强化学习

菠萝包萝卜

2025 年 10 月 18 日

1 基础概念

1.1 状态 State

Definition 1.1. 状态 (state) 是指智能体相对于环境的一个状态 s_i ，可能状态的全体 $\mathcal{S} = \{s_i\}_i$ 称为状态空间 (全集) state space。

1.2 行动 Action

Definition 1.2. 行动 (action) 是指在每个状态下有一些可以进行的行为，每个这样的行为称为当前状态的一个行动，用字母 a_j 表示；类似地，可能的行动全体 $\mathcal{A}(s_i) = \{a_j\}_j$ 称为状态 s_i 下的行动空间；值得一提的是每个状态下能采取的行动不一定是相同的，所以上面的定义式中是有个 s_i 的自变量的。

1.3 状态转移 State Transition

Definition 1.3. 当我们在一定状态下采取了某种行动，智能体会从当前状态转变为行动后的另一种状态，这样的一个过程就是状态转移；它实际上是定义了智能体和环境的一种交互行为。

状态转移可以用表格来表示，在某个 s_i 下做出了一个行动 a_j ，会得到一个新的 $s \in \mathcal{S}$ ，这样的一个二元函数通过表格变能表出，但是它只能表示确定性的情况，并不泛用.

如此，我们引进了状态转移概率 (State transition probability)，通过概率来表示状态转移。

1.4 策略 Policy

Definition 1.4. 在交互过程中会因不同的行为来产生不同的路径来得到结果，这些诸多路径中孰好孰坏我们暂时是不知道的。智能体会在某个状态选择一个行动以便达到目标，所有这些状态-行动对便组成了策略，换句话说策略会告诉智能体在某个状态下采取什么行动。

为了描述策略，我们使用数理表示，仍然可以利用概率来表示，一般我们用 $\pi(\cdot|\cdot)$ 来表示策略。

当然，由于策略给定了，我们在某状态下做出某种行动的概率也就给定了，这个时候这样的条件概率视为一个二元函数，它可以同时表示确定情况和不确定情况，然后做个表格就能表示出这样的策略全体。

1.5 奖励 Reward

奖励是强化学习中最重要的概念之一。

Definition 1.5. 所谓奖励 (reward)，是一个实数，它是在采取某种行动 (action) 后所得到的一个值，它满足:
如果奖励是正的，就代表鼓励这个行为的发生；
反之奖励是负的，就代表对这个行为的发生实施惩罚。

Remark. 如果奖励为零代表什么——就是对行为既不鼓励也不惩罚。

能否把正的奖励视为鼓励——事实上是可以的，如果正数代表惩罚，那么我们就要采取措施来得到更少的惩罚。

奖励是一种人机交互的手段，通过它可以引导智能体应该怎么做，不该怎么做。

1.5.1 表格表示法

对于某种状态，采取某种措施会获得一个 reward，我们把这些状态和这些措施列称表格，表格中间填充对应的 reward 值就能表示出这些情况的 reward。

1.5.2 数学表示法——条件概率

如果在状态 s_i ，我们选择了行动 a_j，得到的奖励（由状态和行动决定了）是 r_{cur}，那么我们就有

$$p(r = r_{cur}|s_i, a_j) = 1, p(r \neq r_{cur}|s_i, a_j) = 0$$

Remark. 刚才的说法其实是确定性的奖励；但是我们的奖励可能是随机的，举个例子，当你学习十分刻苦，你会在成绩上体现出 reward，但是这个 reward 具体是多少是不确定。

1.6 轨迹 Trajectory; 回报 Return

Definition 1.6. 轨迹 (trajectory) 是一个状态-行动-奖励的链，它表述了从一个状态经过若干次行动，每次行动都会得到奖励，并逐次转移到另外一个状态的的过程。

Definition 1.7. 回报 (return) 是在某一个轨迹中的所有奖励之和。

回报可以用来评估一个策略是好还是坏。

Definition 1.8. 回报衰减 (discounted return) 试图解决一个问题：当我们达到目标后可能智能体还会有所行动，继续获得奖励，可能会造成回报发散。我们引入衰减率 (discount rate) $\gamma \in (0, 1]$，来对每步行动奖励乘上重复次数次方的衰减率 γ^i ，进行求和后才是衰减回报。

Definition 1.9. 回合 (episode) 它指的是从环境的初始状态开始，智能体与环境交互直到达到某个终止状态的整个过程，对比轨迹，回合是开始到结束的一个完整轨迹，但是轨迹不一定是回合，它可以节选自某个回合。

