

第七章 策略梯度方法

本章的内容是策略学习 (Policy-Based Reinforcement Learning) 以及策略梯度 (Policy Gradient)。策略学习的意思是通过求解一个优化问题，学出最优策略函数或它的近似（比如策略网络）。第 7.1 节描述策略网络。第 7.2 节把策略学习描述成一个最大化的问题。第 7.3 节推导策略梯度。第 7.4 和 7.5 节用不同的方法近似策略梯度，得到两种训练策略网络的方法——REINFORCE 和 Actor-Critic。

7.1 策略网络

本章考虑离散动作空间，比如 $\mathcal{A} = \{\text{左, 右, 上}\}$ 。策略函数 π 是个条件概率密度函数：

$$\pi(a | s) \triangleq \mathbb{P}(A = a | S = s).$$

策略函数 π 的输入是状态 s 和动作 a ，输出是一个 0 到 1 之间的概率值。举个例子，把超级玛丽游戏当前屏幕上的画面作为 s ，策略函数会输出每个动作的概率值：

$$\pi(\text{左} | s) = 0.5,$$

$$\pi(\text{右} | s) = 0.2,$$

$$\pi(\text{上} | s) = 0.3.$$

如果我们有这样一个策略函数，我们就可以拿它控制智能体。每当观测到一个状态 s ，就用策略函数计算出每个动作的概率值，然后做随机抽样，得到一个动作 a ，让智能体执行 a 。

怎么样才能得到这样一个策略函数呢？当前最有效的方法是用神经网络 $\pi(a|s; \theta)$ 近似策略函数 $\pi(a|s)$ 。神经网络 $\pi(a|s; \theta)$ 被称为策略网络。 θ 表示神经网络的参数；一开始随机初始化 θ ，随后利用收集的状态、动作、奖励去更新 θ 。

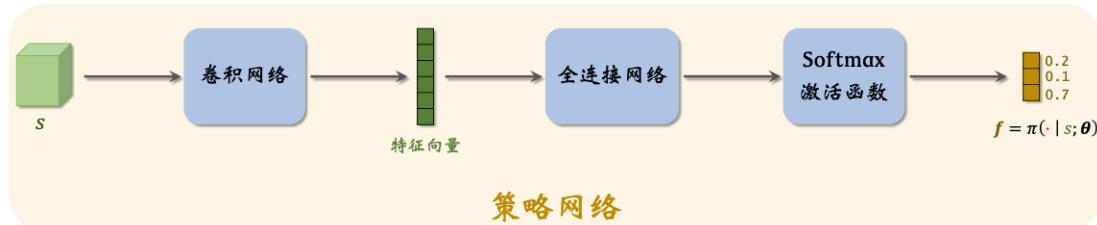


图 7.1：策略网络 $\pi(a|s; \theta)$ 的神经网络结构。输入是状态 s ，输出是动作空间 \mathcal{A} 中每个动作的概率值。

策略网络的结构如图 7.1 所示。策略网络的输入是状态 s 。在 Atari 游戏、围棋等应用中，状态是张量（比如图片），那么应该如图 7.1 所示用卷积网络处理输入。在机器人控制等应用中，状态 s 是向量，它的元素是多个传感器的数值，那么应该把卷积网络换成全连接网络。策略网络输出层的激活函数是 Softmax，因此输出的向量（记作 f ）所有元素都是正数，而且相加等于 1。动作空间 \mathcal{A} 的大小是多少，向量 f 的维度就是多少。在

超级玛丽的例子中， $\mathcal{A} = \{\text{左}, \text{右}, \text{上}\}$ ，那么 f 就是 3 维的向量，比如 $f = [0.2, 0.1, 0.7]$ 。 f 描述了动作空间 \mathcal{A} 上的离散概率分布， f 每个元素对应一个动作：

$$f_1 = \pi(\text{左} | s) = 0.2,$$

$$f_2 = \pi(\text{右} | s) = 0.1,$$

$$f_3 = \pi(\text{上} | s) = 0.7.$$

7.2 策略学习的目标函数

为了推导策略学习的目标函数，我们需要先复习回报和价值函数。回报 U_t 是从 t 时刻开始的所有奖励之和。 U_t 依赖于 t 时刻开始的所有状态和动作：

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

在 t 时刻， U_t 是随机变量，它的不确定性来自于未来未知的状态和动作。动作价值函数的定义是：

$$Q_\pi(s_t, a_t) = \mathbb{E}[U_t \mid S_t = s_t, A_t = a_t].$$

条件期望把 t 时刻状态 s_t 和动作 a_t 看做已知观测值，把 $t + 1$ 时刻后的状态和动作看做未知变量，并消除这些变量。状态价值函数的定义是

$$V_\pi(s_t) = \mathbb{E}_{A_t \sim \pi(\cdot | s_t; \theta)}[Q_\pi(s_t, A_t)].$$

状态价值既依赖于当前状态 s_t ，也依赖于策略网络 π 的参数 θ 。

- 当前状态 s_t 越好，则 $V_\pi(s_t)$ 越大，也就是说明回报 U_t 的期望越大。例如，在超级玛丽游戏中，如果玛丽奥已经接近终点（也就是说当前状态 s_t 很好），那么回报的期望就会很大。
- 策略 π 越好（即参数 θ 越好），那么 $V_\pi(s_t)$ 也会越大。例如，从同一起点出发打游戏，高手（好的策略）的期望回报远高于初学者（差的策略）。

如果一个策略很好，那么对于所有的状态 S ，状态价值 $V_\pi(S)$ 的均值应当很大。因此我们定义目标函数：

$$J(\theta) = \mathbb{E}_S[V_\pi(S)].$$

这个目标函数排除掉了状态 S 的因素，只依赖于策略网络 π 的参数 θ ；策略越好，则 $J(\theta)$ 越大。所以策略学习可以描述为这样一个优化问题：

$$\max_{\theta} J(\theta).$$

我们希望通过策略网络参数 θ 的更新，使得目标函数 $J(\theta)$ 越来越大，也就意味着策略网络越来越强。想要求解最大化问题，显然可以用梯度上升更新 θ ，使得 $J(\theta)$ 增大。设当前策略网络的参数为 θ_{now} 。做梯度上升更新参数，得到新的参数 θ_{new} ：

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} + \beta \cdot \nabla_{\theta} J(\theta_{\text{now}}).$$

此处的 β 是学习率，需要手动调。上面的公式就是训练策略网络的基本想法，其中的梯度

$$\nabla_{\theta} J(\theta_{\text{now}}) \triangleq \frac{\partial J(\theta)}{\partial \theta} \Big|_{\theta=\theta_{\text{now}}}$$

被称作策略梯度。策略梯度可以写成下面定理中的期望形式。之后的算法推导都要基于这个定理，并对其中的期望做近似。

定理 7.1. 策略梯度定理（不严谨的表述）

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_S \left[\mathbb{E}_{A \sim \pi(\cdot|S; \boldsymbol{\theta})} \left[\frac{\partial \ln \pi(A|S; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(S, A) \right] \right].$$



注 上面是策略梯度定理是不严谨的表述，尽管大多数论文和书籍使用这种表述。严格地讲，这个定理只有在“状态 S 服从马尔科夫链的稳态分布 $d(\cdot)$ ”这个假设下才成立。定理中的等号其实是不对的，期望前面应该有一项系数 $1 + \gamma + \dots + \gamma^{n-1} = \frac{1-\gamma^n}{1-\gamma}$ ，其中 γ 是折扣率， n 是一局游戏的长度。策略梯度定理应该是：

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1 - \gamma^n}{1 - \gamma} \cdot \mathbb{E}_{S \sim d(\cdot)} \left[\mathbb{E}_{A \sim \pi(\cdot|S; \boldsymbol{\theta})} \left[\frac{\partial \ln \pi(A|S; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(S, A) \right] \right].$$

在实际应用中，系数 $\frac{1-\gamma^n}{1-\gamma}$ 无关紧要，可以忽略掉。其原因是做梯度上升的时候，系数 $\frac{1-\gamma^n}{1-\gamma}$ 会被学习率 β 吸收。

