强化学习极简笔记（三）：马尔可夫决策过程

〇、概述

MDP 是强化学习问题在数学上的理想化形式，因为在这个框架下，我们可以进行精确的理论说明。

一、“智能体-环境”交互接口

MDP 是一种通过交互式学习来实现目标的理论框架。进行学习及实施决策的机器被称为智能体（agent）。智能体之外所有与其相互作用的事务称为环境（environment）。这些事务之间持续进行交互，智能体选择动作，环境对这些动作做出相应的响应，并像智能体呈现出新的状态。环境也会产生一个收益，通常是特定的数值，这就是智能体在动作选择过程中想要最大化的目标。

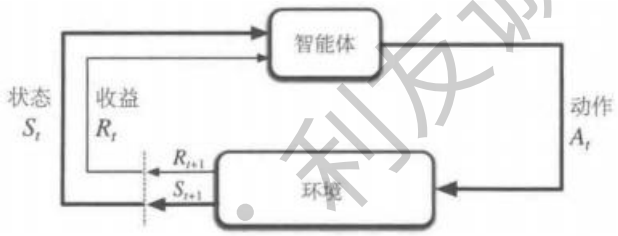


图 1：MDP 中“智能体-环境”交互

经过交互，MDP 和智能体共同给出了一个序列或轨迹

$$S_0, A_0, R_1, S_1, A_1, R_2, \dots \tag{1}$$

MDP 的**动态特性**由函数 p 定义（由 p 给出的概率完全刻画了环境的动态特性）

$$p(s', r \mid s, a) \doteq \Pr\left(S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\right) \tag{2}$$

公式（2）表明，在 MDP 中，S_t, R_t 的每个可能出现的值出现的概率只取决于前一个状态和前一个动作 S_{t-1}, A_{t-1}，并且与更早之前的状态和动作完全无关。这个限制是针对状态的，称为**马尔科夫性**。

有函数 p 出发，构造如下定义：公式（3）为状态转移概率；公式（4）为“状态-动作”二元组的期望收益；公式（5）为“状态-动作-后继状态”三元组的期望收益

$$p(s' \mid s, a) \doteq \Pr\left(S_t = s' \mid S_{t-1} = s, A_{t-1} = a\right) = \sum_r p(s', r \mid s, a) \tag{3}$$

$$r(s, a) \doteq \mathrm{E}\left(R_t \mid S_{t-1} = s, A_{t-1} = a\right) = \sum_r r \sum_s p(s', r \mid s, a) \tag{4}$$

$$r(s, a, s') \doteq \mathrm{E}\left(R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'\right) = \sum_r r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)} \tag{5}$$

总结：MDP 框架是目标导向的**交互式学习**问题的一个高度抽象：任何目标导向的行为的学习都可以概括为智能体及其环境之间来回传递的三个信号：一个信号表示智能体做出的选择（**行动**），一个信号用来表示做出该选择的基础（**状态**），还有一个信号用来定义智能体的目标（**收益**）。

二、目标和收益

智能体的目标被形式化表征一种特殊信号，称为**收益**，它通过环境传递给智能体。每一个时刻，收益都是一个单一标量数值。使用收益信号来形式化目标是强化学习最显著的特征之一。

收益假设：最大化智能体收到的标量信号（称之为收益）累计和的概率期望值。

最重要的一点：**我们设立收益的方式要能真正表明我们的目标**。收益信号只能用来传达什么是你想要实现的目标，而不是如何实现这个目标（**Reward is enough**）

三、回报和分幕

定义带折扣的回报

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{6}$$

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{7}$$

四、分幕式和持续性任务的统一表示法

分幕式任务的结束状态为一个出度指向自身的吸收状态

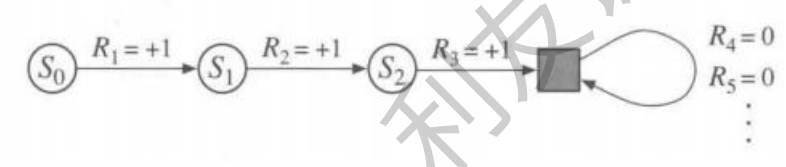


图 2：吸收状态

也可以截断表示回报

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k \tag{8}$$

五、策略和价值函数

状态价值函数

$$v_\pi(s) \doteq \mathrm{E}_\pi\left(G_t \mid S_t = s\right) = \mathrm{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right) \tag{9}$$

动作价值函数

$$q_\pi(s,a) \doteq \mathrm{E}_\pi\left(G_t \mid S_t = s, A_t = a\right) = \mathrm{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right) \tag{10}$$

根据回溯图，可以推导状态价值函数和动作价值函数之间的相互转化关系（注意 r 是在执行动作之后得到的）

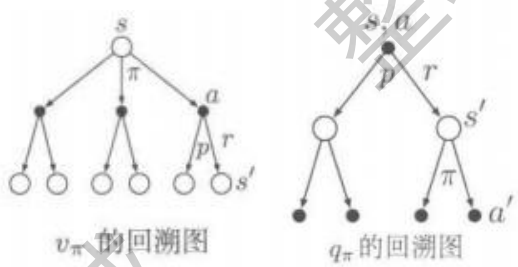


图 3：回溯图

由 $v_\pi(s')$ 表示 $v_\pi(s)$

$$
\begin{aligned}
v_\pi(s) &\doteq \mathrm{E}_\pi(G_t \mid S_t = s) \\
&= \mathrm{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t = s\right) \\
&= \mathrm{E}_\pi(R_{t+1} + \gamma G_{t+1} \mid S_t = s) \\
&= \sum_a \mathrm{E}_\pi(R_{t+1} + \gamma G_{t+1}, A_t = a \mid S_t = s) \\
&= \sum_a \pi(a \mid s)\mathrm{E}_\pi(R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a) \\
&= \sum_a \pi(a \mid s)\sum_r r\sum_{s'} p(s', r \mid s, a) + \gamma\sum_a \pi(a \mid s)\mathrm{E}_\pi(v_\pi(s') \mid S_t = s, A_t = a) \\
&= \sum_a \pi(a \mid s)\sum_r r\sum_{s'} p(s', r \mid s, a) + \gamma\sum_a \pi(a \mid s)\sum_r\sum_{s'} p(s', r \mid s, a)v_\pi(s') \\
&= \sum_a \pi(a \mid s)\sum_{r,s'} p(s', r \mid s, a)\left(r + \gamma v_\pi(s')\right)
\end{aligned}
\tag{11}
$$

由 $q_\pi(s,a)$ 表示 $v_\pi(s)$

$$
\begin{aligned}
v_\pi(s) &\doteq \mathrm{E}_\pi(G_t \mid S_t = s) \\
&= \mathrm{E}_\pi\left(\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t = s\right) \\
&= \mathrm{E}_\pi(R_{t+1} + \gamma G_{t+1} \mid S_t = s) \\
&= \sum_a \mathrm{E}_\pi(R_{t+1} + \gamma G_{t+1}, A_t = a \mid S_t = s) \\
&= \sum_a \pi(a \mid s)\mathrm{E}_\pi(R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a) \\
&= \sum_a \pi(a \mid s)q_\pi(s,a)
\end{aligned}
\tag{12}
$$

由 $v_\pi(s')$ 表示 $q_\pi(s,a)$

$$
\begin{aligned}
q_\pi(s,a) &\doteq \mathrm{E}_\pi(G_t \mid S_t = s, A_t = a) \\
&= \sum_{r,s'} p(s', r \mid s, a)\left(r + \gamma v_\pi(s')\right)
\end{aligned}
\tag{13}
$$

由 $q_\pi(s',a')$ 表示 $q_\pi(s,a)$

$$
\begin{aligned}
q_\pi(s,a) &\doteq \mathrm{E}_\pi(G_t \mid S_t = s, A_t = a) \\
&= \sum_{r,s'} p(s', r \mid s, a)\left(r + \gamma v_\pi(s')\right) \\
&= \sum_{r,s'} p(s', r \mid s, a)\left(r + \gamma\sum_{a'} \pi(a' \mid s')q_\pi(s', a')\right)
\end{aligned}
\tag{14}
$$

六、最优策略和最优价值函数

最优策略：对于所有 $s \in S, \pi \geq \pi'$，那么应当 $v_\pi(s) \geq v_{\pi'}(s)$。尽管最优策略不止一个，

用 π_* 来表示这些策略，他们共享相同的状态价值函数，称之为最优状态价值函数，记为 v_*，其定义为：对于任意 $s \in S$

$$v_*(s) \doteq \max_{\pi} v_\pi(s) \tag{15}$$

最优策略也共享相同的最优动作价值函数，记为 q_*，其定义为：对于任意 $s \in S$

$$q_*(s,a) \doteq \max_{\pi} q_\pi(s,a) \tag{16}$$

用 v_* 来表示 q_*

$$q_*(s,a) = \mathrm{E}\left(R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right) \tag{17}$$

贝尔曼最优方程（最优策略下各个状态的价值一定等于这个状态下最优动作的期望回报）

$$\begin{aligned} v_*(s) &= \max_a q_{\pi_*}(s,a) \\ &= \max_a \mathrm{E}\left(R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right) \\ &= \max_a \mathrm{E}\left(R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a\right) \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)\left(r + \gamma v_*(s')\right) \end{aligned} \tag{18}$$

$$\begin{aligned} q_*(s,a) &= \mathrm{E}\left(R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right) \\ &= \sum_{s',r} p(s', r \mid s, a)\left(r + \gamma \max_{a'} q_*(s', a')\right) \end{aligned} \tag{19}$$

对于最优价值函数 v_* 来说，任何**贪心**策略都是最优策略。

七、最优性和近似解法

最优策略通常需要极大量资源，我们需要面对近似解法。

强化学习极简笔记（四）：动态规划

零、总体目标

已知 MDP 及其动态特性 $p(s',r|s,a)$，如何得到 π_*, q_*, v_*（S^+的情况下）

子问题 1：如何得到 q_*, v_*：求解贝尔曼方程

子问题 2：如何计算贝尔曼方程：直接求解（一个线性方程组，运算量大）；迭代求解（DP）

$$\begin{aligned} v_*(s) &= \max_a q_*(s,a) \\ &= \max_a \mathrm{E}\left(R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a\right) \\ &= \max_a \mathrm{E}\left(R_{t+1} + \gamma v_*(S_{t+1}) | S_t = s, A_t = a\right) \\ &= \max_a \sum_{s',r} p(s',r|s,a)\left(r + \gamma v_*(s')\right) \end{aligned} \tag{1}$$

$$\begin{aligned} q_*(s,a) &= \mathrm{E}\left(R_{t+1} + \gamma \max_a v_*(S_{t+1}) | S_t = s, A_t = a\right) \\ &= \mathrm{E}\left(R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') | S_t = s, A_t = a\right) \\ &= \sum_{s',r} p(s',r|s,a)\left(r + \gamma \max_{a'} q_*(s',a')\right) \end{aligned} \tag{2}$$

直观解释公式（1）和公式（2）：最优状态价值函数等于其最大的后继动作-状态价值函数；最优动作-状态价值函数等于其后继最优状态价值函数的期望。之所以最优动作-价值函数不是其最大的后继状态价值函数，是因为环境转移不受智能体控制。

壹、策略迭代（Policy Iteration）

概述：策略迭代由两个过程交替进行：策略评估（预测）和策略改进

$$\pi_0 \xrightarrow{\mathrm{E}} v_{\pi_0} \xrightarrow{\mathrm{I}} \pi_1 \xrightarrow{\mathrm{E}} v_{\pi_1} \xrightarrow{\mathrm{I}} \pi_2 \xrightarrow{\mathrm{E}} \ldots \xrightarrow{\mathrm{I}} \pi_* \xrightarrow{\mathrm{E}} v_{\pi_*} \tag{3}$$

1.1. 迭代策略评估（给定 π 估计 $v_\pi(s)$）

由回溯图

$$\begin{aligned} v_\pi(s) &\doteq \mathrm{E}_\pi\left(G_t | S_t = s\right) \\ &= \mathrm{E}_\pi\left(R_{t+1} + \gamma G_{t+1} | S_t = s\right) \\ &= \mathrm{E}_\pi\left(R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s\right) \\ &= \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left(r + \gamma v_\pi(s')\right) \end{aligned} \tag{4}$$

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

同理，动作-状态价值函数的策略评估结合公式（2），要求所有动作均要被访问到。

$$
\begin{aligned}
&q_\pi(s,a) \doteq \mathrm{E}_\pi\left(G_t \mid S_t=s, A_t=a\right) \\
&=\mathrm{E}_\pi\left(R_{t+1}+\gamma G_{t+1} \mid S_t=s, A_t=a\right) \\
&=\mathrm{E}_\pi\left(R_{t+1}+\gamma v_\pi\left(S_{t+1}\right) \mid S_t=s, A_t=a\right) \\
&=\mathrm{E}_\pi\left(R_{t+1}+\gamma \sum_{a', s'} \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right) \mid S_t=s, A_t=a\right) \\
&=\sum_{s', r} p\left(s', r \mid s, a\right)\left(r+\gamma \sum_{a'} \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right)\right)
\end{aligned} \tag{6}
$$

$$
\begin{aligned}
&q_{k+1}(s,a) \doteq \mathrm{E}_\pi\left(R_{t+1}+\gamma G_{t+1} \mid S_t=s, A_t=a\right) \\
&=\sum_{s', r} p\left(s', r \mid s, a\right)\left(r+\gamma \sum_{a'} \pi\left(a' \mid s'\right) q_k\left(s', a'\right)\right)
\end{aligned} \tag{7}
$$

1.2. 策略改进

由**策略改进定理**：对于 $\forall s \in S$，满足公式（8），则 π' 相比 π 一样或者更好，即 $\forall s \in S, v_{\pi'}(s) \geq v_\pi(s)$

$$
q_\pi\left(s, \pi'(a)\right) \geq v_\pi(s) \tag{8}
$$

$$
\begin{aligned}
&v_\pi(s) \leq q_\pi\left(s, \pi'(s)\right) \\
&=\mathrm{E}\left(R_{t+1}+\gamma v_\pi\left(S_{t+1}\right) \mid S_t=s, A_t=\pi'(s)\right) \\
&=\mathrm{E}_{\pi'}\left(R_{t+1}+\gamma v_\pi\left(S_{t+1}\right) \mid S_t=s\right) \\
&\leq \mathrm{E}_{\pi'}\left(R_{t+1}+\gamma q_\pi\left(S_{t+1}, \pi'\left(S_{t+1}\right)\right) \mid S_t=s\right) \\
&\vdots \\
&=v_{\pi'}(s)
\end{aligned} \tag{9}
$$

策略改进定理说明了：

① 给定 $v_\pi(s)$ 以及 $\pi(a \mid s)$，可以评估一切 π' 的 $v_{\pi'}(s)$

② 利用①以及策略改进定理，可以根据 $q_\pi(s,a)$ 选最优（考虑所有的状态和动作）。

利用一个单步搜索的贪心策略 π' 改进已有策略

$$
\begin{aligned}
&\pi'(s) \doteq \arg\max_a q_\pi(s,a) \\
&=\arg\max_a \mathrm{E}_\pi\left(R_{t+1}+\gamma v_\pi\left(S_{t+1}\right) \mid S_t=s, A_t=a\right) \\
&=\arg\max_a \mathrm{E}_\pi\left(R_{t+1}+\gamma \sum_{a', s'} \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right) \mid S_t=s, A_t=a\right) \\
&=\arg\max_a \sum_{s', r} p\left(s', r \mid s, a\right)\left(r+\gamma \sum_{a'} \pi\left(a' \mid s'\right) q_\pi\left(s', a'\right)\right)
\end{aligned} \tag{10}
$$

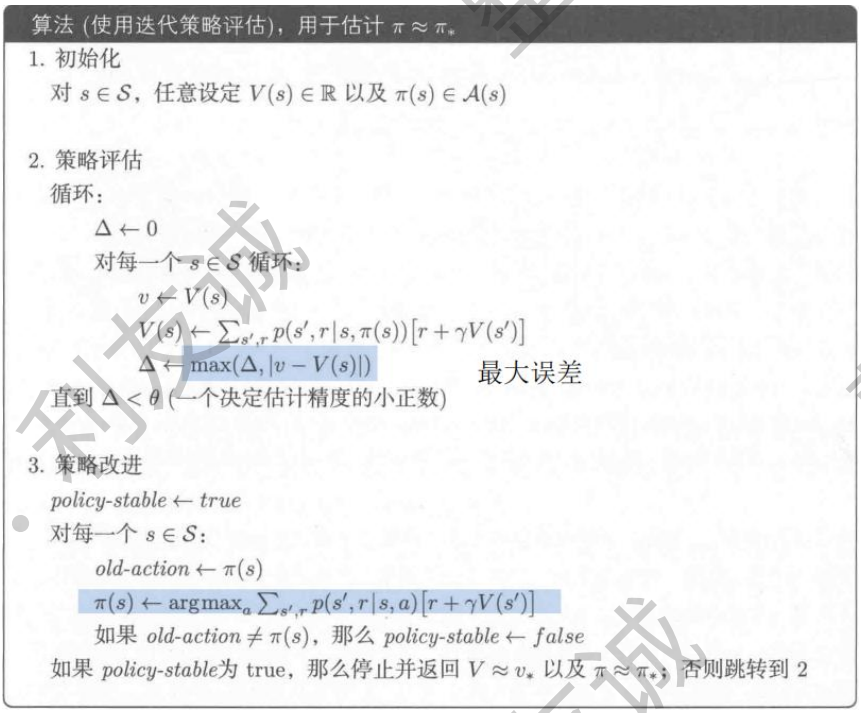


图 1：策略迭代算法

贰、价值迭代（Value Iteration）

概述：提前截断策略评估的过程（甚至一次遍历对每个状态进行一次更新）

表示：结合策略改进与策略评估的简单更新

$$\begin{aligned} v_{k+1}(s) &\doteq \max_a \mathrm{E}_\pi\left(R_{t+1} + \gamma v_k\left(S_{t+1}\right) | S_t = s\right) \\ &= \max_a \sum_{s',r} p(s',r|s,a)\left(r + \gamma v_k(s')\right) \end{aligned} \tag{11}$$