有些任务是没有结束状态的，我们称为持续性任务 (continuing tasks)；事实上，我们会以一种数学方法把持续性任务转化为回合制任务。

选择一：把目标状态视为吸收态；所谓吸收态，只要智能体达到了吸收态，他就不会再离开这个状态了，或者说没有再可选用的 action，之后所有的 reward 全为 0。

选择二：把目标态视为策略的常态；智能体仍然可以离开目标状态并且在回到目标态的时候获得 $+1$ 的奖励。

1.7 马尔科夫决策过程 MDP

Definition 1.10. 马尔可夫决策过程的关键要素:
首先是三个集合：状态空间 $\mathcal{S}$；行动空间 $\mathcal{A}(s), s \in \mathcal{S}$；奖励空间 $\mathcal{R}(s,a)$.
然后是概率分布：状态转移概率，在状态 s 进行行动 a 的条件下然后转化为状态 s' 的条件概率 $p(s'|s,a)$；奖励概率，在状态 s 进行行动 a 的条件下获得奖励 r 的概率 $p(r|s,a)$
策略：在状态 s，我们采取行动 a 的概率 $\pi(a|s)$
马尔可夫性: 无记忆性

$$p(s_{t+1}|a_{t+1}, s_t, \ldots, a_1, s_0) = p(s_{t+1}|a_{t+1}, s_t)$$
$$p(r_{t+1}|a_{t+1}, s_t, \ldots, a_1, s_0) = p(r_{t+1}|a_{t+1}, s_t)$$

2 贝尔曼公式

2.1 状态值 State Value

一些符号说明，考虑以下的一个单步过程

$$S_t \xrightarrow{A_t} R_{t+1}, S_{t+1}$$

$t, t+1$：是离散时间实例
S_t：t 时刻的状态
A_t：状态 S_t 下采取的行动
R_{t+1}：进行 A_t 后所获得的奖励
S_{t+1}：S_t 进行动作 A_t 后转移的状态
值得一提的是 S_t, A_t, R_{t+1} 都是随机变 (向) 量，这样状态转移的过程又可以用以下的语言表示

$$\begin{aligned} S \to A_t &: \pi(A_t = a|S_t = s) \\ S_t, A_t \to R_{t+1} &: p(R_{t+1} = r|S_t = s, A_t = a) \\ S_t, A_t \to S_{t+1} &: p(S_{t+1} = s'|S_t = s, A_t = a) \end{aligned}$$

在这里，我们假设我们是知道模型的这些概率分布的。

再考虑以下的多步骤轨迹

$$S_t \xrightarrow{A_t} R_{t+1}, S_{t+1} \xrightarrow{A_{t+1}} R_{t+2}, S_{t+2} \xrightarrow{A_{t+2}} R_{t+1}, \dots$$

衰减的回报是

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

它仍然是一个随机变量。

Definition 2.1. 我们称上述衰减回报 G_t 的期望是状态值 (函数):

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s]$$

Remark. 状态值是一个 s 的函数，这是由条件期望的定义所决定的；它也受约于策略 π，对于不同的策略，状态值函数可能不一样；它表示的是一个状态的值，如果它越大，说明策略是更优的，因为这意味着可能会获得更多的奖励。

2.2 贝尔曼公式

我们回到 G_t 的表达式

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots, \\ &= R_{t+1} + \gamma(G_{t+1}) \end{aligned}$$

这样我们的状态值也可以是

$$\begin{aligned} v_\pi(s) &= \mathbb{E}[G_t|S_t = s] \\ &= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\ &= \mathbb{E}[R_{t+1}|S_t = s] + \gamma\mathbb{E}[G_{t+1}|S_t = s] \end{aligned}$$

一方面前半部分我们可以由全期望公式，其称为即刻的奖励均值

$$\begin{aligned}\mathbb{E}[R_{t+1}|S_t=s] &= \sum_a \pi(a|s)\mathbb{E}[R_{t+1}|S_t=s,A_t=a] \\ &= \sum_a \pi(a|s)\sum_r p(r|s,a)r\end{aligned}$$

后半部分称为未来奖励的均值，类似地

$$\begin{aligned}\mathbb{E}[G_{t+1}|S_t=s] &= \sum_{s'} \mathbb{E}[G_{t+1}|S_t=s,S_{t+1}=s']p(s'|s) \\ &= \sum_{s'} \mathbb{E}[G_{t+1}|S_{t+1}=s']p(s'|s) \\ &= \sum_{s'} v_\pi(s')p(s'|s) \\ &= \sum_{s'} v_\pi(s')\sum_a p(s'|s,a)\pi(a|s)\end{aligned}$$