7.3 策略梯度定理的证明

策略梯度定理是策略学习的关键所在。本节的内容是证明策略梯度定理。尽管本节数学较多，但还是建议读者认真读完第 7.3.1 小节，理解策略梯度简化的推导。第 7.3.2 小节是策略梯度定理完整的证明。由于完整证明较为复杂，大多数教材中不涉及这部分内容，本书也不建议读者理解、掌握完整证明，除非读者从事强化学习科研。

7.3.1 简化的证明

把策略网络 $\pi(a | s; \theta)$ 看做动作的概率密度函数。状态价值函数 $V_\pi(s)$ 可以写成：

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_{A \sim \pi(\cdot | s; \theta)} [Q_\pi(s, A)] \\ &= \sum_{a \in \mathcal{A}} \pi(a | s; \theta) \cdot Q_\pi(s, a). \end{aligned}$$

状态价值 $V_\pi(s)$ 关于 θ 的梯度可以写作：

$$\begin{aligned} \frac{\partial V_\pi(s)}{\partial \theta} &= \frac{\partial}{\partial \theta} \sum_{a \in \mathcal{A}} \pi(a | s; \theta) \cdot Q_\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \frac{\partial \pi(a | s; \theta) \cdot Q_\pi(s, a)}{\partial \theta}. \end{aligned} \quad (7.1)$$

上面第二个等式把求导放入连加里面；等式成立的原因是求导的对象 θ 与连加的对象 a 不同。回忆一下链式法则：设 $z = f(x) \cdot g(x)$ ，那么

$$\frac{\partial z}{\partial x} = \frac{\partial f(x)}{\partial x} \cdot g(x) + f(x) \cdot \frac{\partial g(x)}{\partial x}.$$

应用链式法则，公式 (7.1) 中的梯度可以写作：

$$\begin{aligned} \frac{\partial V_\pi(s)}{\partial \theta} &= \sum_{a \in \mathcal{A}} \frac{\partial \pi(a | s; \theta)}{\partial \theta} \cdot Q_\pi(s, a) + \sum_{a \in \mathcal{A}} \pi(a | s; \theta) \cdot \frac{\partial Q_\pi(s, a)}{\partial \theta} \\ &= \sum_{a \in \mathcal{A}} \frac{\partial \pi(a | s; \theta)}{\partial \theta} \cdot Q_\pi(s, a) + \underbrace{\mathbb{E}_{A \sim \pi(\cdot | s; \theta)} \left[\frac{\partial Q_\pi(s, A)}{\partial \theta} \right]}_{\text{设为 } x}. \end{aligned}$$

上面公式最右边一项 x 的分析非常复杂，此处不具体分析了。由上面的公式可得：

$$\begin{aligned} \frac{\partial V_\pi(s)}{\partial \theta} &= \sum_{A \in \mathcal{A}} \frac{\partial \pi(A | S; \theta)}{\partial \theta} \cdot Q_\pi(S, A) + x \\ &= \sum_{A \in \mathcal{A}} \pi(A | S; \theta) \cdot \underbrace{\frac{1}{\pi(A | S; \theta)} \cdot \frac{\partial \pi(A | S; \theta)}{\partial \theta}}_{\text{等于 } \partial \ln \pi(A | S; \theta) / \partial \theta} \cdot Q_\pi(S, A) + x. \end{aligned}$$

上面第二个等式成立的原因是添加的两个红色项相乘等于一。公式中用下花括号标出的项等于 $\frac{\partial \ln \pi(A | S; \theta)}{\partial \theta}$ 。由此可得

$$\begin{aligned} \frac{\partial V_\pi(s)}{\partial \theta} &= \sum_{A \in \mathcal{A}} \pi(A | S; \theta) \cdot \frac{\partial \ln \pi(A | S; \theta)}{\partial \theta} \cdot Q_\pi(S, A) + x \\ &= \mathbb{E}_{A \sim \pi(\cdot | S; \theta)} \left[\frac{\partial \ln \pi(A | S; \theta)}{\partial \theta} \cdot Q_\pi(S, A) \right] + x. \end{aligned} \quad (7.2)$$

公式中红色标出的 $\pi(A|S; \theta)$ 被看做概率密度函数，因此连加可以写成期望的形式。根据目标函数的定义 $J(\theta) = \mathbb{E}_S[V_\pi(S)]$ 可得

$$\begin{aligned}\frac{\partial J(\theta)}{\partial \theta} &= \mathbb{E}_S\left[\frac{\partial V_\pi(S)}{\partial \theta}\right] \\ &= \mathbb{E}_S\left[\mathbb{E}_{A \sim \pi(\cdot|S; \theta)}\left[\frac{\partial \ln \pi(A|S; \theta)}{\partial \theta} \cdot Q_\pi(S, A)\right]\right] + \mathbb{E}_S[x].\end{aligned}$$

不严谨的证明通常忽略掉 x ，于是得到定理 7.1。在下一小节中，我们给出严格的证明。除非读者对强化学习的数学很感兴趣，否则没必要阅读下一小节。

7.3.2 完整的证明

本小节给出策略梯度定理的严格数学证明。首先证明几个引理，最后用引理证明策略梯度定理。引理 7.2 分析梯度 $\frac{\partial V_\pi(s)}{\partial \theta}$ ，并把它递归地表示为 $\frac{\partial V_\pi(S')}{\partial \theta}$ 的期望，其中 S' 是下一时刻的状态。

引理 7.2. 递归公式

$$\frac{\partial V_\pi(s)}{\partial \theta} = \mathbb{E}_{A \sim \pi(\cdot|s; \theta)}\left[\frac{\partial \ln \pi(A|s; \theta)}{\partial \theta} \cdot Q_\pi(s, A) + \gamma \cdot \mathbb{E}_{S' \sim p(\cdot|s, A)}\left[\frac{\partial V_\pi(S')}{\partial \theta}\right]\right].$$



证明 设奖励 R 和新状态 S' 是在智能体执行动作 A 之后由环境给出的。新状态 S' 的概率密度函数是状态转移函数 $p(S'|S, A)$ 。设奖励 R 是 S, A, S' 三者的函数，因此可以将其记为 $R(S, A, S')$ 。由贝尔曼方程可得：

$$\begin{aligned}Q_\pi(s, a) &= \mathbb{E}_{S' \sim p(\cdot|s, a)}[R(s, a, S') + \gamma \cdot V_\pi(s')] \\ &= \sum_{s' \in \mathcal{S}} p(s'|s, a) \cdot [R(s, a, s') + \gamma \cdot V_\pi(s')] \\ &= \sum_{s' \in \mathcal{S}} p(s'|s, a) \cdot R(s, a, s') + \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s, a) \cdot V_\pi(s').\end{aligned}\quad (7.3)$$

在观测到 s, a, s' 之后， $p(s'|s, a)$ 和 $R(s, a, s')$ 都与策略网络 π 无关，因此

$$\frac{\partial}{\partial \theta}[p(s'|s, a) \cdot R(s, a, s')] = 0. \quad (7.4)$$

由公式 (7.3) 与 (7.4) 可得：

$$\begin{aligned}\frac{\partial Q_\pi(s, a)}{\partial \theta} &= \sum_{s' \in \mathcal{S}} \underbrace{\frac{\partial}{\partial \theta}[p(s'|s, a) \cdot R(s, a, s')]}_{\text{等于零}} + \gamma \cdot \sum_{s' \in \mathcal{S}} \frac{\partial}{\partial \theta}[p(s'|s, a) \cdot V_\pi(s')] \\ &= \gamma \cdot \sum_{s' \in \mathcal{S}} p(s'|s, a) \cdot \frac{\partial V_\pi(s')}{\partial \theta} \\ &= \gamma \cdot \mathbb{E}_{S' \sim p(\cdot|s, a)}\left[\frac{\partial V_\pi(S')}{\partial \theta}\right].\end{aligned}\quad (7.5)$$

