(r_{22}+\gamma V(S_{24})\right)+0.05*\left(r_{23}+\gamma V(S_{24})\right)\right] \\
&= 0.5*\left[0.8*(3+0.8*0)+0.15*(0+0.8*0)+0.05*(0+0.8*0)\right] \quad ; \\
&\quad +0.5*\left[0.8*(0+0.8*?)+0.15*(0+0.8*0)+0.05*(0+0.8*0)\right] \\
&= ??
\end{aligned}
$$

同理可以计算状态 $S_5$、 $S_{17}$ 的状态值函数：（下面的都改成异步计算方式随机环境的）

$$
\begin{aligned}
V(S_5) &= \tfrac{1}{3}*\left[0.8*\left(r_{11}+\gamma V(S_{10})\right)+0.15*\left(r_{12}+\gamma V(S_5)\right)+0.05*\left(r_{13}+\gamma V(S_0)\right)\right] \\
&\quad +\tfrac{1}{3}*\left[0.8*\left(r_{21}+\gamma V(S_0)\right)+0.15*\left(r_{12}+\gamma V(S_5)\right)+0.05*\left(r_{13}+\gamma V(S_{10})\right)\right] \\
&\quad +\tfrac{1}{3}*\left[0.8*\left(r_{31}+\gamma V(S_6)\right)+0.15*\left(r_{32}+\gamma V(S_5)\right)+0.05*\left(r_{23}+\gamma V(S_5)\right)\right] ; \\
&= ? \ ? \\
&\approx ? \ ? \ ?
\end{aligned}
$$

$$
\begin{aligned}
V(S_{17}) &= \tfrac{1}{4}*\left[0.8*\left(r_{11}+\gamma V(S_{22})\right)+0.15*\left(r_{12}+\gamma V(S_{17})\right)+0.05*\left(r_{13}+\gamma V(S_{17})\right)\right] \\
&\quad +\tfrac{1}{4}*\left[0.8*\left(r_{21}+\gamma V(S_{17})\right)+0.15*\left(r_{12}+\gamma V(S_{17})\right)+0.05*\left(r_{13}+\gamma V(S_{22})\right)\right] \\
&\quad +\tfrac{1}{4}*\left[0.8*\left(r_{31}+\gamma V(S_{16})\right)+0.15*\left(r_{32}+\gamma V(S_{17})\right)+0.05*\left(r_{23}+\gamma V(S_{18})\right)\right] ; \\
&\quad +\tfrac{1}{4}*\left[0.8*\left(r_{41}+\gamma V(S_{18})\right)+0.15*\left(r_{42}+\gamma V(S_{17})\right)+0.05*\left(r_{23}+\gamma V(S_{16})\right)\right] \\
&= ? \ ? \\
&\approx ? \ ? \ ?
\end{aligned}
$$

按顺序计算完一轮后，得到值函数 $V(s)$，如图 4.4（ $\tau=1$ ）所示。

（3）当$\tau=2$时，仍然以状态$S_5$、$S_{17}$、$S_{24}$为例。在每一轮次中，这3个状态的计算顺序为$S_5$、$S_{17}$、$S_{24}$。3个状态值函数分别计算为：

$$\begin{aligned}V(S_5)=&\tfrac{1}{3}*\left[0.8*\left(r_{11}+\gamma V(S_{10})\right)+0.15*\left(r_{12}+\gamma V(S_5)\right)+0.05*\left(r_{13}+\gamma V(S_0)\right)\right]\\&+\tfrac{1}{3}*\left[0.8*\left(r_{21}+\gamma V(S_0)\right)+0.15*\left(r_{12}+\gamma V(S_5)\right)+0.05*\left(r_{13}+\gamma V(S_{10})\right)\right]\\&+\tfrac{1}{3}*\left[0.8*\left(r_{31}+\gamma V(S_6)\right)+0.15*\left(r_{32}+\gamma V(S_5)\right)+0.05*\left(r_{23}+\gamma V(S_5)\right)\right];\\&=??\\&\approx???\end{aligned}$$

$$\begin{aligned}V(S_{17})=&\tfrac{1}{4}*\left[0.8*\left(r_{11}+\gamma V(S_{22})\right)+0.15*\left(r_{12}+\gamma V(S_{17})\right)+0.05*\left(r_{13}+\gamma V(S_{17})\right)\right]\\&+\tfrac{1}{4}*\left[0.8*\left(r_{21}+\gamma V(S_{17})\right)+0.15*\left(r_{12}+\gamma V(S_{17})\right)+0.05*\left(r_{13}+\gamma V(S_{22})\right)\right]\\&+\tfrac{1}{4}*\left[0.8*\left(r_{31}+\gamma V(S_{16})\right)+0.15*\left(r_{32}+\gamma V(S_{17})\right)+0.05*\left(r_{23}+\gamma V(S_{18})\right)\right];\\&+\tfrac{1}{4}*\left[0.8*\left(r_{41}+\gamma V(S_{18})\right)+0.15*\left(r_{42}+\gamma V(S_{17})\right)+0.05*\left(r_{23}+\gamma V(S_{16})\right)\right]\\&=??\\&\approx???\end{aligned}$$

$$\begin{aligned}V(S_{24})=&0.5*\left[0.8*\left(r_{11}+\gamma V(S_{19})\right)+0.15*\left(r_{12}+\gamma V(S_{24})\right)+0.05*\left(r_{13}+\gamma V(S_{24})\right)\right]\\&+0.5*\left[0.8*\left(r_{21}+\gamma V(S_{23})\right)+0.15*\left(r_{22}+\gamma V(S_{24})\right)+0.05*\left(r_{23}+\gamma V(S_{24})\right)\right]\\&=0.5*\left[0.8*(3+0.8*0)+0.15*(0+0.8*0)+0.05*(0+0.8*0)\right];\\&+0.5*\left[0.8*(0+0.8*?)+0.15*(0+0.8*0)+0.05*(0+0.8*0)\right]\\&=??\end{aligned}$$

按顺序计算完一轮后，得到状态值函数$V(s)$，如图4.4（$\tau=2$）所示。

（4）当$\tau=??$时，$|V_\tau(s)-V_{\tau-1}(s)|_\infty<\theta$，认为$V_\tau(s)$已经收敛于$v_\pi(s)$，计算得到的$v_\pi(s)$就是在策略$\pi$下的有效评估。

随着迭代的进行，每轮状态值函数更新，如图4.4所示。

类似图4.2的图

图4.4 面向随机环境扫地机器人任务的状态值函数$v_\pi$的评估过程（异步计算方式）

#### 4.1.1.2 基于动作值函数的策略评估

与基于状态值函数的策略评估一样，根据迭代思想，可以将动作值函数的贝尔曼方程（3.19）转化为迭代式（4.2），即基于动作值（Q值）函数的策略评估迭代式：

$$\begin{aligned}q_\tau(s,a)&=\mathbb{E}_\pi(G_t\mid S_t=s,A_t=a)\\&=\sum_{s',r}p(s',r\mid s,a)\left[r+\gamma\sum_{a'}\pi(a'\mid s')q_{\tau-1}(s',a')\right]\end{aligned}\qquad(4.2)$$

使用Q值函数的策略评估，在当前策略$\pi$下，从任意Q值函数$q_0$开始，在每一轮迭代中，利用式（4.2）来更新Q值函数。

算法 4.2 给出了在有穷状态空间 MDP 中，基于动作值函数的策略评估算法。

**算法 4.2** 基于动作值函数的策略评估算法

| | |
|---|---|
| **输　入** | 初始策略 $\pi(a\mid s)$，动态性 $p$，奖赏函数 $r$，折扣因子 $\gamma$ |
| **初始化** | 1. 对任意 $s\in\mathcal{S}$，$a\in\mathcal{A}(s)$，初始化动作值函数，如 $Q(s,a)=0$<br>2. 阈值 $\theta$ 设置为一个较小的实数值，如 $\theta=0.01$ |
| **策　略<br>评　估** | 3. **repeat** 对每一轮策略评估 $\tau=1,2,\cdots$<br>4. $delta\leftarrow 0$<br>5. **for** 每个状态-动作对 $(s,a)$ **do**<br>7. $q\leftarrow Q(s,a)$<br>8. $Q(s,a)\leftarrow\sum_{s',r}p(s',r\mid s,a)\left[r+\gamma\sum_{a'}\left(\pi(a'\mid s')Q(s',a')\right)\right]$<br>9. $delta\leftarrow\max(delta,\mid q-Q(s,a)\mid)$<br>10. **end for**<br>11. **until** $delta<\theta$ |
| **输　出** | $q_*=Q$ |

**例 4.3** 利用算法 4.2，基于 Q 值函数对确定环境扫地机器人任务进行策略评估。如图 4.1 所示，设定策略评估的起始位置为左下角，即充电桩位置，按序号顺序进行评估。动作按 *Up, Down, Left, Right* 顺序评估。折扣系数 $\gamma=0.8$，在等概率策略 $\pi$ 下，计算确定环境扫地机器人任务的动作值函数。