我们对和式进行一个换序就有

$$v_\pi(s) = \sum_a \pi(a|s)[\sum_r p(r|s,a)r + \gamma\sum_{s'} p(s'|s,a)v_\pi(s')]$$

它便是贝尔曼公式，中括号外是一个策略评估，而中括号内其实是一个动态模型。

我们把上式子的中括号拆开，并且再进行一下变换

$$\begin{aligned}v_\pi(s) &= \sum_a \pi(a|s)[\sum_r p(r|s,a)r + \gamma\sum_{s'} p(s'|s,a)v_\pi(s')] \\ &= \sum_a \pi(a|s)\sum_r p(r|s,a)r + \gamma\sum_a\sum_{s'} \pi(a|s)p(s'|s,a)v_\pi(s') \\ &= r_\pi(s) + \gamma\sum_{s'} p_\pi(s'|s)v_\pi(s')\end{aligned}$$

这里 $r_\pi(s) = \sum_a \pi(a|s)\sum_r p(r|s,a)r, p_\pi(s'|s) = \sum_a \pi(a|s)p(s'|s,a)$，把诸多的 s_i 的 $v_\pi(s_i)$ 等式拼起来

$$\boldsymbol{v_\pi} = \boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v_\pi}$$

其中

$$\begin{aligned}&\boldsymbol{v_\pi} = [v_\pi(s_1),\ldots,v_\pi(s_n)]^T \in \mathbb{R}^n \\ &\boldsymbol{r_\pi} = [r_\pi(s_1),\ldots,r_\pi(s_n)]^T \in \mathbb{R}^n \\ &P_\pi \in \mathbb{R}^{n\times n}, (P_\pi)_{ij} = p_\pi(s_j|s_i)\text{称为状态转移矩阵}\end{aligned}$$

2.3 利用贝尔曼公式解状态值

给定一个策略，找出相关的状态值就叫策略评估，这是强化学习的一个基础问题，是找到更好的策略的基石。

由于我们的方程组是

$$\boldsymbol{v_\pi} = \boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v_\pi}$$

由线性方程组的解法是可以得到

$$\boldsymbol{v_\pi} = (I - \gamma P_p i)^{-1}\boldsymbol{r_\pi}$$

这样的解析解，但是这在实际运算中并不是个容易事，实际用数值逼近的办法来解

$$\boldsymbol{v_{k+1}} = \boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v_k}$$

Theorem 2.1. 上述的迭代过程使得 $\boldsymbol{v_k}$ 收敛。

Proof. 定义误差 $\boldsymbol{\delta_k} = \boldsymbol{v_k} - \boldsymbol{v_\pi}$，我们只需说明 $\boldsymbol{\delta_k} \to \mathbf{0}$。

我们有 $\boldsymbol{v_{k+1}} = \boldsymbol{\delta_{k+1}} + \boldsymbol{v_\pi}$，和 $\boldsymbol{v_k} = \boldsymbol{\delta_k} + \boldsymbol{v_\pi}$，结合更新式 $\boldsymbol{v_{k+1}} = \boldsymbol{r_\pi} + \gamma P_\pi v_k$ 就能够得到

$$\boldsymbol{\delta_{k+1}} + \boldsymbol{v_\pi} = \boldsymbol{r_\pi} + \gamma P_\pi(\boldsymbol{\delta_k} + \boldsymbol{v_\pi})$$

重写一下就是

$$\boldsymbol{\delta_{k+1}} = -\boldsymbol{v_\pi} + \boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{\delta_k} + \gamma P_\pi \boldsymbol{v_\pi} = \gamma P_\pi \boldsymbol{\delta}_k$$

重复迭代就是

$$\boldsymbol{\delta_{k+1}} = \gamma^{k+1} P_\pi^{k+1} \boldsymbol{\delta_0}$$

注意到 $P_\pi \mathbf{1} = \mathbf{1}$ ，这样就有 $P_\pi^k \mathbf{1} = \mathbf{1}$ 对任意 $k \in \mathbb{N}$ 成立；另一方面，由于 $|\gamma| < 1$,k 充分大的时候 $\gamma^k \to 0$，由于 $\boldsymbol{\delta_0}$ 是迭代开始既定的，它的各项可以小于一个 M，这样

$$||\boldsymbol{\delta_{k+1}}|| \leq ||M\gamma^{k+1} P_\pi^{k+1} \mathbf{1}|| \to 0$$