由上一小节的公式 (7.2) 可得：

$$\begin{aligned}\frac{\partial V_\pi(s)}{\partial \theta} &= \mathbb{E}_{A \sim \pi(\cdot|S; \theta)}\left[\frac{\partial \ln \pi(A|S; \theta)}{\partial \theta} \cdot Q_\pi(S, A)\right] + \mathbb{E}_{A \sim \pi(\cdot|S; \theta)}\left[\frac{\partial Q_\pi(s, a)}{\partial \theta}\right].\end{aligned}\quad (7.6)$$

结合公式(7.5)、(7.6)可得引理7.2 □

引理7.3. 策略梯度的连加形式

设 $\mathbf{g}(s, a; \boldsymbol{\theta}) \triangleq Q_\pi(s, a) \cdot \frac{\partial \ln \pi(a|s; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ 。设一局游戏在第 n 步之后结束。那么

$$\begin{aligned} \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \mathbb{E}_{S_1, A_1} [\mathbf{g}(S_1, A_1; \boldsymbol{\theta})] \\ &\quad + \gamma \cdot \mathbb{E}_{S_1, A_1, S_2, A_2} [\mathbf{g}(S_2, A_2; \boldsymbol{\theta})] \\ &\quad + \gamma^2 \cdot \mathbb{E}_{S_1, A_1, S_2, A_2, S_3, A_3} [\mathbf{g}(S_3, A_3; \boldsymbol{\theta})] \\ &\quad + \dots \\ &\quad + \gamma^{n-1} \cdot \mathbb{E}_{S_1, A_1, S_2, A_2, S_3, A_3, \dots, S_n, A_n} [\mathbf{g}(S_n, A_n; \boldsymbol{\theta})]. \end{aligned}$$



证明 设 S 、 A 为当前状态和动作， S' 为下一个状态。引理7.2 证明了下面的结论：

$$\frac{\partial V_\pi(S)}{\partial \boldsymbol{\theta}} = \mathbb{E}_A \left[\underbrace{\frac{\partial \ln \pi(A|S; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \cdot Q_\pi(S, A)}_{\text{定义为 } \mathbf{g}(S, A; \boldsymbol{\theta})} + \gamma \cdot \mathbb{E}_{S'} \left[\frac{\partial V_\pi(S')}{\partial \boldsymbol{\theta}} \right] \right].$$

这样我们可以把 $\frac{\partial V_\pi(S_1)}{\partial \boldsymbol{\theta}}$ 写成递归的形式：

$$\frac{\partial V_\pi(S_1)}{\partial \boldsymbol{\theta}} = \mathbb{E}_{A_1} [\mathbf{g}(S_1, A_1; \boldsymbol{\theta})] + \gamma \cdot \mathbb{E}_{A_1, S_2} \left[\frac{\partial V_\pi(S_2)}{\partial \boldsymbol{\theta}} \right]. \quad (7.7)$$

同理， $\frac{\partial V_\pi(S_2)}{\partial \boldsymbol{\theta}}$ 可以写成

$$\frac{\partial V_\pi(S_2)}{\partial \boldsymbol{\theta}} = \mathbb{E}_{A_2} [\mathbf{g}(S_2, A_2; \boldsymbol{\theta})] + \gamma \cdot \mathbb{E}_{A_2, S_3} \left[\frac{\partial V_\pi(S_3)}{\partial \boldsymbol{\theta}} \right]. \quad (7.8)$$

把等式(7.8)插入等式(7.7)，得到

$$\begin{aligned} \frac{\partial V_\pi(S_1)}{\partial \boldsymbol{\theta}} &= \mathbb{E}_{A_1} [\mathbf{g}(S_1, A_1; \boldsymbol{\theta})] \\ &\quad + \gamma \cdot \mathbb{E}_{A_1, S_2, A_2} [\mathbf{g}(S_2, A_2; \boldsymbol{\theta})] \\ &\quad + \gamma^2 \cdot \mathbb{E}_{A_1, S_2, A_2, S_3} \left[\frac{\partial V_\pi(S_3)}{\partial \boldsymbol{\theta}} \right]. \end{aligned}$$

按照这种规律递归下去，可得：

$$\begin{aligned} \frac{\partial V_\pi(S_1)}{\partial \boldsymbol{\theta}} &= \mathbb{E}_{A_1} [\mathbf{g}(S_1, A_1; \boldsymbol{\theta})] \\ &\quad + \gamma \cdot \mathbb{E}_{A_1, S_2, A_2} [\mathbf{g}(S_2, A_2; \boldsymbol{\theta})] \\ &\quad + \gamma^2 \cdot \mathbb{E}_{A_1, S_2, A_2, S_3, A_3} [\mathbf{g}(S_3, A_3; \boldsymbol{\theta})] \\ &\quad + \dots \\ &\quad + \gamma^{n-1} \cdot \mathbb{E}_{A_1, S_2, A_2, S_3, A_3, \dots, S_n, A_n} [\mathbf{g}(S_n, A_n; \boldsymbol{\theta})] \\ &\quad + \gamma^n \cdot \underbrace{\mathbb{E}_{A_1, S_2, A_2, S_3, A_3, \dots, S_n, A_n, S_{n+1}} \left[\frac{\partial V_\pi(S_{n+1})}{\partial \boldsymbol{\theta}} \right]}_{\text{等于零}}. \end{aligned}$$

上式中最后一项等于零，原因是游戏在 n 时刻后结束，而 $n+1$ 时刻之后没有奖励，所以 $n+1$ 时刻的回报和价值都是零。最后，由上面的公式和

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \mathbb{E}_{S_1} \left[\frac{\partial V_\pi(S_1)}{\partial \boldsymbol{\theta}} \right]$$

可得引理 7.3. □

稳态分布: 想要严格证明策略梯度定理, 需要用到马尔科夫链 (Markov Chain) 的稳态分布 (Stationary Distribution)。设状态 s' 是这样得到的: $s \rightarrow a \rightarrow s'$ 。回忆一下, 状态转移函数 $p(s'|s, a)$ 是一个概率密度函数。设 $d(s)$ 是状态 s 的概率密度函数。那么状态 s' 的边缘分布是

$$\tilde{d}(s') = \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p(s'|s, a) \cdot \pi(a|s; \theta) \cdot d(s).$$

如果 $\tilde{d}(\cdot)$ 与 $d(\cdot)$ 是相同的概率密度函数, 则意味着马尔科夫链达到稳态, 而 $d(\cdot)$ 就是稳态时的概率密度函数。

引理 7.4

设 $d(\cdot)$ 是马尔科夫链稳态时的概率密度函数。那么对于任意函数 $f(S')$,

$$\mathbb{E}_{S \sim d(\cdot)} [\mathbb{E}_{A \sim \pi(\cdot|S; \theta)} [\mathbb{E}_{S' \sim p(\cdot|s, A)} [f(S')]]] = \mathbb{E}_{S' \sim d(\cdot)} [f(S')].$$
♡

证明 把引理中的期望写成连加的形式:

$$\begin{aligned} & \mathbb{E}_{S \sim d(\cdot)} [\mathbb{E}_{A \sim \pi(\cdot|S; \theta)} [\mathbb{E}_{S' \sim p(\cdot|s, A)} [f(S')]]] \\ &= \sum_{s \in \mathcal{S}} d(s) \sum_{a \in \mathcal{A}} \pi(a|s; \theta) \sum_{s' \in \mathcal{S}} p(s'|s, a) \cdot f(s') \\ &= \sum_{s' \in \mathcal{S}} f(s') \underbrace{\sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} p(s'|s, a) \cdot \pi(a|s; \theta) \cdot d(s)}_{\text{等于 } d(s')} \end{aligned}$$

上面等式最右边标出的项等于 $d(s')$, 这是根据稳态分布的定义得到的。于是有

$$\begin{aligned} \mathbb{E}_{S \sim d(\cdot)} [\mathbb{E}_{A \sim \pi(\cdot|S; \theta)} [\mathbb{E}_{S' \sim p(\cdot|s, A)} [f(S')]]] &= \sum_{s' \in \mathcal{S}} f(s') \cdot d(s') \\ &= \mathbb{E}_{S' \sim d(\cdot)} [f(S')]. \end{aligned}$$