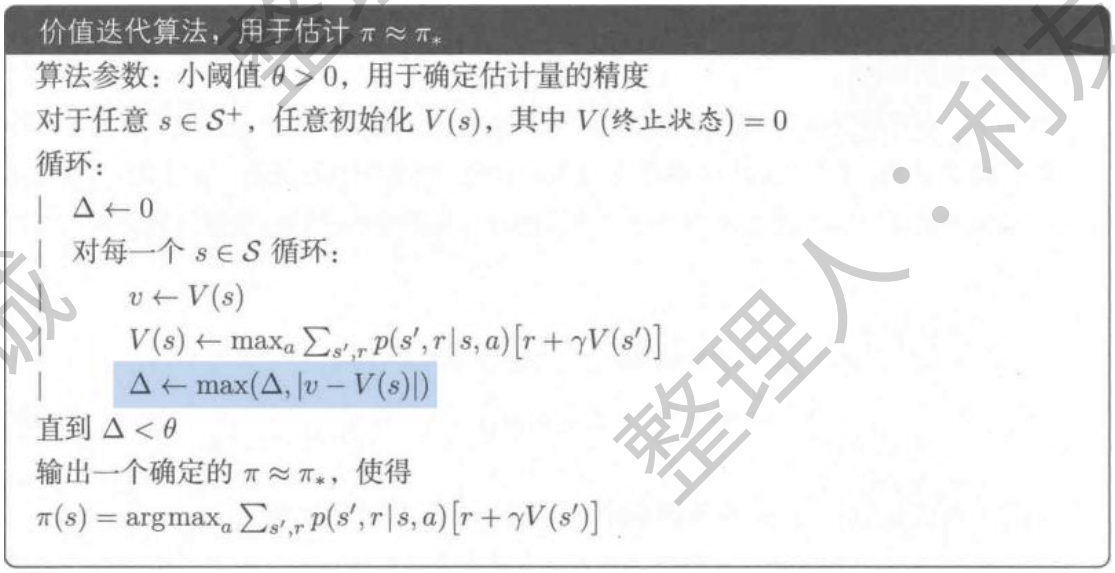


图 2：价值迭代算法

叁、异步动态规划（就地迭代的 DP）

按照公式（11）**乱序更新** $v_\pi(s)$，利于和实际环境交互。

肆、广义策略迭代（GPI）

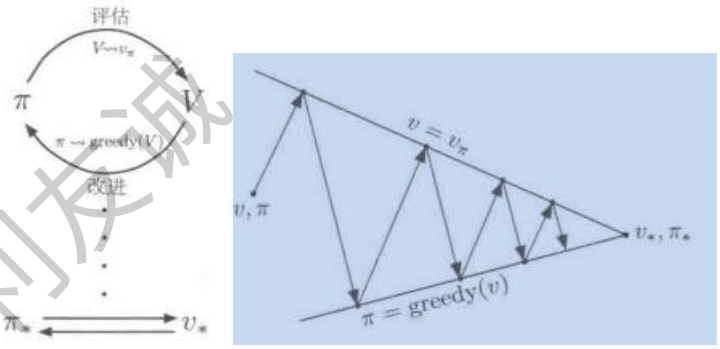


图 3：广义策略迭代

我们将 GPI 的评估和改进流程视作两个约束或目标之间相互作用的过程。

强化学习极简笔记（五）：蒙特卡洛搜索

核心：不假设拥有完备的环境知识，仅需要 trajectory（状态、动作、收益序列）
概述：分幕式任务的 MC 算法（逐幕作出改进，而非每一步（在线）作出改进）；随机估计方法（对完整的回报取平均）；采用 GPI 处理非平稳问题（参考赌博机开始的时候的非平稳上升过程）。

一、蒙特卡洛预测（给定策略由 MC 方法学习状态价值函数）

给定策略 π 下途经状态 s 的多幕数据，想要估计策略 π 下状态 s 的价值函数 $v_\pi(s)$。在给定的某一幕中，第一次访问为 s 的首次**访问**。

首次访问型 MC 算法用 s 的所有首次访问的汇报的平均值估计 $v_\pi(s)$。**每次访问型 MC 算法**则使用所有访问的回报的平均值。伪代码上两种方法的差异在于每次访问型 MC 算法不需要检查 S_t 是否在当前幕早期时段出现。两种算法有不同的数学基础，但都可以保证当 s 的访问次数（或首次访问次数）趋向无穷时，均收敛到 $v_\pi(s)$。

首次访问型 MC：算法中每个回报值都是对 $v_\pi(s)$ 的一个独立同分布的估计，且估计的方差有限。根据大数定理，这一平均值的序列会收敛到其期望值，每次平均都是一个无偏估计，其误差的标准差以 $1/\sqrt{n}$ 衰减，这里的 n 为被平均的回报值的个数。

每次访问型 MC：估计值会二阶收敛到 $v_\pi(s)$（Singh & Sutton，1996）。

首次访问型 MC 预测算法，用于估计 $V \approx v_\pi$

输入：待评估的策略 π

初始化：

对所有 $s \in \mathcal{S}$，任意初始化 $V(s) \in \mathbb{R}$

对所有 $s \in \mathcal{S}$，$Returns(s) \leftarrow$ 空列表

无限循环 (对每幕)：

根据 π 生成一幕序列：$S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

对本幕中的每一步进行循环，$t = T-1, T-2, \ldots, 0$：　倒序遍历

$G \leftarrow \gamma G + R_{t+1}$

除非 S_t 在 $S_0, S_1, \ldots, S_{t-1}$ 中已出现过：

将 G 加入 $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$

图 1.1：首次访问型 MC

Algorithm 1: First-Visit MC Prediction

Input: policy π, positive integer *num_episodes*
Output: value function V ($\approx v_\pi$, if *num_episodes* is large enough)
Initialize $N(s) = 0$ for all $s \in \mathcal{S}$
Initialize Returns$(s) = 0$ for all $s \in \mathcal{S}$
for *episode* $e \leftarrow 1$ ***to*** $e \leftarrow num\_episodes$ **do**
  Generate, using π, an episode $S_0, A_0, R_1, S_1, A_1, R_2 \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for *time step* $t = T-1$ ***to*** $t = 0$ *(of the episode e)* **do**
    $G \leftarrow G + R_{t+1}$
    if *state* S_t *is* ***not*** *in the sequence* $S_0, S_1, \ldots, S_{t-1}$ **then**
      Returns$(S_t) \leftarrow$ Returns$(S_t) + G_t$
      $N(S_t) \leftarrow N(S_t) + 1$
  end
end
$V(s) \leftarrow \frac{\text{Returns}(s)}{N(s)}$ for all $s \in \mathcal{S}$
return V

图 1.2：首次访问型 MC（详细版本）

Algorithm 2: Every-Visit MC Prediction

Input: policy π, positive integer *num_episodes*
Output: value function V ($\approx v_\pi$, if *num_episodes* is large enough)
Initialize $N(s) = 0$ for all $s \in \mathcal{S}$
Initialize Returns$(s) = 0$ for all $s \in \mathcal{S}$
for *episode* $e \leftarrow 1$ ***to*** $e \leftarrow num\_episodes$ **do**
  Generate, using π, an episode $S_0, A_0, R_1, S_1, A_1, R_2 \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  for *time step* $t = T-1$ ***to*** $t = 0$ *(of the episode e)* **do**
    $G \leftarrow G + R_{t+1}$
    Returns$(S_t) \leftarrow$ Returns$(S_t) + G_t$
    $N(S_t) \leftarrow N(S_t) + 1$
  end
end
$V(s) \leftarrow \frac{\text{Returns}(s)}{N(s)}$ for all $s \in \mathcal{S}$
return V

图 1.3：每次访问型 MC（详细版本）

二、动作价值的 MC 估计

动机：如果无法获得环境的模型，那么计算动作价值比计算状态价值更有用。

难点：一些“状态-动作”二元组可能永远无法访问到。

改进：估计一个状态中可采取的所有动作的价值，而不是仅仅是当前更偏好的某个特定动作的价值函数。

改进策略一（试探性出发）：将指定的“状态-动作”二元组作为起点开始一幕采样，同时保证所有“状态-动作”二元组都有非零的概率可以被选为起点。

改进策略二：只考虑那些在每个状态下所有动作都有非零概率被选中的随机策略。

三、MC 控制

目标：采用 MC 估计解决控制问题，即如何近似最优的策略。

基本思想：GPI（广义策略迭代），同时维护一个近似的策略和近似的价值函数。

收敛假设：1. 试探性出发假设；2. 进行策略评估时有无限多幕样本序列进行试探。

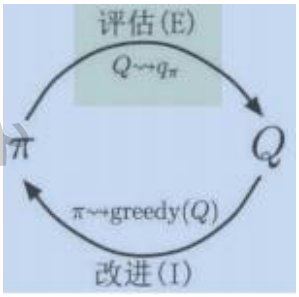


图 3.1：GPI

经典策略迭代算法的 MC 版本如下，经历了很多幕后，近似的动作价值函数会渐进趋向真实的动作价值函数（**假设观测到无限多幕的序列，并且每一幕保证试探性出发，对于一切 π，MC 算法都可以保证估计到对应的 q_π**）

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \ldots \xrightarrow{I} \pi_* \xrightarrow{E} q_{\pi_*} \tag{3.1}$$

策略改进的方法在于**在当前价值函数上贪心地选择动作**，即

$$\pi(s) \doteq \arg\max_a q(s,a) \tag{3.2}$$

$$\begin{aligned} q_{\pi_k}\left(s, \pi_{k+1}(s)\right) &= q_{\pi_k}\left(s, \arg\max_a q_{\pi_k}(s,a)\right) \\ &= \max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}\left(s, \pi_k(s)\right) \\ &\geq v_\pi(s) \end{aligned} \tag{3.3}$$

<important> 公式（3.3）满足策略改进定理，这个过程保证了 π_{k+1} 一定比 π_k 更优，除非 π_k 已经是最优策略，这一情况下二者均为最优策略。这个过程反过来保证了整个流程一定收敛到最优的策略和最优的价值函数。这样，在只能得到若干幕采样序列而不知道环境动态知识时，MC 算法可以用来寻找最优策略。

如何去除无限多幕采样的假设：1. 借鉴 DP，设定一个测度，来分析逼近误差的幅度和出现概率的上下界，然后采取足够多的步数来保证这些界足够小（需要大量的幕序列以用于计算）。2. 参考价值迭代，不再要求在策略改进前完成策略评估，选择逐幕交替进行评估与改进（**基于试探性出发的蒙特卡洛（MCES）**）。

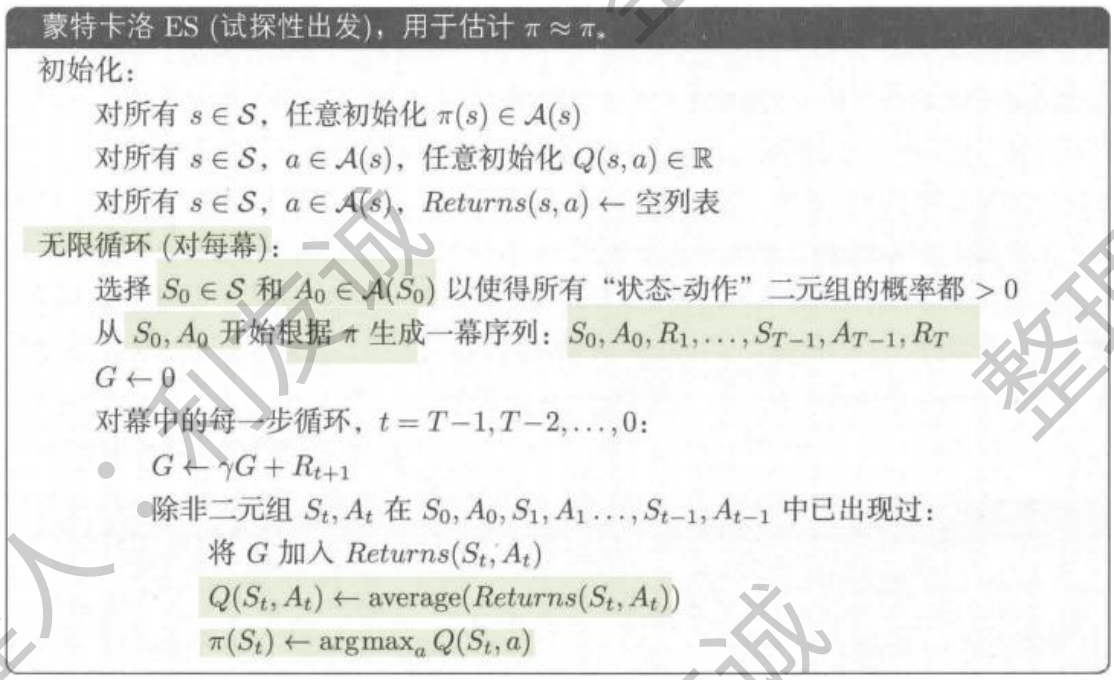


图 3.2：MCES

四、没有试探性出发假设的 MC 控制（同轨策略）

替代试探性出发假设：Agent 能够持续不断地选择所有可能动作。

方法：同轨策略（on-policy）&离轨策略（off-policy）（区别在于采样的策略和目标策略是否一致）

同轨策略方法中，**策略一般是“软性”的，即对与任意的** $s \in S, a \in A(s)$ **都有** $\pi(a|s) > 0$**，但是他们会逐渐逼近一个确定性策略。**在所有 e-软性策略中，e-贪心策略在某种层面上是最接近贪心策略的，即对于所有非贪心动作，都有 $\varepsilon / |A(s)|$ 的概率被选中，贪心策略选中的概率为 $1 - \varepsilon + \varepsilon / |A(s)|$。

同轨策略算法的 MC 控制的总体思想依然是 GPI。使用首次访问型 MC 算法估计当前策略的动作价值函数。由于**缺乏试探性出发的假设**，不能简单地通过对当前价值函数进行贪心优化来改进策略，否则无法进一步试探贪心动作。注意，**GPI 不要求优化策略一定是贪心**的，只需要逐渐逼近贪心策略即可。对于任意一个 e-软性策略 π，根据 q_π 生成的任意一个 e-贪心策略保证优于或等于 π

同轨策略的首次访问型 MC 控制算法 (对于 ε-软性策略)，用于估计 $\pi \approx \pi_*$

算法参数：很小的 $\varepsilon > 0$

初始化：

$\pi \leftarrow$ 一个任意的 ε-软性策略

对所有 $s \in \mathcal{S}$，$a \in \mathcal{A}(s)$，任意初始化 $Q(s,a) \in \mathbb{R}$

对所有 $s \in \mathcal{S}$，$a \in \mathcal{A}(s)$，$Returns(s,a) \leftarrow$ 空列表

无限循环 (对每幕)：

根据 π 生成一幕序列：$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

对幕中的每一步循环，$t = T-1, T-2, \ldots, 0$：

- $G \leftarrow \gamma G + R_{t+1}$

  除非“状态-动作”二元组 S_t, A_t 在 $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$ 已出现过：

  把 G 加入 $Returns(S_t, A_t)$

  $Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

  $A^* \leftarrow \text{argmax}_a Q(S_t, a)$　　(有多个最大值时任意选取)

  对所有 $a \in \mathcal{A}(S_t)$：

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

图 4.1：同轨策略的首次访问型 MC 控制算法（对于 e-软性策略）

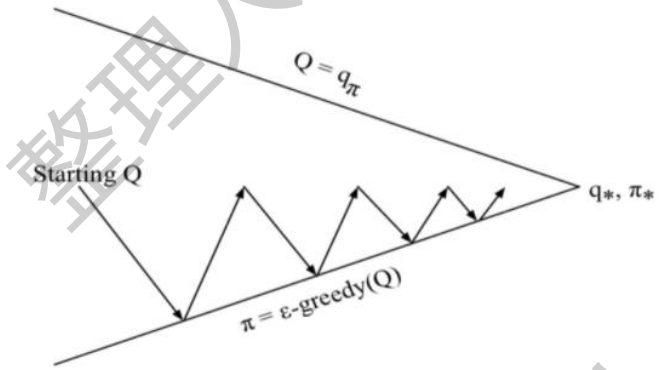


图 4.2：同轨策略的首次访问型 MC 控制算法（对于 e-软性策略）GPI

根据策略改进定理，e-软性策略可以对策略进行改进

$$
\begin{aligned}
&q_\pi\left(s,\pi'(s)\right)=\sum_a \pi'(a|s)q_\pi(s,a)\\
&=\frac{\varepsilon}{|A(s)|}\sum_a q_\pi(s,a)+(1-\varepsilon)\max_a q_\pi(s,a)\\
&\geq\frac{\varepsilon}{|A(s)|}\sum_a q_\pi(s,a)+(1-\varepsilon)\sum_a\frac{\pi(a|s)-\frac{\varepsilon}{|A(s)|}}{1-\varepsilon}q_\pi(s,a)\\
&=\frac{\varepsilon}{|A(s)|}\sum_a q_\pi(s,a)-\frac{\varepsilon}{|A(s)|}\sum_a q_\pi(s,a)+\sum_a \pi(a|s)q_\pi(s,a)\\
&=v_\pi(s)
\end{aligned}
\tag{4.1}
$$

$$
\begin{aligned}
\tilde{v}_*(s)&=(1-\varepsilon)\max_a \tilde{q}_*(s,a)+\frac{\varepsilon}{|A(s)|}\sum_a \tilde{q}_*(s,a)\\
&=(1-\varepsilon)\max_a\sum_{s',r}p(s',r|s,a)\left(r+\gamma\tilde{v}_*(s')\right)+\\
&\frac{\varepsilon}{|A(s)|}\sum_a\tilde{q}_*(s,a)\sum_{s',r}p(s',r|s,a)\left(r+\gamma\tilde{v}_*(s')\right)
\end{aligned}
\tag{4.2}
$$

需要注意，MC 控制学习到的是一个 e-软性策略，和 e 有关，但是实际执行的时候可以轻松转化成确定性策略。

五、基于重要度采样的离轨策略

目标：希望学到的动作可以使随后的智能体行为是最优的，但是为了搜索所有的动作（以保证找到最优动作），需要采取非最优行动。

难点：如何遵循试探策略采取行动的同时学习到最优策略。同轨策略方法实际上是一种妥协——它并不学习最优策略的动作值，二十学习一个接近最优且能进行试探的策略的动作值。

方法：采用两个策略，一个用来学习并最终成为最优策略，另一个更加具有试探性，用来产生智能体的行动样本。用来学习的策略称为**目标策略**，用于生成行动样本的策略被称为**行动策略**。我们认为学习所用的数据“离开”了待学习的目标策略，因此整个过程被称为**离轨策略学习**。该方法方差更大，收敛更慢，更加通用。离轨策略学习也被视作学习外部世界动态特性的多步预测模型中的关键思想。

现在讨论**预测**问题。现有两个固定策略，π 为目标策略，b 为行动策略。

基本方法：重要度采样。重要度采样是一种在给定来自其他分布的条件下，估计某种分布期望值的通用方法。给定起始状态 S_t，后续的状态-动作轨迹 $A_t,S_t,\ldots,S_T$ 在策略 π 下发生的概率，定义 $\rho_{t:T-1}$ 为**重要度采样比**，使用该比例系数可以**调整回报到正确的期望值**

$$
\begin{aligned}
&\Pr\left(A_t,S_t,\ldots,S_T|S_t,A_{t:T-1}\sim\pi\right)\\
&=\pi(A_t|S_t)p(S_{t+1}|S_t,A_t)\pi(A_{t+1}|S_{t+1})\cdots p(S_T|S_{T-1},A_{T-1})\\
&=\prod_{k=t}^{T-1}\pi(A_k|S_k)p(S_{k+1}|S_k,A_k)
\end{aligned}
\tag{5.1}
$$

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k \mid S_k)\, p(S_{k+1} \mid S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k \mid S_k)\, p(S_{k+1} \mid S_k, A_k)} = \frac{\prod_{k=t}^{T-1} \pi(A_k \mid S_k)}{\prod_{k=t}^{T-1} b(A_k \mid S_k)} \tag{5.2}$$

有两种重要度采样实现：普通重要度采样（简单平均）和加权重要度采样

$$V(s) \doteq \frac{\sum_{t \in \Gamma(s)} \rho_{t:T(t)-1} G_t}{|\Gamma(s)|} \tag{5.3}$$

$$V(s) \doteq \frac{\sum_{t \in \Gamma(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \Gamma(s)} \rho_{t:T(t)-1}} \tag{5.4}$$

数学上，两种重要度采样算法在首次访问型方法下的差异可以用偏差与方差来表示。普通重要度采样的估计是无偏的，而加权重要度采样的估计是有偏的（偏差渐进收敛到 0）。另一方面，普通重要度采样的方差一般是无界的，因为重要度采样比的方差是无界的。而在加权估计中任何汇报的最大权值为 1.