根据算法 4.2，在随机策略 $\pi(a\mid s)$ 下，确定环境扫地机器人任务的评估过程如下：

（1）当 $\tau=0$ 时（$\tau$ 为迭代的次数），对于所有的 $(s,a)$ 初始化为：$Q(s,a)=0$；

（2）当 $\tau=1$ 时，以状态 $S_7$ 的 4 个动作 $\{Up,Down,Left,Right\}$ 为例。根据扫地机器人的确定环境 MDP 模型，$(S_7,Up)$ 遇到障碍物，保持原地 $S_7$ 不动，得到−10 的惩罚，状态 $S_7$ 中可以采取 $\{Up,Down,Left,Right\}$ 4 个动作，概率各为 $\frac{1}{4}$；$(S_7,Down)$ 到达状态 $S_2$，得到 0 的奖赏，状态 $S_2$ 中可以采取 $\{Up,Left,Right\}$ 3 个动作，概率各为 $\frac{1}{3}$；$(S_7,Left)$ 到达状态 $S_6$，得到 0 的奖赏，状态 $S_6$ 中可以采取 $\{Up,Down,Left,Right\}$ 4 个动作，概率各为 $\frac{1}{4}$；$(S_7,Right)$ 到达状态 $S_8$，得到 0 的奖赏，状态 $S_8$ 中可以采取 $\{Up,Down,Left,Right\}$ 4 个动作，概率各为 $\frac{1}{4}$。终上所述，根据算法 4.2 计算得：

$$
\begin{aligned}
Q(S_7,Up)&=r+\gamma\sum_{a'}\left(\pi(a'\mid S_7)Q(S_7,a')\right)\\
&=-10+0.8*\left[\tfrac{1}{4}*Q(S_7,Up)+\tfrac{1}{4}*Q(S_7,Down)+\tfrac{1}{4}*Q(S_7,Left)+\tfrac{1}{4}*Q(S_7,Right)\right];\\
&=-10+0.8*(0.25*0.25*0+0.25*0.25*0+0.25*0+0.25*0)\\
&=-10.00
\end{aligned}
$$

$$Q(S_7,Down)=r+\gamma\sum_{a'}\left(\pi(a'\mid S_2)Q(S_2,a')\right)$$
$$=0+0.8*\left[\tfrac{1}{3}*Q(S_2,up)+\tfrac{1}{3}*Q(S_2,Left)+\tfrac{1}{3}*Q(S_2,Right)\right]$$
$$=0+0.8*(\tfrac{1}{3}*0+\tfrac{1}{3}*0+\tfrac{1}{3}*0)$$
$$=0.00$$
$$Q(S_7,Left)=r+\gamma\sum_{a'}\left(\pi(a'\mid S_6)Q(S_6,a')\right)$$
$$=0+0.8*\left[\tfrac{1}{4}*Q(S_6,up)+\tfrac{1}{4}*Q(S_6,Down)+\tfrac{1}{4}*Q(S_6,Left)+\tfrac{1}{4}*Q(S_6,Right)\right]$$
$$=0+0.8*(\tfrac{1}{4}*?\ ?+\tfrac{1}{4}*?\ ?+\tfrac{1}{4}*?\ ?+\tfrac{1}{4}*?\ ?)$$
$$=?\ ?$$
$$Q(S_7,Right)=r+\gamma\sum_{a'}\left(\pi(a'\mid S_8)Q(S_8,a')\right)$$
$$=0+0.8*\left[\tfrac{1}{4}*Q(S_8,Up)+\tfrac{1}{4}*Q(S_8,Down)+\tfrac{1}{4}*Q(S_8,Left)+\tfrac{1}{4}*Q(S_8,Right)\right]$$
$$=0.00$$

按顺序计算完一轮后，得到动作值函数$Q(s,a)$，如表 4.2（$\tau=1$）所示。

（3） 当$\tau=?\ ?$时，$|Q_\tau(s,a)-Q_{\tau-1}(s,a)|_\infty<\theta$，认为$Q_\tau(s,a)$已经收敛于$q_\pi(s,a)$，计算得到的$q_\pi(s,a)$就是在策略$\pi$下的有效评估。

随着迭代的进行，每轮动作值函数更新，如表 4.1 所示。表中动作按*Up, Down, Left, Right*顺序评估，不同动作值之间用“;”分割开。

表 4.1 面向确定环境扫地机器人任务的动作值函数$q_\pi$评估过程（状态 S11 改为状态 S7，数据要改）

| 状态 / $q_k$ | $S_1$ | $S_2$ | $S_3$ | … | $S_7$ | … | $S_{24}$ |
|---|---|---|---|---|---|---|---|
| $q_0$ | 0.00;×;0.00;0.00 | 0.00;×;0.00;0.00 | 0.00;×;0.00;0.00 | … | 0.00;0.00;0.00;0.00 | … | ×;0.00;0.00;× |
| $q_1$ | -1.73;×;1.00;-1.42 | -3.72;×;-0.57;-1.02 | -1.73;×;-1.42;-0.69 | … | -1.90;-1.73;-1.46;-10.00 | … | ×;3.00;-0.26;× |
| $q_2$ | 0.64;×;1.00;0.64 | 0.51;×;0.80;1.23 | 1.54;×;0.64;1.54 | … | 0.00;0.80;0.51;0.51 | … | ×;3.00;1.92;× |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $q_5$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.64;-10.00 | … | ×;3.00;1.92;× |
| $q_6$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.98;-10.00 | … | ×;3.00;1.92;× |
| $q_\pi$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |

**例 4.4** 利用算法 4.2，使用 Q 值函数对随机环境扫地机器人任务进行策略评估。在其他条件与例 4.2 不变的情况下，采用异步计算方式，给出在策略$\pi$下，面向随机环境扫地机器人任务的动作值函数$q_\pi$评估过程。

根据算法 4.2，在随机策略$\pi(a\mid s)$下，利用 Q 值函数，随机环境扫地机器人任务的评估过程如下：

（1）当$\tau=0$时（$\tau$是迭代的次数），对于所有的$(s,a)$初始化为：$Q(s,a)=0$；

（2）当$\tau=1$时，以状态$S_7$的 4 个动作$\{Up,Down,Left,Right\}$为例。根据扫地机器人的

随机环境 MDP 模型，$(S_7,Up)$ 以 0.8 的概率撞到障碍物，保持原地 $S_7\left(Q(S_7,Up)=?,Q(S_7,Down)=?,Q(S_7,Left)=?,Q(S_7,Right)=?\right)$ 不动，得到−10 的惩罚；以 0.15 保持原地 $S_7\left(Q(S_7,Up)=?,Q(S_7,Down)=?,Q(S_7,Left)=?,Q(S_7,Right)=?\right)$ 不动，获得 0 的奖赏；以 0.05 滑到状态 $S_2\left(Q(S_2,Up)=?,Q(S_2,Left)=?,Q(S_2,Right)=?\right)$，获得 0 的奖赏。

根据算法 4.2 计算得：

$$
\begin{aligned}
Q(S_7,Up) &= p(s',r\mid s,Up)\left[r+\gamma\sum_{a'}\left(\pi(a'\mid s')Q(s',a')\right)\right]\\
&= p(S_7,r\mid S_7,Up)\left[r_1+\gamma\sum_{a'}\left(\pi(a'\mid S_7)Q(S_7,a')\right)\right]\\
&+p(S_7,r_2\mid S_7,Up)\left[r_2+\gamma\sum_{a'}\left(\pi(a'\mid S_7)Q(S_7,a')\right)\right]\\
&+p(S_2,r_3\mid S_7,Up)\left[r_3+\gamma\sum_{a'}\left(\pi(a'\mid S_2)Q(S_2,a')\right)\right]\\
&= 0.8*\left[-10+0.8*\left(\tfrac{1}{4}*Q(S_7,Up)+\tfrac{1}{4}*Q(S_7,Down)+\tfrac{1}{4}*Q(S_7,Left)+\tfrac{1}{4}*Q(S_7,Right)\right)\right]\\
&+0.15*\left[0+0.8*\left(\tfrac{1}{4}*Q(S_7,Up)+\tfrac{1}{4}*Q(S_7,Down)+\tfrac{1}{4}*Q(S_7,Left)+\tfrac{1}{4}*Q(S_7,Right)\right)\right]\\
&+0.05*\left[0+0.8*\left(\tfrac{1}{3}*Q(S_2,Up)+\tfrac{1}{3}*Q(S_2,Left)+\tfrac{1}{3}*Q(S_2,Right)\right)\right]\\
&= 0.8*\left[-10+0.8*\left(\tfrac{1}{4}*??+\tfrac{1}{4}*??+\tfrac{1}{4}*??+\tfrac{1}{4}*??\right)\right]\\
&+0.15*\left[0+0.8*\left(\tfrac{1}{4}*??+\tfrac{1}{4}*??+\tfrac{1}{4}*??+\tfrac{1}{4}*??\right)\right]\\
&+0.05*\left[0+0.8*\left(\tfrac{1}{3}*??+\tfrac{1}{3}*??+\tfrac{1}{3}*??\right)\right]\\
&= ???
\end{aligned}
$$