由此可得引理 7.4 □

定理 7.5. 策略梯度定理 (严谨的表述)

设目标函数为 $J(\theta) = \mathbb{E}_{S \sim d(\cdot)} [V_\pi(S)]$, 设折扣率 $\gamma < 1$, 设马尔科夫链稳态分布的概率密度函数为 $d(s)$ 。那么

$$\frac{\partial J(\theta)}{\partial \theta} = \left(1 + \gamma + \gamma^2 + \dots + \gamma^{n-1}\right) \cdot \mathbb{E}_{S \sim d(\cdot)} \left[\mathbb{E}_{A \sim \pi(\cdot|S; \theta)} \left[\frac{\partial \ln \pi(A|S; \theta)}{\partial \theta} \cdot Q_\pi(S, A) \right] \right].$$
♡

证明 设初始状态 S_1 服从马尔科夫链的稳态分布, 它的概率密度函数是 $d(S_1)$ 。对于所有的 $t = 1, \dots, n$, 动作 A_t 根据策略网络抽样得到:

$$A_t \sim \pi(\cdot | S_t; \theta),$$

新的状态 S_{t+1} 根据状态转移函数抽样得到:

$$S_{t+1} \sim p(\cdot | S_t, A_t).$$

对于任意函数 f , 反复应用引理 7.4 可得:

$$\begin{aligned}
 & \mathbb{E}_{S_1 \sim d} \left\{ \mathbb{E}_{A_1 \sim \pi, S_2 \sim p} \left\{ \mathbb{E}_{A_2, S_3, A_3, S_4, \dots, A_{t-1}, S_t} [f(S_t)] \right\} \right\} \\
 &= \mathbb{E}_{S_2 \sim d} \left\{ \mathbb{E}_{A_2, S_3, A_3, S_4, \dots, A_{t-1}, S_t} [f(S_t)] \right\} \quad (\text{由引理 7.4 得出}) \\
 &= \mathbb{E}_{S_2 \sim d} \left\{ \mathbb{E}_{A_2 \sim \pi, S_3 \sim p} \left\{ \mathbb{E}_{A_3, S_4, A_4, S_5, \dots, A_{t-1}, S_t} [f(S_t)] \right\} \right\} \\
 &= \mathbb{E}_{S_3 \sim d} \left\{ \mathbb{E}_{A_3, S_4, A_4, S_5, \dots, A_{t-1}, S_t} [f(S_t)] \right\} \quad (\text{由引理 7.4 得出}) \\
 &\vdots \\
 &= \mathbb{E}_{S_{t-1} \sim d} \left\{ \mathbb{E}_{A_{t-1} \sim \pi, S_t \sim p} \left\{ f(S_t) \right\} \right\} \\
 &= \mathbb{E}_{S_t \sim d} \left\{ f(S_t) \right\}. \quad (\text{由引理 7.4 得出})
 \end{aligned}$$

设 $\mathbf{g}(s, a; \boldsymbol{\theta}) \triangleq Q_\pi(s, a) \cdot \frac{\partial \ln \pi(a|s; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ 。设一局游戏在第 n 步之后结束。由引理 7.3 与上面的公式可得:

$$\begin{aligned}
 \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \mathbb{E}_{S_1, A_1} [\mathbf{g}(S_1, A_1; \boldsymbol{\theta})] \\
 &\quad + \gamma \cdot \mathbb{E}_{S_1, A_1, S_2, A_2} [\mathbf{g}(S_2, A_2; \boldsymbol{\theta})] \\
 &\quad + \gamma^2 \cdot \mathbb{E}_{S_1, A_1, S_2, A_2, S_3, A_3} [\mathbf{g}(S_3, A_3; \boldsymbol{\theta})] \\
 &\quad + \dots \\
 &\quad + \gamma^{n-1} \cdot \mathbb{E}_{S_1, A_1, S_2, A_2, S_3, A_3, \dots, S_n, A_n} [\mathbf{g}(S_n, A_n; \boldsymbol{\theta})] \\
 &= \mathbb{E}_{S_1 \sim d(\cdot)} \left\{ \mathbb{E}_{A_1 \sim \pi(\cdot|S_1; \boldsymbol{\theta})} [\mathbf{g}(S_1, A_1; \boldsymbol{\theta})] \right\} \\
 &\quad + \gamma \cdot \mathbb{E}_{S_2 \sim d(\cdot)} \left\{ \mathbb{E}_{A_2 \sim \pi(\cdot|S_2; \boldsymbol{\theta})} [\mathbf{g}(S_2, A_2; \boldsymbol{\theta})] \right\} \\
 &\quad + \gamma^2 \cdot \mathbb{E}_{S_3 \sim d(\cdot)} \left\{ \mathbb{E}_{A_3 \sim \pi(\cdot|S_3; \boldsymbol{\theta})} [\mathbf{g}(S_3, A_3; \boldsymbol{\theta})] \right\} \\
 &\quad + \dots \\
 &\quad + \gamma^{n-1} \cdot \mathbb{E}_{S_n \sim d(\cdot)} \left\{ \mathbb{E}_{A_n \sim \pi(\cdot|S_n; \boldsymbol{\theta})} [\mathbf{g}(S_n, A_n; \boldsymbol{\theta})] \right\} \\
 &= (1 + \gamma + \gamma^2 + \dots + \gamma^{n-1}) \cdot \mathbb{E}_{S \sim d(\cdot)} \left\{ \mathbb{E}_{A \sim \pi(\cdot|S; \boldsymbol{\theta})} [\mathbf{g}(S, A; \boldsymbol{\theta})] \right\}.
 \end{aligned}$$

由此可得定理 7.5。 □

7.3.3 近似策略梯度

先复习一下前两小节的内容。策略学习可以表述为这样一个优化问题：

$$\max_{\theta} \left\{ J(\theta) \triangleq \mathbb{E}_S [V_\pi(S)] \right\}.$$

求解这个最大化问题最简单的算法就是梯度上升：

$$\theta \leftarrow \theta + \beta \cdot \nabla_{\theta} J(\theta).$$

其中的 $\nabla_{\theta} J(\theta)$ 是策略梯度。策略梯度定理证明：

$$\nabla_{\theta} J(\theta) = \mathbb{E}_S \left[\mathbb{E}_{A \sim \pi(\cdot | S; \theta)} \left[Q_\pi(S, A) \cdot \nabla_{\theta} \ln \pi(A | S; \theta) \right] \right].$$

解析求出这个期望是不可能的，因为我们并不知道状态 S 概率密度函数；即使我们知道 S 的概率密度函数，能够通过连加或者定积分求出期望，我们也不愿意这样做，因为连加或者定积分的计算量非常大。

回忆一下，第 2 章介绍了期望的蒙特卡洛近似，可以将这种方法用来近似策略梯度中的期望。每次从环境中观测到一个状态 s ，它相当于随机变量 S 的观测值。然后再根据当前的策略网络（策略网络的参数必须是最新的）随机抽样得出一个动作：

$$a \sim \pi(\cdot | s; \theta).$$

计算随机梯度：

$$g(s, a; \theta) \triangleq Q_\pi(s, a) \cdot \nabla_{\theta} \ln \pi(a | s; \theta).$$

很显然， $g(s, a; \theta)$ 是策略梯度 $\nabla_{\theta} J(\theta)$ 的无偏估计：

$$\nabla_{\theta} J(\theta) = \mathbb{E}_S \left[\mathbb{E}_{A \sim \pi(\cdot | S; \theta)} \left[g(s, a; \theta) \right] \right].$$

于是我们得到下面的结论：

结论 7.1

随机梯度 $g(s, a; \theta) \triangleq Q_\pi(s, a) \cdot \nabla_{\theta} \ln \pi(a | s; \theta)$ 是策略梯度 $\nabla_{\theta} J(\theta)$ 的无偏估计。