□

2.4 行动值 Action Value

Definition 2.2. 所谓行动值 (action value)，它是指智能体从一个状态采取了某个行动后所能得到的平均回报

$$q_\pi(s, a) = \mathbb{E}[G_t | S_t = s, A_t = a]$$

它是元组 (s, a) 的一个二元函数，且依旧受约于 π

我们容易得到

$$\mathbb{E}[G_t | S_t = s] = \sum_a \mathbb{E}[G_t | S_t = s, A_t = a]\pi(a|s)$$

也就是说

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a)$$

同样回到贝尔曼公式的话可以看出

$$q_\pi(s, a) = \sum_r p(r|s, a)r + \gamma \sum_{s'} p(s'|s, a) v_\pi(s')$$

3 贝尔曼最优公式

Definition 3.1. 状态值会被用来评估策略是好还是坏，我们称策略 π_1 比 π_2 更好，如果满足

$$v_{\pi_1}(s) \geq v_{\pi_2}(s), \forall s \in \mathcal{S}$$

进一步，我们称 π^* 是最优的，如果

$$v_{\pi^*}(s) \geq v_{\pi}(s), \text{对所有 s ，任意策略}\pi\text{成立}$$

我们所说的贝尔曼公式

$$v_\pi(s) = \sum_a \pi(a|s)[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_\pi(s')]$$

它是在每个固定的 $\pi(\cdot|\cdot)$ 下给出来的，而所谓的贝尔曼最优公式就是在上述的这个公式下求解一个最优化问题

$$v(s) = \max_\pi \sum_a \pi(a|s)[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v(s')]$$

值得一提的是，我们的 $p(r|s,a), p(s'|s,a)$ 都是已知的，我们的 $v(s), v(s')$ 是未知的并且是待计算的。

Definition 3.2. 矩阵形式的贝尔曼最优公式

$$\boldsymbol{v} = \max_\pi(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v})$$

其中有

$$(\boldsymbol{r_\pi})_s = \sum_a \pi(a|s) \sum_r p(r|s,a)r,$$
$$(P_\pi)_{s,s'} = p(s'|s) = \sum_a \pi(a|s)p(s'|s,a)$$

这里的 $\max_\pi$ 其实是逐元素的最大化。

Remark. 在上面的向量记号下，不难看出 $P_\pi \boldsymbol{v}$ 的某一个元素是

$$c_i = \sum_k (P_\pi)_{i,k} v_k = \sum_k \sum_a \pi(a|i)p(k|i,a)v_k$$

或者用更为适配的符号是

$$c_s = \sum_{s'} (P_\pi)_{s,s'} v_{s'} = \sum_{s'} \sum_a \pi(a|s)p(s'|s,a)v_{s'}$$

我们再聚焦于 $v(s) = \max_\pi \sum_a \pi(a|s)[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v(s')]$ ，这个问题并不是件易事，但是优化问题中我们一般不是直接求解结果而是采用数值逼近的办法，这样在每次逼近的过程中，我们会把右边的 $v(s')$ 视为每次迭代前的初值，这样它是给定的，在第二章的讨论中知道中括号中的部分可以写为一个 $q(s,a)$，这样问题就是去调整

$$\sum_a \pi(a|s)q(s,a)$$

中的 $\pi(a|s)$，再由于 π 是个概率权数，我们若能找到若干个 $q(s,a)$ 中最大的那个就好了，如此把对应权数设置为 1，问题就是

$$\max_{a \in \mathcal{A}(s)} q(s,a)$$

现在来到向量形式，由于最优的策略 π 是由值 $\boldsymbol{v}$ 决定的，所以优化问题

$$\boldsymbol{v} = \max_{\pi}(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v})$$

的右边实际上是一个 $\boldsymbol{v}$ 的函数，那我们可以直接写作

$$\boldsymbol{f(v)} = \max_{\pi}(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v})$$

这样贝尔曼最优就是解决

$$\boldsymbol{v} = \boldsymbol{f(v)}$$

这样一个问题。

我们需要区分一个问题，前文说 $\boldsymbol{v}$ 是由 π 决定的，是因为我们求解析解 $\boldsymbol{v}$ 的时候对于不同的每个 π 是可以计算出不同的相应 value 的；但是对于这样的一系列可调整的 π，对应有一系列的 value，其中最大的那个 value 是一定存在的，和这些 π 并没有关系而仅和 π 的可调整空间有关系，所以说我们才做出了 $\boldsymbol{f(v)} = \max_\pi(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v})$ 的简化。