六、增量式实现

假设有一个回报序列 $G_1, G_2, \ldots, G_{n-1}$，都从相同状态开始，且每一个回报都对应一个随机权重 W_i，我们希望得到如下估计，并且在获得一个额外汇报 G_n 时能保持更新

$$V_n \doteq \frac{\sum_{t \in \Gamma(s)} W_k G_t}{\sum_{t \in \Gamma(s)} W_k}, n \geq 2 \tag{6.1}$$

为了能不断跟踪 V_n 的变化，必须为每一个状态维护前 n 个回报对应的权值的累加和 C_n。V_n 的更新方法是，这里定义 $C_0 = 0$

$$V_{n+1} \doteq V_n + \frac{W_n}{C_n}(G_n - V_n) \tag{6.2}$$

$$C_{n+1} \doteq C_n + W_{n+1} \tag{6.3}$$

离轨策略 MC 预测算法（策略评估）为

离轨策略 MC 预测算法 (策略评估)，用于估计 $Q \approx q_\pi$

输入：一个任意的目标策略 π

初始化，对所有 $s \in \mathcal{S}$，$a \in \mathcal{A}(s)$：

　　任意初始化 $Q(s,a) \in \mathbb{R}$

　　$C(s,a) \leftarrow 0$

无限循环 (对每幕)：

　　$b \leftarrow$ 任何能包括 π 的策略

　　根据 b 生成一幕序列：$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

　　$G \leftarrow 0$

　　$W \leftarrow 1$

　　对幕中的每一步循环，$t = T-1, T-2, \ldots, 0$，当 $W \neq 0$ 时：

　　　　$G \leftarrow \gamma G + R_{t+1}$

　　　　$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

　　　　$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$

　　　　$W \leftarrow W \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$

　　　　如果 $W = 0$，则退出内层循环

图 6.1：离轨策略 MC 预测算法（策略评估）

七、离轨策略 MC 控制

离轨策略 MC 控制算法，用于估计 $\pi \approx \pi_*$

初始化，对所有 $s \in \mathcal{S}$，$a \in \mathcal{A}(s)$：

　　$Q(s,a) \in \mathbb{R}$ (任意值)

　　$C(s,a) \leftarrow 0$

　　$\pi(s) \leftarrow \operatorname{argmax}_a Q(s,a)$ (出现平分的情况下选取方法应保持一致)

无限循环 (对每幕)：

　　$b \leftarrow$ 任意软性策略

　　根据 b 生成一幕数据：$S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

　　$G \leftarrow 0$

　　$W \leftarrow 1$

　　对幕中的每一时刻循环，$t = T-1, T-2, \ldots, 0$：

　　　　$G \leftarrow \gamma G + R_{t+1}$

　　　　$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$

　　　　$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$

　　　　$\pi(S_t) \leftarrow \operatorname{argmax}_a Q(S_t, a)$ (出现平分的情况下选取方法应保持一致)

　　　　如果 $A_t \neq \pi(S_t)$ 那么退出内层循环 (处理下一幕数据)

　　　　$W \leftarrow W \frac{1}{b(A_t|S_t)}$

图 7.1：离轨策略 MC 控制算法

一个潜在的问题是，当幕中某时刻后剩下的所有动作都是贪心的时候，这种方法也只会

从幕的尾部进行学习。如果非贪心的行为较为普遍，则**学习的速度会很慢**，尤其是对于那些在很长的幕中较早出现的状态就是如此。这可能会极大降低学习速度。处理这个问题的方法可能是**时序差分学习**。

强化学习极简笔记（六）：时序差分学习

〇、算法概述

概述：时序差分学习（TD）结合了蒙特卡洛方法和同代规划方法的思想：TD 可以**直接从与环境互动的经验中学习策略**，而不需要构建关于环境动态特性的模型（MC）；TD **无需等待交互的最终结果**（DP），利用了自举思想（从估计中估计），而可以基于已得到的其他状态的估计值来更新当前状态的价值函数。

核心：策略评估（预测）问题；控制问题。这些方法之间的主要**区别在于他们解决预测问题的不同方式**。

一、时序差分预测

TD 和 MC 的区别：TD 和蒙特卡洛方法都利用经验来解决预测问题。MC 方法需要一直等到一次访问后的回报知道之后，再用这个回报作为状态价值函数的目标进行估计，一个使用于非平稳环境的简单的每次访问型蒙特卡洛方法（**常量 α MC**）可以表示成

$$V\left(S_t\right) \leftarrow V\left(S_t\right)+\alpha\left(G_t-V\left(S_t\right)\right) \tag{1}$$

MC 必须等到一幕的末尾才能确定对 $V\left(S_t\right)$ 的增量（因为只有这时 G_t 才是已知的。在 t+1 时刻，TD 方法立刻就能构造出目标，并使用观察到的收益 G_t 和估计值 $V\left(S_{t+1}\right)$ 来进行一次有效更新。最简单的 TD 方法在状态转移到 S_{t+1} 并收到 R_{t+1} 的收益时会立刻作出如下更新

$$V\left(S_t\right) \leftarrow V\left(S_t\right)+\alpha\left(R_t+\gamma V_t\left(S_{t+1}\right)-V\left(S_t\right)\right) \tag{2}$$

实际上，MC 的目标是 G_t，而 TD 更新的目标是 $R_t+\gamma V_t\left(S_{t+1}\right)$。这种 TD 方法被称为 TD(0)，或单步 TD，算法如下

表格型 TD(0) 算法，用于估计 v_π

输入：待评估的策略 π

算法参数：步长 $\alpha \in (0,1]$

对于所有 $s \in \mathcal{S}^+$，任意初始化 $V(s)$，其中 $V(\text{终止状态})=0$

对每幕循环：

　初始化 S

　对幕中的每一步循环：

　　$A \leftarrow$ 策略 π 在状态 S 下做出的决策动作

　　执行动作 A，观察到 R，S'

　　$V(S) \leftarrow V(S)+\alpha[R+\gamma V(S')-V(S)]$

　　$S \leftarrow S'$

　直到 S 为终止状态

图 1：表格型 TD(0)

由于

$$v_\pi(s) \doteq \mathrm{E}_\pi\left(G_t \mid S_t=s\right) \tag{3}$$

$$=\mathrm{E}_\pi\left(R_{t+1}+\gamma G_{t+1} \mid S_t=s\right) \tag{4}$$

$$= \mathrm{E}_{\pi}\left(R_{t+1} + \gamma v_{\pi}\left(S_{t+1}\right) | S_t = s\right) \tag{5}$$

MC 方法把对公式（3）的估计值作为目标，DP 方法把对（5）的估计值作为目标。MC 的目标是所以是一个估计值，是因为（3）中的期望值是未知的，我们用样本回报来代替实际的期望回报。DP 的目标之所以是一个估计值则不是因为期望值的原因，其会假设由环境模型完整地提供期望值，真正的原因是因为真实的 $v_{\pi}\left(S_{t+1}\right)$ 是未知的，因此要使用当前的估计值 $V\left(S_{t+1}\right)$ 来替代。TD 的目标也是一个估计值：它采样得到（4）的期望值，并且使用当前的估计值 V 来替代真实值 v_{π}。因此，**TD 结合了 MC 和 DP**。

TD 误差如下

$$\delta_t \doteq R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right) \tag{6}$$

每个时刻的 TD 误差是当前时刻估计的误差，由于 TD 误差取决于下一个状态和下一个收益，所以要到一个时刻步长之后才可获得。如果价值函数数组 V 在一幕中没有改变（MC 中如此），则 MC 误差可以写成 TD 误差的和

$$\begin{aligned} G_t - V\left(S_t\right) &= R_{t+1} + \gamma G_{t+1} - V\left(S_t\right) + \gamma V\left(S_{t+1}\right) - \gamma V\left(S_{t+1}\right) \\ &= \delta_t + \gamma\left(G_{t+1} - V\left(S_{t+1}\right)\right) \\ &= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k \end{aligned} \tag{7}$$

如果 V 在该幕中变化了（如 TD(0)的情况，V 不断地更新），那么这个等式就不准确，大如果时刻步长较小，则等式仍能近似成立。

二、时序差分预测方法的优势

优势 1：TD 不需要环境模型，即描述受益和下一状态联合概率分布的模型。

优势 2：相比 MC，TD 自然地运用了一种在线的、完全递增的方法来实现。MC 必须等到一幕的结束，因为只有那时我们才能知道确切的回报值，而 TD 方法只需要等到下一时刻即可。

TD 的收敛性：对于任意固定的策略 π，TD(0)都已经被证明能够收敛到 v_{π}。如果步长参数是一个足够小的常数，那么它的均值能够收敛到 v_{π}；如果步长参数根据随机近似条件（8）逐渐变小，则它能够以概率 1 收敛（**概率 1 收敛一定均值收敛，均值收敛不一定概率 1 收敛**）

$$\sum_{n=1}^{\infty} \alpha_n(a) = 0 \wedge \sum_{n=1}^{\infty} \alpha_n^2(a) < \infty \tag{8}$$

TD 的收敛速度：在实践中，TD 方法在随机任务上通常比常量 α MC 方法收敛更快。

三、TD(0)的最优性

批量更新：只有在处理了整批的训练数据后才进行更新。给定近似价值函数 V，在访问非终止状态的每个时刻 t，使用公式（1）和（2）计算相应的增量，但价值函数仅根据所有增量的和改变一次。然后，利用新的价值函数再次处理所有可用的经验，产生新的总增量，以此类推，直到价值函数收敛。

在批量更新下，只要选择足够小的步长参数 α，TD(0)就能确定地收敛到与 α 无关的唯

一结果。常数 α MC 方法在相同的条件下也能确定地收敛，但会收敛到不同的结果。在批量训练下，常数 α MC 的收敛值 $V(s)$ 是经过状态 s 后得到的实际回报的样本平均值。可以认为他们是最优估计，因为他们最小化了与训练集中实际回报的均方根误差。但是，批量 TD 方法在均方根误差上表现得更好：**MC 方法只是从某些有限的方面来说最优，而 TD 方法的最优性则与预测回报这个任务更为相关**。

批量 MC 方法总是找出最小化训练集上均方误差的估计，而**批量 TD(0)总是找出完全符合马尔可夫过程模型的最大似然估计参数**。通常，一个参数的最大似然估计是使得生成训练数据的概率最大的参数值。这种估计被称为确定性等价估计，因为它等价于假设潜在过程参数的估计是确定性的而不是近似的。批量 TD(0)通常收敛到确定性等价估计。

为什么 TD 方法比 MC 方法更快收敛：**在以批量的形式学习的时候，TD(0)比 MC 方法更快是因为它计算的是真正的确定性等价估计**。

对于状态空间巨大的任务，TD 方法可能是**唯一**可行的逼近确定性等价解的方法。

四、Sarsa：同轨策略下的时序差分控制（on-policy TD）

一幕数据指的是一个状态和动作交替出现的序列

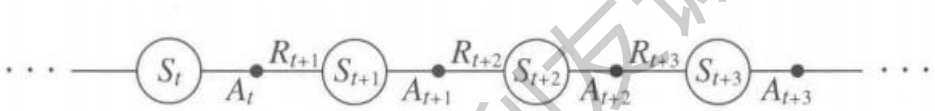
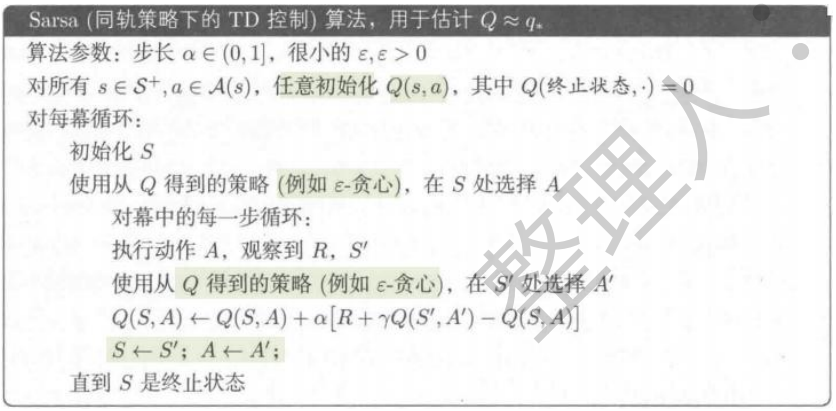


图 2：一幕数据

确保状态值在 TD(0)下收敛的定理也同样适用于对应的关于动作值的算法上

$$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\alpha\left(R_{t+1}+\gamma Q(S_{t+1},A_{t+1})-Q(S_t,A_t)\right) \tag{9}$$

这个更新规则用到了描述这个事件的五元组 $(S_t,A_t,R_{t+1},S_{t+1},A_{t+1})$，因此得名 **Sarsa**。Sarsa 算法的收敛性取决于策略对于 Q 的依赖程度。**若使用 e-贪心或 e-软性策略，只要所有状态-动作二元组都被无限多次地访问到，并且贪心策略在极限的情况下能够收敛（这个收敛过程可以通过令 $\varepsilon=1/t$ 来实现），Sarsa 就能够以 1 的概率收敛到最优策略和动作价值函数**


Sarsa (同轨策略下的 TD 控制) 算法，用于估计 $Q\approx q_*$

算法参数：步长 $\alpha\in(0,1]$，很小的 $\varepsilon,\varepsilon>0$

对所有 $s\in\mathcal{S}^+,a\in\mathcal{A}(s)$，任意初始化 $Q(s,a)$，其中 $Q(\text{终止状态},\cdot)=0$

对每幕循环：

　初始化 S

　使用从 Q 得到的策略 (例如 ε-贪心)，在 S 处选择 A

　　对幕中的每一步循环：

　　执行动作 A，观察到 R，S'

　　使用从 Q 得到的策略 (例如 ε-贪心)，在 S' 处选择 A'

　　$Q(S,A)\leftarrow Q(S,A)+\alpha[R+\gamma Q(S',A')-Q(S,A)]$

　　$S\leftarrow S';\ A\leftarrow A';$

　直到 S 是终止状态

图 3：Sarsa（同轨策略下的 TD 控制）算法

五、Q 学习：离轨策略下的时序差分控制

Q 学习定义为

$$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\alpha\left(R_{t+1}+\gamma\max_a Q(S_{t+1},a)-Q(S_t,A_t)\right) \tag{10}$$

在这里，待学习的动作价值函数 Q 采用了对最优动作价值函数 q_* 的直接近似作为学习的目标，而与用于生成智能体决策序列轨迹的行动策略是什么无关。Q 学习算法的流程如下

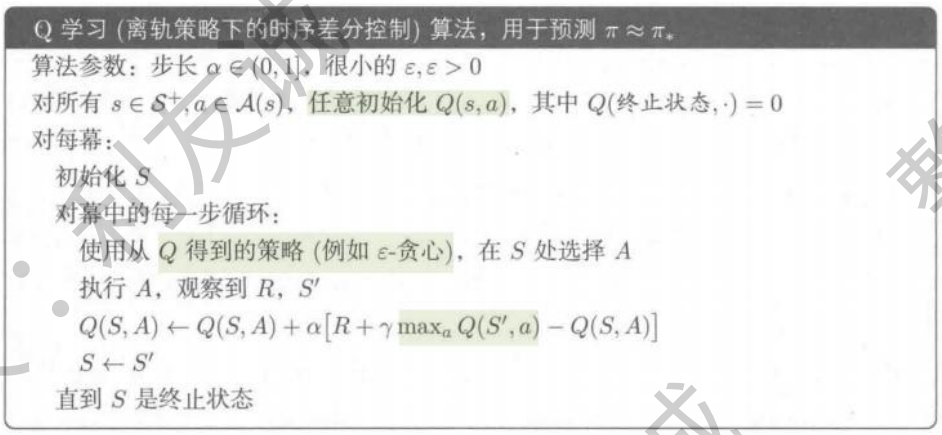


图 4：Q 学习（离轨策略下的时序差分控制）算法

为什么 Q 学习是一种离轨策略控制方法，而 Sarsa 是一种同轨策略的控制方法

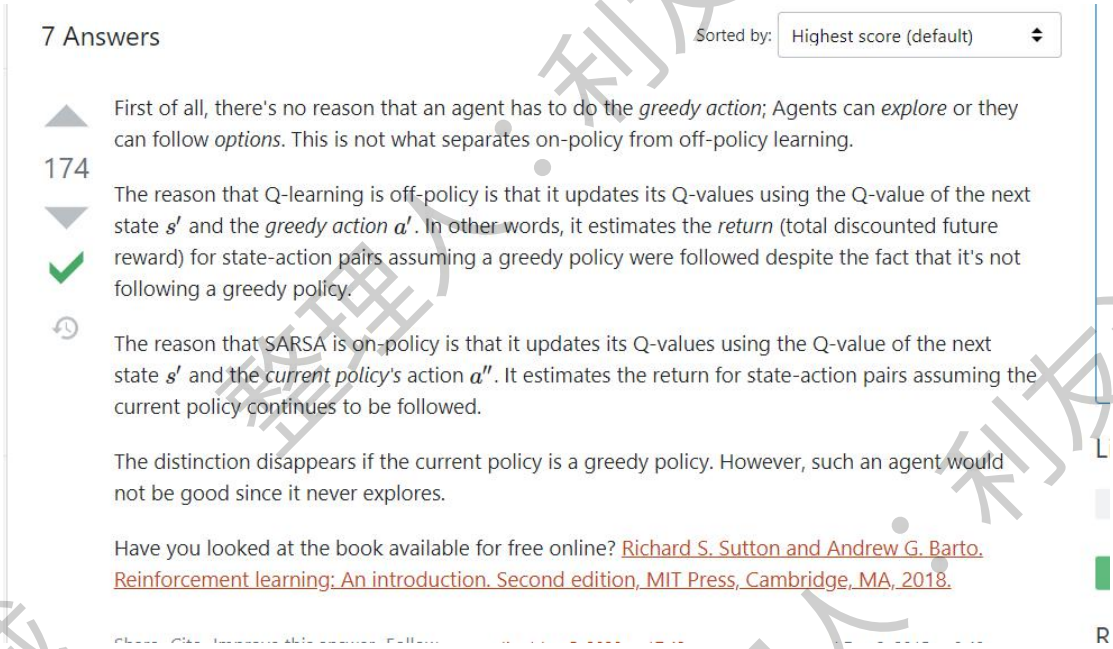


图 5：为什么 Q 学习是离轨策略控制方法，而 Sarsa 是一种同轨策略控制方法

六、期望 Sarsa（一种离轨策略控制方法）

考虑一种和 Q 学习类似，但把对于下一个状态-动作二元组取最大值一步换成取**期望**的学习算法

$$\begin{aligned} Q(S_t,A_t)&\leftarrow Q(S_t,A_t)+\alpha\left(R_{t+1}+\gamma \mathrm{E}\left(Q(S_{t+1},A_{t+1})\mid S_{t+1}\right)-Q(S_t,A_t)\right)\\ &\leftarrow Q(S_t,A_t)+\alpha\left(R_{t+1}+\gamma\sum_a \pi(a\mid S_{t+1})Q(S_{t+1},a)-Q(S_t,A_t)\right)\end{aligned} \tag{11}$$

