强化学习经典算法解读：价值迭代与策略迭代

szq (根据 PPT 内容整理)

2025 年 7 月 26 日

目录

1 引言

本文档根据提供的 PPT 内容，详细解读了强化学习中两种经典的动态规划（Dynamic Programming, DP）算法：**价值迭代（Value Iteration）**和**策略迭代（Policy Iteration）**。这两种算法都是在已知环境模型（即状态转移概率 P 和奖励 R）的前提下，求解马尔可夫决策过程（MDP）最优策略和最优价值函数的核心方法。

2 价值迭代 (Value Iteration)

2.1 核心思想与算法引出

价值迭代的目标是求解**贝尔曼最优方程 (Bellman Optimality Equation)**：

$$v^*(s) = \max_{a\in\mathcal{A}} \sum_{s',r} p(s',r|s,a)[r + \gamma v^*(s')]$$

在向量形式下，可以写作：

$$\mathbf{v} = \max_{\pi}(\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v})$$

由于该方程中存在 max 算子，导致其为非线性方程组，无法直接求解。

根据**收缩映射定理 (Contraction Mapping Theorem)**，我们可以将贝尔曼最优方程的右侧视为一个更新算子，通过反复迭代来逼近最优解。这便引出了价值迭代算法。

2.2 算法描述

价值迭代从一个任意的初始价值函数 $\mathbf{v}_0$ 开始，通过以下公式进行迭代更新：

$$\mathbf{v}_{k+1} = \max_{\pi}(\mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_k), \quad k = 0, 1, 2, \ldots$$

这个过程会持续进行，直到价值函数的变化足够小（即 $\|\mathbf{v}_{k+1} - \mathbf{v}_k\| < \epsilon$），此时的 $\mathbf{v}_{k+1}$ 就近似为最优价值函数 $\mathbf{v}^*$。

2.3 算法分解与元素形式

价值迭代的每一步更新 ‘v$_{k+1}(s) = ...$‘

Step 1: **策略更新 (Policy Update) (隐式)**

在这一步，我们基于当前的价值估计 $\mathbf{v}_k$，找到一个临时的贪心策略 π_{k+1}。对于每一个状态 $s \in \mathcal{S}$，该策略选择能够最大化期望回报的动作：

$$\pi_{k+1}(s) = \underset{a\in\mathcal{A}}{\operatorname{argmax}} \left\{ \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')] \right\}$$

我们通常将括号内的部分定义为动作价值函数 $q_k(s,a)$:

$$q_k(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')]$$

因此，策略更新可以简化为：$\pi_{k+1}(s) = \text{argmax}_a\, q_k(s,a)$。

Step 2: **价值更新 (Value Update)**
接着，我们使用上一步找到的贪心策略来更新状态的价值。由于该策略总是选择最优动作，所以新的价值就是该状态下所有动作价值中的最大值：

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} q_k(s,a)$$

重要说明：在迭代过程中（收敛前），$\mathbf{v}_k$ 并不是任何一个固定策略下的真实价值函数。它仅仅是通往最优价值函数 $\mathbf{v}^*$ 的一个中间估计值。

3 策略迭代 (Policy Iteration)

策略迭代是另一种求解 MDP 的经典算法。与价值迭代直接优化价值函数不同，策略迭代在**策略空间**中进行搜索，通过交替评估和改进策略来找到最优策略。

3.1 算法描述

算法从一个随机的初始策略 π_0 开始，循环执行以下两个步骤，直到策略不再发生变化。

Step 1: **策略评估 (Policy Evaluation, PE)**
对于当前固定的策略 π_k，精确地计算其对应的状态价值函数 $\mathbf{v}_{\pi_k}$。这需要求解**贝尔曼期望方程**:

$$\mathbf{v}_{\pi_k} = \mathbf{r}_{\pi_k} + \gamma \mathbf{P}_{\pi_k} \mathbf{v}_{\pi_k}$$

这是一个线性方程组。对于大型问题，通常使用迭代法求解：

$$\mathbf{v}_{\pi_k}^{(j+1)} = \mathbf{r}_{\pi_k} + \gamma \mathbf{P}_{\pi_k} \mathbf{v}_{\pi_k}^{(j)}$$

这个内部迭代会进行直到 $\mathbf{v}_{\pi_k}^{(j)}$ 收敛，得到精确的 $\mathbf{v}_{\pi_k}$。

Step 2: **策略改进 (Policy Improvement, PI)**
利用上一步计算出的精确价值函数 $\mathbf{v}_{\pi_k}$，找到一个更好的新策略 π_{k+1}。这个新策略对于每个状态 s 都是贪心的：

$$\pi_{k+1}(s) = \underset{a \in \mathcal{A}}{\text{argmax}} \left\{ \sum_{s',r} p(s',r|s,a)[r + \gamma v_{\pi_k}(s')] \right\}$$

根据**策略改进定理**，这样得到的新策略 π_{k+1} 总是优于或等于旧策略 π_k。

这个过程 ‘$\pi_0 \xrightarrow{PE} \mathbf{v}_{\pi_0} \xrightarrow{PI} \pi_1 \xrightarrow{PE} \mathbf{v}_{\pi_1} \xrightarrow{PI} \dots$‘ 会持续进行，直到 $\pi_{k+1} = \pi_k$。此时，算法收敛，找到了最优策略 π^* 和最优价值函数 $\mathbf{v}^*$。

4 总结与对比

价值迭代和策略迭代是解决 MDP 问题的两种 foundational 算法。它们都保证收敛到最优解，但在计算过程和效率上有所不同。

表 1: 价值迭代与策略迭代的对比

特性	**价值迭代 (Value Iteration)**	**策略迭代 (Policy Iteration)**
迭代对象	价值函数 $\mathbf{v}_k$	策略 π_k
核心操作	一步完成价值更新和隐式策略改进。 $\mathbf{v}_{k+1} = \max_a Q(s, a; \mathbf{v}_k)$	分为两步：策略评估和策略改进。 1. PE: 求解线性方程组 (开销大)。 2. PI: 贪心更新策略 (开销小)。
每次迭代的复杂度	计算简单，只需遍历所有状态和动作。	计算复杂，PE 步骤本身需要一个完整的迭代过程直至收敛。
收敛速度	通常需要较多次迭代才能收敛。	外部循环（策略改进）的迭代次数通常很少，能很快找到最优策略。
联系	可看作“截断的”策略迭代，即 PE 步骤只迭代一次。	每次改进策略前，都要求对当前策略的价值进行充分评估。

理解这两种算法的原理、区别与联系，对于学习后续更高级的强化学习算法，如 Q-Learning 和 Actor-Critic 等，至关重要。