应用上述结论，我们可以做随机梯度上升来更新 θ ，使得目标函数 $J(\theta)$ 逐渐增长：

$$\theta \leftarrow \theta + \beta \cdot g(s, a; \theta).$$

此处的 β 是学习率，需要手动调。但是这种方法仍然不可行，原因在于我们不知道动作价值函数 $Q_\pi(s, a)$ 。在后面两节中，我们用两种方法对 $Q_\pi(s, a)$ 做近似：一种方法是 REINFORCE，用实际观测的回报 u 近似 $Q_\pi(s, a)$ ；另一种方法是 Actor-Critic，用神经网络 $q(s, a; w)$ 近似 $Q_\pi(s, a)$ 。

7.4 REINFORCE

策略梯度方法用策略梯度 $\nabla_{\theta} J(\theta)$ 更新策略网络参数 θ , 从而增大目标函数。上一节中, 我们推导出策略梯度 $\nabla_{\theta} J(\theta)$ 的无偏估计, 即下面的随机梯度:

$$\mathbf{g}(s, a; \theta) \triangleq Q_{\pi}(s, a) \cdot \nabla_{\theta} \ln \pi(a | s; \theta).$$

但是其中的动作价值函数 Q_{π} 是未知的, 导致无法直接计算 $\mathbf{g}(s, a; \theta)$ 。REINFORCE 进一步对 Q_{π} 做蒙特卡洛近似, 把它替换成回报 u 。REINFORCE 属于策略梯度方法。

7.4.1 REINFORCE 的简化推导

设一局游戏有 n 步, 一局中的奖励记作 R_1, \dots, R_n 。回忆一下, t 时刻的折扣回报定义为:

$$U_t = \sum_{k=t}^n \gamma^{k-t} \cdot R_k.$$

而动作价值定义为 U_t 的条件期望:

$$Q_{\pi}(s_t, a_t) = \mathbb{E}[U_t | S_t = s_t, A_t = a_t].$$

我们可以用蒙特卡洛近似上面的条件期望。从时刻 t 开始, 智能体完成一局游戏, 观测到全部奖励 r_t, \dots, r_n , 然后可以计算出 $u_t = \sum_{k=t}^n \gamma^{k-t} \cdot r_k$ 。因为 u_t 是随机变量 U_t 的观测值, 所以 u_t 是上面公式中期望的蒙特卡洛近似。在实践中, 可以用 u_t 代替 $Q_{\pi}(s_t, a_t)$, 那么随机梯度 $\mathbf{g}(s_t, a_t; \theta)$ 可以近似成

$$\tilde{\mathbf{g}}(s_t, a_t; \theta) = u_t \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta).$$

$\tilde{\mathbf{g}}$ 是 \mathbf{g} 的无偏估计, 所以也是策略梯度 $\nabla_{\theta} J(\theta)$ 的无偏估计; $\tilde{\mathbf{g}}$ 也是一种随机梯度。

我们可以用反向传播计算出 $\ln \pi$ 关于 θ 的梯度, 而且可以实际观测到 u_t , 于是我们可以实际计算出随机梯度 $\tilde{\mathbf{g}}$ 的值。有了随机梯度的值, 我们可以做随机梯度上升更新策略网络参数 θ :

$$\theta \leftarrow \theta + \beta \cdot \tilde{\mathbf{g}}(s_t, a_t; \theta). \quad (7.9)$$

根据上述推导, 我们得到了训练策略网络的方法, 这种方法叫做 REINFORCE。

7.4.2 训练流程

当前策略网络的参数是 θ_{now} 。REINFORCE 执行下面的步骤对策略网络的参数做一次更新:

1. 用策略网络 θ_{now} 控制智能体从头开始玩一局游戏, 得到一条轨迹 (Trajectory):

$$s_1, a_1, r_1, \quad s_2, a_2, r_2, \quad \dots, \quad s_n, a_n, r_n.$$

2. 计算所有的回报:

$$u_t = \sum_{k=t}^n \gamma^{k-t} \cdot r_k, \quad \forall t = 1, \dots, n.$$

3. 用 $\{(s_t, a_t)\}_{t=1}^n$ 作为数据，做反向传播计算：

$$\nabla_{\theta} \ln \pi(a_t | s_t; \theta_{\text{now}}), \quad \forall t = 1, \dots, n.$$

4. 做随机梯度上升更新策略网络参数：

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} + \beta \cdot \sum_{t=1}^n \gamma^{t-1} \cdot \underbrace{u_t \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta_{\text{now}})}_{\text{即随机梯度 } \tilde{g}(s_t, a_t; \theta_{\text{now}})}.$$

注 在算法最后一步中，随机梯度前面乘以系数 γ^{t-1} 。读者可能会好奇，为什么需要这个系数呢？前面 REINFORCE 的推导是简化的，而非严谨的数学推导；按照我们简化的推导，不应该乘以系数 γ^{t-1} 。下一小节做严格的数学推导，得出的 REINFORCE 算法需要系数 γ^{t-1} 。读者只要知道这个事实就行了，不必读懂下一小节的数学推导。

注 REINFORCE 是一种同策略 (On-Policy) 方法，要求行为策略 (Behavior Policy) 与目标策略 (Target Policy) 相同，两者都必须是策略网络 $\pi(a|s; \theta_{\text{now}})$ ，其中 θ_{now} 是策略网络当前的参数。所以经验回放不适用于 REINFORCE。

7.4.3 REINFORCE 严格的推导

第 7.4.1 小节对策略梯度做近似，推导出 REINFORCE 方法。那种推导是简化过的，帮助读者理解 REINFORCE 算法，但实际上那种推导并不够严谨。¹ 本小节做严格的数学推导，对策略梯度做近似，得出真正的 REINFORCE 方法。建议对数学证明不感兴趣的读者跳过本小节。

根据定义， $\mathbf{g}(s, a; \theta) \triangleq Q_{\pi}(s, a) \cdot \nabla_{\theta} \ln \pi(a | s; \theta)$ 。引理 7.3 把策略梯度 $\nabla_{\theta} J(\theta)$ 表示成期望的连加：

$$\begin{aligned} \nabla_{\theta} J(\theta) &= \mathbb{E}_{S_1, A_1} [\mathbf{g}(S_1, A_1; \theta)] \\ &\quad + \gamma \cdot \mathbb{E}_{S_1, A_1, S_2, A_2} [\mathbf{g}(S_2, A_2; \theta)] \\ &\quad + \gamma^2 \cdot \mathbb{E}_{S_1, A_1, S_2, A_2, S_3, A_3} [\mathbf{g}(S_3, A_3; \theta)] \\ &\quad + \dots \\ &\quad + \gamma^{n-1} \cdot \mathbb{E}_{S_1, A_1, S_2, A_2, S_3, A_3, \dots, S_n, A_n} [\mathbf{g}(S_n, A_n; \theta)]. \end{aligned} \quad (7.10)$$

我们可以对期望做蒙特卡洛近似。首先观测到第一个状态 $S_1 = s_1$ 。然后用最新的策略网络 $\pi(a|s; \theta_{\text{now}})$ 控制智能体与环境交互，观测到到轨迹

$$s_1, a_1, r_1, \quad s_2, a_2, r_2, \quad \dots, \quad s_n, a_n, r_n.$$

对公式 (7.10) 中的期望做蒙特卡洛近似，得到：

$$\nabla_{\theta} J(\theta_{\text{now}}) \approx \mathbf{g}(s_1, a_1; \theta_{\text{now}}) + \gamma \cdot \mathbf{g}(s_2, a_2; \theta_{\text{now}}) + \dots + \gamma^{n-1} \cdot \mathbf{g}(s_n, a_n; \theta_{\text{now}}).$$

进一步把 $\mathbf{g}(s_t, a_t; \theta_{\text{now}}) \triangleq Q_{\pi}(s_t, a_t) \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta_{\text{now}})$ 中的 $Q_{\pi}(s_t, a_t)$ 替换成 u_t ，那么 $\mathbf{g}(s_t, a_t; \theta_{\text{now}})$ 就被近似成为

$$\mathbf{g}(s_t, a_t; \theta_{\text{now}}) \approx u_t \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta_{\text{now}}).$$