给定下一个状态，这个算法确定地向期望意义上的 Sarsa 算法所决定的方向上移动。因此被称作期望 Sarsa。

期望 Sarsa 在计算上比 Sarsa 更为复杂，但是它消除了因为随机选择动作而产生的方差。在相同数量的经验上，期望 Sarsa 的表现优于 Sarsa，且可以随机设置一个 α 而不需要单行长期稳态性能的损失。

七、最大化偏差与双学习

目前为止讨论的所有控制算法在构建目标策略时都包含了最大化操作，在估计值的基础上进行最大化也可以被看作隐式地对最大值进行估计，而这回带来一个显著的正偏差，我们将其称作**最大化偏差**。

最大化偏差会损坏 TD 控制算法的性能，可以使用双学习来避免。双学习一共学习两个估计值，以双 Q 学习举例：**一个估计 Q_1 来确定最大的动作，再用另一个 Q_2 来计算其价值的估计**

$$Q_1(S_t,A_t) \leftarrow Q_1(S_t,A_t)+\alpha\left(R_{t+1}+\gamma Q_2\left(S_{t+1},\arg\max_a Q_1(S_{t+1},a)\right)-Q_1(S_t,A_t)\right) \tag{12}$$

双学习消除了最大化偏差带来的性能损失，过程如下

双 Q 学习，用于估计 $Q_1 \approx Q_2 \approx q_*$

算法参数：步长 $\alpha \in (0,1]$，很小的 $\varepsilon, \varepsilon > 0$

对所有 $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$，初始化 $Q_1(s,a)$ 和 $Q_2(s,a)$，其中 $Q(\text{终止状态},\cdot)=0$

对每幕循环：

初始化 S

对幕中的每一步循环：

基于 Q_1+Q_2，使用 ε-贪心策略在 S 中选择 A

执行动作 A，观察到 R，S'

以 0.5 的概率执行：

$Q_1(S,A) \leftarrow Q_1(S,A)+\alpha\left(R+\gamma Q_2\left(S',\text{argmax}_a\, Q_1(S',a)\right)-Q_1(S,A)\right)$

或者执行：

$Q_2(S,A) \leftarrow Q_2(S,A)+\alpha\left(R+\gamma Q_1\left(S',\text{argmax}_a\, Q_2(S',a)\right)-Q_2(S,A)\right)$

$S \leftarrow S'$

直到 S 是终止状态

图 6：双 Q 学习

八、游戏、后位状态和其他特殊例子

传统的状态价值估计的是在某个状态中，当智能体可以选择执行一个动作时，这个状态的价值，但是也可以评估智能体执行这一动作之后的状态的价值，我们称之为**后位状态**（afterstates），并将它们的价值函数称之为后位状态价值函数。后位状态价值函数就是利用了先验知识的更加高效的学习方法。

强化学习极简笔记（九）：基于函数逼近的同轨策略预测

〇、表格型近似求解方法

目标：将表格型方法扩展到任意大的状态空间的问题上。我们的目标从期望能够找到最优的策略或最优价值函数，转而为**使用有限的计算资源找到一个比较好的近似解**。

要求：泛化能力：如何将状态空间中的一个有限子集的经验推广来很好地近似一个大得多的子集。

函数逼近：从样例中进行泛化，从一个预期的函数（如一个价值函数）中获得实例，并试图对他们进行泛化来逼近整个函数。函数逼近是一个有监督学习的例子。

新的问题：非平稳性、自举和延时目标。

一、价值函数逼近

近似价值函数：我们将给定权值向量 w 在状态 s 的近似价值函数写作

$$\hat{v}(s,w)\approx v_{\pi}(s) \tag{1}$$

权值的数量（w 的维度）远远小于状态的数量（d《|S|），另外改变一个权值将改变许多状态的估计值。因此，**当一个状态更新时，许多其他状态的价值函数也会被更改**。这样的泛化能力可能使得学习能力更为强大，但也可能更加难以控制和理解。

将 RL 扩展到函数逼近有助于解决“部分观测”问题，即智能体无法观测到完整的状态的情况。如果参数化函数 $\hat{v}$ 的形式化表达并不依赖于状态的某些特定的部分，那么可以认为状态的这些部分是**不可观测**的。

定义符号（2）表示一次单独的更新，s 表示更新的状态，而 u 表示 s 的估计价值朝向的更新目标，如 MC 的价值函数预测的更新为 $s\mapsto G_t$。值得注意的是：**MC 和 TD 只更新遇到的状态；DP 更新任意状态。**

$$s\mapsto u \tag{2}$$

自然地，将每一次更新解释为给价值函数指定一个理想的“输入-输出”范例样本。更新 $s\mapsto u$ 意味着状态 s 的估计价值需要更接近更新目标 u。学习拟合“输入-输出”范例的机器学习方法叫**有监督学习**，当输出 u 为数字时，这个过程通常被称作**函数逼近**。

并不是所有的函数逼近方法都适用于 RL：RL 需要**在线学习**，即智能体需要与环境或环境的模型进行交互，这要求算法能够逐步得到的数据中有效地进行学习。此外，RL 需要能够处理**非平稳目标函数（即随时间变化的目标函数）**的函数逼近方法。

二、预测目标（$\overline{VE}$）

表格型：不需要对预测质量的连续函数进行衡量，这是因为学习到的价值函数完全可以与真是的价值函数精确相等。每个状态下学习的价值函数都是解耦的：一个状态的更新不会影响其他状态。

函数逼近：一个状态的更新会影响到许多其他状态，而且不可能让所有状态的价值完全正确，这是因为状态的数量远多于权值的数量，所以一个状态的估计价值越准确就意味着别

的状态的估计价值变得不那么准确，所以我们需要给出哪些状态是我们最关心的。

指定一个状态分布（3），表示对每个状态 s 的误差的关心程度

$$\mu(s)\geq 0, \sum_s \mu(s)=1 \tag{3}$$

我们得到一个目标函数，即**均方价值误差**，记作 $\overline{VE}$

$$\overline{VE}(w)\doteq \sum_{s\in S}\mu(s)\left(v_\pi(s)-\hat{v}(s,w)\right)^2 \tag{4}$$

这个度量的平方根，即 $\overline{VE}$ 方根，给出了一个粗略的对近似价值函数与真实价值差异大小的度量。我们**将在状态 s 上消耗的计算时间的比例定义为** $\mu(s)$。同轨策略训练中称其为同轨策略分布。在持续性任务中，同轨策略分布是 π 的平稳分布。

分幕式任务中的同轨策略分布

在分幕式任务中，同轨策略分布与上文有些区别，因为它与幕的初始状态的选取有关。令 $h(s)$ 表示从状态 s 开始一幕交互序列的概率，令 $\eta(s)$ 表示 s 在单幕交互中消耗的平均时间 (时刻步数)。所谓“在状态 s 上的消耗时间”是指，由 s 开始的一幕序列，或者发生了一次由前导状态 $\bar{s}$ 到 s 的转移

$$\eta(s)=h(s)+\sum_{\bar{s}}\eta(\bar{s})\sum_a \pi(a|\bar{s})p(s|\bar{s},a),\ \text{对所有}s\in\mathcal{S}. \tag{9.2}$$

可以求解这个方程组得到期望访问次数 $\eta(s)$。同轨策略 分布被定义为其归一化的时间消耗的比例

$$\mu(s)=\frac{\eta(s)}{\sum_{s'}\eta(s')},\ \text{对所有}s\in\mathcal{S}. \tag{9.3}$$

这是一个对没有折扣问题的自然考虑。如果存在折扣 ($\gamma<1$)，那么可以将它表示为某种形式的幕终止条件 (软性终止)，为此，可以简单地在式 (9.2) 中的第二项加入因子 γ。

图 1：分幕式任务中的同轨策略分布

在近似求解持续性和分幕式任务的情况下需要进行分类讨论，这使得学习目标的确定更加规范。

对于 $\overline{VE}$ 而言，最理想的目标是对于所有可能的 w 找到一个**全局最优**的权值向量 w^*，满足

$$\overline{VE}(w^*)\leq \overline{VE}(w) \tag{5}$$

对于复杂的函数逼近模型，转而寻找**局部最优**，即找到一个权值向量 w^* 及其附近所有向量 w，满足（5）。

三、随机梯度和半梯度方法

随机梯度下降（SGD）：一类函数逼近方法，适用于在线强化学习。SGD 方法对于每一个样本，将权值向量朝着能够**减小这个样本的误差的方向移动**

$$w_{t+1} \doteq w_t - \frac{1}{2}\alpha\nabla\left(v_\pi(S_t) - \hat{v}(S_t, w_t)\right)^2 \tag{6}$$

$$= w_t - \frac{1}{2}\alpha\left(\frac{\partial\left(v_\pi(S_t) - \hat{v}(S_t, w_t)\right)^2}{\partial w_1}, \ldots, \frac{\partial\left(v_\pi(S_t) - \hat{v}(S_t, w_t)\right)^2}{\partial w_d}\right)^{\mathrm{T}} \tag{7}$$

$$= w + \alpha\left(v_\pi(S_t) - \hat{v}(S_t, w_t)\right)\nabla\hat{v}(S_t, w_t) \tag{8}$$

SGD 的“随机”体现在：**更新仅仅依赖于一个样本来完成**，而且该样本很可能是**随机选择**的。如果 α 以满足标准随机近似条件的方式较小，那么 SGD 方法（8）可以保证收敛到局部最优解。

若目标输出为 $U_t \in \mathbb{R}$ 不是真实的价值 $v_\pi(S_t)$，而是其随机近似。SGD 表示为

$$w_{t+1} \doteq w + \alpha\left(U_t - \hat{v}(S_t, w_t)\right)\nabla\hat{v}(S_t, w_t) \tag{9}$$

如果 U_t 是一个无偏估计，即满足对任意 t，有

$$\mathrm{E}\left(U_t \mid S_t = s\right) = v_\pi(S_t) \tag{10}$$

那么在 α 的下降满足随机近似条件的情况下，w_t **一定会收敛到一个局部最优解**。

根据以上，MC 状态价值函数预测的梯度下降版本可以办证找到一个局部最优解

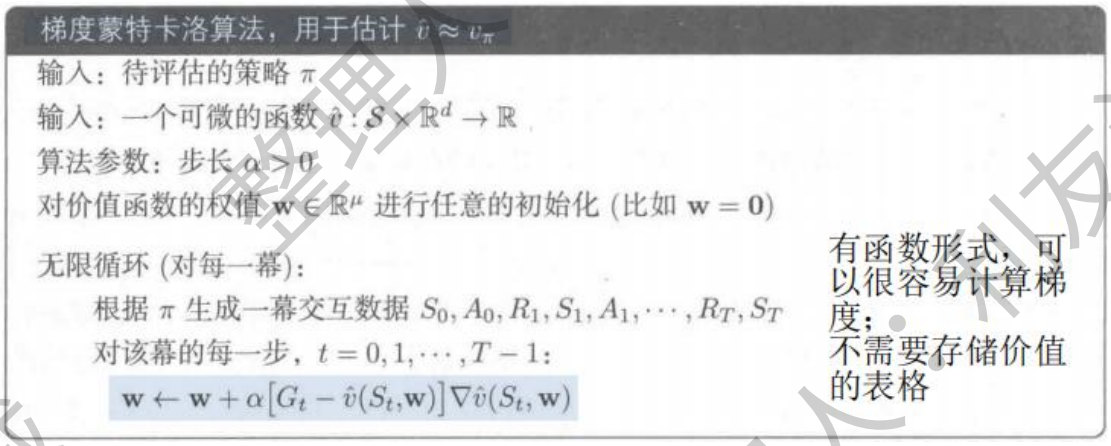


图 2：梯度 MC

注意，若使用 $v_\pi(S_t)$ 的自举估计值作为目标，则无法得到相同的收敛性保证。自举目标都取决于权值向量 w_t 的当前值，这意味着有偏的，所以无法实现真正的梯度下降法。自举法实际上并不是真正意义上的梯度下降，它只考虑了改变权值向量 w_t 对估计的影响，却忽略了对目标的影响。由于只包含了一部分梯度，我们称之为**半梯度方法**。

尽管半梯度（自举法）并不像梯度方法那样稳健收敛，但在一些情况下依然可以稳健收敛。其优势在于：**1. 学习速度比较快；2. 支持持续地和在线地学习**，而不需要等待一幕的结束。这使得可以用于持续性问题，并有一些计算优势。

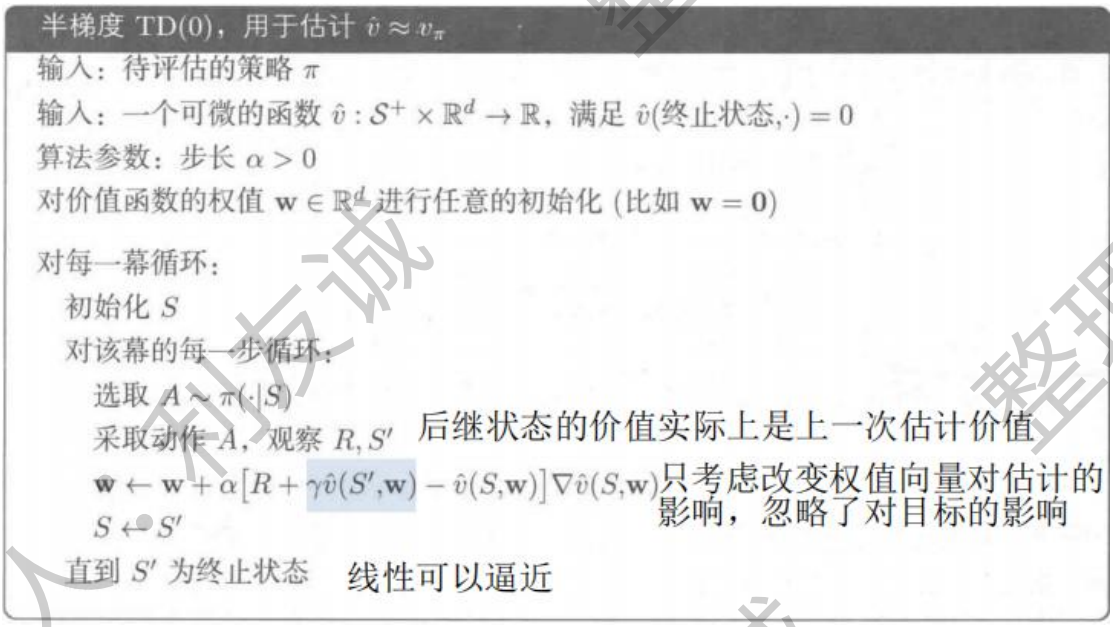


图 3：半梯度 TD(0)

状态聚合是一种简单形式的泛化函数逼近，这里状态被分为不同组，每个组对应一个估计值（对应权值向量 w 的一个维度分量）。状态的价值估计就是所在组对应的分量，当状态被更新时，只有这个对应分量被单独更新。

四、线性方法

线性近似的状态价值函数可以写作 w 和 $x(s)$ 的内积

$$\hat{v}(s,w) \doteq w^{\mathrm{T}} x(s) \doteq \sum_{i=1}^{d} w_i x_i(s) \tag{11}$$

在这种情况下，我们说近似价值函数是**依据权值线性**的，或者简单地说是**线性**的。状态 s 对应的函数值称作 s 的特征。对于线性方法，特征被称作基函数，这是因为它们构成了可能的近似函数集合的线性基。

SGD 在线性情况下可以简化为一个特别简单的形式

$$\nabla \hat{v}(S_t, w_t) = x(s) \tag{12}$$

$$w_{t+1} \doteq w + \alpha\left(U_t - \hat{v}(S_t, w_t)\right)x(s) \tag{13}$$

在线性情况下，函数只存在一个最优质，因此保证收敛到或接近局部最优值的任何方法，也都自动地保证收敛到或接近于全局最优质。

TD(0)在线性函数的情况下逼近收敛，权值向量也不是收敛到全局最优，而是靠近局部最优的点。每个时刻 t 的更新是

$$\begin{aligned} w_{t+1} &\doteq w_t + \alpha\left(U_t - \hat{v}(S_t, w_t)\right)x(s) \\ &\doteq w_t + \alpha\left(R_{t+1} + \gamma w_t^{\mathrm{T}} x_{t+1} - w_t^{\mathrm{T}} x_t\right)x_t \\ &= w_t + \alpha\left(R_{t+1}x_t - x_t\left(x_t - \gamma x_{t+1}\right)^{\mathrm{T}} w_t\right) \end{aligned} \tag{14}$$

一旦系统达到一个稳定的状态，对于任意给定的 w_t，下一个更新的权值向量的期望可

以写作

$$\mathrm{E}\left(w_{t+1}\mid w_t\right)=w_t+\alpha\left(b-Aw_t\right) \tag{15}$$

$$b\doteq\mathrm{E}\left(R_{t+1}x_t\right)\in\mathbb{R}^d$$

$$A\doteq\mathrm{E}\left(x_t\left(x_t-\gamma x_{t+1}\right)^{\mathrm{T}}\right)\in\mathbb{R}^d\times\mathbb{R}^d \tag{16}$$

从（15）可以看出，如果系统收敛，它必须收敛于满足下式的权值向量 w_{TD}

$$\begin{aligned}&b-Aw_{TD}=0\\&\Rightarrow b=Aw_{TD}\\&\Rightarrow w_{TD}\doteq A^{-1}b\end{aligned} \tag{17}$$

这个量被称为 **TD 不动点**。事实上，线性半梯度 TD(0)就收敛到这个点上，下面给出收敛性以及逆的存在性的证明

线性 TD(0) 收敛性的证明

是什么保证了线性 TD(0) 算法 (式 9.9) 的收敛性？我们可以通过重写式 (9.10) 得到

$$\mathbb{E}[\mathbf{w}_{t+1}|\mathbf{w}_t]=(\mathbf{I}-\alpha\mathbf{A})\mathbf{w}_t+\alpha\mathbf{b}. \tag{9.13}$$

注意，是矩阵 $\mathbf{A}$ 与 $\mathbf{w}_t$ 相乘而不是 $\mathbf{b}$，所以只有 $\mathbf{A}$ 与收敛性有关。为了更直观地考虑，不妨设 $\mathbf{A}$ 是对角矩阵。如果任意的对角线元素是负的，那么 $\mathbf{I}-\alpha\mathbf{A}$ 中对应的元素将大于 1，则 $\mathbf{w}_t$ 对应元素会被放大，最终造成发散。换言之，如果 $\mathbf{A}$ 的所有对角线元素为正，那么选取比最大元素的倒数小的 α，这样 $\mathbf{I}-\alpha\mathbf{A}$ 的对角线元素均在 0~1 之间。这样在更新中会促使 $\mathbf{w}_t$ 收缩，从而保证稳定性。一般，当 $\mathbf{A}$ 是正定的时候 $\mathbf{w}_t$ 会趋向于零，正定指的是对任意实向量 y 满足 $y^\top\mathbf{A}y>0$。正定同样可以保证逆 $\mathbf{A}^{-1}$ 存在。

对于线性 TD(0)，在 $\gamma<1$ 的连续情况中，矩阵 $\mathbf{A}$ (式 9.11) 可以写作

$$\begin{aligned}\mathbf{A}&=\sum_s\mu(s)\sum_a\pi(a|s)\sum_{r,s'}p(r,s'|s,a)\mathbf{x}(s)\big(\mathbf{x}(s)-\gamma\mathbf{x}(s')\big)^\top\\&=\sum_s\mu(s)\sum_{s'}p(s'|s)\mathbf{x}(s)\big(\mathbf{x}(s)-\gamma\mathbf{x}(s')\big)^\top\\&=\sum_s\mu(s)\mathbf{x}(s)\Big(\mathbf{x}(s)-\gamma\sum_{s'}p(s'|s)\mathbf{x}(s')\Big)^\top\\&=\mathbf{X}^\top\mathbf{D}(\mathbf{I}-\gamma\mathbf{P})\mathbf{X},\end{aligned}$$