为了解决这样的一个问题，我们先来回顾一下几个分析的内容

Definition 3.3. 称 $\boldsymbol{x^*}$ 为 $\boldsymbol{f}$ 的一个不动点，如果对于 $\boldsymbol{x} \in \mathbb{R}^d, \boldsymbol{f}: \mathbb{R}^d \to \mathbb{R}^d$ 有

$$\boldsymbol{f(x^*)} = \boldsymbol{x}^*$$

Definition 3.4. 称映射 $\boldsymbol{f(x)}: \mathbb{R}^d \to \mathbb{R}^d$ 是李普希兹连续，如果它满足存在一个 $L > 0$ 使得

$$||\boldsymbol{f(x_1)} - \boldsymbol{f(x_2)}|| \leq L||\boldsymbol{x_1} - \boldsymbol{x_2}||$$

对任意 $\boldsymbol{x_1}, \boldsymbol{x_2}$ 成立；
进一步地，我们称 $L \in (0, 1)$ 的时候的李普希兹连续映射是一个压缩映射。

Theorem 3.1. 压缩映射一定存在唯一的不动点。

Proof. 证明暂时从略，这是容易通过两边夹说明的，并且这不是我们关注的。 □

Theorem 3.2. 然后回到 $\boldsymbol{v} = \boldsymbol{f(v)}$，它是一个压缩映射。

Proof. 考虑任意两个向量 $\boldsymbol{v_1}, \boldsymbol{v_2} \in \mathbb{R}^{|S|}$，并且假设

$$\pi_1^* = \arg\max_{\pi}(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v_1}), \pi_2^* = \arg\max_{\pi}(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v_2})$$

于是就有

$$\boldsymbol{f(v_1)} = \max_{\pi}(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v_1}) = \boldsymbol{r_{\pi_1^*}} + \gamma P_{\pi_1^*} \boldsymbol{v_1} \geq \boldsymbol{r_{\pi_2^*}} + \gamma P_{\pi_2^*} \boldsymbol{v_1}$$

$$\boldsymbol{f(v_2)} = \max_{\pi}(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v_2}) = \boldsymbol{r_{\pi_2^*}} + \gamma P_{\pi_2^*} \boldsymbol{v_2} \geq \boldsymbol{r_{\pi_1^*}} + \gamma P_{\pi_1^*} \boldsymbol{v_2}$$

其中 $\geq$ 是在逐元素的情形下定义的，再进行差运算就是

$$\boldsymbol{f(v_1)} - \boldsymbol{f(v_2)} = \boldsymbol{r_{\pi_1^*}} + \gamma P_{\pi_1^*} \boldsymbol{v_1} - \boldsymbol{r_{\pi_2^*}} + \gamma P_{\pi_2^*} \boldsymbol{v_2} \leq \boldsymbol{r_{\pi_1^*}} + \gamma P_{\pi_1^*} \boldsymbol{v_1} - (\boldsymbol{r_{\pi_1^*}} + \gamma P_{\pi_1^*} \boldsymbol{v_2}) = \gamma P_{\pi_1^*}(\boldsymbol{v_1} - \boldsymbol{v_2})$$

它仍然是逐元素下定义的，类似地

$$\boldsymbol{f(v_2)} - \boldsymbol{f(v_1)} \leq \gamma P_{\pi_2^*}(\boldsymbol{v_2} - \boldsymbol{v_1})$$

从而

$$\gamma P_{\pi_2^*}(\boldsymbol{v_1}-\boldsymbol{v_2}) \leq \boldsymbol{f(v_1)}-\boldsymbol{f(v_2)} \leq \gamma P_{\pi_1^*}(\boldsymbol{v_1}-\boldsymbol{v_2})$$

定义一个

$$\boldsymbol{z} = \max\{|\gamma P_{\pi_2^*}(\boldsymbol{v_1}-\boldsymbol{v_2})|, |\gamma P_{\pi_1^*}(\boldsymbol{v_1}-\boldsymbol{v_2})|\} \in \mathbb{R}^{|\mathcal{S}|}$$

这里 $\max, \geq$ 都是逐元素定义的；由定义，$\boldsymbol{z} > 0$，另一方面，容易看出

$$-\boldsymbol{z} \leq \gamma P_{\pi_2^*}(\boldsymbol{v_1}-\boldsymbol{v_2}) \leq \boldsymbol{f(v_1)}-\boldsymbol{f(v_2)} \leq \gamma P_{\pi_1^*}(\boldsymbol{v_1}-\boldsymbol{v_2}) \leq \boldsymbol{z}$$