；

同理可以计算动作值函数 $Q(S_7,Down)=??$，$Q(S_7,Left)=??$，$Q(S_7,Right)=??$。

按顺序计算完一轮后，得到动作值函数 $Q(s,a)$，如表 4.2（$\tau=1$）所示。

（3） 当 $\tau=??$ 时，$|Q_\tau(s,a)-Q_{\tau-1}(s,a)|_\infty<\theta$，认为 $Q_\tau(s,a)$ 已经收敛于 $q_\pi(s,a)$，计算得到的 $q_\pi(s,a)$ 就是在策略 $\pi$ 下的有效评估。

随着迭代的进行，每轮动作值函数更新，如表 4.2 所示。

表 4.2 面向随机环境扫地机器人任务的动作值函数 $q_\pi$ 评估过程

| 状态 / $q_\tau$ | $S_1$ | $S_2$ | $S_3$ | … | $S_7$ | … | $S_{24}$ |
|---|---|---|---|---|---|---|---|
| $q_0$ | 0.00;×;0.00;0.00 | 0.00;×;0.00;0.00 | 0.00;×;0.00;0.00 | … | 0.00;0.00;0.00;0.00 | … | ×;0.00;0.00;× |
| $q_1$ | -1.73;×;1.00;-1.42 | -3.72;×;-0.57;-1.02 | -1.73;×;-1.42;-0.69 | … | -1.90;-1.73;-1.46;-10.00 | … | ×;3.00;-0.26;× |
| $q_2$ | 0.64;×;1.00;0.64 | 0.51;×;0.80;1.23 | 1.54;×;0.64;1.54 | … | 0.00;0.80;0.51;0.51 | … | ×;3.00;1.92;× |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $q_5$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.64;-10.00 | … | ×;3.00;1.92;× |
| $q_6$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.98;-10.00 | … | ×;3.00;1.92;× |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| $q_\pi$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |

### 4.1.2 策略改进

策略的优劣性可以由值函数来评价。通过策略评估迭代得到值函数，再利用动作值函数来寻找更好的策略。假设已知某一策略 $\pi$ 的值函数 $v_\pi$ 或 $q_\pi$，目的是寻找一个更优策略 $\pi'$。下面从特殊情况到一般情况对策略改进方法进行说明，最后推导出最优策略的获取方法。

（1）特殊情况

针对单一状态 $s$ 和特定动作 $a$，制定如下约定以获得新策略 $\pi'\ (\pi' \neq \pi)$：

- 在状态 $s$ 下选择一个新动作 $a(a \neq \pi(s))$；
- 保持后续（其他）状态所执行的动作与原策略 $\pi$ 给出的动作相同。

在这种情况下，根据动作值函数贝尔曼方程，得到的价值为：

$$\begin{aligned} q_\pi(s,a) &= \mathbb{E}_\pi\left(R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a\right) \\ &= \sum_{s',r} p(s',r \mid s,a)\left[r + \gamma v_\pi(s')\right] \end{aligned} \tag{4.3}$$

若 $q_\pi(s,a) \geq v_\pi(s)$ 成立，则说明满足以上约定的策略 $\pi'$ 优于或等价于 $\pi$。

（2）一般情况

将单一状态和特定动作的情况进行拓展。对任意状态 $s \in \mathcal{S}$，若存在任意的两个确定策略 $\pi$ 和 $\pi'$（$\pi' \neq \pi$）满足策略改进定理（policy improvement theorem），则说明在状态 $s$ 处采取策略 $\pi'$ 时，能得到更大的值函数，即 $\pi'$ 优于或等价于 $\pi$。策略改进定理为：

$$q_\pi\left(s,\pi'(s)\right) \geq v_\pi(s) \tag{4.4}$$

此时，对任意状态 $s$，执行策略 $\pi'$ 时，必定能够得到更大的期望回报，即存在：

$$v_{\pi'}(s) \geq v_\pi(s) \tag{4.5}$$

对策略改进定理进行推导，利用式（4.3），基于式（4.2）等号左侧的 $q_\pi$ 进行多次展开，并不断应用式（4.3）直到得到 $v_{\pi'}(s)$：

$$\begin{aligned} v_\pi(s) &\leq q_\pi\left(s,\pi'(s)\right) \\ &= \mathbb{E}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)\right] && \text{由式（4.2）推出} \\ &= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s\right] && \text{将条件}\pi'(s)\text{提出} \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma q_\pi\left(S_{t+1},\pi'(S_{t+1})\right) \mid S_t = s\right] && \text{由式（4.4）推出，} v_\pi(S_{t+1}) \leq q_\pi\left(S_{t+1},\pi'(S_{t+1})\right) \\ &= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma \mathbb{E}_{\pi'}\left[R_{t+2} + \gamma\pi(S_{t+2})\right] \mid S_t = s\right] && \text{由式（4.2）推出} \\ &= \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \mid S_t = s\right] && \text{期望展开} \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \mid S_t = s\right] \\ &\cdots \\ &\leq \mathbb{E}_{\pi'}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] = \mathbb{E}_{\pi'}\left[G_t \mid S_t = s\right] = v_{\pi'}(s) \end{aligned}$$

由上述推导可知，给定一个策略及其值函数估计值，就能判定改变某个状态的动作对策

略的影响。于是对每个状态 $s$ 都选择最优动作，即基于贪心策略选取最大动作值函数对应的动作，构建一个新的贪心策略 $\pi$ ，满足如下等式：

$$\begin{aligned}\pi'(s) &= \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}\left(R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a\right) \\ &= \arg\max_a \sum_{s',r} p(s',r \mid s,a)[r + \gamma v_\pi(s')]\end{aligned} \tag{4.6}$$

其中， $\arg\max_a$ 表示能够使得表达式值最大化的 $a$，如果结果有多个满足条件的值时，则随机取其一。

贪心策略选取了短期内最优的动作，即根据 $v_\pi$ 向前单步搜索，式（4.6）构造出的策略满足策略改进定理：

$$v_{\pi'}(s) = q_\pi\left(s, \pi'(s)\right) = \max_a q_\pi(s,a) \geq q_\pi\left(s, \pi(s)\right) = v_\pi(s) \tag{4.7}$$

这样一种贪心策略的概率分布呈 *one - hot* 分布，该策略也称为确定贪心策略。其满足如下关系：

$$\pi'(a \mid s) = \begin{cases} 1 & a = \arg\max_a q_\pi(s,a) \\ 0 & \text{其他} \end{cases} \tag{4.8}$$

另一种贪心策略称为随机贪心策略，其概率分布为：对于所有的贪心动作 $a'$ ，赋予一个 $(0,1]$ 的值，且其满足如下关系：

$$\pi'(a \mid s) = \begin{cases} \pi'(a' \mid s) & \pi'(a' \mid s) \in (0,1] \text{且} \sum_{a'} \pi'(a' \mid s) = 1 \\ 0 & \text{其他} \end{cases} \tag{4.9}$$

（3）策略迭代中的最优策略

通过策略改进，可以得到更优策略 $\pi'$ ，而强化学习的目标是获得最优策略 $\pi_*$ 。假设贪心策略 $\pi'$ 与原策略 $\pi$ 一样好，根据式（4.6）可得，对任意 $s \in \mathcal{S}$ ，存在：

$$v_{\pi'}(s) = \max_a \sum_{s',\mathrm{r}} p(s',r \mid s,a)\left[r + \gamma v_{\pi'}(s')\right]$$

该式与贝尔曼最优方程（3.24）的形式完全相同，因此 $v_{\pi'}$ 一定等于 $v_*$ ，且 $\pi$ 和 $\pi'$ 均为最优策略。综上所述，策略改进可以得到最优策略。

### 4.1.3 策略迭代

策略迭代的关键部分是策略评估，首先评估状态的价值，然后根据状态的动作值进行相应的策略改进，并进行下一轮评估和改进，直到策略稳定。策略改进可以通过求解静态最优化问题来实现，通过状态动作值来选择动作，通常比策略评估容易。

#### 4.1.3.1 基于状态值函数的策略迭代

基于状态值函数的策略迭代算法主要包括以下 3 个阶段：

（1）初始化策略函数 $\pi(a|s)$ 和状态值函数 $V_0(s)$；

（2）策略评估：在当前策略 $\pi$ 下，使用贝尔曼方程更新状态值函数 $V_\tau(s)$，直到收敛于 $v_\pi(s)$，再根据式（4.3）计算出 $q_\pi(s,a)$。

（3）策略改进：基于 $q_\pi(s,a)$，通过贪心策略得到更优策略 $\pi'$，满足式（4.9）。

直到策略不再发生变化，迭代产生的状态值函数渐近地收敛到 $v_*(s)$，同时得到最优策略 $\pi_*(s)$ 。

通常基于状态值函数的策略改进是通过状态值函数 $v_\pi(s)$ 进行的，但需要经过一步搜索，得到动作值函数 $q_\pi(s,a)$，而后进行改进策略。基于状态值函数估算最优策略的策略迭代算法，如算法 4.3 所示。