这里 $\mu(s)$ 是 π 下的平稳分布，$p(s'|s)$ 是策略 π 下从 s 到 s' 的转移概率，$\mathbf{P}$ 是这个概率对应的 $|\mathcal{S}|\times|\mathcal{S}|$ 的矩阵，$\mathbf{D}$ 是对角线为 $\mu(s)$ 的 $|\mathcal{S}|\times|\mathcal{S}|$ 的对角阵，$\mathbf{X}$ 是行向量为 $\mathbf{x}(s)$ 的 $|\mathcal{S}|\times d$ 的矩阵。可以发现，中间的矩阵 $\mathbf{D}(\mathbf{I}-\gamma\mathbf{P})$ 是 $\mathbf{A}$ 正定性的关键。

对于这样的关键矩阵，当它所有列的和为非负数时，正定性可以得到保证。Sutton (1988，27 页) 中基于两个前提定理给出了证明，其中一个定理是，任意矩阵 $\mathbf{M}$ 是正定的条件为当且仅当其对称矩阵 $\mathbf{S}=\mathbf{M}+\mathbf{M}^\top$ 正定 (Sutton 1998，附录)。第二个定理是，对于任意一个对称实矩阵 $\mathbf{S}$，如果它的所有对角线元素为正且大于对应的非对角线元素绝对值之和，那么它是正定的 (Varga 1962，23 页)。对于我们的关键矩阵 $\mathbf{D}(\mathbf{I}-\gamma\mathbf{P})$，对角线元素为正，非对角线元素为负，所以问题转化为证明每个行和加上对应的列和为正。因为 $\mathbf{P}$ 是一个概率矩阵且 $\gamma<1$，所以行和为正。所以问题变为证明列和非负。注意，任意矩阵 $\mathbf{M}$ 对应的分量为列和的行向

量，可以写作 $\mathbf{1}^\top\mathbf{M}$，这里 $\mathbf{1}$ 是所有分量为 1 的列向量。令 μ 为 $\mu(s)$ 组成的 $|\mathcal{S}|$ 维向量，由于 μ 是一个平稳分布，则有 $\mu=\mathbf{P}^\top\mu$。我们的关键矩阵的列和，可以写作

$$\begin{aligned}\mathbf{1}^\top\mathbf{D}(\mathbf{I}-\gamma\mathbf{P}) &= \mu^\top(\mathbf{I}-\gamma\mathbf{P}) \\ &= \mu^\top-\gamma\mu^\top\mathbf{P} \\ &= \mu^\top-\gamma\mu^\top \qquad \text{(因为 } \mu \text{ 是平稳分布)} \\ &= (1-\gamma)\mu^\top,\end{aligned}$$

所有分量均为正。所以，关键矩阵与矩阵 $\mathbf{A}$ 都是正定的，进而同轨策略 TD(0) 是稳定的。（如果需要证明概率版本的收敛性，则需要考虑一些额外情况以及 α 随时间减小的方式。）

图 4：线性 TD(0)收敛性的证明

在 TD 不动点出，已经证明（在持续性任务的情况下）$\overline{VE}$ 在可能的最小误差的一个扩展边界内，即 TD 的渐进误差不超过 MC 的最小可能误差的 $\frac{1}{1-\gamma}$

$$\overline{VE}\left(w_{TD}\right)\le\frac{1}{1-\gamma}\min_{w}\overline{VE}\left(w\right) \tag{18}$$

以上结果的关键在于状态是按照**同轨策略分布**来更新的，对于按照其他分布的更新，例如使用函数逼近的自举法可能发散到无穷大。尽管如此，TD 方法在学习速率方面有巨大的潜在优势。

n 步半梯度 TD 算法，用于估计 $\hat{v}\approx v_\pi$

输入：待评估的策略 π

输入：一个可微的函数 $\hat{v}:\mathcal{S}^+\times\mathbb{R}^d\to\mathbb{R}$，满足 $\hat{v}(\text{终止状态},\cdot)=0$

算法参数：步长 $\alpha>0$，正整数 n

对价值函数权值 $\mathbf{w}$ 进行任意的初始化（比如 $\mathbf{w}=0$）

所有的存取操作（对 S_t 和 R_t）的索引可以对 $n+1$ 取模

对每一幕循环：

  初始化并存储 $S_0\neq$ 终止状态

  $T\leftarrow\infty$

  对 $t=0,1,2,\ldots$ 循环：

  | 如果 $t<T$，那么：

  | 根据 $\pi(\cdot|S_t)$ 采取动作

  | 观察并存储下一个收益 R_{t+1} 和下一个状态 S_{t+1}

  | 如果 S_{t+1} 为终止状态，那么 $T\leftarrow t+1$

  | $\tau\leftarrow t-n+1$（τ 是正在被更新的状态所在的时刻）

  | 如果 $\tau\geqslant 0$：

  | $G\leftarrow\sum_{i=\tau+1}^{\min(\tau+n,T)}\gamma^{i-\tau-1}R_i$

  | 如果 $\tau+n<T$，那么：$G\leftarrow G+\gamma^n\hat{v}(S_{\tau+n},\mathbf{w})\quad(G_{\tau:\tau+n})$

  | $\mathbf{w}\leftarrow\mathbf{w}+\alpha\left[G-\hat{v}(S_\tau,\mathbf{w})\right]\nabla\hat{v}(S_\tau,\mathbf{w})$

  直到 $\tau=T-1$

图 5：n 步半梯度 TD

五、线性方法的特征构造

线性方法的优势：1. 收敛性保证；2. 数据和计算方面高效。第二点取决于我们如何选用来表达状态的特征。

特征应该是提取状态空间中最通用的信息。线性形式的一个局限性在于无法表征之间的相互作用，直观解释是：单个特征维度可能无意义，特征维度的组合才有意义。

有多种方法增加线性特征的表示能力

- **多项式**（Polynomials）：基本的多项式族（Polynomial Families）可以用作函数拟合的特征矢量（Feature Vectors）。假设每一个状态 $\boldsymbol{s}=(S_1,S_2,\cdots,S_d)^{\mathrm{T}}$ 是一个 d 维向量，那么我们有一个 d 维的多项式基（Polynomial Basis）$\phi_i(\boldsymbol{s})=\prod_{j=1}^{d}S_j^{c_{i,j}}$，其中每个 $c_{i,j}$ 是集合 $\{0,1,\cdots,N\}$ 中的一个整数。这构成秩（Order）为 N 的多项式基和 $(N+1)^d$ 个不同的函数。
  - **傅立叶基**（Fourier Basis）：傅立叶变换（Fourier Transformation）经常用于表示在时间域或频率域的序列信号。有 $N+1$ 个函数的一维秩为 N 的傅立叶余弦（Cosine）基为 $\phi_i(s)=\cos(i\pi s)$，其中 $s\in[0,1]$ 且 $i=0,1,\cdots,N$。
  - **粗略编码**（Coarse Coding）：状态空间可以从高维缩减到低维，例如用一个区域覆盖决定过程（Determination Process）来进行二值化表示（Binary Representation），这被称为粗略编码。
  - **瓦式编码**（Tile Coding）：在粗略编码中，瓦式编码对于多维连续空间是一种高效的特征表示方式。瓦式编码中特征的感知域（Receptive Field）被指定成输入空间的不同分割（Partitions）。每一个分割称为一个瓦面（Tilling），而分割中的每一个元素称为一个瓦片（Tile）。许多有着重叠感知域的瓦面往往被结合使用，以得到实际的特征矢量。
  - **径向基函数**（Radial Basis Functions，RBF）：径向基函数自然地泛化了粗略编码，粗略编码是二值化的，而径向基函数可用于 $[0,1]$ 内的连续值特征。典型的 RBF 是以高斯函数（Gaussian）的形式 $\phi_i(s)=\exp(-\frac{\|s-c_i\|^2}{2\sigma_i^2})$，其中 s 是状态，c_i 是特征的原型（Prototypical）或核心状态（Center State），而 σ_i 是特征宽度（Feature Width）。
- **非线性方法**（Non-Linear Methods）：
  - **人工神经网络**（Artificial Neural Networks）：不同于以上的函数拟合方法，人工神经网络被广泛用作非线性函数拟合器，它被证明在一定条件下有普遍的拟合能力（Universal Approximation Ability）(Leshno et al., 1993)。基于深度学习技术，人工神经网络构成了现代基于函数拟合的深度强化学习方法的主体。一个典型的例子是 DQN 算法，使用人工神经网络来对 Q 值进行拟合。
- **其他方法**：
  - **决策树**（Decision Trees）：决策树 (Pyeatt et al., 2001) 可以用来表示状态空间，通过使用决策节点（Decision Nodes）对其分割。这构成了一种重要的状态特征表示方法。
  - **最近邻**（Nearest Neighbor）方法：它测量了当前状态和内存中之前状态的差异，并用内存中最接近状态的值来近似当前状态的值。

图 6：线性方法中构建特征的不同方式

六、最小二乘时序差分（LSTD）

根据（19），我们可以迭代计算 TD 不动点

$$w_{TD}\doteq A^{-1}b \tag{19}$$

迭代计算或许效率不高，我们可以通过对 A 和 b 的估计，直接计算 TD 的不动点。LSTD 方法如下

$$\hat{A}_t \doteq \sum_{k=0}^{t-1} x_k \left(x_k - \gamma x_{k+1}\right)^{\mathrm{T}} + \varepsilon \mathrm{I}, \hat{b}_t \doteq \sum_{k=0}^{t-1} R_{t+1} x_k \tag{20}$$

$$w_t \doteq \hat{A}_t^{-1} \hat{b}_t \tag{21}$$

上式是线性 TD(0)数据效率最高的一种形式，但是计算复杂度更高。

$O(d^2)$ 版本 LSTD，用于估计 $\hat{v} = \mathbf{w}^\top \mathbf{x}(\cdot) \approx v_\pi$

输入：特征表示 $\mathbf{x}\ \mathcal{S}^+ \to \mathbb{R}^d$ 满足 $\mathbf{x}(\text{终止状态}) = \mathbf{0}$

算法参数：很小的 $\varepsilon, \varepsilon > 0$

$\widehat{\mathbf{A}^{-1}} \leftarrow \varepsilon^{-1}\mathbf{I}$　　一个 $d \times d$ 的矩阵

$\widehat{\mathbf{b}} \leftarrow \mathbf{0}$　　一个 d 维向量

对每一幕循环：

　初始化 S；$\mathbf{x} \leftarrow \mathbf{x}(S)$

　对该幕的每一步循环：

　　选择并采取动作 $A \sim \pi(\cdot|S)$，观测 $R\ S'$；　　$\mathbf{x}' \leftarrow \mathbf{x}(S')$

　　$\mathbf{v} \leftarrow \widehat{\mathbf{A}^{-1}}^\top (\mathbf{x} - \gamma \mathbf{x}')$

　　$\widehat{\mathbf{A}^{-1}} \leftarrow \widehat{\mathbf{A}^{-1}} - \left(\widehat{\mathbf{A}^{-1}}\mathbf{x}\right)\mathbf{v}^\top / \left(1 + \mathbf{v}^\top \mathbf{x}\right)$

　　$\widehat{\mathbf{b}} \leftarrow \widehat{\mathbf{b}} + R\mathbf{x}$

　　$\mathbf{w} \leftarrow \widehat{\mathbf{A}^{-1}}\widehat{\mathbf{b}}$

　　$S \leftarrow S'$；$\mathbf{x} \leftarrow \mathbf{x}'$

　直到 S' 为终止状态

图 7：$O\left(d^2\right)$版本 LSTD，用于估计 $\hat{v} = w^{\mathrm{T}} x\left(\cdot\right) \approx v_\pi$

七、基于记忆的函数逼近

基于记忆的函数笔记方法不同于上面的参数化方法：仅仅在记忆中保存考到过的训练样本（至少保存样本的子集）而不更新任何参数。每当需要知道查询状态的价值估计值时，就从记忆中检索查找出一组样本，然后使用这些样本来计算查询状态的估计值。这种方法被称为拖延学习，因为处理训练样本被推迟到了系统被查询的时候。这是**非参数化方法**的主要例子。

八、基于核函数的函数逼近

核函数回归是一种基于记忆的方法，它对于记忆中存储的所有样本的对应目标计算其和函数加权平均值，并将结果返回给查询状态。

九、深入了解同轨策略学习：“兴趣”与“强调”

对于一个 MDP，我们可以有很多同轨策略分布，而不是仅仅有一个。所有的分布都遵从目标策略运行时在轨迹中遇到的状态分布，但是他们在不同的轨迹中是不同的，从某种意义上来说，就是源于轨迹的初始化不同。

兴趣值：一个非负随机标量变量 I_t，表示我们在 t 时刻有多大兴趣要精确估计一个状态（或“状态-动作”二元组）的价值。根本不在意，兴趣值为 0；非常在意，兴趣值为 1.

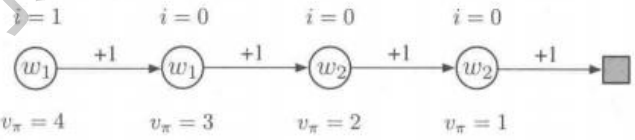
强调值：一个非负随机标量变量 M_t，这个标量会被乘上学习过程中的更新量，因此决定了在 t 时刻强调或者不强调“学习”。我们可以用更一般的 n 步学习法来替代

$$w_{t+n} \doteq w_{t+n-1} + \alpha\left(G_{t:t+n} - \hat{v}\left(S_t, w_{t+n-1}\right)\right)\nabla \hat{v}\left(S_t, w_{t+n-1}\right), 0 \le t < T \tag{22}$$

$$M_t = I_t + \gamma^n M_{t-n}, 0 \le t < T \tag{23}$$

例 9.4　兴趣与强调

为了观察使用兴趣值和强调值的潜在优势，考虑下图的四状态的马尔可夫收益过程：



所有幕都从最左边的状态开始，然后每次向右传递一个状态，并且收获 +1 的收益，直到到达终止状态。因此，第一个状态的真实价值为 4，第二个状态的真实价值为 3，等等，如同每个状态下方所示。这些是状态的真实价值，而估计值只能近似这些值，因为它们是受到参数化的约束的。参数向量有两个元素，即 $\mathbf{w} = (w_1, w_2)^\top$，并且参数也被写在每个状态里面。前两个状态的估计价值都由 w_1 单独给出，因此即便它们的真实价值不同，这两个值也必然相等。类似地，第三个和第四个状态的估计价值也由 w_2 单独决定，无论真实价值如何，这两个估计值也必然相等。假设我们只对最左边的状态的准确估计值感兴趣，我们将它的兴趣值分配为 1，而将其他所有的状态的兴趣值都分配为 0，正如状态上方所表示的。

首先，考虑将梯度蒙特卡洛算法应用于这个问题。在本章前面提出的这种算法没有考虑兴趣值与强调值 (式 9.7 和 200 页框中的算法)，该算法将收敛 (通过步长参数的减小) 到参数向量 $\mathbf{w}_\infty = (3.5, 1.5)$，这将给第一个状态，也就是我们唯一感兴趣的状态 3.5 的估计价值 (在第一个状态和第二个状态的真实价值的中间)。本节提出的使用兴趣值和强调值的方法，在另一方面，将正确学习第一个状态的价值。w_1 将收敛至 4，而 w_2 永不被更新，因为除了最左边的状态以外，其他所有状态的强调值都为 0。

现在考虑使用两步半梯度 TD 法。在本章之前讨论的未使用兴趣值与强调值的该方法 (式 9.15、式 9.16 和第 206 页框中的算法)，将再次收敛到 $\mathbf{w}_\infty = (3.5, 1.5)$，而使用兴趣值和强调值的该方法将收敛于 $\mathbf{w}_\infty = (4, 2)$。后者产生了第一个状态和第三个状态的实际真实价值而不会对第二个或者第四个状态做出任何更新。

图 8：兴趣与强调

强化学习极简笔记（十）：基于函数逼近的同轨策略控制

〇、概述

关注点：动作价值函数 $\hat{q}(s,a,w) \approx q_*(s,a)$ 进行参数化逼近的控制问题，这里 $w \in \mathbb{R}^d$ 是有限维权值向量。令人惊讶的是，一旦我们有了真正的函数逼近，我们必须放弃折扣并将控制问题的定义转换为一个新的“**平均收益**”的形式，这个形式有一种新的“差分”价值函数。

一、分幕式半梯度控制

动作价值函数预测的梯度下降更新的一般形式是

$$w_{t+1} \doteq w_t + \alpha\left(U_t - \hat{q}(S_t,A_t,w_t)\right)\nabla \hat{q}(S_t,A_t,w_t) \tag{1}$$

作为一个一般形式的例子，单步 Sarsa 方法的更新可以表示为（分幕式半梯度单步 Sarsa）

$$w_{t+1} \doteq w_t + \alpha\left(R_{t+1} + \gamma \hat{q}(S_{t+1},A_{t+1},w_t) - \hat{q}(S_t,A_t,w_t)\right)\nabla \hat{q}(S_t,A_t,w_t) \tag{2}$$

对于一个固定的策略，这个方法的收敛情况和 TD(0)一样，具有相同的误差边界。

为了得到控制方法，我们需要将这种动作价值函数预测与策略改进及动作选择的技术结合起来（适用于连续动作或从大型离散集合中选取动作的技术尚未有完美的解决方案）。另一方面，如果动作集合是离散的并且不是太大，那么我们可以使用前几章中已经提及的技术（根据动作价值函数贪心选择动作，之后转化为柔性近似策略）。完整算法如下

分幕式半梯度 Sarsa，用于估计 $\hat{q} \approx q_*$

输入：一个参数化的可微动作价值函数 $\hat{q}: \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$

算法参数：步长 $\alpha > 0$，很小的 $\varepsilon, \varepsilon > 0$

任意初始化价值函数的权值 $\mathbf{w} \in \mathbb{R}^d$（比如 $\mathbf{w} = \mathbf{0}$）

对每一幕循环：

  $S, A \leftarrow$ 幕的初始状态和动作（如 $\varepsilon-$ 贪心策略）

  对该幕的每一步循环：

    采取动作 A，观察 R, S'

    如果 S' 为终止状态：

      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\left[R - \hat{q}(S,A,\mathbf{w})\right]\nabla \hat{q}(S,A,\mathbf{w})$

      到下一幕

    通过 $\hat{q}(S',\cdot,\mathbf{w})$ 选取 A'（如 ε-贪心策略）

    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\left[R + \gamma \hat{q}(S',A',\mathbf{w}) - \hat{q}(S,A,\mathbf{w})\right]\nabla \hat{q}(S,A,\mathbf{w})$

    $S \leftarrow S'$

    $A \leftarrow A'$

图 1：分幕式半梯度 Sarsa

二、半梯度 n 步 Sarsa

通过使用 n 步回报作为半梯度公式中的更新目标来获得分幕式半梯度 Sarsa 的 n 步版本

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^{n} \hat{q}\left(S_{t+n}, A_{t+n}, w_{t+n-1}\right), t+n < T \tag{2}$$