也就是说

$$|\boldsymbol{f(v_1)}-\boldsymbol{f(v_2)}| \leq \boldsymbol{z}\text{逐元素成立}$$

也就得到了

$$||\boldsymbol{f(v_1)}-\boldsymbol{f(v_2)}||_\infty \leq ||\boldsymbol{z}||_\infty$$

现在假设 z_i 是 $\boldsymbol{z}$ 的第 i 个分量，$\boldsymbol{p_i^T}, \boldsymbol{q_i^T}$ 是 $P_{\pi_1^*}, P_{\pi_2^*}$ 的第 i 行。如此就有

$$z_i = \max\{\gamma|\boldsymbol{p_i^T(v_1-v_2)}|, \gamma|\boldsymbol{q_i^T(v_1-v_2)}|\}$$

因为 $\boldsymbol{p_i}$ 是一个非负元列向量，并且它的和是归一的，这就说明了

$$|\boldsymbol{p_i^T(v_1-v_2)}| \leq \boldsymbol{p_i^T}|\boldsymbol{v_1}-\boldsymbol{v_2}| \leq ||\boldsymbol{v_1}-\boldsymbol{v_2}||_\infty$$

解释一下上面这个，最左边的是 $\boldsymbol{p_i^T}$ 与 $\boldsymbol{v_1}-\boldsymbol{v_2}$ 做内积的值; 第二个表示的是逐元素绝对值后做内积的结果，这个不等号是三角不等式; 第三个是最大值。同理有

$$|\boldsymbol{q_i^T}(\boldsymbol{v_1}-\boldsymbol{v_2})| \leq ||\boldsymbol{v_1}-\boldsymbol{v_2}||_\infty$$

因此 $z_i \leq \gamma||\boldsymbol{v_1}-\boldsymbol{v_2}||_\infty$ 并且由此得出

$$||\boldsymbol{z}||_\infty = \max_i |z_i| \leq \gamma||\boldsymbol{v_1}-\boldsymbol{v_2}||_\infty$$

也就是

$$||\boldsymbol{f(v_1)}-\boldsymbol{f(v_2)}|| \leq \gamma||\boldsymbol{v_1}-\boldsymbol{v_2}||_\infty$$

证毕。 □

我们说明了 $\boldsymbol{f(v)}$ 是个压缩映射，那么贝尔曼最优公式就可以迭代求解，更新式

$$\boldsymbol{v_{k+1}} = \boldsymbol{f(v_k)} = \max_\pi(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v_k})$$

当然，得到上述解 $\boldsymbol{v^*}$ 后就可以得到对应的策略

$$\pi^* = \arg\max_{\pi\in\Pi}(\boldsymbol{r_\pi} + \gamma P_\pi \boldsymbol{v^*})$$

既然这个最优的策略求出来了，那么 $\boldsymbol{v^*}$ 也可以换个表述

$$\boldsymbol{v^*} = \boldsymbol{r_{\pi^*}} + \gamma P_{\pi^*}\boldsymbol{v^*}$$

也就是说 $\boldsymbol{v^*}$ 是 π^* 策略的状态值，并且贝尔曼最优公式是一个特殊的贝尔曼公式，它是策略为 π^* 的贝尔曼公式。

到这里为止，尽管我们通过数值方法求出了不动点 $\boldsymbol{v^*}$ 以及对应策略 $\boldsymbol{\pi^*}$，但是我们还未能保证它们确为最优的。

Theorem 3.3. 上述求解的 $\boldsymbol{v}^*$ 确实是最优的状态值，并且 π^* 也确为最优的策略。也就是说，对于任意策略 π，我们总能得到

$$\boldsymbol{v}^* = \boldsymbol{v}_{\boldsymbol{\pi}^*} \geq \boldsymbol{v}_{\boldsymbol{\pi}}$$

这里的 $\boldsymbol{v}_{\boldsymbol{\pi}}$ 是 π 策略下的状态值，并且大于号是定义在逐元素序上的。

Proof. 对任意的策略 π，总有

$$\boldsymbol{v}_{\boldsymbol{\pi}} = \boldsymbol{r}_{\boldsymbol{\pi}} + \gamma P_\pi \boldsymbol{v}_{\boldsymbol{\pi}}$$

因为

$$\boldsymbol{v}^* = \max_\pi(\boldsymbol{r}_{\boldsymbol{\pi}} + \gamma P_\pi \boldsymbol{v}^*) = \boldsymbol{r}_{\boldsymbol{\pi}^*} + \gamma P_{\pi^*} \boldsymbol{v}^*$$

