强化学习原理

第四章读书笔记

姓名：石若川 学号：2111381 专业：智能科学与技术

4 动态规划

对于强化学习问题，传统的 DP 算法的作用有限。其原因有二: 一是完备的环境模型只是一个假设；二是它的计算复杂度极高。但是，它依然是一个非常重要的理论。事实上，所有其他方法都是对 DP 的一种近似，只不过降低了计算复杂度以及减弱了对环境模型完备性的假设。

本章中假设环境是一个有限 MDP，假设状态集合 $\mathcal{S}$、动作集合 $\mathcal{A}$ 和收益集合 $\mathcal{R}$ 是有限的，并且整个系统的动态特性由对于任意 $s \in \mathcal{S}$、$a \in \mathcal{A}$、$r \in \mathcal{R}$ 和 $s^{'} \in \mathcal{S}^{+}$($S^{+}$ 表示在分幕式任务下 S 加上一个终止状态) 的四参数概率分布 $p(s', r|s, a)$ 给出。

在强化学习中，DP 的核心思想是使用价值函数来结构化地组织对最优策略的搜索。对于任意 $s \in \mathcal{S}$、$a \in \mathcal{A}$ 和 $s^{'} \in \mathcal{S}^{+}$，有

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

$$\begin{aligned} q_*(s, a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \Big| S_t = s, A_t = a\Big] \\ &= \sum_{s'.r} p(s', r|s, a)\Big[r + \gamma \max_{a'} q_*(s', a')\Big] \end{aligned} \tag{2}$$

4.1 策略评估

策略评估问题：对于任意一个策略 π，如何计算其状态价值函数 v_π。对于任意 $s \in \mathcal{S}$

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi[G_t \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \qquad (3) \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_\pi(s')\Big] \qquad (4) \end{aligned}$$

只要 $\gamma < 1$ 或者任何状态在 π 下都能保证最后终止，那么 v_π 唯一存在。如果环境的动态特性完全已知，那么式4就是一个有着 $|S|$ 个未知数 $(v_\pi(s), s \in S)$ 以及 $|S|$ 个等式的联立线性方程组。理论上，这

个方程的解可以被直接解出，但是过于繁琐。使用迭代法求解

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi[R_{t+1} + \gamma v_k(S_{t+1}) | S_t = s] \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_k(s') \Big] \end{aligned} \tag{5}$$

在保证 v_π 存在的条件下，序列 v_k 在 $k \to \infty$ 是将会收敛到 v_π。

迭代策略评估算法，用于估计 $V \approx v_\pi$

输入待评估的策略 π
算法参数：小阈值 $\theta > 0$, 用于确定估计量的精度
对于任意 $s \in S^+$, 任意初始化 $V(s)$, 其中 V (终止状态) =0
循环：
  $\Delta \leftarrow 0$
  对每一个 $s \in S$ 循环：
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
直到 $\Delta < \theta$

4.2 策略改进

$$\begin{aligned} q_\pi(s, a) &\doteq \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r|s, a) \Big[r + \gamma v_\pi(s') \Big]. \end{aligned} \tag{6}$$

策略改进定理：一般来说，如果 π 和 π' 是任意的两个确定的策略，对任意 $s \in \mathcal{S}$,

$$q_\pi(s, \pi'(s)) \geqslant v_\pi(s) \tag{7}$$

那么我们称策略 π' 相比于 π 一样好或者更好。也就是说，对任意状态 $s \in S$, 这样肯定能得到一样或更好的期望回报

$$v_{\pi'}(s) \geqslant v_\pi(s) \tag{8}$$

并且，如果式7中的不等式在某个状态下是严格不等的，那么式8在这个状态下也会是严格不等的。

证明过程：

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

给定一个策略及其价值函数，可以很容易评估一个状态中某个特定动作的改变会产生怎样的后果，在每个状态下根据 $q_\pi(s, a)$ 选择一个最优的。换言之，考虑一个新的贪心策略 $\pi_\prime$，满足

$$
\begin{aligned}
\pi'(s) &\doteq \arg\max_a q_\pi(s, a) \\
&= \operatorname{argmax}_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a)\Big[r + \gamma v_\pi(s')\Big]
\end{aligned} \tag{9}
$$

这个贪心策略采取在短期内看上去最优的动作，即根据 v_π 向前单步搜索。这样构造出的贪心策略满足策略改进定理的条件7，它和原策略相比一样好，甚至更好。这种根据原策略的价值函数执行贪心算法，来构造一个更好策略的过程，称为策略改进。

假设新的贪心策略 π' 和原有的策略 π 一样好，但不是更好。那么一定有 $v_\pi = v_{\pi'}$，再通过式9可以得到，对任意 $s \in \mathcal{S}$

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

但这和贝尔曼最优方程1完全相同，因此 $v_{\pi'}$ 一定与 v_* 相同，而且 π 与 π' 均必须为最优策略。因此在除了原策略即为最优策略的情况下，策略改进一定会给出一个更优的结果。对于随机策略而言，该策略改进定理同样成立，只要满足所有其他非最优动作的概略和为零即可。