**算法 4.3** 基于随机 MDP 的状态值函数 $v_\pi$ 策略迭代算法

| | | |
|---|---|---|
| 输　入 | | 初始策略 $\pi(a\|s)$，动态性 $p$，奖赏函数 $r$，折扣因子 $\gamma$ |
| 初始化 | | 1. 对任意 $s\in\mathcal{S}$，初始化状态值函数：如 $V(s)=0$<br>2. 阈值 $\theta$ 设置为一个较小的实值 |
| 策略迭代 | | 3. **repeat** 对每一轮策略迭代 $l=0,1,2,\cdots$ |
| | 策略评估 | 4. 在当前策略 $\pi(a\|s)$ 下，通过算法 4.1 策略评估迭代，得到值函数 $v_\pi$：$V(s)$ |
| | 策略改进 | 5. $policy\_stable \leftarrow$ **True**<br>6. **for** 每个状态 $s$ **do**<br>7. $old\_policy \leftarrow \pi(a\|s)$<br>8. $Q(s,a)=\sum_{s',r} p(s',r\|s,a)\left[r+\gamma V(s')\right]$<br>9. $max\_action \leftarrow \arg\max_a Q(s,a)$；$max\_count \leftarrow \text{count}(\max(Q(s,a)))$<br>10. **for** 每个状态-动作对 $(s,a')$ **do**<br>11. **if** $a'==max\_action$ **then**<br>12. $\pi(a'\|s)=1/(max\_count)$<br>13. **else**<br>14. $\pi(a'\|s)=0$<br>15. **end if**<br>16. **end for**<br>17. **if** $old\_policy\neq\pi(a\|s)$ **then**<br>18. $policy\text{-}stable \leftarrow$ **False**<br>19. **end for** |
| | | 20. **until** $policy\_stable=$ **True** |

| | |
|---|---|
| 输　出 | $v_* = V$ ， $\pi_* = \pi$ |

**例 4.5** 将基于状态值函数的策略迭代算法 4.3 应用于例 4.1 的确定环境扫地机器人任务中，经过多轮迭代后，得到表 4.3 所示的值函数和策略迭代更新过程。

表 4.3 面向确定环境扫地机器人任务的状态值函数策略迭代更新过程

| | $S_1$ | $S_2$ | $S_3$ | … | $S_7$ | … | $S_{24}$ |
|---|---|---|---|---|---|---|---|
| $v_0$ | 0.00 | 0.00 | 0.00 | … | 0.00 | … | 0.00 |
| $\pi_0$ | 0.33;0.00;0.33;0.33 | 0.33;0.00;0.33;0.33 | 0.33;0.00;0.33;0.33 | … | 0.25;0.25;0.25;0.25 | … | 0.00;0.5;0.5;0.00 |
| $v_1$ | -0.72 | -1.77 | -1.28 | … | -4.72 | … | 1.37 |
| $q_1$ | -1.73;×;1.00;-1.42 | -3.72;×;-0.57;-1.02 | -1.73;×;-1.42;-0.69 | … | -1.90;-1.73;-1.46;-10.00 | … | ×;3.00;-0.26;× |
| $\pi_1$ | 0.00;0.00;1.00;0.00 | 0.00;0.00;1.00;0.00 | 0.00;0.00;0.00;1.00 | … | 0.00;0.00;1.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| $v_2$ | 1.00 | 0.80 | 1.54 | … | 0.64 | … | 3.00 |
| $q_2$ | 0.64;×;1.00;0.64 | 0.51;×;0.80;1.23 | 1.54;×;0.64;1.54 | … | 0.00;0.80;0.51;0.51 | … | ×;3.00;1.92;× |
| $\pi_2$ | 0.00;0.00;1.00;0.00 | 0.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 0.00;1.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $\pi_4$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| $v_5$ | 1.00 | 1.23 | 1.54 | … | 1.54 | … | 3.00 |
| $q_5$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.64;-10.00 | … | ×;3.00;1.92;× |
| $\pi_5$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| $v_6$ | 1.00 | 1.23 | 1.54 | … | 1.54 | … | 3.00 |
| $q_6$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.98;-10.00 | … | ×;3.00;1.92;× |
| $\pi_6$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| $\pi_*$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |

图 4.5 给出了面向确定环境扫地机器人任务的状态值函数及策略迭代更新过程图。

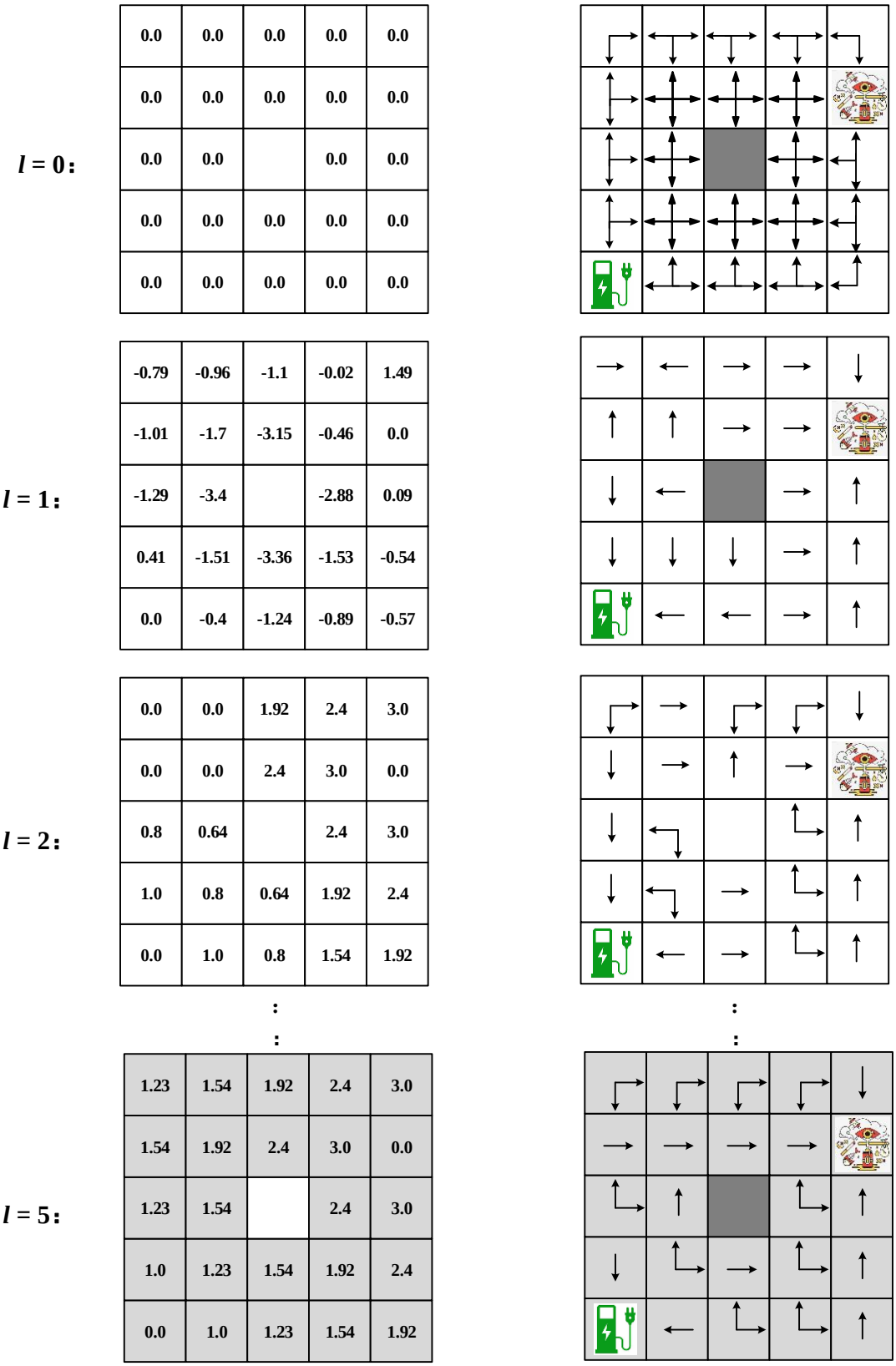



图 4.5 面向确定环境扫地机器人任务的状态值函数及策略迭代更新图

在图 4.5 中，左边一列给出了每一轮（$l = 0, 1, 2, 3, 4, 5$）状态值函数的迭代更新过程。当$l = 4$和5时，策略不再发生变化（这里状态值也不再变化），策略评估迭代收敛。右侧一