完整伪代码如下

分幕式半梯度n步 Sarsa，用于估计 $\hat{q} \approx q_*$ 或 q_π

输入：一个参数化的可微动作价值函数 $\hat{q}: \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
输入：一个策略 π（如果要估计 q_π）
算法参数：步长 $\alpha > 0$，很小的 $\varepsilon, \varepsilon > 0$，正整数 n
任意初始化价值函数权值 $\mathbf{w} \in \mathbb{R}^d$（如 $\mathbf{w} = \mathbf{0}$）
所有的存取操作（对 S_t、A_t 和 R_t）的索引可以对 $n+1$ 取模
对每一幕循环：
  初始化并存储 $S_0, S_0 \neq$ 终止状态
  选取并存储动作 $A_0 \sim \pi(\cdot|S_0)$ 或根据 $\hat{q}(S_0, \cdot, \mathbf{w})$ 进行 ε-贪心选择
  $T \leftarrow \infty$
  循环 $t = 0, 1, 2, \ldots$：
  | 如果 $t < T$，那么：
  | 采取动作 A_t
  | 观察并存储下一个收益 R_{t+1} 和下一个状态 S_{t+1}
  | 如果 S_{t+1} 为终止状态，那么
  | $T \leftarrow t+1$
  | 否则：
  | 选取并存储 $A_{t+1} \sim \pi(\cdot|S_{t+1})$ 或根据 $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$ 进行 ε-贪心选择
  | $\tau \leftarrow t - n + 1$（$\tau$ 是估计值被更新的时刻）
  | 如果 $\tau \geqslant 0$：
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
  | 如果 $\tau + n < T$，那么 $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ $\quad (G_{\tau:\tau+n})$
  | $\mathbf{w} \leftarrow \mathbf{w} + \alpha [G - \hat{q}(S_\tau, A_\tau, \mathbf{w})] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
  直到 $\tau = T - 1$

图 2：分幕式半梯度 n 步 Sarsa

使用中等程度的自举法往往能够得到最好的性能，对应的便是大于 1 的 n。

三、平均收益：持续性任务中的新的问题设定

现在介绍第三种经典的马尔可夫决策问题（MDP）的目标设定：“平均收益”设定。前两种为“分幕式”设定和“折扣”设定。正如折扣设定，平均收益也适用于持续性问题，即智能体与环境之间的交互一直持续而没有对应的终止或开始状态。平均收益和折扣收益不同，智能体对于延迟收益的重视程度与即时收益相同。

折扣设定对函数逼近来说是有问题的，因此需要用平均收益设定来替换。

在平均收益设定中，一个策略 π 的质量被定义为在遵循该策略时的收益率的平均值，简称平均收益，表示为 $r(\pi)$

$$
\begin{aligned}
r(\pi) &\doteq \lim_{h\to\infty}\frac{1}{h}\sum_{t=1}^{h}\mathrm{E}\left(R_t \mid S_0, A_{0:t-1}\sim\pi\right) \\
&= \lim_{t\to\infty}\mathrm{E}\left(R_t \mid S_0, A_{0:t-1}\sim\pi\right) \\
&= \sum_s \mu_\pi(s)\sum_a \pi(a\mid s)\sum_{s',r} p(s',r\mid s,a)r
\end{aligned}
\tag{3}
$$

这里的期望根据初始状态 S_0 和遵循 π 生成的动作序列 $A_0, A_1, \ldots, A_{t-1}$ 来决定，μ_π 是一个稳态分布，假设对于每一个 π 都存在并是独立于 S_0 的，则

$$
\mu_\pi(s) \doteq \lim_{t\to\infty}\Pr\left(S_t = s \mid A_{0:t-1}\sim\pi\right) \tag{4}
$$

这种关于 MDP 的假设称为遍历性。这意味着 MDP 开始的位置或智能体的早期决定只具有临时的效果；从长远看，一个状态的期望只与策略本身以及 MDP 的转移概率相关。遍历性足以保证上述公式中极限的存在。

特别地，我们认为所有达到 $r(\pi)$ 最大值的策略都是最优的。注意，“稳态分布”是一个特殊的分布，即如果你按照 π 来选择动作，依然会获得同样的分布

$$
\sum_s \mu_\pi(s)\sum_a \pi(a\mid s)\,p(s'\mid s,a) = \mu_\pi(s') \tag{5}
$$

在平均收益设定中，回报是根据即使收益和平均收益定义的

$$
G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + \ldots \tag{6}
$$

这被成为差分回报，相应的价值函数被成为差分价值函数。它们的定义方式和之前相同，与状态价值函数/动作价值函数形式兼容。差分价值函数也有贝尔曼方程（和之前的差别在于：去掉所有的折扣 γ，并用即时收益和真实平均收益之间的差来代替原来的即时收益

$$
v_\pi(s) = \sum_a \pi(a\mid s)\sum_{r,s'} p(s',r\mid s,a)\left(r - r(\pi) + v_\pi(s')\right) \tag{7}
$$

$$
q_\pi(s,a) = \sum_{r,s'} p(s',r\mid s,a)\left(r - r(\pi) + \sum_{a'}\pi(a'\mid s')q_\pi(s',a')\right) \tag{8}
$$

$$
v_*(s) = \max_a \sum_{s',r} p(s',r\mid s,a)\left(r - \max_\pi r(\pi) + v_*(s')\right) \tag{9}
$$

$$
q_*(s,a) = \sum_{s',r} p(s',r\mid s,a)\left(r - \max_\pi r(\pi) + \max_{a'} q_*(s',a')\right) \tag{10}
$$

对于两类 TD 误差也有对应的差分形式

$$
\delta_t \doteq R_{t+1} - \overline{R}_{t+1} + \hat{v}(S_{t+1}, w_t) - \hat{v}(S_t, w_t) \tag{11}
$$

$$
\delta_t \doteq R_{t+1} - \overline{R}_{t+1} + \hat{q}(S_{t+1}, A_{t+1}, w_t) - \hat{q}(S_t, A_t, w_t) \tag{12}
$$

这里 $\overline{R}_{t+1}$ 是在时刻 t 对平均收益 $r(\pi)$ 的估计。有了这些改进的定义，大多数算法和理论的结果都无需改变，就适用于平均收益设定。例如，半梯度 Sarsa 的平均收益版本的定义式只在 TD 误差定义有区别

$$
w_{t+1} \doteq w_t + \alpha\delta_t \nabla \hat{q}(S_t, A_t, w_t) \tag{13}
$$

完整算法伪代码如下

差分半梯度 Sarsa 算法，用于估计 $\hat{q} \approx q_*$

输入：一个参数化的可微动作价值函数 $\hat{q}: \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$

算法参数：步长 $\alpha, \beta > 0$

任意初始化价值函数权值 $\mathbf{w} \in \mathbb{R}^d$ (如 $\mathbf{w} = \mathbf{0}$)

任意初始化平均收益的估计值 $\bar{R} \in \mathbb{R}$ (如 $\bar{R} = 0$)

初始化状态 S 和动作 A

对每一个步循环：

采取动作 A，观察 R, S'

通过 $\hat{q}(S', \cdot, \mathbf{w})$ 选取 A' (如 ε-贪心策略)

$\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$

$\bar{R} \leftarrow \bar{R} + \beta\delta$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$

$S \leftarrow S'$

$A \leftarrow A'$

图 3：差分半梯度 n 步 Sarsa

四、弃用折扣

持续性的带折扣问题的公式化表达在表格型的情况下非常有用，因为每个状态的回报可以被分别地识别和取平均。但是在函数逼近的情况下，是否将折扣用在问题的公式化表达中则需要打一个问号。

由于在较长（近似无限长）的时间里的带折扣的累计回报和等价于平均回报，即对于策略 π，折扣后的均值总是 $r(\pi)/(1-\gamma)$，也就是说，它本质上就是平均收益 $r(\pi)$。折扣率 γ 在问题中没有任何作用，实质上可以是零，而这个排序仍然保持不变。

下面给出详细证明，说明在持续性问题（伴随无限长度的序列）中，折扣是无用的（随便取值都可以）

持续性问题中折扣的无用性

我们可以设定策略排序的准则为折后价值的概率加权和，概率分布是给定策略下的状态分布。这时，我们也许能够把折扣系数的影响去掉。

$$J(\pi) = \sum_s \mu_\pi(s) v_\pi^\gamma(s) \qquad \text{(这里 } v_\pi^\gamma \text{ 是折后价值函数)}$$

$$= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) [r + \gamma v_\pi^\gamma(s')] \qquad \text{(贝尔曼公式)}$$

$$
\begin{aligned}
&= r(\pi) + \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \gamma v_\pi^\gamma(s') && \text{由式 (10.7) 得到} \\
&= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s'|s, a) && \text{由式 (3.4) 得到} \\
&= r(\pi) + \gamma \sum_{s'} v_\pi^\gamma(s') \mu_\pi(s') && \text{由式 (10.8) 得到} \\
&= r(\pi) + \gamma J(\pi) \\
&= r(\pi) + \gamma r(\pi) + \gamma^2 J(\pi) \\
&= r(\pi) + \gamma r(\pi) + \gamma^2 r(\pi) + \gamma^3 r(\pi) + \cdots \\
&= \frac{1}{1-\gamma} r(\pi).
\end{aligned}
$$

根据折扣准则得到的策略的排序与无折扣的 (平均收益) 准则下的排序完全相同。注意，折扣率 γ 完全不影响顺序!

图 4：持续性问题中折扣的无用性

使用函数逼近的折扣控制设定困难的根本原因在于我们**失去了策略改进定理**。我们在单个状态上改进折后价值函数不再保证我们会改进整个策略。而这个保证是强化学习控制方法的核心，使用函数逼近让我们失去了这个核心！

事实上，策略改进定理的缺失也是分幕式设定以及平均收益设定的理论缺陷：**一旦引入函数逼近，我们无法保证在任何设定下都一定会有策略的改进**。而在基于参数化策略的 RL 中，有**策略梯度定理**在理论上进行保证。

五、差分半梯度 n 步 Sarsa

为了推广到 n 步自举法，我们需要一个 n 步版本的 TD 误差。首先将 n 步回报推广到差分形式，使用函数逼近有

$$G_{t:t+n} \doteq R_{t+1} - \overline{R}_{t+1} + R_{t+2} - \overline{R}_{t+2} + \ldots + R_{t+n} - \overline{R}_{t+n} + \hat{q}\left(S_{t+n}, A_{t+n}, w_{t+n-1}\right) \tag{14}$$

其中，$\overline{R}$ 是对 $r(\pi)$ 的一个估计，$n \geq 1, t+n < T$。如果 $t+n \geq T$，和往常一样定义 $G_{t:t+n} \doteq G_t$。n 步 TD 误差变为：

$$\delta_t \doteq G_{t:t+n} - \hat{q}\left(S_t, A_t, w\right) \tag{15}$$

之后，我们可以使用常规的半梯度 Sarsa 更新，完整算法如下

差分半梯度n步 Sarsa 算法，用于估计 $\hat{q} \approx q_\pi$ 或 q_*

输入：一个参数化的可微动作价值函数 $\hat{q}\ \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$，一个策略 π

任意初始化价值函数权值 $\mathbf{w} \in \mathbb{R}^d$ (如 $\mathbf{w} = \mathbf{0}$)

任意初始化平均收益估计 $\bar{R} \in \mathbb{R}$ (如 $\bar{R} = 0$)

算法参数：步长 $\alpha, \beta > 0$，正整数 n

所有的存取操作 (对 S_t、A_t 和 R_t) 的索引可以对 $n+1$ 取模

初始化并存储 S_0 和 A_0

对每一步循环，$t = 0, 1, 2, \ldots$：

  采取动作 A_t

  观察并存储下一个收益 R_{t+1} 和下一个状态 S_{t+1}

  选取并存储 $A_{t+1} \sim \pi(\cdot|S_{t+1})$ 或根据 $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$ 进行 ε-贪心选择

  $\tau \leftarrow t - n + 1$ (τ 是估计值被更新的时刻)

  如果 $\tau \geqslant 0$：

    $\delta \leftarrow \sum_{i=\tau+1}^{\tau+n}(R_i - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(S_\tau, A_\tau, \mathbf{w})$

    $\bar{R} \leftarrow \bar{R} + \beta\delta$

    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S_\tau, A_\tau, \mathbf{w})$

图 5：差分半梯度 n 步 Sarsa 算法

强化学习极简笔记（十三）：策略梯度方法

〇、概述

目标：直接学习参数化策略的方法。记为

$$\pi(a|s,\theta)=\Pr(A_t=a|S_t=s,\theta_t=\theta) \tag{1}$$

策略参数的学习方法基于某种**性能度量$J(\theta)$的梯度**，这些梯度是标量$J(\theta)$对策略参数的梯度。这些方法的目标是最大化性能指标，所以他们的更新近似于 J 的梯度上升

$$\theta_{t+1}=\theta_t+\alpha\nabla\hat{J}(\theta_t) \tag{2}$$

其中，$\nabla\hat{J}(\theta_t)$是一个随机估计，其维度与 θ 维度相同。它的期望是性能指标对参数 θ 的梯度的近似。这一类方法成为**策略梯度法**。同时学习策略和价值函数的方法一般被称为**行动器-评判器方法**，其中行动器指的是学习到的**策略**；判别器指学习到的**价值函数**。

一、策略近似及其优势

在策略梯度方法中，策略可以用任意的方式参数化，只要$\pi(a|s,\theta)$对参数可导。实际中，为了便于探索，一**般要求策略永远不会变成确定的**。回顾赌博机为题，引入**偏好函数**$h(s,a,\theta)$表示对动作的偏好，此时策略可以表示为**偏好函数的指数柔性最大化（softmax）分布**

$$\pi(a|s,\theta)\doteq\frac{e^{h(s,a,\theta)}}{\sum_b e^{h(s,b,\theta)}} \tag{3}$$

这些动作偏好可以被任意形式参数化表示，如神经网络或线性拟合。

以偏好柔性最大化分布选择动作的一个直接好处就是近似策略可以接近于一个确定的分布，尽管基于 e-贪心的动作选择中会有 e 的概率选择随机动作。我们也可以根据动作价值的柔性最大化分布选择动作，但是这不会使策略趋向于一个确定的策略。相反，动作价值的估计值会收敛于对应的真实值，而真实值之间的差异是有限的，因此各个动作会对应到一个特定的概率而不是 0 和 1.而动作偏好不同，因为它们不趋向于任何特定的值；相反，它们趋向于最优的随机策略。**如果最优策略是确定的，则最优动作的偏好值将可能趋向无限大于所有次优动作。**

利用动作偏好的柔性最大化分布的参数化策略还有另外一个优势，即**可以以任意的概率来选择动作**。在一些情况下，最好的近似策略可能是一个随即策略，基于动作价值函数的方法没有一种自然的途径来求解随机最优策略，但是基于策略近似的方法可以。

策略参数化相对于动作价值函数参数化的一个最简单的优势是可以用**更简单的函数近似**。

最后，策略参数化形式的选择有时是**在基于强化学习系统中引入理想中的策略形式的先验知识的一个好方法**。

二、策略梯度定理

策略参数化相对于 e-贪心动作选择法有理论上的优势：**对于连续的策略参数化，选择**

动作的概率作为被优化的参数会平滑地变化，但是 e-贪心方法则不同，只要估计的动作价值函数的变化导致了最大动作价值对应的动作发生了变化，则选择某个动作的概率可能会突变，即使估计的动作价值函数只发生了很小的变化。由于这个原因，基于策略梯度的方法能够比基于动作价值函数的方法有**更强的收敛保证。特别地，策略关于参数变化的连续性使得基于梯度的方法能够近似梯度公式（1）。**

在分幕式和持续性的两种不同情况下，他们的性能指标 $J(\theta)$ 定义不同，在一定程度上不得不分开处理。

先考虑分幕式的情况，将性能指标定义为初始状态的价值。假设每幕都从某个（非随机的）状态 s_0 开始，则将性能指标定义为

$$J(\theta) \doteq v_{\pi_\theta}(s_0) \tag{4}$$

其中，v_{π_θ} 是在策略 π_θ 下的真实价值函数，策略由参数 θ 决定。

策略梯度定理提供了一个性能指标相对于策略参数的解析表达式，在分幕式情况下策略梯度定理表达式如下

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta) \tag{5}$$

下面给出证明

策略梯度定理的证明 (分幕式情况)

只需要基本的微积分知识和重排公式中的各项，我们就能根据一些基本原理来证明策略梯度定理。为了使符号简单，在所有公式中我们隐去了 π 对 $\boldsymbol{\theta}$ 的依赖，同样，也隐去了梯度对 $\boldsymbol{\theta}$ 的依赖。首先，注意，状态价值函数的梯度可以写为动作价值函数的形式

$$\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \quad \text{对所有} s \in \mathcal{S} && \text{(练习 3.18)} \\
&= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \right] && \text{(微积分的乘法法则)} \\
&= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a)\left(r + v_\pi(s')\right) \right] \\
& && \text{(练习 3.19 和式 3.2)} \\
&= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \right] && \text{(式 3.4)} \\
&= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \right. && \text{(展开)} \\
&\quad \left. \sum_{a'} \left[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \right] \right] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a),
\end{aligned}$$

在重新展开后，其中 $\Pr(s \to x, k, \pi)$ 是在策略 π 下，状态 s 在 k 步内转移到状态 x 的概率。所以，我们可以马上得到

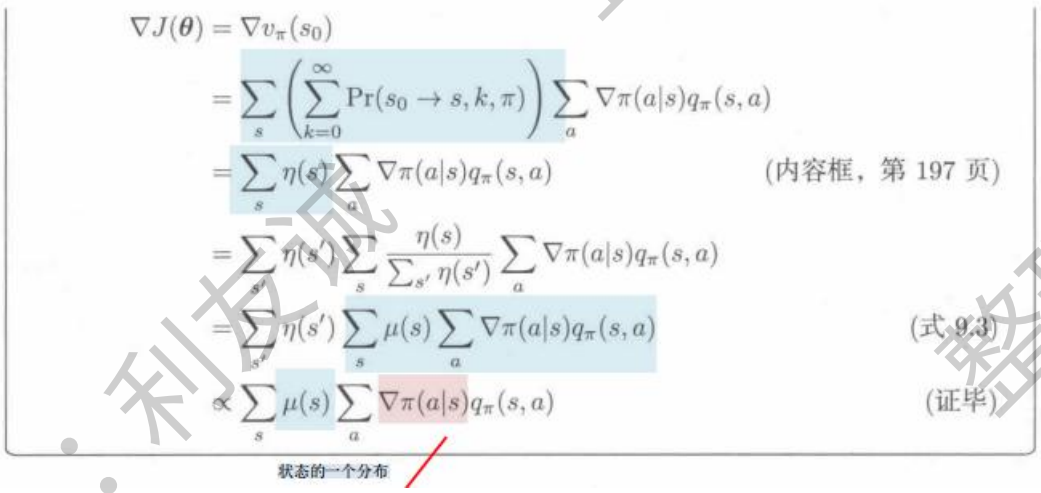


图 1：策略梯度定理的证明（分幕式情况）

三、REINFORCE：蒙特卡洛策略梯度

公式（1）中的随机梯度上升，其中需要一个获取样本的方法，这些采样的样本梯度的期望正比于性能指标对于策略参数的实际梯度。这些样本梯度只需要正比于实际的梯度，因为任何常熟的正比系数显然都可以被吸收到步长参数 α 中。策略梯度定理给出了一个正比于梯度的精确表达式，现在只需要一种采样方法，使得采样样本的梯度近似或者等于这个表达式

$$\begin{aligned}\nabla J(\theta) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta) \\ &= \mathrm{E}_\pi \left(\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\theta) \right)\end{aligned} \tag{6}$$

