强化学习笔记：价值函数与贝尔曼方程

State Value (状态价值函数)

- **状态价值 (State Value)**：在遵循策略 π 的前提下，从状态 s 出发所能获得的**期望回报 (Expected Return)**。
- **回报 (Return)**：从某个时刻开始，后续所有奖励的折扣总和，这是一条具体轨迹的产出。
- **价值 (Value)**：对所有可能轨迹的**回报**求期望，是一个统计量。

> **状态价值函数 (State-Value Function)** 定义为： $$ v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] $$ 其中：
>
> - $v_\pi(s)$：在策略 π下，状态s的价值。
> - G_t：从时刻 t 开始的总折扣回报，即 $G_t = \sum_{k=0}^\infty \gamma^k R_{t+k+1}$。
> - $\mathbb{E}_\pi[\cdot]$: 表示在策略 π 下的期望。

Bellman Equation (贝尔曼方程)

贝尔曼方程建立了当前状态的价值与后继状态价值之间的递归关系。

状态价值函数的贝尔曼展开

状态价值函数可以分解为**立即奖励 (Immediate Reward)** 和**后继状态的折扣价值 (Discounted Value of Successor State)**。

$$ v_\pi(s) = \mathbb{E}|pi |left[R{t+1} + \gamma G_{t+1} \mid S_t = s \right] $$

通过对动作 a 和后继状态 s'进行边缘化，可以得到贝尔曼方程的完整形式:

> **贝尔曼方程 (Bellman Equation for v_π)** $$ v_\pi(s) = \sum_a \pi(a \mid s) \left(\mathcal{R}s^a + |gamma |sum{s'} \mathcal{P}{ss'}^a v\pi(s') \right) $$ 或者写成期望形式: $$ v_\pi(s) = \sum_a \pi(a \mid s) \mathbb{E} \left[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t=s, A_t=a \right] $$ 其中:
>
> - $\pi(a \mid s)$ 是在状态 s下选择动作 a 的概率。
> - $\mathcal{R}s^a = |mathbb{E}[R{t+1} \mid S_t=s, A_t=a]$ 是期望立即奖励。
> - $\mathcal{P}_{ss'}^a = p(s' \mid s, a)$ 是状态转移概率。

矩阵形式的贝尔曼方程

将贝尔曼方程写成矩阵形式，更便于分析和求解。

首先，为所有状态 $s \in \mathcal{S}$ 写出方程: $$ v_\pi(s) = \underbrace{\sum_a \pi(a \mid s) \mathcal{R}s^a}{\text{期望立即奖励}} + \gamma \underbrace{\sum_a \pi(a \mid s) \sum_{s'} \mathcal{P}{ss'}^a v\pi(s')}_{\text{期望未来价值}} $$

定义:

- **价值向量** $\mathbf{v}|pi$：一个列向量，其中第 s 个元素是$v\pi(s)$。

- **奖励向量** $\mathbf{r}_\pi$：一个列向量，其中第 s 个元素是 $r_\pi(s) = \sum_a \pi(a \mid s) \mathcal{R}_s^a$。
- **转移概率矩阵** $\mathbf{P}_\pi$: 一个$|\mathcal{S}| \times |\mathcal{S}|$ 的矩阵，其中元素 (s, s') 是 $P_\pi(s, s') = \sum_a \pi(a \mid s) p(s' \mid s, a)$。

代入后得到贝尔曼方程的矩阵形式:

> **贝尔曼方程(矩阵形式)** $$ \mathbf{v}_\pi = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}_\pi $$

这是一个线性方程组，可以直接求解： $$ (I - \gamma \mathbf{P}_\pi) \mathbf{v}_\pi = \mathbf{r}_\pi $$ $$ \mathbf{v}_\pi = (I - \gamma \mathbf{P}_\pi)^{-1} \mathbf{r}_\pi $$ 由于矩阵求逆的计算复杂度是 $O(|\mathcal{S}|^3)$，对于状态空间很大的问题，通常采用迭代法求解。

价值函数的迭代求解方法

1. 策略评估 (Policy Evaluation)

目标：计算给定策略 π 的价值函数v_π。 **方法**：使用贝尔曼方程作为迭代更新规则。

> **迭代更新公式**: 从任意初始值 $\mathbf{v}^{(0)}$ (通常为全零向量)开始，反复迭代: $$ \mathbf{v}^{(k+1)} = \mathbf{r}_\pi + \gamma \mathbf{P}_\pi \mathbf{v}^{(k)} $$ 直到收敛, 即 $\|\mathbf{v}^{(k+1)} - \mathbf{v}^{(k)}\|_\infty < \epsilon$。

2. 策略迭代 (Policy Iteration)

目标: 找到最优策略 π^*。 **方法**: 交替进行**策略评估**和**策略改进**。

1. **初始化**: 选择一个任意的初始策略 π_0。
2. **策略评估 (Policy Evaluation)**: 使用上述迭代法计算当前策略 π_k 的价值函数 v_{π_k}。
3. **策略改进 (Policy Improvement)**: 根据 v_{π_k} 生成一个新策略 π_{k+1}，该策略在每个状态下都选择能最大化期望回报的动作(贪心策略): $$ \pi_{k+1}(s) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v_{\pi_k}(s') \right) $$
4. **循环**: 如果 π_{k+1} 与 π_k 相同，则算法收敛，找到了最优策略和最优价值函数。否则，返回第2步。