列给出了相应的策略改进过程。依据对动作值函数的贪心，获得较好策略，对策略进行改进。从图 4.5 可以看出，每个状态的最优策略始终选择动作值最大的动作。

**例 4.6** 汽车租赁问题。杰克经营两家异地的汽车租赁场（A，B 租赁场），每天都会有客户来租车。如果每个租赁场有可供外租的汽车，每租出一辆汽车，杰克都会获得 10 美元的租金。为了保证每个租赁场有足够的车辆可租，每天晚上杰克会在两个租赁场之间移动车辆，每辆车的移车费用为 2 美元。假设每个租赁场的租车和还车的数量是一个泊松随机量，即期望数量 $n$ 的概率为 $\frac{\lambda^n}{n!}e^{-\lambda}$ ，其中 $\lambda$ 为期望值。假设在两个租赁场租车的 $\lambda$ 分别是 $\lambda_{\mathrm{A1}}=3$，$\lambda_{\mathrm{B1}}=4$；还车的 $\lambda$ 分别为 $\lambda_{\mathrm{A2}}=3$， $\lambda_{\mathrm{B2}}=2$。

为了简化问题，给定 3 个假设：（1）任何一个租赁场车辆总数不超过 20 辆车；（2）当天还回的车辆第 2 天才能出租。（3）两个租车场之间每天最多可移车数量为 5 辆。

利用策略迭代方法，在折扣率 $\gamma=0.9$ 时，计算两个租赁场点之间的最优移车策略。

将汽车租赁问题描述为一个持续的有限 MDP 模型。其中时刻 $t$ 按天计算；状态为每天租还车结束时，两个租赁场的车辆数；动作为每天晚上在两个租赁场之间移动的车辆数目。

该问题的 MDP 模型为：

（1）状态空间：两个租赁场每天租还结束时，可供出租的车辆数组成的二维向量，状态共 $21\times21=441$ 个，即 $\mathcal{S}=\{[0,0],[0,1],[0,2],\cdots,[10,10],[10,11],\cdots,[20,19],[20,20]\}$ 。用 $s$ 表示状态空间中的某一个状态 $[s_{\mathrm{A}},s_{\mathrm{B}}]$。例如，$[3,5]$ 表示每天租还结束后，A 租赁场可供出租 3 辆车、B 租赁场可供出租 5 辆车的状态。

（2）动作空间：每天晚上在两个租赁场之间移动车辆的数目。根据任务的假设，可移动车辆数目不超过 5 辆。设 A 租赁场向 B 租赁场移车为“−”，B 租赁场向 A 租赁场移车为“+”。该问题的动作空间中共包含 11 个动作，即：$\mathcal{A}(s)=\{-5,-4,-3,-2,-1,0,+1,+2,+3,+4,+5\}$。用 $a$ 表示动作空间中的某一个动作。例如，−2 表示 A 租赁场向 B 租赁场移车 2 辆；+3 表示 B 租赁场向 A 租赁场移车 3 辆。

（3）状态转移函数：

令 $s=[s_{\mathrm{A}},s_{\mathrm{B}}]$，$s'=[s'_{\mathrm{A}},s'_{\mathrm{B}}]$，那么状态转移函数为：$p([s_{\mathrm{A}},s_{\mathrm{B}}],a,[s'_{\mathrm{A}},s'_{\mathrm{B}}])$，即在当前状态 $s=[s_{\mathrm{A}},s_{\mathrm{B}}]$ 下，采取动作 $a$，到达下一状态 $s'=[s'_{\mathrm{A}},s'_{\mathrm{B}}]$ 的概率。设 $n_{\mathrm{A}}$ 为 A 租赁场当天租掉的车辆数；$n_{\mathrm{B}}$ 为 B 租赁场当天租掉的车辆数。$m_{\mathrm{A}}$ 为 A 租赁场当天还回的车辆数；$m_{\mathrm{B}}$ 为 B 租赁场当天还回的车辆数。假设当前状态为 $s=[2,5]$，采取+1 动作后，中间状态为 $s_{\mathrm{L}}=[3,4]$，

当下一状态为$s'=[1,2]$，对于 A 租赁场来说，一天的租掉、还回车辆$(n_A,m_A)$可能为：$(2,0)$、$(3,1)$共 2 种情况；对于 B 租赁场来说，一天的租掉、还回车辆$(n_B,m_B)$可能为：$(2,0)$、$(3,1)$、$(4,2)$共 3 种情况。那么在该情况下，状态转移函数$p([2,5],+1,[1,2])$计算为：

$$\begin{aligned}p([2,5],+1,[1,2])=&\left(\frac{\lambda_{A1}^2}{2!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{A1}^0}{0!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{B1}^2}{2!}e^{-\lambda_{B1}}\right)*\left(\frac{\lambda_{B1}^0}{0!}e^{-\lambda_{B1}}\right)\\&+\left(\frac{\lambda_{A1}^2}{2!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{A1}^0}{0!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{B1}^3}{3!}e^{-\lambda_{B1}}\right)*\left(\frac{\lambda_{B1}^1}{1!}e^{-\lambda_{B1}}\right)\\&+\left(\frac{\lambda_{A1}^2}{2!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{A1}^0}{0!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{B1}^4}{4!}e^{-\lambda_{B1}}\right)*\left(\frac{\lambda_{B1}^2}{2!}e^{-\lambda_{B1}}\right)\\&+\left(\frac{\lambda_{A1}^3}{3!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{A1}^1}{1!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{B1}^2}{2!}e^{-\lambda_{B1}}\right)*\left(\frac{\lambda_{B1}^0}{0!}e^{-\lambda_{B1}}\right)\\&+\left(\frac{\lambda_{A1}^3}{3!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{A1}^1}{1!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{B1}^3}{3!}e^{-\lambda_{B1}}\right)*\left(\frac{\lambda_{B1}^1}{1!}e^{-\lambda_{B1}}\right)\\&+\left(\frac{\lambda_{A1}^3}{3!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{A1}^1}{1!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{B1}^4}{4!}e^{-\lambda_{B1}}\right)*\left(\frac{\lambda_{B1}^2}{2!}e^{-\lambda_{B1}}\right)\end{aligned}$$

根据任务可知，这里$\lambda_{A1}=3$，$\lambda_{B1}=4$，$\lambda_{A2}=3$，$\lambda_{B2}=2$。

综上所述，如果$0\le S'_A-S_A-a+n_A\le 20$且$0\le S'_B-S_B+a+n_B\le 20$，则有：

$$\begin{aligned}p(s,a,s')&=p([s_A,s_B],a,[s'_A,s'_B])\\&=\sum_{n_A=0}^{s_A+a}\sum_{n_B=0}^{S_B-a}\left(\frac{\lambda_{A1}^{n_A}}{n_A!}e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{A2}^{(S'_A-S_A-a+n_A)}}{(S'_A-S_A-a+n_A)!}e^{-\lambda_{A2}}\right)*\left(\frac{\lambda_{B1}^{n_B}}{n_B!}e^{-\lambda_{B1}}\right)*\left(\frac{\lambda_{B2}^{(S'_B-S_B+a+n_B)}}{(S'_B-S_B+a+n_B)!}e^{-\lambda_{B2}}\right)\end{aligned}$$

（4）立即奖赏：该问题的立即奖赏函数为：$r([s_A,s_B],a,[s'_A,s'_B])$，即在当前状态$s=[s_A,s_B]$下，采取动作$a$，到达下一状态$s'=[s'_A,s'_B]$得到的立即奖赏。