此时公式（1）可以实例化为

$$\theta_{t+1} = \theta_t + \alpha \sum_a \hat{q}(S_t,a,w) \nabla \pi(a|S_t,\theta) \tag{7}$$

这一算法称为全部动作算法，因为它的更新涉及了所有可能的动作。经典的 REINFORCE 算法，在时刻 t 的更新仅仅涉及了 A_t，即在时刻 t 被实际采用的动作。在公式（6）基础上进行 REINFORCE 的推导

$$\begin{aligned}\nabla J(\theta) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\theta) \\ &= \mathrm{E}_\pi \left(\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\theta) \right) \\ &= \mathrm{E}_\pi \left(\sum_a \pi(a|S_t,\theta) q_\pi(S_t,a) \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)} \right) \\ &= \mathrm{E}_\pi \left(q_\pi(S_t,A_t) \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)} \right) \\ &= \mathrm{E}_\pi \left(G_t \frac{\nabla \pi(a|S_t,\theta)}{\pi(a|S_t,\theta)} \right)\end{aligned} \tag{8}$$

此时公式（1）的随机梯度上升表达如下

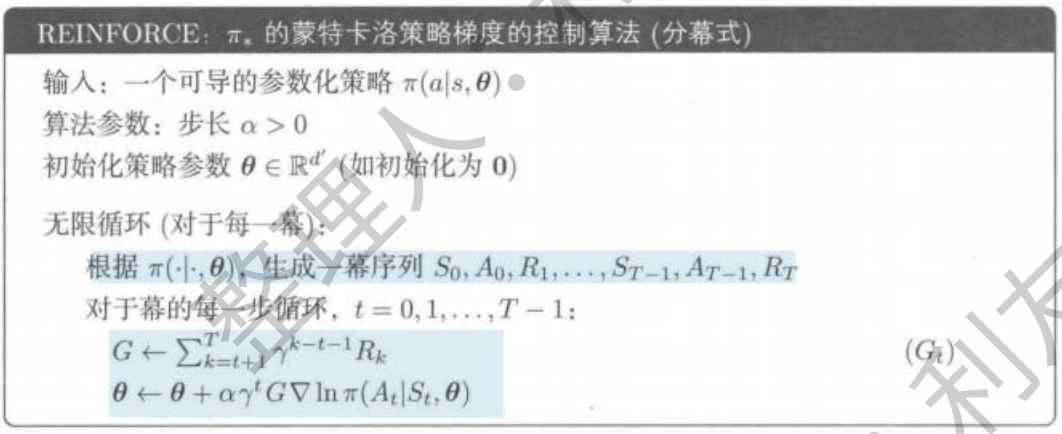
$$\theta_{t+1}=\theta_t+\alpha G_t\frac{\nabla\pi\left(A_t\mid S_t,\theta\right)}{\pi\left(A_t\mid S_t,\theta\right)} \tag{9}$$

REINFORCE 算法每一个增量更新正比于回报 G_t 和一个向量的乘积，这个向量是选择动作的概率的梯度除以这个概率本身。这个向量是参数空间中使得将来在状态 S_t 下重复选择动作 A_t 的概率增加最大的方向。这个更新使得参数向量沿着这个方向增加，更新大小正比于回报，反比于选择动作的概率。前者的意义在于它使得参数向着更有利于产生最大回报的动作的方向更新。后者有意义是因为如果不这样的话，频繁被选择的动作会占优（在这些方向更新更频繁），即使这些动作不是产生最大回报的动作，最后可能也会胜出，这就会影响性能指标的优化。

在实际使用中 $\frac{\nabla\pi\left(a\mid S_t,\theta\right)}{\pi\left(a\mid S_t,\theta\right)}$ 会表达为一个更加紧凑的形式

$$\frac{\nabla\pi\left(A_t\mid S_t,\theta_t\right)}{\pi\left(A_t\mid S_t,\theta_t\right)}=\nabla\ln\pi\left(A_t\mid S_t,\theta_t\right) \tag{10}$$

这个向量称为迹向量，这是算法中策略参数唯一出现的地方


REINFORCE：π_* 的蒙特卡洛策略梯度的控制算法 (分幕式)

输入：一个可导的参数化策略 $\pi(a|s,\boldsymbol{\theta})$

算法参数：步长 $\alpha>0$

初始化策略参数 $\boldsymbol{\theta}\in\mathbb{R}^{d'}$（如初始化为 $\mathbf{0}$）

无限循环（对于每一幕）：

根据 $\pi(\cdot|\cdot,\boldsymbol{\theta})$ 生成一幕序列 $S_0,A_0,R_1,\ldots,S_{T-1},A_{T-1},R_T$

对于幕的每一步循环，$t=0,1,\ldots,T-1$：

$G\leftarrow\sum_{k=t+1}^{T}\gamma^{k-t-1}R_k$ (G_t)

$\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}+\alpha\gamma^t G\nabla\ln\pi(A_t|S_t,\boldsymbol{\theta})$

图 2：REINFORCE：π_* 的蒙特卡洛策略梯度的控制算法（分幕式）

REINFOECE 有很好的理论收敛保证，这是其和其他随机梯度方法相比的优势。每幕的期望更新和性能指标的真实梯度方向相同。这可以保证期望意义下的性能指标的改善，只要 α 足够小，在标准的随机近似条件（减小 α ）下，它可以收敛到局部最优。然而，作为 MC 方法，REINFORCE 可能有较高的方差，因此导致学习较慢。

四、带有基线的 REINFORCE

将策略梯度定理推广，加入任意一个与动作价值函数对比的基线 $b(s)$

$$\nabla J\left(\theta\right)\propto\sum_s\mu\left(s\right)\sum_a\left(q_\pi\left(s,a\right)-b\left(s\right)\right)\nabla\pi\left(a\mid s,\theta\right) \tag{11}$$

基线函数 $b(s)$ 与动作 a 无关

$$\sum_a b(s)\nabla\pi(a\mid s,\theta)=b(s)\nabla\sum_a\pi(a\mid s,\theta)=b(s)\nabla 1=0 \tag{12}$$

包含基线的 REINFORCE 更新如下

$$\theta_{t+1}=\theta_t+\alpha\left(G_t-b(S_t)\right)\frac{\nabla\pi(A_t\mid S_t,\theta)}{\pi(A_t\mid S_t,\theta)} \tag{13}$$

在赌博机问题中，baseline 是一个数值（目前为止的平均收益），对于 MDP，这个 baseline 应该是根据状态的变化而变化。在一些状态下，所有动作的价值可能都比较大，因此我们需要一个较大的基线用以区分拥有更大值的动作和相对值不那么高的动作；其他状态下当所有动作的值都比较低的时候，baseline 应该较小。

状态价值函数 $\hat{v}(S_t,w)$ 是一个比较直观的 baseline

有两个参数需要变化，一个是 θ（策略梯度），一个是w（价值梯度）

带基线的 REINFORCE 算法（分幕式），用于估计 $\pi_\theta\approx\pi_*$

输入：一个可微的参数化策略 $\pi(a|s,\boldsymbol{\theta})$

输入：一个可微的参数化状态价值函数 $\hat{v}(s,\mathbf{w})$

算法参数：步长 $\alpha^{\boldsymbol{\theta}}>0$，$\alpha^{\mathbf{w}}>0$

初始化策略参数 $\boldsymbol{\theta}\in\mathbb{R}^{d'}$ 和状态价值函数的权重 $\mathbf{w}\in\mathbb{R}^d$（如初始化为 $\mathbf{0}$）

无限循环（对于每一幕）：

根据 $\pi(\cdot|\cdot,\boldsymbol{\theta})$，生成一幕序列 $S_0,A_0,R_1,\ldots,S_{T-1},A_{T-1},R_T$

对于幕中的每一步循环，$t=0,1,\ldots,T-1$：

$G\leftarrow\sum_{k=t+1}^{T}\gamma^{k-t-1}R_k$ $\quad(G_t)$

$\delta\leftarrow G-\hat{v}(S_t,\mathbf{w})$

$\mathbf{w}\leftarrow\mathbf{w}+\alpha^{\mathbf{w}}\delta\nabla\hat{v}(S_t,\mathbf{w})$

$\boldsymbol{\theta}\leftarrow\boldsymbol{\theta}+\alpha^{\boldsymbol{\theta}}\gamma^t\delta\nabla\ln\pi(A_t|S_t,\boldsymbol{\theta})$

图 3：带有基线的 REINFORCE 算法（分幕式）

注意区分 REINFORCE 算法和 AC 算法的区别：尽管带基线的 RL 方法既学习了一个策略函数也学习了一个状态价值函数，我们也不认为它是一种 AC 方法，因为它的状态价值函数仅被用于 baseline，而不是作为一个“判别器”。也就是说，它没有被用于自举操作，而只是作为正被更新的状态价值的 baseline。只有采用自举法的时候，才会出现依赖于函数逼近质量的偏差和渐进性收敛（降低了方差并加快了学习）。

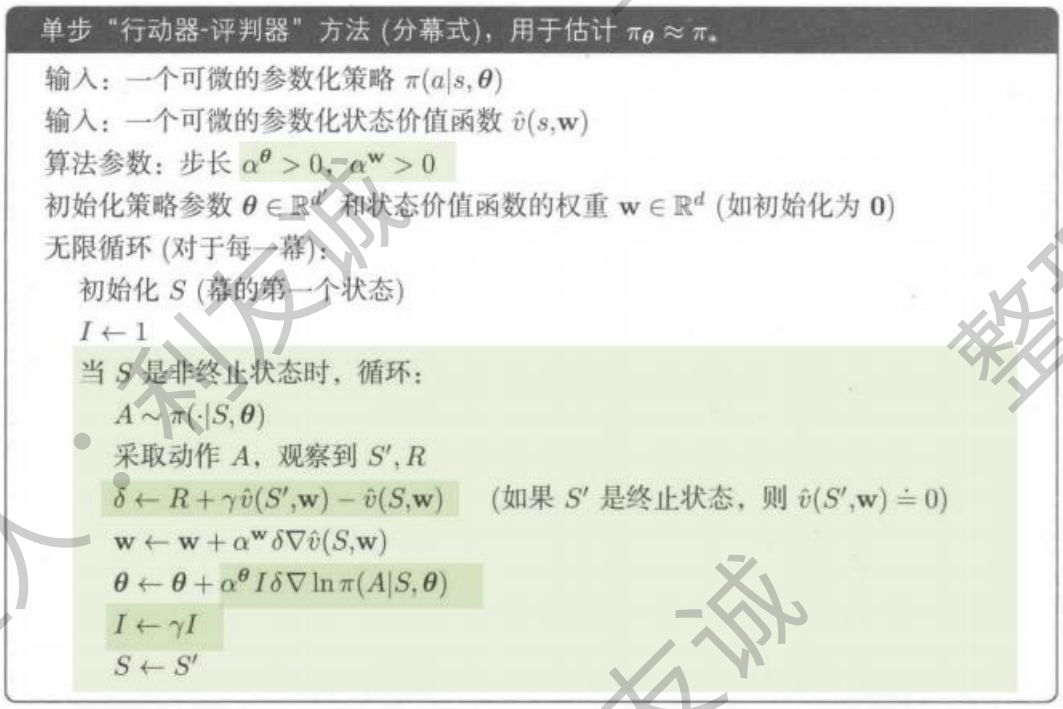
五、“行动器-判别器”方法

使用单步回报（并使用估计的状态价值函数作为基线）来代替 REINFORCE 算法的整个回报

$$\begin{aligned}\theta_{t+1}&\doteq\theta_t+\alpha\left(G_{t:t+1}-\hat{v}(S_t,w)\right)\frac{\nabla\pi(A_t\mid S_t,\theta)}{\pi(A_t\mid S_t,\theta)}\\&=\theta_t+\alpha\left(R_{t+1}+\gamma\hat{v}(S_{t+1},w)-\hat{v}(S_t,w)\right)\frac{\nabla\pi(A_t\mid S_t,\theta)}{\pi(A_t\mid S_t,\theta)}\\&=\theta_t+\alpha\delta_t\frac{\nabla\pi(A_t\mid S_t,\theta)}{\pi(A_t\mid S_t,\theta)}\end{aligned} \tag{14}$$

这是一个**完全在线、增量式的算法**，其状态、动作和收益都只会在它们第一次被收集到

时候使用，之后都不会再次使用



单步“行动器-评判器”方法 (分幕式)，用于估计 $\pi_{\boldsymbol{\theta}} \approx \pi_*$

输入：一个可微的参数化策略 $\pi(a|s,\boldsymbol{\theta})$

输入：一个可微的参数化状态价值函数 $\hat{v}(s,\mathbf{w})$

算法参数：步长 $\alpha^{\boldsymbol{\theta}} > 0$，$\alpha^{\mathbf{w}} > 0$

初始化策略参数 $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ 和状态价值函数的权重 $\mathbf{w} \in \mathbb{R}^d$ (如初始化为 $\mathbf{0}$)

无限循环 (对于每一幕)：

初始化 S (幕的第一个状态)

$I \leftarrow 1$

当 S 是非终止状态时，循环：

$A \sim \pi(\cdot|S,\boldsymbol{\theta})$

采取动作 A，观察到 S', R

$\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (如果 S' 是终止状态，则 $\hat{v}(S',\mathbf{w}) \doteq 0$)

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla\hat{v}(S,\mathbf{w})$

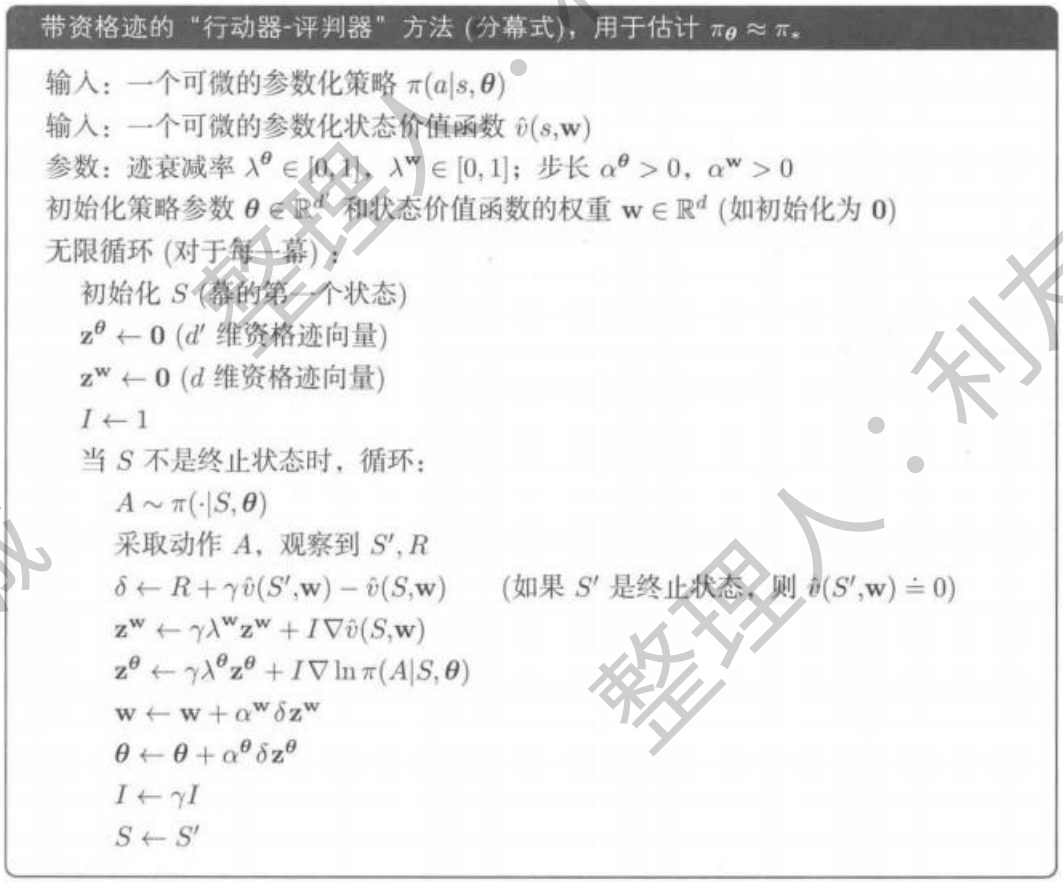
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I\delta\nabla \ln\pi(A|S,\boldsymbol{\theta})$

$I \leftarrow \gamma I$

$S \leftarrow S'$

图 4：单步“行动器-判别器”方法（分幕式）

可以很容易扩展到 n 步方法和 λ-回报方法



带资格迹的“行动器-评判器”方法 (分幕式)，用于估计 $\pi_{\boldsymbol{\theta}} \approx \pi_*$

输入：一个可微的参数化策略 $\pi(a|s,\boldsymbol{\theta})$

输入：一个可微的参数化状态价值函数 $\hat{v}(s,\mathbf{w})$

参数：迹衰减率 $\lambda^{\boldsymbol{\theta}} \in [0,1]$，$\lambda^{\mathbf{w}} \in [0,1]$；步长 $\alpha^{\boldsymbol{\theta}} > 0$，$\alpha^{\mathbf{w}} > 0$

初始化策略参数 $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ 和状态价值函数的权重 $\mathbf{w} \in \mathbb{R}^d$ (如初始化为 $\mathbf{0}$)

无限循环 (对于每一幕)：

初始化 S (幕的第一个状态)

$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d' 维资格迹向量)

$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d 维资格迹向量)

$I \leftarrow 1$

当 S 不是终止状态时，循环：

$A \sim \pi(\cdot|S,\boldsymbol{\theta})$

采取动作 A，观察到 S', R

$\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (如果 S' 是终止状态，则 $\hat{v}(S',\mathbf{w}) \doteq 0$)

$\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla\hat{v}(S,\mathbf{w})$

$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla \ln\pi(A|S,\boldsymbol{\theta})$

$\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$

$I \leftarrow \gamma I$

$S \leftarrow S'$

图 5：带资格迹的“行动器-判别器”方法（分幕式）

六、持续性问题的策略梯度

对于没有分幕式边界的持续性问题，我们需要根据每个时刻的平均收益来定义性能指标

$$J(\theta) \doteq r(\pi) \doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathrm{E}\left(R_t \mid S_0, A_{0:t-1} \sim \pi\right)$$
$$= \lim_{h\to\infty} \mathrm{E}\left(R_t \mid S_0, A_{0:t-1} \sim \pi\right) \tag{15}$$
$$= \sum_s \mu(s) \sum_a \pi(a \mid s) \sum_{s',r} p(s', r \mid s, a) r$$

$$\mu(s) \doteq \lim_{t\to\infty} \Pr\left(S_t = s \mid A_{0:t} \sim \pi\right) \tag{16}$$