4.3 策略迭代

一旦一个策略 π 根据 v_π 产生了一个更好的策略 π'，我们就可以通过计算 $v_{\pi'}$ 来得到一个更优的策略 π''。这样一个链式的方法可以得到一个不断改进的策略和价值函数的序列

$$
\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{1} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{1} \pi_2 \xrightarrow{E} \cdots \xrightarrow{1} \pi_* \xrightarrow{E} v_*,
$$

这里 $\xrightarrow{E}$ 代表策略评估，而 $\xrightarrow{1}$ 表示策略改进。每一个策略都能保证比前一个更优 (除非前一个已经是最优的)。由于一个有限 MDP 必然只有有限种策略，所以在有限次的迭代后，这种方法一定收敛到一个最优的策略与最优价值函数。这种寻找最优策略的方式成为策略迭代。

算法（使用迭代策略评估），用于估计 $\pi \approx \pi_*$

输入待评估的策略 π

1. 初始化

对 $s \in S$, 任意设定 $V(s) \in \mathbb{R}$ 以及 $\pi(s) \in \mathcal{A}(s)$

2. 策略评估

循环:

$\Delta \leftarrow 0$

对每一个 $s \in S$ 循环:

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

直到 $\Delta < \theta$ (一个决定估计精度的小正数)

3. 策略改进

policy-stable $\leftarrow$ *true*

对每一个 $s \in S$:

$old - action \leftarrow \pi(s)$

$\pi(s) \leftarrow \text{argmax}_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

如果 *old-action* $\neq \pi(s)$, *policy-stable* $\leftarrow$ *false*

如果 *policy* $-$ *stable* 为 true, 那么停止并返回 $V \approx v_*$ 以及 $\pi \approx \pi_*$; 否则跳转到 2

4.4 价值迭代

策略迭代算法的一个缺点是每一次迭代都涉及了策略评估，这本身就是一个需要多次遍历状态集合的迭代过程。如果策略评估是迭代进行的，那么收敛到 v_π 理论上在极限处才成立。但是当未收敛时，策略可能已经是最优策略，所以可以提前阶段策略评估过程。

一种截断策略评估的算法为价值迭代，在一次遍历后立即停止策略评估（对每个状态进行一次更新）。对于任意 $s \in \mathcal{S}$

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r|s, a)\Big[r + \gamma v_k(s')\Big] \end{aligned} \tag{10}$$

在每一次遍历中，价值迭代都有效结合了策略评估的遍历和策略改进的遍历。在每次策略改进遍历的迭代中间进行多次策略评估遍历经常收敛得更快。一般来说，可以将截断策略迭代算法看作一系列的遍历序列，其中某些进行策略评估更新，而另一些则进行价值迭代更新。

价值迭代算法，用于估计 $\pi \approx \pi_*$

算法参数：小阈值 $\theta > 0$，用于确定估计量的精度
对于任意 $s \in \mathcal{S}^+$，任意初始化 $V(s)$，其中 $V(\text{终止状态}) = 0$
循环：
  $\Delta \leftarrow 0$
  对每一个 $s \in S$ 循环：
    $v \leftarrow V(s)$
    $V(s) \leftarrow max_a \sum_{s',r} p(s', r|s, \pi(s))\big[r + \gamma V(s')\big]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
直到 $\Delta < \theta$
输出一个确定的 $\pi \approx \pi_*$，使得
$\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\big[r + \gamma V(s')\big]$

4.5 异步动态规划

异步 DP 算法是一类就地迭代的 DP 算法，其不以系统遍历状态集的形式来组织算法。这些算法使用任意可用的状态值，以任意顺序来更新状态值。在某些状态的值更新一次之前，另一些状态的值可能已经更新了好几次。然而，为了正确收敛，异步算法必须要不断地更新所有状态的值：在某个计算节点后，它不能忽略任何一个状态。在选择要更新的状态方面，异步 DP 算法具有极大的灵活性。

异步算法还使计算和实时交互的结合变得更加容易。为了解决给定的 MDP 问题，我们可以在一个智能体实际在 MDP 中进行真实交互的同时，执行迭代 DP 算法。这个智能体的经历可用于确定 DP 算法要更新哪个状态。与此同时，DP 算法的最新值和策略信息可以指导智能体的决策。比如，我们可以在智能体访问状态时将其更新。这使得将 DP 算法的更新聚焦到部分与智能体最相关的状态集成为可能。这种聚焦是强化学习中不断重复的主题。

4.6 广义策略迭代

广义策略迭代 (GPI) 一词指代让策略评估和策略改进相互作用的一般思路，与这两个流程的粒度和其他细节无关。几乎所有的强化学习方法都可以被描述为 GPI。也就是说，几乎所有方法都包含明确定义的策略和价值函数，且如图4.1所示，策略总是基于特定的价值函数进行改进，价值函数也始终会向对应特定策略的真实价值函数收敛。如果评估流程和改进流程都很稳定，即不再产生变化，那么价值函数和策略必定是最优的。价值函数只有在与当前策略一致时才稳定并且策略只有在对当前价值函数是贪心策略时才稳定。因此，只有当一个策略发现它对自己的评估价值函数是贪心策略时，这两个流程才能稳定下来。这意味着贝尔曼最优方程1成立，也因此这个策略和价值函数都是最优的。