如果$0\le S'_A-S_A-a+n_A\le 20$且$0\le S'_B-S_B+a+n_B\le 20$，则立即奖赏由 3 部分组成，分别是：

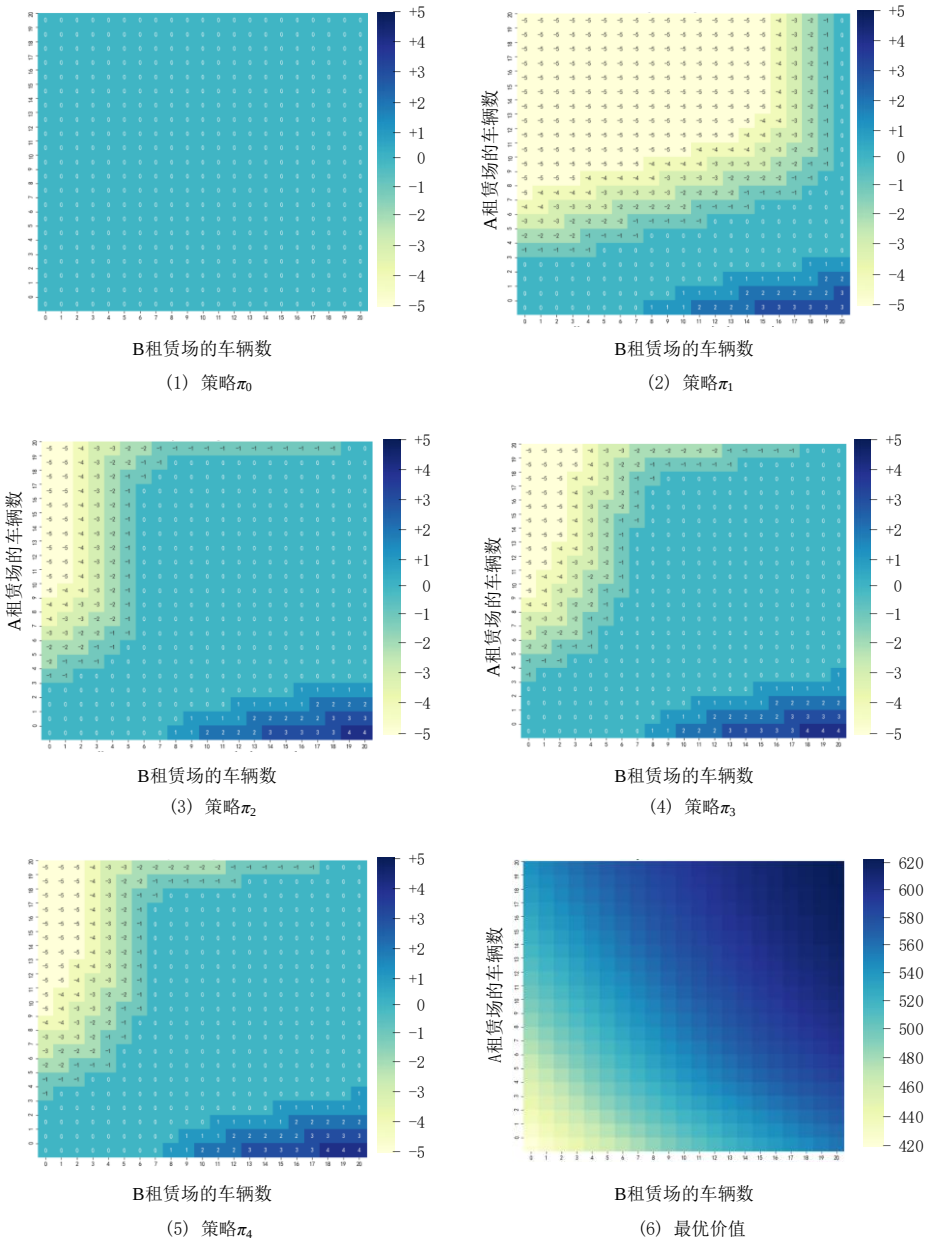
- 两个租赁场之间的移车费用：$r_1=-2*|a|$；
- 两个租赁场的租车收益：

$$r_2=\sum_{n_A=0}^{S_A+a}\sum_{n_B=0}^{S_B-a}\left[\begin{array}{l}\left(\frac{\lambda_{A1}^{n_A}}{n_A!}*e^{-\lambda_{A1}}\right)*\left(\frac{\lambda_{A2}^{(S'_A-S_A-a+n_A)}}{(S'_A-S_A-a+n_A)!}*e^{-\lambda_{A2}}\right)*\left(\frac{\lambda_{B1}^{n_B}}{n_B!}*e^{-\lambda_{B1}}\right)\\*\left(\frac{\lambda_{B2}^{(S'_B-S_B+a+n_B)}}{(S'_B-S_B+a+n_B)!}*e^{-\lambda_{B2}}\right)*(n_A+n_B)*10\end{array}\right];$$

这样，获得的立即奖赏为：$r([s_A,s_B],a,[s'_A,s'_B])=r_1+r_2$。

（5）折扣因子：$\gamma = 0.9$。

图 4.6 为关于杰克租车问题的策略迭代过程，经过 5 轮的策略评估和改进后，得到该问题的最优策略及最优价值，如图 4.6（5）、图 4.6（6）所示。(有问题，横坐标为 A 租赁场的，纵坐标为 B,另外图中不要标出动作了，字太小，看不清)




（1）策略$\pi_0$　（2）策略$\pi_1$

（3）策略$\pi_2$　（4）策略$\pi_3$

（5）策略$\pi_4$　（6）最优价值

图 4.6 关于杰克租车问题的策略迭代过程

该问题需要从一个确定策略出发进行策略迭代。本实验的初始策略为：在任何状态下，不考虑两个租赁场的需求，选择不移动车辆（动作为 0）。然后进行策略评估，并根据值函数进行策略改进。如此反复，直到策略收敛，找到最优策略。图 4.6（5）即为最优策略图。

#### 4.1.3.2 基于动作值函数的策略迭代

与基于状态值函数的策略迭代不同，基于动作值函数的策略迭代是在当前策略 $\pi(a|s)$ 下，利用式（4.2）对状态-动作对进行评估。具体如算法 4.4 所示。

**算法 4.4** 基于随机 MDP 的动作值函数 $q_\pi$ 策略迭代算法

| | | |
|---|---|---|
| 输　入 | | 初始策略 $\pi(a\|s)$，动态性 $p$，奖赏函数 $r$，折扣因子 $\gamma$ |
| 初始化 | | 1. 对任意 $s\in\mathcal{S}$，$a\in\mathcal{A}(s)$，初始化动作值函数：如 $Q(s,a)=0$<br>2. 阈值 $\theta$ 设置为一个较小的实值 |
| 策略迭代 | | 3. **repeat** 对每一轮策略迭代 $l=1,2,\cdots$ |
| | 策略评估 | 4. 在当前策略 $\pi(a\|s)$ 下，通过算法 4.2 策略评估迭代，得到动作值函数 $q_\pi$：$Q(s,a)$ |
| | 策略改进 | 5. $policy\_stable \leftarrow$ **True**<br>6. **for** 每个状态 $s$ **do**<br>7. $old\_policy \leftarrow \pi(a\|s)$<br>8. $max\_ction \leftarrow \arg\max_a Q(s,a)$；$max\_count \leftarrow \text{count}(\max(Q(s,a)))$<br>9. **for** 每个状态-动作对 $(s,a')$ **do**<br>10. **if** $a'==max\_action$ **then**<br>11. $\pi(a'\|s)=1/(max\_count)$<br>12. **else**<br>13. $\pi(a'\|s)=0$<br>14. **end if**<br>15. **end for**<br>16. **if** $old\_policy\neq\pi(a\|s)$ **then**<br>17. $policy\_stable \leftarrow$ **False**<br>18. **end for** |
| | | 19. **until** $policy\_stable=$ **True** |
| 输　出 | | $q_*=Q$，$\pi_*=\pi$ |

算法 4.4 与算法 4.3 的区别在于：算法 4.3 是基于状态值函数的策略迭代，利用 $V(s)$ 值函数和转移动态，计算该状态 $s$ 下所有动作 $a$ 的动作值 $Q(s,a)$，然后根据 $Q(s,a)$，选择较优的策略，再进行改进。而算法 4.2 是基于动作值函数 $Q(s,a)$ 进行的策略迭代，因为在 $Q(s,a)$ 中包含了关于动作的信息，因此较优策略可以根据 $Q(s,a)$ 直接得到。

**例 4.7** 将基于动作值函数的策略迭代算法 4.4 应用于例 4.1 的确定环境扫地机器人任务中，经过多轮迭代后，得到表 4.4 所示的动作值函数和策略迭代更新过程。

表 4.4 面向确定环境扫地机器人任务的基于动作值函数的策略迭代更新过程

| | $S_1$ | $S_2$ | $S_3$ | … | $S_7$ | … | $S_{24}$ |
|---|---|---|---|---|---|---|---|
| $q_0$ | 0.00;×;0.00;0.00 | 0.00;×;0.00;0.00 | 0.00;×;0.00;0.00 | … | 0.00;0.00;0.00;0.00 | … | ×;0.00;0.00;× |
| $\pi_0$ | 0.33;0.00;0.33;0.33 | 0.33;0.00;0.33;0.33 | 0.33;0.00;0.33;0.33 | … | 0.25;0.25;0.25;0.25 | … | 0.00;0.5;0.5;0.00 |
| $q_1$ | -1.73;×;1.00;-1.42 | -3.72;×;-0.57;-1.02 | -1.73;×;-1.42;-0.69 | … | -1.90;-1.73;-1.46;-10.00 | … | ×;3.00;-0.26;× |
| $\pi_1$ | 0.00;0.00;1.00;0.00 | 0.00;0.00;1.00;0.00 | 0.00;0.00;0.00;1.00 | … | 0.00;0.00;1.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| $q_2$ | 0.64;×;1.00;0.64 | 0.51;×;0.80;1.23 | 1.54;×;0.64;1.54 | … | 0.00;0.80;0.51;0.51 | … | ×;3.00;1.92;× |
| $\pi_2$ | 0.00;0.00;1.00;0.00 | 0.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 0.00;1.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $\pi_4$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| $q_5$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.64;-10.00 | … | ×;3.00;1.92;× |
| $\pi_5$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| $q_6$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.98;-10.00 | … | ×;3.00;1.92;× |
| $\pi_6$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |
| $\pi_*$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |

## 4.2 值迭代

在策略迭代中，每轮策略改进之前都涉及策略评估，每次策略评估都需要多次遍历才能保证状态值函数在一定程度上得到收敛，这将消耗大量的时间和计算资源。从例 4.1 中可以看出，当 $\tau \geq 4$ 时，值函数的变化对策略不再有影响。于是根据迭代次数与策略稳定的相互关系，考虑在单步评估之后就进入改进过程，即采取截断式策略评估，在一次遍历完所有的状态后立即停止策略评估，进行策略改进，这种方法称为值迭代。基于状态值函数的值迭代公式为：

$$\begin{aligned} v_\ell(s) &= \max_a \mathbb{E}_\pi(R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a) \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma v_{\ell-1}(s')\right] \end{aligned} \tag{4.10}$$