¹ 第 7.4.1 小节把相邻状态 S_t 和 S_{t+1} 视作独立的变量，但其实它们并不独立。

经过上述两次近似，策略梯度被近似成为下面的随机梯度

$$\nabla_{\theta} J(\theta_{\text{now}}) \approx \sum_{t=1}^n \gamma^{t-1} \cdot u_t \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta_{\text{now}}).$$

这样就得到了 REINFORCE 算法的随机梯度上升公式：

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} + \beta \cdot \sum_{t=1}^n \gamma^{t-1} \cdot u_t \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta_{\text{now}}).$$

7.5 Actor-Critic

策略梯度方法用策略梯度 $\nabla_{\theta} J(\theta)$ 更新策略网络参数 θ , 从而增大目标函数。第 7.2 节推导出策略梯度 $\nabla_{\theta} J(\theta)$ 的无偏估计, 即下面的随机梯度:

$$\mathbf{g}(s, a; \theta) \triangleq Q_{\pi}(s, a) \cdot \nabla_{\theta} \ln \pi(a | s; \theta).$$

但是其中的动作价值函数 Q_{π} 是未知的, 导致无法直接计算 $\mathbf{g}(s, a; \theta)$ 。上一节的 REINFORCE 用实际观测的回报近似 Q_{π} , 本节的 Actor-Critic 方法用神经网络近似 Q_{π} 。

7.5.1 价值网络

Actor-Critic 方法中用一个神经网络近似动作价值函数 $Q_{\pi}(s, a)$, 这个神经网络叫做“价值网络”, 记为 $q(s, a; w)$, 其中的 w 表示神经网络中可训练的参数。价值网络的输入是状态 s , 输出是每个动作的价值。动作空间 \mathcal{A} 中有多少种动作, 那么价值网络的输出就是多少维的向量, 向量每个元素对应一个动作。举个例子, 动作空间是 $\mathcal{A} = \{\text{左}, \text{右}, \text{上}\}$, 价值网络的输出是

$$\begin{aligned} q(s, \text{左}; w) &= 219, \\ q(s, \text{右}; w) &= -73, \\ q(s, \text{上}; w) &= 580. \end{aligned}$$

神经网络的结构见图 7.2。

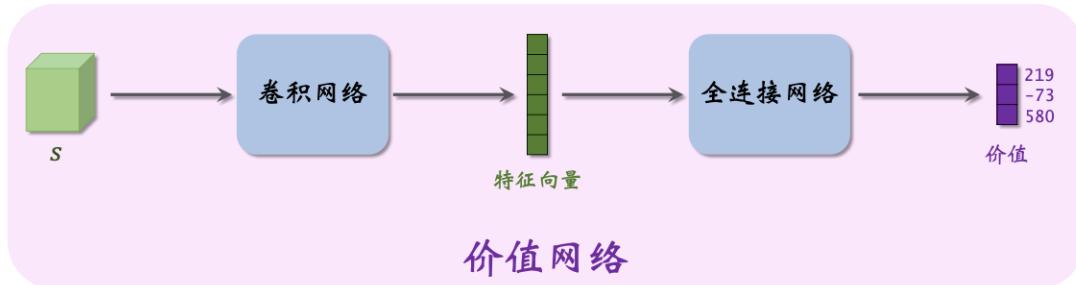


图 7.2: 价值网络 $q(s, a; w)$ 的结构。输入是状态 s ; 输出是每个动作的价值。

虽然价值网络 $q(s, a; w)$ 与之前学的 DQN 有相同的结构, 但是两者的意义不同, 训练算法也不同。

- 价值网络是对动作价值函数 $Q_{\pi}(s, a)$ 的近似。而 DQN 则是对最优动作价值函数 $Q_{\star}(s, a)$ 的近似。
- 对价值网络的训练使用的是 SARSA 算法, 它属于同策略, 不能用经验回放。对 DQN 的训练使用的是 Q 学习算法, 它属于异策略, 可以用经验回放。

7.5.2 算法的推导

Actor-Critic 翻译成“演员—评委”方法。策略网络 $\pi(a|s; \theta)$ 相当于演员，它基于状态 s 做出动作 a 。价值网络 $q(s, a; \theta)$ 相当于评委，它给演员的表现打分，量化在状态 s 的情况下做出动作 a 的好坏程度。策略网络（演员）和价值网络（评委）的关系如图 7.3 所示。

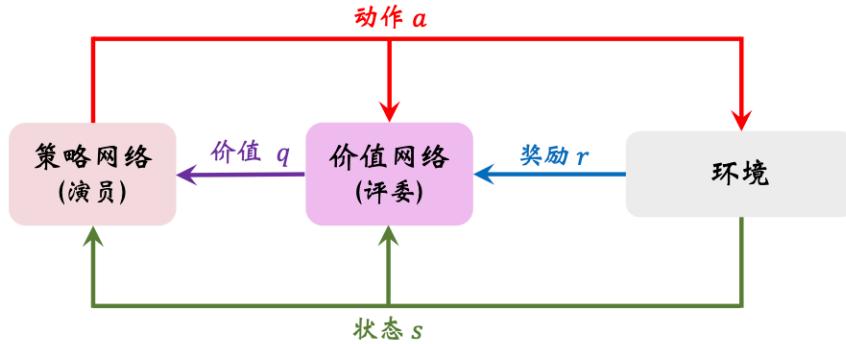


图 7.3: Actor-Critic 方法中策略网络（演员）和价值网络（评委）的关系图。

读者可能会对图 7.3 感到不解：为什么不直接把奖励 R 反馈给策略网络（演员），而要用价值网络（评委）这样一个中介呢？原因是这样的。策略学习的目标函数 $J(\theta)$ 是回报 U 的期望，而不是奖励 R 的期望；注意回报 U 和奖励 R 的区别。虽然能观测到当前的奖励 R ，但是它对价值网络是毫无意义的；价值网络在乎的是回报 U ，也就是未来所有奖励的加权和。价值网络能够估算出回报 U 的期望，因此能帮助训练策略网络。

训练策略网络（演员）：策略网络（演员）想要改进自己的演技，但是演员自己不知道什么样的表演才算更好，所以需要价值网络（评委）的帮助。在演员做出动作 a 之后，评委打一个分数 $\hat{q} \triangleq q(s, a; \theta)$ ，并把分数反馈给演员，帮助演员做出改进。演员利用当前状态 s ，自己的动作 a ，以及评委的打分 \hat{q} ，计算近似策略梯度，然后更新自己的参数 θ （相当于改变自己的技术）。通过这种方式，演员的表现越来越受评委的好评，于是演员的获得的评分 \hat{q} 越来越高。

训练策略网络的基本想法是用策略梯度 $\nabla_{\theta} J(\theta)$ 的近似来更新参数 θ 。之前我们推导过策略梯度的无偏估计：

$$\mathbf{g}(s, a; \theta) \triangleq Q_{\pi}(s, a) \cdot \nabla_{\theta} \ln \pi(a | s; \theta).$$

价值网络 $q(s, a; \theta)$ 是对动作价值函数 $Q_{\pi}(s, a)$ 的近似，所以把上面公式中的 Q_{π} 替换成价值网络，得到近似策略梯度：

$$\widehat{\mathbf{g}}(s, a; \theta) \triangleq \underbrace{q(s, a; \theta)}_{\text{评委的打分}} \cdot \nabla_{\theta} \ln \pi(a | s; \theta). \quad (7.11)$$

最后做梯度上升更新策略网络的参数：

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \beta \cdot \hat{\mathbf{g}}(s, a; \boldsymbol{\theta}). \quad (7.12)$$

注 用上述方式更新参数之后，会让评委打出的分数越来越高，原因是这样的。状态价值函数 $V_\pi(s)$ 可以近似成为：

$$v(s; \boldsymbol{\theta}) = \mathbb{E}_{A \sim \pi(\cdot|s; \boldsymbol{\theta})} [q(s, A; \mathbf{w})].$$