我们有

$$\boldsymbol{v}^* - \boldsymbol{v}_{\boldsymbol{\pi}} \geq (\boldsymbol{r}_{\boldsymbol{\pi}} + \gamma P_\pi \boldsymbol{v}^*) - (\boldsymbol{r}_{\boldsymbol{\pi}} + \gamma P_\pi \boldsymbol{v}_{\boldsymbol{\pi}}) = \gamma P_\pi (\boldsymbol{v}^* - \boldsymbol{v}_{\boldsymbol{\pi}})$$

重复上述过程就有

$$\boldsymbol{v}^* - \boldsymbol{v}_{\boldsymbol{\pi}} \geq \gamma^n P_\pi^n (\boldsymbol{v}^* - \boldsymbol{v}_{\boldsymbol{\pi}}) \to \mathbf{0}$$

因此 $\boldsymbol{v}^* \geq \boldsymbol{v}_{\boldsymbol{\pi}}$ 对任意策略 π 成立 □

Theorem 3.4. 对于任意 $s \in \mathcal{S}$ ，确定的贪心策略

$$\pi^*(a|s) = \begin{cases} 1, a = a^*(s) \\ 0, a \neq a^*(s) \end{cases}$$

是一个最优策略，这里

$$a^*(s) = \arg\max_a q^*(a, s), q^*(s, a) := \sum_r p(r|s, a)r + \gamma \sum_{s'} p(s'|s, a)v^*(s')$$

Proof. 当 $\pi^* = \arg\max_\pi(\boldsymbol{r}_{\boldsymbol{\pi}} + \gamma P_\pi \boldsymbol{v}^*)$ 是最优策略，他的逐元素形式是

$$\pi^*(s) = \arg\max_{\pi \in \Pi} \sum_{a \in \mathcal{S}} \pi(a|s) \left(\sum_{r \in \mathcal{R}} p(r|s, a)r + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a)v^*(s') \right), s \in \mathcal{S}$$

显然，我们取最大的那个 $q^*(s, a)$ 的条概为 1，其它的为 0 得到的 $\sum_a \pi(a|s)q^*(s, a) = \max q^*(s, a)$ 是最大的。 □

注意，对每个 r 进行某种线性变换不会改编策略。

Theorem 3.5. *考虑一个 Markov 决策过程，它以 $\boldsymbol{v}^*$ 为最优的状态值，满足 $\boldsymbol{v}^* = \max_\pi(\boldsymbol{r}_{\boldsymbol{\pi}} + \gamma P_\pi \boldsymbol{v}^*)$；如果对每个奖励 r 实施一个变换 $\alpha r + \beta, \alpha > 0, \beta \in \mathbb{R}$，那么对应的最优状态值为*

$$\boldsymbol{v}' = \alpha \boldsymbol{v}^* + \frac{\beta}{1 - \gamma} \mathbf{1}$$

仍然是新的模型下的最优状态值。

Proof. 证明暂时从略，因为结论比较直观而且更偏向结论导向。 □

4 值迭代算法和策略迭代算法

4.1 值迭代算法

其实上值迭代算法是一个贪心的算法。回顾前面的内容，我们是如何对贝尔曼最优公式求解的?

$$\boldsymbol{v} = \boldsymbol{f}(\boldsymbol{v}) = \max_{\pi}(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v})$$

我们证明了它是一个压缩映射，然后利用数值迭代的方法求解

$$\boldsymbol{v}_{k+1} = \boldsymbol{f}(\boldsymbol{v}_k) = \max_{\pi}(\boldsymbol{r}_\pi + \gamma P_\pi \boldsymbol{v}_k), k = 1, 2, 3 \cdots$$

这里 $\boldsymbol{v}_0$ 是可以随机给定的，总之迭代过后会收敛到值；在值迭代算法中我们是对于某一个 v_k 然后去求出诸 $q_k(s,a)$ ，找到最大的那个 q_k ，把它的系数规定为 1，从而更新为新的策略 $\pi_{k+1}(a|s)$ ，然后在新的策略 π_{k+1} 下找到新的局部最优值 $v_{k+1}(s) = \max_a q_k(a,s)$ 再仿照之前的，把 v_{k+1} 代入算出新的 $q_{k+1}(s,a)$ 以此类推.