算法伪代码如下

带资格迹的“行动器-评判器”方法 (持续的)，用于估计 $\pi_{\boldsymbol{\theta}} \approx \pi_*$

输入：一个可微的参数化策略 $\pi(a|s,\boldsymbol{\theta})$
输入：一个可微的参数化状态价值函数 $\hat{v}(s,\mathbf{w})$
算法参数：$\lambda^{\mathbf{w}} \in [0,1]$，$\lambda^{\boldsymbol{\theta}} \in [0,1]$，$\alpha^{\mathbf{w}} > 0$，$\alpha^{\boldsymbol{\theta}} > 0$，$\alpha^{\bar{R}} > 0$
初始化 $\bar{R} \in \mathbb{R}$ (如初始化为 0)
初始化状态价值函数权重 $\mathbf{w} \in \mathbb{R}^d$ 和策略参数 $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (如初始化为 $\mathbf{0}$)
初始化 $S \in \mathcal{S}$ (如初始化为 s_0)

$\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d 维资格迹向量)
$\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d' 维资格迹向量)
无限循环 (对于每一个时刻)：
  $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
  采取动作 A，观察到 S', R
  $\delta \leftarrow R - \bar{R} + \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \alpha^{\bar{R}}\delta$
  $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{v}(S,\mathbf{w})$
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln\pi(A|S,\boldsymbol{\theta})$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
  $S \leftarrow S'$

图 6：带资格迹的“行动器-判别器”方法（持续的）

自然地，在持续性问题中，我们使用差分回报定义价值函数

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + \ldots \tag{17}$$

下面给出持续性问题中的策略梯度定理的证明

策略梯度定理证明 (持续性问题)

持续性问题的策略梯度定理的证明和分幕式问题的证明非常类似。这里我们再一次明确 π 是关于 $\boldsymbol{\theta}$ 的一个函数，所求的是关于 $\boldsymbol{\theta}$ 的梯度。回顾一下，在持续性问题中 $J(\boldsymbol{\theta}) = r(\pi)$ (式 13.15)，并且 v_π 和 q_π 是用差分回报表示的价值函数。对于任意 $s \in S$，状态价值函数的梯度可以表示为

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla\left[\sum_a \pi(a|s) q_\pi(s,a)\right], \text{ 对所有} s \in \mathcal{S} && \text{(练习 3.18)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s)\nabla q_\pi(s,a)\Big] && \text{(微积分的乘法法则)} \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s)\nabla \sum_{s',r} p(s',r|s,a)\big(r - r(\boldsymbol{\theta}) + v_\pi(s')\big)\Big] \\
&= \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s)\big[-\nabla r(\boldsymbol{\theta}) + \sum_{s'} p(s'|s,a)\nabla v_\pi(s')\big]\Big].
\end{aligned}
$$

重新组合公式后，我们可以获得

$$
\nabla r(\boldsymbol{\theta}) = \sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s)\sum_{s'} p(s'|s,a)\nabla v_\pi(s')\Big] - \nabla v_\pi(s).
$$

注意，等式左边可以写为 $\nabla J(\boldsymbol{\theta})$，它与 s 无关。因此等式右边也与 s 无关。我们可以放心地用 $\mu(s)$ $(s \in \mathcal{S})$ 进行加权求和，而不会改变原来的结果 (因为 $\sum_s \mu(s) = 1$)。所以，

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \sum_s \mu(s)\left(\sum_a \Big[\nabla\pi(a|s) q_\pi(s,a) + \pi(a|s)\sum_{s'} p(s'|s,a)\nabla v_\pi(s')\Big] - \nabla v_\pi(s)\right) \\
&= \sum_s \mu(s)\sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&\quad + \sum_s \mu(s)\sum_a \pi(a|s)\sum_{s'} p(s'|s,a)\nabla v_\pi(s') - \sum_s \mu(s)\nabla v_\pi(s) \\
&= \sum_s \mu(s)\sum_a \nabla\pi(a|s) q_\pi(s,a) \\
&\quad + \sum_{s'} \underbrace{\sum_s \mu(s)\sum_a \pi(a|s) p(s'|s,a)}_{\mu(s') \text{ (13.16)}} \nabla v_\pi(s') - \sum_s \mu(s)\nabla v_\pi(s) \\
&= \sum_s \mu(s)\sum_a \nabla\pi(a|s) q_\pi(s,a) + \sum_{s'} \mu(s')\nabla v_\pi(s') - \sum_s \mu(s)\nabla v_\pi(s) \\
&= \sum_s \mu(s)\sum_a \nabla\pi(a|s) q_\pi(s,a). \qquad \text{证毕}
\end{aligned}
$$

图 7：策略梯度定理证明（持续性问题）

七、针对连续动作的策略参数化方法

基于参数化策略函数的方法还提供了解决动作空间大甚至动作空间连续（动作无限多）的实际途径。我们不直接计算每一个动作的概率，而是**学习概率分布的统计量**。

$$\pi(a \mid s,\theta) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} \exp\left(-\frac{\left(a-\mu(s,\theta)\right)^2}{2\sigma(s,\theta)^2}\right) \tag{18}$$

我们将策略的参数向量划分为两个部分，一部分用来**近似均值**，一部分用来**近似标准差**（均值可以用一个线性啊还能输来逼近，标准差必须为正数，因此使用线性函数的指数形式比较好）

$$\theta = \left[\theta_\mu, \theta_\sigma\right]^{\mathrm{T}} \tag{19}$$

$$\mu(s,\theta) \doteq \theta_\mu{}^{\mathrm{T}} x_\mu(s) \tag{20}$$

$$\sigma(s,\theta) \doteq \exp\left(\theta_\sigma{}^{\mathrm{T}} x_\sigma(s)\right) \tag{21}$$

深度强化学习极简笔记（四）：深度 Q 网络

〇、概述

Q-learning 算法在使用**表格（Tabular）**的情况下或使用线性函数逼近 Q 函数时，Q-learning 已被证明可以收敛于最优解。在使用非线性函数逼近器（如神经网络）来表示 Q 函数时，Q-learning 并**不稳定，甚至发散**。深度 Q 网络（Deep Q-Networks，DQN）算法解决了以上问题。

无模型（Model-Free）方法为解决基于 MDP 的决策问题提供了一种通用的方法。其中给“模型”是指显式地对 MDP 相关的转移概率分布和回报函数建模，而 TD 学习是一类无模型的方法。TD 学习遵循的思想是：**即使对子问题的估计并非一直是最优的，我们可以通过自举（Bootstrapping）来估计子问题的值**。

$$v_\pi(s)=\mathrm{E}_\pi\left(R_t+\gamma v_\pi\left(S_{t+1}\right)|S_t=s\right) \tag{1}$$

$$V\left(S_t\right)\leftarrow V\left(S_t\right)+\alpha\left(R_t+\gamma V\left(S_{t+1}\right)-V\left(S_t\right)\right) \tag{2}$$

一、Sarsa 和 Q-learning

Q-learning（one-step）遵循如下更新规则

$$Q\left(S_t,A_t\right)\leftarrow Q\left(S_t,A_t\right)+\alpha\left(R_t+\gamma\max_{A_{t+1}}Q\left(S_{t+1},A_{t+1}\right)-Q\left(S_t,A_t\right)\right) \tag{3}$$

可以使用**多步回报**提高逼近精度。

二、为什么使用 DL 做价值函数逼近

使用函数拟合器逼近价值函数，将原问题转为一个**优化问题**

$$\theta_t\leftarrow\arg\max_\theta Loss\left(Q\left(S_t,A_t;\theta\right),R_{t+1}+\gamma Q\left(S_{t+1},A_{t+1};\theta\right)\right) \tag{4}$$

Loss()代表**损失函数**，如**均方误差（MSE）**。有**拟合 Q 迭代（Fitted Q Iteration）**和**在线 Q 迭代（Online Q Iteration）**两种算法

算法 4.15 拟合 Q 迭代

for 迭代数 $i=1,T$ **do**
  收集 D 份采样 $\{(S_i,A_i,R_i,S_i')\}_{i=1}^D$
  for t $=1,K$ **do**
    设置 $Y_i\leftarrow R_i+\gamma\max_a Q(S_i',a;\theta)$
    设置 $\theta\leftarrow\arg\min_{\theta'}\frac{1}{2}\sum_{i=1}^D(Q(S_i,A_i;\theta')-Y_i)^2$
  end for
end for

算法 4.16 在线 Q 迭代

for 迭代数 $=1,T$ **do**
  选择动作 a 与环境交互，并得到观察数据 (s,a,r,s')
  设置 $y\leftarrow r+\gamma\max_{a'}Q(s',a';\theta)$
  设置 $\theta\leftarrow\theta-\alpha(Q(s,a;\theta)-y)\frac{\mathrm{d}Q(s,a;\theta)}{\mathrm{d}\theta}$
end for

图 1：拟合 Q 迭代和在线 Q 迭代算法

值得注意的是，拟合 Q 迭代和在线 Q 迭代都是**离轨**的，因此，它们可以**多次重用过去的经验**。我们将在下一节对此进行讨论。

三、DQN

DQN 通过使用两个关键技术结合 Q-learning 和深度学习来解决**不稳定性问题：**

回放缓存（Replay Buffer）：这是一种被称为**经验重演的生物学启发机制**。每个时间步 t 中，DQN 先将智能体获得的经验(St,At,Rt,St+1)存入回放缓存中，然后从该缓存中**均匀采样**小批量样本用于 Q-Learning 更新。回放缓存相较于拟合 Q 迭代有几个优势。首先，它可以**重用每个时间步的经验**来学习 Q 函数，这样可以**提高数据使用效率**。其次，如果像拟合 Q 迭代那样没有回放缓存，那么一个批次中的样本将会是连续采集的，即**样本高度相关**。这样会增加更新的方差。最后，**经验回放防止用于训练的样本只来自上一个策略，这样能平滑学习过程并减少参数的震荡或发散**。在实践中，为了节省内存，我们往往只将最后 N 个经验存入回放缓存（**FIFO 缓存**）。

目标网络：它作为一个独立的网络，用来代替所需的 Q 网络来生成 Q-learning 的目标，进一步提高神经网络的稳定性。此外，目标网络每 C 步将通过**直接复制（硬更新）或者指数衰减平均（软更新）**的方式与主 Q 网络同步。目标网络通过使用**旧参数**生成 Q-learning 目标，使目标值的产生不受最新参数的影响，从而大大减少发散和震荡的情况。例如，在动作(St,At)上的更新使得 Q 值增加，此时 St 和 St+1 的相似性可能会导致所有动作 a 的 Q(St+1,a)值增加，从而使得由 Q 网络产生的训练目标值被过估计。但是如果使用目标网络产生训练目标，就能避免过估计的问题。

由于将任意长度的历史数据作为神经网络的输入较为复杂，**DQN 转而处理由函数φ生成的固定长度表示的历史数据**。准确来说，**φ集合了当前帧和前三帧的数据，这对于跟踪时间相关信息（如对象的移动）非常有用**。完整的算法展示在算法 4.17 中。其中原始帧被调整为 84×84 的灰度图像。函数φ堆叠了最近 4 帧的数据作为神经网络的输入。此外，神经网络的结构由三个卷积层和两个完全连接的层组成，每个有效动作只有一个输出。我们将在 4.8 节讨论更多的训练细节。

算法 4.17 DQN

超参数: 回放缓存容量 N，奖励折扣因子 γ，用于目标状态-动作值函数更新的延迟步长 C，ϵ-greedy 中的 ϵ。
输入: 空回放缓存 $\mathcal{D}$，初始化状态-动作值函数 Q 的参数 θ。
使用参数 $\hat{\theta} \leftarrow \theta$ 初始化目标状态-动作值函数 $\hat{Q}$。
for 片段 $= 0, 1, 2, \cdots$ **do**
  初始化环境并获取观测数据 O_0。
  初始化序列 $S_0 = \{O_0\}$ 并对序列进行预处理 $\phi_0 = \phi(S_0)$。
  for t $= 0, 1, 2, \cdots$ **do**
    通过概率 ϵ 选择一个随机动作 A_t，否则选择动作 $A_t = \arg\max_a Q(\phi(S_t), a; \theta)$。
    执行动作 A_t 并获得观测数据 O_{t+1} 和奖励数据 R_t。
    如果本局结束，则设置 $D_t = 1$，否则 $D_t = 0$。
    设置 $S_{t+1} = \{S_t, A_t, O_{t+1}\}$ 并进行预处理 $\phi_{t+1} = \phi(S_{t+1})$。
    存储状态转移数据 $(\phi_t, A_t, R_t, D_t, \phi_{t+1})$ 到 $\mathcal{D}$ 中。
    从 $\mathcal{D}$ 中随机采样小批量状态转移数据 $(\phi_i, A_i, R_i, D_i, \phi'_i)$。
    若 $D_i = 0$，则设置 $Y_i = R_i + \gamma \max_{a'} \hat{Q}(\phi'_i, a'; \hat{\theta})$，否则，设置 $Y_i = R_i$。
    在 $(Y_i - Q(\phi_i, A_i; \theta))^2$ 上对 θ 执行梯度下降步骤。
    每 C 步对目标网络 $\hat{Q}$ 进行同步。
    如果片段结束，则跳出循环。
  end for
end for

图 2：DQN

四、Double DQN

Double DQN 是对 DQN 在减少过拟合方面的改进。注意到 Q-learning 的目标包含一个最

大化算子 max 的操作。而 Q 又由于环境、非稳态、函数近似或者其他原因，可能带有噪声。最大噪声的期望并不会小于再生的最大期望，即

$$\mathrm{E}\left(\max\left(\varepsilon_1,\ldots,\varepsilon_n\right)\right)\geq\max\left(\mathrm{E}\left(\varepsilon_1\right),\ldots,\mathrm{E}\left(\varepsilon_n\right)\right) \tag{5}$$

因此，下一个 Q 值往往被过估计了。标准 Q 学习的目标可以被重写如下

$$R_t+\gamma\hat{Q}\left(S_{t+1},\arg\max_a\hat{Q}\left(S_{t+1},a;\hat{\theta}\right);\hat{\theta}\right) \tag{6}$$

公式（6）存在**自举半梯度**的问题，因此增加一个额外的网络

$$R_t+\gamma\hat{Q}\left(S_{t+1},\arg\max_a Q\left(S_{t+1},a;\theta\right);\hat{\theta}\right) \tag{7}$$

五、Dueling DQN

对于某些状态来说，不同的动作与预期值无关，因此我们不需要学习各个动作对该状态的影响。Dueling DQN 提出了一种新的网络结构来实现这一思想。确切地说，Q 值可以被分为**状态值和动作优势**两个部分。

$$Q_\pi\left(s,a\right)=V_\pi\left(s\right)+A_\pi\left(s,a\right) \tag{8}$$

之后，Dueling DQN 通过如下方法将两部分的表示分开

$$Q\left(s,a;\theta,\theta_v,\theta_a\right)=V\left(s;\theta,\theta_v\right)+\left(A\left(s,a;\theta,\theta_a\right)-\max_{a'}A\left(s,a';\theta,\theta_a\right)\right) \tag{9}$$

六、优先经验回放

标准 DQN 中还剩下的一个可改进的地方就是，使用更好经验回放采样策略。优先经验回放（Prioritized Experience Replay，PER）是一种将经验进行优先排序的技术。通过该技术可以使重要的状态转移经验被更加频繁地回放(Schauletal.,2015)。PER 的核心思想是通过 TD 误差δ来考虑不同状态转移数据的重要性。TD 误差δ是一个令人惊喜的衡量标准。该方法之所以能有效，是由于某些经验数据相较于其他经验数据，可能包含更多值得学习的信息，所以**给予这些包含更丰富信息量的经验更多的回放机会，有助于使得整个学习进度更为快速和高效**。

当然，**直接使用 TD 误差做优先排序**是最直接能想到的方法。然而这种方法有一些问题。首先，扫描整个回放缓存空间非常低效。其次，这种方法对近似误差和随机回报的噪声十分敏感。最后，这种贪心的方法会使误差收敛缓慢，可能导致刚开始训练时有着高误差的状态转移被频繁地回放。为了克服这些问题，文献(Schauletal.,2015)提出了使用如下方法计算状态转移 i 的采样概率：

$$P\left(i\right)=\frac{p_i^\alpha}{\sum_k p_k^\alpha} \tag{10}$$

其中，pi 指状态转移 i 的优先级，它是一个正数，即 pi>0。α是一个指数超参数，α=0 对应均匀采样情况，而 k 表示对采样的状态转移进行枚举。

七、更多改进内容：多步学习、噪声网络和值分布强化学习

Rainbow 在包含 Double Q-Learning、Dueling 结构和 PER 之外，还包含了另外 3 个 DQN 的扩展，并在雅达利游戏上取得了显著的成果(Hesseletal.,2018)。在本节中，我们将对此展开讨论，并进一步讨论它们的延伸内容。

第一个**扩展是多步学习（Multi-Step Learning）**。使用 n 步回报将使估计更加准确，也被证明可以通过适当调整 n 值来加快学习速度(Suttonetal.,2018)。然而，在离线策略学习过

程中，目标策略和行为策略在多个步骤中的行为选择可能并不匹配。我们可以在文献(Hernandez-Garciaetal.,2019)中找到一个系统性的研究方法来纠正此类错配问题。Rainbow 直接使用了来自给定状态 St 的截断的 n 步回报 Rt(k)(Castroetal.,2018;Hesseletal.,2018)，其中 Rt(k)由以下公式定义。

$$R_t^{(k)} = \sum_{k=0}^{n-1} \gamma^k R_{t+k} \tag{11}$$

接着，Q-Learning 多步学习变体的目标通过下式定义

$$R_t^{(k)} + \gamma^k \max_a Q\left(S_{t+k}, a\right) \tag{12}$$

第二个扩展是**噪声网络(Fortunatoetal.,2017)**。它是另一种ϵ贪心的探索算法，对于像《蒙特祖玛的复仇》这样需要大量探索的游戏十分有效。我们使用一个额外的噪声流将噪声加入线性层 y=(Wx+b)中。

$$y = \left(Wx + b\right) + \left(\left(W_{noisy} \odot \varepsilon_w\right)x + b_{noisy} \odot \varepsilon_b\right) \tag{13}$$

实验表明，噪声网络相比于许多基线算法，使得众多雅达利游戏的得分有了大幅提升。

最后一个扩展是**值分布强化学习(Bellemareetal.,2017)**。该方法为值估计提供了一个新的视角。文献(Bellemareetal.,2017)提出了分布式贝尔曼算子 Tπ用于估计回报 Z 的分布，以改进过去只考虑 Z 的期望的做法:

$$\Gamma^\pi Z = R + \gamma P^\pi Z \tag{14}$$

下图展示 Γ^π 一种连续分布情况

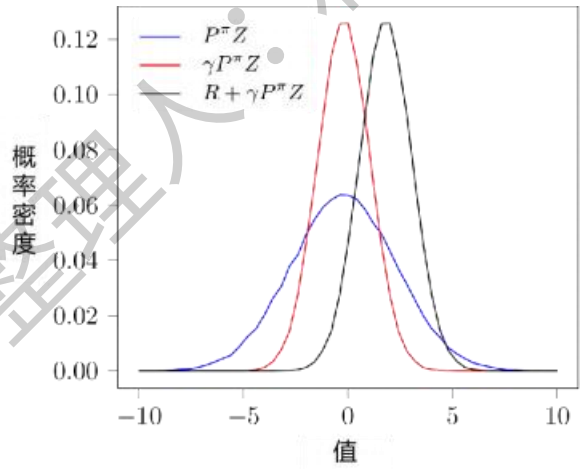


图 3：一种分布式贝尔曼算子在连续分布上的情况。它提供了在策略 π 下，下个状态的回报分布。它将先被折扣因子 γ 折损，然后被当前时间步中的奖励移动