因此可以将 $v(s; \boldsymbol{\theta})$ 看做评委打分的均值。不难证明，公式 7.11 中定义的近似策略梯度 $\hat{\mathbf{g}}(s, a; \boldsymbol{\theta})$ 的期望等于 $v(s; \boldsymbol{\theta})$ 关于 $\boldsymbol{\theta}$ 的梯度；

$$\nabla_{\boldsymbol{\theta}} v(s; \boldsymbol{\theta}) = \mathbb{E}_{A \sim \pi(\cdot|s; \boldsymbol{\theta})} [\hat{\mathbf{g}}(s, A; \boldsymbol{\theta})].$$

因此，用公式 7.12 中的梯度上升更新 $\boldsymbol{\theta}$ ，会让 $v(s; \boldsymbol{\theta})$ 变大，也就是让评委打分的均值更高。

训练价值网络（评委）：通过以上分析，我们不难发现上述训练策略网络（演员）的方法不是真正让演员表现更好，只是让演员更迎合评委的喜好而已。因此，评委的水平也很重要，只有当评委的打分 \hat{q} 真正反映出动作价值 Q_π （也就是回报 U 的期望），演员的水平才能真正提高。初始的时候，价值网络的参数 \mathbf{w} 是随机的，也就是说评委的打分是瞎猜。可以用 SARSA 算法（一种 TD 算法）来提高评委的水平。每次从环境中观测到一个奖励 r ，把 r 看做是真相，用 r 来校准评委的打分。

SARSA 算法的原理是这样的。根据贝尔曼公式，动作价值 $Q_\pi(s_t, a_t)$ 可以写成

$$Q_\pi(s_t, a_t) = \mathbb{E}_{S_{t+1} \sim p(\cdot|s_t, a_t)} \left[\mathbb{E}_{A_{t+1} \sim \pi(\cdot|S_{t+1})} \left[R_t + \gamma \cdot Q_\pi(S_{t+1}, A_{t+1}) \right] \right]$$

由于价值网络 $q(s, a; \mathbf{w})$ 是对动作价值函数 $Q_\pi(s, a)$ 的近似，我们希望价值网络满足这个公式：

$$q(s_t, a_t; \mathbf{w}) = \mathbb{E}_{S_{t+1} \sim p(\cdot|s_t, a_t)} \left[\mathbb{E}_{A_{t+1} \sim \pi(\cdot|S_{t+1})} \left[R_t + \gamma \cdot q(S_{t+1}, A_{t+1}; \mathbf{w}) \right] \right]$$

当实际观测到 r_t, s_{t+1}, a_{t+1} 之后，可以对上面的期望做蒙特卡洛近似，得到

$$\hat{y}_t \triangleq r_t + \gamma \cdot q(s_{t+1}, a_{t+1}; \mathbf{w}),$$

把它叫作 TD 目标。 \hat{y}_t 与 $q(s_t, a_t; \mathbf{w})$ 都是对 t 时刻的状态价值的估计。由于 \hat{y}_t 部分基于实际观测到的奖励 r_t ，我们认为 \hat{y}_t 比 $q(s_t, a_t; \mathbf{w})$ 更接近事实真相。所以把 \hat{y}_t 固定住，鼓励 $q(s_t, a_t; \mathbf{w})$ 去接近 \hat{y}_t 。

SARSA 算法具体这样更新价值网络参数 \mathbf{w} 。定义损失函数：

$$L(\mathbf{w}) \triangleq \frac{1}{2} \left[q(s_t, a_t; \mathbf{w}) - \hat{y}_t \right]^2,$$

设 $\hat{q}_t \triangleq q(s_t, a_t; \mathbf{w})$ 。损失函数的梯度是：

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = (\hat{q}_t - \hat{y}_t) \cdot \nabla_{\mathbf{w}} q(s_t, a_t; \mathbf{w}).$$

做一轮梯度下降更新 \mathbf{w} ：

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \nabla_{\mathbf{w}} L(\mathbf{w}).$$

这样可以让 $q(s_t, a_t; \mathbf{w})$ 更接近 \hat{y}_t 。SARSA 通过这样的方式用观测到的奖励 r_t “校准” 评委的打分 $q(s_t, a_t; \mathbf{w})$ 。

7.5.3 训练流程

最后概括 Actor-Critic 训练流程。设当前策略网络参数是 θ_{now} , 价值网络参数是 \mathbf{w}_{now} 。执行下面的步骤, 将参数更新成 θ_{new} 和 \mathbf{w}_{new} :

1. 观测到当前状态 s_t , 根据策略网络做决策: $a_t \sim \pi(\cdot | s_t; \theta_{\text{now}})$, 并让智能体执行动作 a_t 。
2. 从环境中观测到奖励 r_t 和新的状态 s_{t+1} 。
3. 根据策略网络做决策: $\tilde{a}_{t+1} \sim \pi(\cdot | s_{t+1}; \theta_{\text{now}})$, 但不让智能体执行动作 \tilde{a}_{t+1} 。
4. 让价值网络打分:

$$\hat{q}_t = q(s_t, a_t; \mathbf{w}_{\text{now}}) \quad \text{和} \quad \hat{q}_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \mathbf{w}_{\text{now}})$$

5. 计算 TD 目标和 TD 误差:

$$\hat{y}_t = r_t + \gamma \cdot \hat{q}_{t+1} \quad \text{和} \quad \delta_t = \hat{q}_t - \hat{y}_t.$$

6. 更新价值网络:

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{now}} - \alpha \cdot \delta_t \cdot \nabla_{\mathbf{w}} q(s_t, a_t; \mathbf{w}_{\text{now}}).$$

7. 更新策略网络:

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} + \beta \cdot \hat{q}_t \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta_{\text{now}}).$$

注 此处训练策略网络和价值网络的方法属于同策略 (On-policy), 要求行为策略 (Behavior Policy) 与目标策略 (Target Policy) 相同, 都是最新的策略网络 $\pi(a|s; \theta_{\text{now}})$ 。不能使用经验回放, 因为经验回放数组中的数据是用旧的策略网络 $\pi(a|s; \theta_{\text{old}})$ 获取的, 不能在当前重复利用。

7.5.4 用目标网络改进训练

第 6.2 节讨论了 Q 学习中的自举问题, 这种问题在 SARA 算法中也同样存在。上述训练价值网络的算法是 SARSA, 它存在自举问题——即用价值网络自己的估值 \hat{q}_{t+1} 去更新价值网络自己。缓解自举问题的方法是用目标网络 (Target Network) 计算 TD 目标。把目标网络记作 $q(s, a; \mathbf{w}^-)$, 它的结构与价值网络的结构相同, 但是参数不同。使用目标网络计算 TD 目标, 那么 Actor-Critic 的训练就变成了:

1. 观测到当前状态 s_t , 根据策略网络做决策: $a_t \sim \pi(\cdot | s_t; \theta_{\text{now}})$, 并让智能体执行动作 a_t 。
2. 从环境中观测到奖励 r_t 和新的状态 s_{t+1} 。
3. 根据策略网络做决策: $\tilde{a}_{t+1} \sim \pi(\cdot | s_{t+1}; \theta_{\text{now}})$, 但是不让智能体执行动作 \tilde{a}_{t+1} 。
4. 让价值网络给 (s_t, a_t) 打分:

$$\hat{q}_t = q(s_t, a_t; \mathbf{w}_{\text{now}}).$$

5. 让目标网络给 $(s_{t+1}, \tilde{a}_{t+1})$ 打分:

$$\widehat{q}_{t+1} = q(s_{t+1}, \tilde{a}_{t+1}; \mathbf{w}_{\text{now-}}).$$

6. 计算 TD 目标和 TD 误差:

$$\widehat{y}_t^- = r_t + \gamma \cdot \widehat{q}_{t+1} \quad \text{和} \quad \delta_t = \widehat{q}_t - \widehat{y}_t^-.$$

7. 更新价值网络:

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{now}} - \alpha \cdot \delta_t \cdot \nabla_{\mathbf{w}} q(s_t, a_t; \mathbf{w}_{\text{now}}).$$