具体的算法如下:

Algorithm 1: 值迭代

Input: 在状态 s 下采取行动 a 会获得的奖励 r 的分布 $p(r|s,a)$；在状态 s 下采取行动 a 转移到新状态 s' 的分布 $p(s'|s,a)$ ，这里所有的 (s,a) 视为既定的，并且初始给定一个 $\boldsymbol{v}_0$

Output: 最优状态值 $\boldsymbol{v}_\pi^*$，以及最优的策略 π^*

/* 当 $\boldsymbol{v}_k$ 不收敛（也就是说范数 $||\boldsymbol{v}_k - \boldsymbol{v}_{k-1}||$）还没有在预定义的一个阈值之下的时候，我们对 k 进行遍历 */

for $s \in \mathcal{S}$ **do**
  for $a \in \mathcal{A}(s)$ **do**
    $q_k(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_k(s')$
  end
  $a_k^*(s) = \arg\max_a q_k(a,s)$;
  $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$ else $\pi_{k+1}(a|s) = 0$;
  $v_{k+1}(s) = \max_a q_k(a,s)$
end

4.2 策略迭代算法

我们说过值迭代算法是贪心的；对应地肯定有个精度更大的算法，但是它的复杂度相应地应该会更高。我们称这种算法为策略迭代算法，如果我们是从一个任意初始的策略 π_0 出发，然后进行以下几步：在某种既定的策略 π_k 下，我们不断迭代 $\boldsymbol{v}$ 的值来求出 π_k 下的 $\boldsymbol{v}_{\pi_k}$ 良好估计；然后在这样的 $\boldsymbol{v}_{\pi_k}$ 下再去求逐行的 $\max q_{\pi_k}(a,s)$ ，找到最大的 q 后把系数设置为 1 从而做到对 $\pi_k \to \pi_{k+1}$，此后重复上述步骤.

具体的算法如下:

Algorithm 2: 策略迭代

Input: 在状态 s 下采取行动 a 会获得的奖励 r 的分布 $p(r|s,a)$ 在状态 s 下采取行动 a 转移到新状态 s' 的分布 $p(s'|s,a)$ ，这里所有的 (s,a) 视为既定的，并且出示给定一个 π_0

Output: 最优状态值 $\boldsymbol{v}_\pi^*$ ，以及最优的策略 π^*

/* 只要策略还没有收敛，我们遍历 k */

初始化一个 $v_{\pi_k}^{(0)}$;

while $v_{\pi_k}^*$ 不收敛，遍历 j **do**

  for $s \in \mathcal{S}$ **do**

    $v_{\pi_k}^{(j+1)}(s) = \sum_a \pi_k a|s[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_{\pi_k}^{(j)}(s')]$

  end

end

for $s \in \mathcal{S}$ **do**

  for $a \in \mathcal{A}(s)$ **do**

    $q_{\pi_k}(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_{\pi_k}(s')$

  end

  $a_k^*(s) = \arg\max_a q_{\pi_k}(s,a)$;

  $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$ else $\pi_{k+1}(a|s) = 0$

end

4.3 剪枝迭代

在上述两个算法之间选择一个居中复杂度和居中精度的算法，在策略迭代中我们要求的是遍历 j 来直到我们的状态值收敛，然后再去找下一步的策略；而在值迭代中我们进进行了一部状态值的计算就贪心地去找新的状态值下的更优策略，相当于 $j=1$；而在剪枝中我们把 j 的遍历次数给定，这样便是一个居中的情况.

具体算法如下:

Algorithm 3: 剪枝迭代

Input: 在状态 s 下采取行动 a 会获得的奖励 r 的分布 $p(r|s,a)$ 在状态 s 下采取行动 a 转移到新状态 s' 的分布 $p(s'|s,a)$ ，这里所有的 (s,a) 视为既定的，并且出示给定一个 π_0，再给定一个 $j_{truncate}$

Output: 最优状态值 $\boldsymbol{v}_\pi^*$ ，以及最优的策略 π^*

/* 居中情况 */

初始化一个 $v_{\pi_k}^{(0)}$;

while $j \le j_{truncate}$ 遍历 j **do**

  for $s \in \mathcal{S}$ **do**

    $v_{\pi_k}^{(j+1)}(s) = \sum_a \pi_k a|s[\sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_{\pi_k}^{(j)}(s')]$

  end

end

for $s \in \mathcal{S}$ **do**

  for $a \in \mathcal{A}(s)$ **do**

    $q_{\pi_k}(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_{\pi_k}(s')$

  end

  $a_k^*(s) = \arg\max_a q_{\pi_k}(s,a)$;

  $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$ else $\pi_{k+1}(a|s) = 0$

end