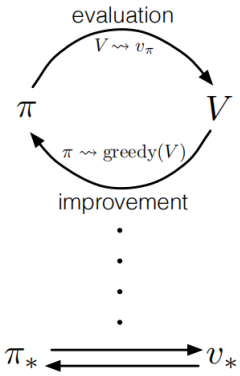

图 4.1: 策略与价值函数之间的关系

可以将 GPI 的评估和改进流程看作竞争与合作。竞争是指它们朝着相反的方向前进。让策略对价值函数贪心通常会使价值函数与当前策略不匹配，而使价值函数与策略一致通常会导致策略不再贪心。然而，从长远来看，这两个流程会相互作用以找到一个联合解决方案: 最优价值函数和一个最优策略。

也可以将 GPI 的评估和改进流程视为两个约束或目标之间的相互作用的流程，如图4.2所示的二维空间中的两条线。每个流程都把价值函数或策略推向其中的一条线，该线代表了对于两个目标中的某一个目标的解决方案。因为这两条线不是正交的，所以两个目标之间会产生相互作用。该图中的箭头对应于策略迭代的行为，即每个箭头都表示系统每次都会完全地达到其中一个目标。评估和改进这两个流程结合在一起，就可以最终达到最优的总目标。

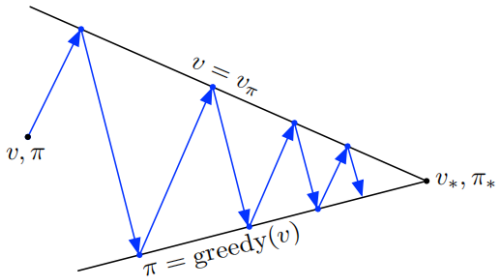

图 4.2: 评估与改进相互作用

4.7 动态规划的效率

DP 算法和其他解决马尔可夫决策过程的方法相比，DP 算法实际上效率很高。动态规划算法找到最优策略的时间 (最坏情况下) 与状态和动作的数量是呈多项式级关系的。如果我们分别用 n 和 k 来表示状态和动作的数量，则意味着一个 DP 方法所需要的计算操作的数量少于 n 与 k 的某个多项式函数。即使 (确定) 策略的总量达到 k^n，DP 算法也能保证在多项式时间内找到一个最优策略。从这个意义上来说，DP 算法的效率指数级优于任何直接在策略空间搜索的方法，这是因为，为了找到最优策略，直接搜索的方法需要穷举每一种策略。线性规划方法同样可以用于解决马尔可夫决策过程，且在某些例子中，它们在最坏情况下的收敛是优于 DP 方法的。但是当状态数量较少时，线性规划方法就不如 DP 方法实用 (大约需要 100 倍的时间)。对那些极大规模的问题，只有动态规划算法是可行的。

DP 算法有时会由于维度灾难而被认为缺乏实用性。维度灾难指的是，状态的总数量经常随着状态变量的增加而指数级上升。状态集合太大的确会带来一些困难，但是这些是问题本身的困难，而非 DP 作为一种解法所带来的困难。事实上，相比于直接搜索和线性规划，DP 更加适合于解决大规模状

态空间的问题。

4.8 总结

在这一章中，我们熟悉了利用动态规划方法解决有限马尔可夫决策过程的基本思想和算法。策略评估通常指的是对一个策略的价值函数进行迭代计算。策略改进指的是给定某个策略的价值函数，计算一个改进的策略。将这两种计算统一在一起，我们得到了策略迭代和价值迭代，这也是两种最常用的 DP 算法。这两种方法中的任意一种都可以在给定马尔可夫决策过程完整信息的条件下，计算出有限马尔可夫决策过程的最优策略和价值函数。

传统的 DP 方法要遍历整个状态集合，并对每一个状态进行期望更新操作。每一次操作都基于所有可能的后继状态的价值函数以及它们可能出现的概率，来更新一个状态的价值。当更新对结果不再产生数值上的变化时，就表示已经收敛并满足对应的贝尔曼方程。

GPI 的主要思想是采用两个相互作用的流程围绕着近似策略和近似价值函数循环进行。一个流程根据给定的策略进行策略评估，使得价值函数逼近这个策略真实的价值函数。另一个流程根据给定的价值函数进行策略改进，在假设价值函数固定的条件下，使得这个策略更好。尽管每一个流程都改变了另一个流程，但它们的相互作用总体上可以一起找到一个联合的解：不会被任意一个流程改变的策略和价值函数，即最优解。

有时我们并不需要用 DP 算法来遍历整个状态集合。异步 DP 算法就是一种以任意顺序更新状态的就地迭代方法，更新顺序甚至可以随机确定并可以使用过时的信息。其中许多方法可以被看作 GP 的细粒度形式。

最后，我们注意到 DP 算法的一个特殊性质。所有的方法都根据对后继状态价值的估计，来更新对当前状态价值的估计。也就是说，它们基于其他估计来更新自己的估计。我们把这种普遍的思想称为自举法。