8. 更新策略网络:

$$\boldsymbol{\theta}_{\text{new}} \leftarrow \boldsymbol{\theta}_{\text{now}} + \beta \cdot \widehat{q}_t \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t; \boldsymbol{\theta}_{\text{now}}).$$

9. 设 $\tau \in (0, 1)$ 是需要手动调的超参数。做加权平均更新目标网络的参数:

$$\mathbf{w}_{\text{new}}^- \leftarrow \tau \cdot \mathbf{w}_{\text{now}} + (1 - \tau) \cdot \mathbf{w}_{\text{now-}}.$$

相关文献

REINFORCE 方法由 Williams 在 1992 年提出 [22-23]。Actor-Critic 方法在 Barto 等人 1983 年的论文 [24] 中提出。很多论文分析过 Actor-Critic 方法的收敛，比如 [25-29]。策略梯度定理由 Marbach 和 Tsitsiklis 1999 年的论文 [30] 和 Sutton 等人 2000 年的论文 [31] 独立提出。

参考文献

- [1] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.
- [2] MNIH V, KAVUKCUOGLU K, SILVER D, et al. Human-level control through deep reinforcement learning [J]. nature, 2015, 518(7540): 529-533.
- [3] BAIRD L. Residual algorithms: reinforcement learning with function approximation[M]//Machine Learning Proceedings 1995. [S.I.]: Elsevier, 1995: 30-37.
- [4] TSITSIKLIS J N, VAN ROY B. An analysis of temporal-difference learning with function approximation[J]. IEEE transactions on automatic control, 1997, 42(5): 674-690.
- [5] WATKINS C J C H. Learning from delayed rewards[J]. 1989.
- [6] WATKINS C J, DAYAN P. Q-learning[J]. Machine learning, 1992, 8(3-4): 279-292.
- [7] JAAKKOLA T, JORDAN M I, SINGH S P. On the convergence of stochastic iterative dynamic programming algorithms[J]. Neural computation, 1994, 6(6): 1185-1201.
- [8] TSITSIKLIS J N. Asynchronous stochastic approximation and Q-learning[J]. Machine learning, 1994, 16(3): 185-202.
- [9] LIN L J. Reinforcement learning for robots using neural networks[R]. [S.I.]: Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.
- [10] RUMMERY G A, NIRANJAN M. On-line q-learning using connectionist systems: volume 37[M]. [S.I.]: University of Cambridge, Department of Engineering Cambridge, UK, 1994.
- [11] SUTTON R S. Generalization in reinforcement learning: Successful examples using sparse coarse coding[C]// Advances in Neural Information Processing Systems (NIPS). [S.I.: s.n.], 1996.
- [12] SUTTON R S, BARTO A G. Reinforcement learning: An introduction[M]. [S.I.]: MIT press, 2018.
- [13] MNIH V, BADIA A P, MIRZA M, et al. Asynchronous methods for deep reinforcement learning[C]// International Conference on Machine Learning (ICML). [S.I.: s.n.], 2016.
- [14] VAN SEIJEN H. Effective multi-step temporal-difference learning for non-linear function approximation[J]. arXiv preprint arXiv:1608.05151, 2016.
- [15] HESSEL M, MODAYIL J, VAN HASSELT H, et al. Rainbow: Combining improvements in deep reinforcement learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 32. [S.I.: s.n.], 2018.
- [16] SCHAUL T, QUAN J, ANTONOGLOU I, et al. Prioritized experience replay[C]//International Conference on Learning Representations (ICLR). [S.I.: s.n.], 2015.
- [17] VAN HASSELT H. Double q-learning[C]//Advances in Neural Information Processing Systems (NIPS). [S.I.: s.n.], 2010.
- [18] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI conference on artificial intelligence: volume 30. [S.I.: s.n.], 2016.
- [19] WANG Z, SCHAUL T, HESSEL M, et al. Dueling network architectures for deep reinforcement learning[C]// International Conference on Machine Learning (ICML). [S.I.: s.n.], 2016.
- [20] BELLEMARE M G, DABNEY W, MUNOS R. A distributional perspective on reinforcement learning[C]// International Conference on Machine Learning (ICML). [S.I.: s.n.], 2017.
- [21] FORTUNATO M, AZAR M G, PIOT B, et al. Noisy networks for exploration[C]//International Conference on Learning Representations (ICLR). [S.I.: s.n.], 2018.
- [22] WILLIAMS R J. Reinforcement-learning connectionist systems[M]. [S.I.]: College of Computer Science, Northeastern University, 1987.
- [23] WILLIAMS R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning[J]. Machine learning, 1992, 8(3-4): 229-256.
- [24] BARTO A G, SUTTON R S, ANDERSON C W. Neuronlike adaptive elements that can solve difficult learning

- control problems[J]. IEEE transactions on systems, man, and cybernetics, 1983(5): 834-846.
- [25] KONDA V R, TSITSIKLIS J N. Actor-critic algorithms[C]//Advances in Neural Information Processing Systems (NIPS). [S.l.: s.n.], 2000.
- [26] BHATNAGAR S, KUMAR S. A simultaneous perturbation stochastic approximation-based actor-critic algorithm for markov decision processes[J]. IEEE Transactions on Automatic Control, 2004, 49(4): 592-598.
- [27] ABDULLA M S, BHATNAGAR S. Reinforcement learning based algorithms for average cost markov decision processes[J]. Discrete Event Dynamic Systems, 2007, 17(1): 23-52.
- [28] BHATNAGAR S, SUTTON R S, GHAVAMZADEH M, et al. Natural actor-critic algorithms[J]. Automatica, 2009, 45(11): 2471-2482.
- [29] YANG Z, CHEN Y, HONG M, et al. Provably global convergence of actor-critic: A case for linear quadratic regulator with ergodic cost[C]//Advances in Neural Information Processing Systems (NeurIPS). [S.l.: s.n.], 2019: 8353-8365.
- [30] MARBACH P, TSITSIKLIS J N. Simulation-based optimization of markov reward processes: Implementation issues[C]//Proceedings of the 38th IEEE Conference on Decision and Control. [S.l.: s.n.], 1999.
- [31] SUTTON R S, MCALLESTER D A, SINGH S P, et al. Policy gradient methods for reinforcement learning with function approximation[C]//Advances in Neural Information Processing Systems. [S.l.: s.n.], 2000.
- [32] SCHULMAN J, LEVINE S, ABBEEL P, et al. Trust region policy optimization[C]//International Conference on Machine Learning (ICML). [S.l.: s.n.], 2015.
- [33] NOCEDAL J, WRIGHT S. Numerical optimization[M]. [S.l.]: Springer Science & Business Media, 2006.
- [34] CONN A R, GOULD N I, TOINT P L. Trust region methods[M]. [S.l.]: SIAM, 2000.
- [35] BERTSEKAS D P. Constrained optimization and lagrange multiplier methods[M]. [S.l.]: Academic press, 2014.
- [36] BOYD S, BOYD S P, VANDENBERGHE L. Convex optimization[M]. [S.l.]: Cambridge university press, 2004.
- [37] SILVER D, LEVER G, HEESS N, et al. Deterministic policy gradient algorithms[C]//International Conference on Machine Learning (ICML). [S.l.: s.n.], 2014.
- [38] LILlicrap T P, HUNT J J, PRITZEL A, et al. Continuous control with deep reinforcement learning.[C]// International Conference on Learning Representations (ICLR). [S.l.: s.n.], 2016.
- [39] HAFNER R, RIEDMILLER M. Reinforcement learning in feedback control[J]. Machine learning, 2011, 84 (1-2): 137-169.
- [40] PROKHOROV D V, WUNSCH D C. Adaptive critic designs[J]. IEEE transactions on Neural Networks, 1997, 8(5): 997-1007.
- [41] HAUSKNECHT M, STONE P. Deep recurrent Q-learning for partially observable MDPs[C]//AAAI Fall Symposium on Sequential Decision Making for Intelligent Agents. [S.l.: s.n.], 2015.
- [42] RASHID T, SAMVELYAN M, SCHROEDER C, et al. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning[C]//International Conference on Machine Learning (ICML). [S.l.: s.n.], 2018.