该式与策略评估方程（4.1）的唯一区别在于：对状态值函数进行更新时，仅使用最大动作值函数，而非所有动作值函数的期望。可以证明：当 $v_*$ 存在时，对任意初始 $v_0$，序列 $\{v_\ell\}$ 一定能收敛到 $v_*$。

值迭代将贝尔曼最优方程变成一条更新规则，其状态值函数的计算过程仅体现了状态转移的随机性，而与策略无关。理论上值迭代依然需要迭代无数次才能收敛，但与策略迭代一样，如果一次遍历后值函数的变化低于阈值，可以提前停止。由于策略迭代的收敛速度更快一些，因此在状态空间较小时，最好选用策略迭代方法。当状态空间较大时，值迭代的计算量更小些。

算法 4.5 给出了在有穷状态空间 MDP 中，基于状态值函数的值迭代算法。

**算法 4.5** 基于状态值函数的值迭代算法

| | |
|---|---|
| 输　入 | 动态性 $p$，奖赏函数 $r$，折扣因子 $\gamma$ |
| 初始化 | 1. 对任意 $s \in \mathcal{S}$，初始化状态值函数，如 $V(s)=0$<br>2. 阈值 $\theta$ 设置为一个较小的实值 |
| 评　估<br>过　程 | 3. **repeat** 对每一轮值迭代 $l=1,2,\cdots$<br>4. $delta \leftarrow 0$<br>5. **for** 每个状态 $s$ **do**<br>6. $v \leftarrow V(s)$<br>7. $V(s) \leftarrow \max\limits_{a} \sum\limits_{s',r} p(s',r \mid s,a)\left[r+\gamma V(s')\right]$<br>8. $delta \leftarrow \max(delta, \mid v-V(s) \mid)$<br>9. **end for**<br>10. **until** $delta < \theta$ |
| 最　优<br>策　略 | 11. **for** 每个状态 $s$ **do**<br>12. $max\_action \leftarrow \arg\max\limits_{a} \sum\limits_{s',r} p(s',r \mid s,a)\left[r+\gamma V(s')\right]$;<br>13. $max\_count \leftarrow \text{count}(max\_action)$<br>14. **for** 每个状态-动作对 $(s,a')$ **do**<br>15. **if** $a' == max\_action$ **then**<br>16. $\pi(a' \mid s) = 1 \big/ (max\_count)$<br>17. **else**<br>18. $\pi(a' \mid s) = 0$<br>19. **end if**<br>20. **end for**<br>21. **end for** |
| 输　出 | $v_* = V$，$\pi_* = \pi$ |

与策略迭代类似，也可以直接基于动作值函数进行值迭代，迭代公式为：

$$Q(s,a) = \sum_{s',r} p(s',r \mid s,a)\left[ r + \gamma \max_{a'} Q(s',a') \right] \tag{4.11}$$

算法 4.6 给出了在有穷状态空间 MDP 中，基于动作值函数的值迭代算法。

**算法 4.6** 基于动作值函数的值迭代算法

| | |
|---|---|
| 输　入 | 动态性 $p$，奖赏函数 $r$，折扣因子 $\gamma$ |
| 初始化 | 1. 对任意 $s \in \mathcal{S}$，$a \in \mathcal{A}(s)$，初始化动作值函数，如 $Q(s,a)=0$<br>2. 阈值 $\theta$ 设置为一个较小的实值 |
| 评　估<br>过　程 | 3. **repeat** 对每一轮值迭代 $l=1,2,\cdots$<br>4. $delta \leftarrow 0$ |

| | |
|---|---|
| | 5. **for** 每个状态-动作对 $(s,a)$ **do**<br>6. $q \leftarrow Q(s,a)$<br>7. $Q(s,a)=\sum_{s',r} p(s',r \mid s,a)\left[r+\gamma \max_{a'} Q(s',a')\right]$<br>8. $delta \leftarrow \max(delta, \mid q-Q(s,a) \mid)$<br>9. **until** $\Delta < \theta$ |
| 最 优<br>策 略 | 10. **for** 每个状态 $s$ **do**<br>11. $max\_ction \leftarrow \arg\max_a Q(s,a)$； $max\_count \leftarrow \text{count}(\max(Q(s,a)))$<br>12. **for** 每个状态-动作对 $(s,a')$ **do**<br>13. **if** $a' == max\_action$ **then**<br>14. $\pi(a' \mid s)=1 \diagup (max\_count)$<br>15. **else**<br>16. $\pi(a' \mid s)=0$<br>17. **end if**<br>18. **end for**<br>19. **end for** |
| 输 出 | $q_* = Q$， $\pi_* = \pi$ |

算法 4.5 和算法 4.6 既可用于确定环境 MDP 又可用于随机环境 MDP。

**例 4.8** 将基于状态值函数的值迭代算法 4.5 应用于例 4.1 的确定环境扫地机器人任务中，可以得到以下状态值迭代更新过程：

（1）当 $l=0$ 时（$l$ 为值迭代的次数），对于所有的 $s$ 初始化为：$V(s)=0$；

（2）当 $l=1$ 时，以状态 $S_{24}$ 为例。在策略 $\pi$ 下，只能采取向下和向左 2 个动作，概率各为 0.5。采取向下的动作时，到达状态 $S_{19}\left(V(S_{19})=0\right)$，并可以捡到垃圾，获得 $r_1=+3$ 的奖赏；采取向左的动作时，到达状态 $S_{23}\left(V(S_{23})=2.40\right)$，获得 $r_2=0$ 的奖赏。根据算法 4.2 计算得：

$$\begin{aligned} V(S_{24}) &= \max\left(r_1+\gamma V(S_{19}), r_2+\gamma V(S_{23})\right) \\ &= \max(3+0.8*0, 0+0.8*2.40) \\ &= \max(3, 1.92) \\ &= 3.00 \end{aligned} ;$$

同理可以计算状态 $S_5$ 和 $S_6$ 的状态值函数：

$$\begin{aligned} V(S_5) &= \max\left(r_1+\gamma V(S_{10}), r_2+\gamma V(S_0), r_3+\gamma V(S_6)\right) \\ &= \max(0+0.8*0, 1+0.8*0, 0+0.8*0) \\ &= 1.00 \end{aligned} ;$$

$$\begin{aligned} V(S_6) &= \max\left(r_1+\gamma V(S_{11}), r_2+\gamma V(S_1), r_3+\gamma V(S_5), r_4+\gamma V(S_7)\right) \\ &= \max(0+0.8*0, 0+0.8*1.00, 0+0.8*1.00, 0+0.8*0) \\ &= 0.80 \end{aligned} ;$$

按顺序计算完一轮后，得到值函数$V(s)$，如图 4.7（$\ell=1$）所示。

（3）当$l=2$时，以状态$S_{22}$、$S_{23}$、$S_{24}$为例，计算状态值函数。异步计算方式，通常与迭代的计算顺序有关，根据例 4.1 规定，在每一轮次中，这 3 个状态的计算顺序为$S_{22}$、$S_{23}$、$S_{24}$。

$$\begin{aligned}V(S_{22}) &= \max\left(r_1+\gamma V(S_{17}), r_2+\gamma V(S_{21}), r_3+\gamma V(S_{23})\right)\\ &= \max(0+0.8*2.40, 0+0.8*0.41, 0+0.8*2.40)\\ &= \max(1.92, 0.33, 1.92)\\ &= 1.92\end{aligned}$$；

$$\begin{aligned}V(S_{23}) &= \max\left(r_1+\gamma V(S_{18}), r_2+\gamma V(S_{22}), r_3+\gamma V(S_{24})\right)\\ &= \max(0+0.8*3.00, 0+0.8*1.92, 0+0.8*3.00)\\ &= \max(2.40, 1.54, 2.40)\\ &= 2.40\end{aligned}$$；

$$\begin{aligned}V(S_{24}) &= \max\left(r_1+\gamma V(S_{19}), r_2+\gamma V(S_{23})\right)\\ &= \max(3+0.8*0.00, 0+0.8*2.40)\\ &= 3.00\end{aligned}$$

按顺序计算完一轮后，得到值函数$V(s)$，如图 4.7（$\ell=2$）所示。

（4）当$l=6$时，$|v_l(s)-v_{l-1}(s)|_\infty<\theta$，认为$v_l(s)$已经收敛于$v_*(s)$，计算得到的$v_*(s)$就是最优状态值函数。