3. 值迭代 (Value Iteration)

目标: 直接计算最优价值函数 v^*。 **方法**: 将策略评估和策略改进融合在一个更新步骤中。

> **值迭代更新公式**: 从任意初始值 $v^{(0)}$ 开始，反复迭代: $$ v^{(k+1)}(s) = \max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v^{(k)}(s') \right), \quad \forall s \in \mathcal{S} $$ 当 $v^{(k)}$ 收敛到 v^* 后，可以通过一次策略提取得到最优策略: $$ \pi^*(s) = \arg\max_a \left(\mathcal{R}_s^a + \gamma \sum_{s'} \mathcal{P}_{ss'}^a v^*(s') \right) $$

Action Value (动作价值函数)

- **动作价值 (Action Value)**: 在遵循策略 π 的前提下，于状态 s 执行动作 a 后，所能获得的期望回报。
- 它就是状态价值贝尔曼方程中，括号里的部分。

> **动作价值函数(Action-Value Function)** 定义为: $$ q_\pi(s, a) = \mathbb{E}\pi \left[G_t \mid S_t = s, A_t = a \right] $$ $$ q\pi(s, a) = \mathcal{R}s^a + \gamma \sum{s'} \mathcal{P}{ss'}^a v\pi(s') $$

状态价值和动作价值的关系:

- $v_\pi(s) = \sum_a \pi(a \mid s) q_\pi(s, a)$
- $q_\pi(s, a) = \mathcal{R}s^a + \gamma \sum{s'} \mathcal{P}{ss'}^a \sum{a'} \pi(a' \mid s') q_\pi(s', a')$

贝尔曼最优方程 (Bellman Optimality Equation)

贝尔曼最优方程描述了最优价值函数 v^* 和最优动作价值函数 q^* 之间的关系。

> **最优状态价值函数 v^***: $$ v^(s) = \max_a q^(s, a) $$
>
> **最优动作价值函数 q^***: $$ q^(s, a) = \mathcal{R}s^a + \gamma \sum{s'} \mathcal{P}_{ss'}^a v^(s') $$

将两者结合, 得到**贝尔曼最优方程**: $$ v^(s) = \max_a \left(\mathcal{R}s^a + \gamma \sum{s'} \mathcal{P}_{ss'}^a v^(s') \right) $$ $$ q^(s, a) = \mathcal{R}s^a + \gamma \sum{s'} \mathcal{P}{ss'}^a \max{a'} q^(s', a') $$

这个方程是非线性的(因为有`max`算子)，通常没有解析解，需要通过值迭代等方法求解。

不动点定理与收缩映射

1. 收缩映射与巴拿赫不动点定理

> **定义: 收缩映射 (Contraction Mapping)**
>
> 设 (X, d) 是一个完备度量空间, 若映射 $f: X \to X$ 满足: $$ d(f(x), f(y)) \leq k \cdot d(x, y), \quad \forall x, y \in X $$ 其中 $k \in [0, 1)$ 是一个常数，则称 f 是一个**收缩映射**。

> **巴拿赫不动点定理 (Banach Fixed-Point Theorem)**
>
> 在完备度量空间中，任意收缩映射 f 都存在**唯一**的不动点 x^*，使得: $$ f(x^) = x^* $$ 并且，从任意初始点 $x_0 \in X$ 出发，通过迭代 $x_{n+1} = f(x_n)$ 生成的序列 ${x_n}$ 必定收敛于不动点 x^*。

2. 贝尔曼最优算子

我们可以将贝尔曼最优方程看作一个算子 $\mathcal{T}$ 作用于价值函数 v: $$ (\mathcal{T}v)(s) = \max_a \left(\mathcal{R}s^a + \gamma \sum{s'} \mathcal{P}_{ss'}^a v(s') \right) $$ 这个**贝尔曼最优算子** $\mathcal{T}$ 是一个 γ-收缩映射。根据巴拿赫不动点定理:

1. 存在唯一的不动点 v^*，满足 $v^ = \mathcal{T}v^*$。这个不动点就是**最优状态价值函数**。
2. 值迭代算法 $v^{(k+1)} = \mathcal{T}v^{(k)}$ 保证收敛到唯一的 v^*。

3. 最优策略

一旦我们得到了最优价值函数 v^* 或 $q*^$，就可以确定最优策略 π^*。

> **最优策略 π^***
>
> 对于任意状态 $s \in \mathcal{S}$，存在一个确定性的最优策略 π^*，它选择能使 $q*^(s,a)$ 最大化的动作: $$ \pi^*(s) = \arg\max_a q*^(s, a) = \arg\max_a \left(\mathcal{R}*s^a + \gamma \sum*{s'} \mathcal{P}_{ss'}^a v^*(s') \right) $$ 这意味着，只要知道了最优价值函数，就可以通过**贪心**选择来获得最优行为。