图 4.7 为确定环境扫地机器人任务通过状态值迭代得到的结果。

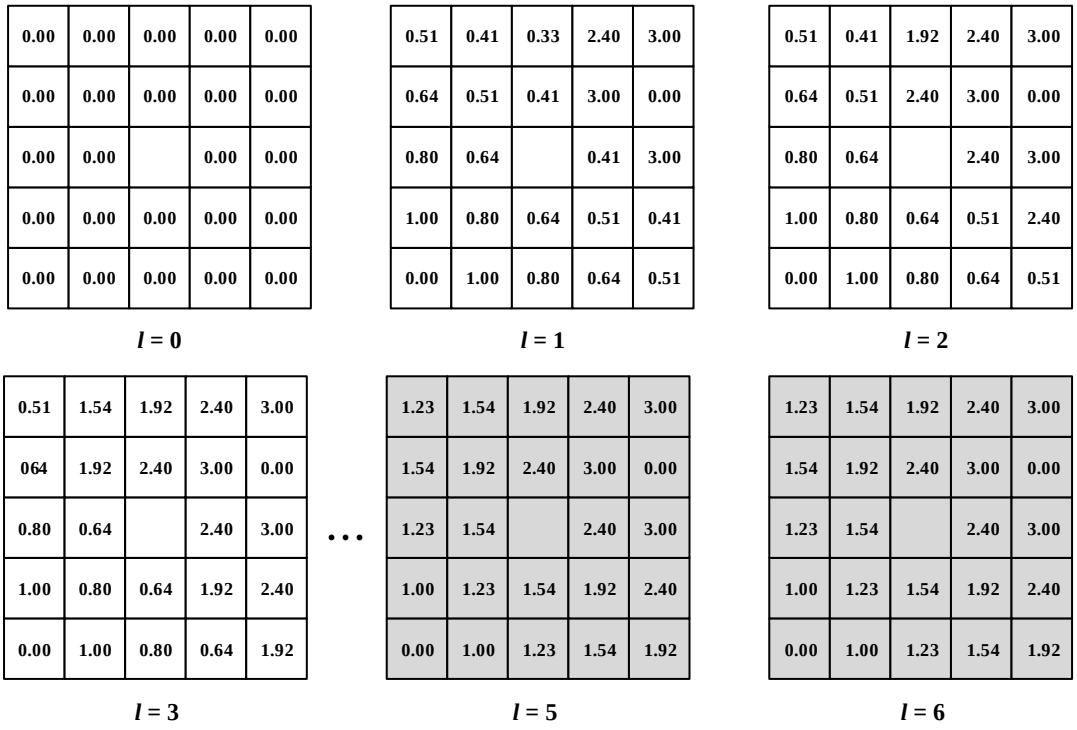



图 4.7 面向扫地机器人任务的基于状态值函数的值迭代更新过程

表 4.5 和表 4.6 分别给出了通过异步计算方式得到的，基于状态值函数和动作值函数的扫地机器人任务的值迭代更新过程。

表 4.5 面向确定环境扫地机器人任务的基于状态值函数的值迭代更新过程

| | $S_1$ | $S_2$ | $S_3$ | … | $S_7$ | … | $S_{24}$ |
|---|---|---|---|---|---|---|---|
| $v_0$ | 0.00 | 0.00 | 0.00 | … | 0.00 | … | 0.00 |
| $v_*$ | 1.00 | 1.23 | 1.54 | … | 1.54 | … | 3.00 |
| $q_*$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.98;-10.00 | … | ×;3.00;1.92;0.00 |
| $\pi_*$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |

表 4.5 面向确定扫地机器人任务的基于动作值函数的值迭代更新过程

| | $S_1$ | $S_2$ | $S_3$ | … | $S_7$ | … | $S_{24}$ |
|---|---|---|---|---|---|---|---|
| $q_0$ | 0.00;×;0.00;0.00 | 0.00;×;0.00;0.00 | 0.00;×;0.00;0.00 | … | 0.00;0.00;0.00;0.00 | … | ×;0.00; 0.00;0.00 |
| $q_*$ | 0.98;×;1.00;0.98 | 1.23;×;0.80;1.23 | 1.54;×;0.98;1.54 | … | 1.54;0.98;0.98;-10.00 | … | ×;3.00;1.92;0.00 |
| $\pi_*$ | 0.00;0.00;1.00;0.00 | 1.00;0.00;0.00;1.00 | 1.00;0.00;0.00;1.00 | … | 1.00;0.00;0.00;0.00 | … | 0.00;1.00;0.00;0.00 |

虽然策略迭代、值迭代等DP方法可以有效解决强化学习问题，但它仍然存在较多缺点：

（1）在进行最优策略计算时，必须知道状态转移概率 $p$。

（2）DP 的推演是整个树状展开的，计算量大，存储消耗资源多。

（3）每次回溯，所有可能的下一状态和相应动作都要被考虑在内，存在维度灾难问题。

（4）由于策略初始化的随机性，不合理的策略可能会导致算法无法收敛。

**例 4.9** 赌徒问题 一个赌徒投掷一个骰子来累加骰子点数之和。赌徒可以选择重新投掷骰子或者结束整局游戏。如果选择结束整局游戏骰子总和数刚好 18 点则赌徒赢得 10 元，骰子点数总和超过了 18 则输掉（骰子点数总数-18）的资金，少于 18 则输掉（$\frac{18-\text{骰子点数总数}}{2}$）的资金。当点数超过或者等于 18 时，会自动结束整局游戏。

利用值迭代方法，在折扣率 $\gamma=0.9$ 时，计算赌徒在不同状态最优策略。

该问题的 MDP 模型为：

（1）状态空间：当前赌徒共扔掷了骰子点数的总和，即 $\mathcal{S}=\{0,1,2,3,4,5...,10,..,23\}$ 共 24 个状态。

（2）动作空间：赌徒可以选择重新投掷骰子或是结束整局比赛，即两个动作 $\mathcal{A}(s)=\{0,1\}$ 0 代表结束游戏，1 代表掷骰子。

（3）状态转移：即在当前状态 $s$ 下，采取动作 a，到达下一状态 $s'$ 的概率。假设 $s=1$ 时执行了动作 1。那么 $s'$ 有可能的状态是 $\{2,3,4,5,6,7\}$ 共 6 种状态，每个状态的概率为 $\frac{1}{6}$ 。但

当 $s \geq 36$，会自动执行动作 0 来结束整个回合并获得回报。故状态转移函数可写成：

$$p(s,a,s') = \begin{cases} \frac{1}{6}, & a=1,\ s \neq [18,23] \text{且} s' \in [s+1, s+6] \\ 1, & a=0,\ s=s' \\ 0, & \text{其他} \end{cases}$$

（4）立即奖赏：赌徒重新投掷筛子会获得 0 的立即奖赏。当整局比赛结束，立即回报

$$r = \begin{cases} -(s-18), & s>18 \\ 10, & s=18 \\ -\frac{(18-s)}{2}, & s<18 \end{cases}$$

（5）折扣因子：$\gamma = 0.9$。

### 4.3 广义策略迭代

在策略迭代算法中，策略评估与策略改进两个流程交替进行，且每个流程都在另一个开始前完成，这样的方法被称为经典策略迭代（classical policy iteration）。广义策略迭代（Generalized Policy Iteration，GPI）则体现了策略评估与策略改进交替进行的一般性，强调策略评估和策略改进的交互关系，而不关心策略评估到底迭代了多少次，或具体的策略评估和策略改进的细节。在 GPI 中，策略评估没结束，就可以进行策略改进，只要这两个过程都能不断地更新，就能收敛到最优值函数和最优策略。从这一角度看，值迭代也属于 GPI，而实际上几乎所有的强化学习方法都可以被描述为 GPI。

GPI 体现了评估和改进之间相互竞争与合作的关系：基于贪心策略，使得值函数与当前策略不匹配，而保持值函数与策略一致就无法更新策略。在长期的博弈后，两个流程会趋于一个目标，即最优值函数和最优策略。

策略总是基于特定的值函数进行改进的，值函数始终会收敛于对应的特定策略的真实值函数，当评估和改进都稳定时，贝尔曼最优方程便可成立，此时得到最优值函数和最优策略。换句话说，值函数只有与当前策略一致时才稳定，且策略只有是当前值函数的贪心策略时才稳定。

### 4.4 小结

在环境已知的前提下，基于马尔可夫决策过程，动态规划可以很好的完成强化学习任务。策略评估通常对于给定的策略，不断迭代计算每个状态（或状态-动作对）的价值。其迭代方法主要是利用对后继状态（或状态-动作对）价值的估计，来更新当前状态（或状态-动作对）价值的估计，也就是用自举的方法。策略改进是采用贪心算法，利用动作值函数获得更优的策略，每次都选择最好的动作。策略迭代是重复策略评估和策略改进的迭代，直到策略收敛，找到最优的策略。但是策略迭代需要多次使用策略评估才能得到收敛的状态（或状态-动作对）值函数，即策略评估是迭代进行的，只有在 $v_\pi$ 收敛时，才能停止迭代。值迭代无

需等到其完全收敛，提早的计算出贪心策略，截断策略评估，在一次遍历后即刻停止策略评估，并对每个状态进行更新。实践证明，值迭代算法收敛速度优于策略迭代算法。广义策略迭代则体现了策略评估与策略改进交替进行的一般性。在 GPI 中，策略评估和策略改进同时进行，只要这两个过程都能不断地更新，就能收敛到最优值函数和最优策略。从这一角度看，值迭代也属于 GPI，而实际上几乎所有的强化学习方法都可以被描述为 GPI。