

第八章 带基线的策略梯度方法

上一章推导出策略梯度, 并介绍了两种策略梯度方法——REINFORCE 和 Actor-Critic。虽然上一章的方法在理论上是正确的, 但是在实践中效果并不理想。本章介绍带基线的策略梯度 (Policy Gradient with Baseline) 可以大幅提升策略梯度方法的表现。使用基线 (Baseline) 之后, REINFORCE 变成 REINFORCE with Baseline, Actor-Critic 变成 Advantage Actor-Critic (A2C)。

8.1 策略梯度中的基线

首先回顾上一章的内容。策略学习通过最大化目标函数 $J(\theta) = \mathbb{E}_S[V_\pi(S)]$, 得到策略网络 $\pi(a|s; \theta)$, 用于控制智能体。可以用策略梯度 $\nabla_\theta J(\theta)$ 来更新参数 θ :

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

策略梯度定理证明:

$$\nabla_\theta J(\theta) = \mathbb{E}_S \left[\mathbb{E}_{A \sim \pi(\cdot|S; \theta)} \left[Q_\pi(S, A) \cdot \nabla_\theta \ln \pi(A|S; \theta) \right] \right]. \quad (8.1)$$

REINFORCE 和 Actor-Critic 都是通过对策略梯度 $\nabla_\theta J(\theta)$ 做近似得出的; 两种方法区别在于具体如何做近似。

8.1.1 基线 (Baseline)

基于策略梯度公式 (8.1) 得出的 REINFORCE 和 Actor-Critic 方法效果通常不好。但是只需对策略梯度公式 (8.1) 做一个微小的改动, 就能大幅提升表现: 把 b 作为动作价值函数 $Q_\pi(S, A)$ 的基线 (Baseline), 用 $Q_\pi(S, A) - b$ 替换掉 Q_π 。设 b 是任意的函数, 只要不依赖于动作 A 就可以; 例如, b 可以是状态价值函数 $V_\pi(S)$ 。

定理 8.1. 带基线的策略梯度定理

设 b 是任意的函数, 只要不依赖于动作 A 就可以。把 b 作为动作价值函数 $Q_\pi(S, A)$ 的基线, 对策略梯度没有影响:

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

定理 8.1 说明 b 的取值不影响策略梯度的正确性。不论是让 $b = 0$ 还是让 $b = V_\pi(S)$, 对期望的结果毫无影响, 期望的结果都会等于 $\nabla_\theta J(\theta)$ 。其原因在于

$$\mathbb{E}_S \left[\mathbb{E}_{A \sim \pi(\cdot|S; \theta)} \left[b \cdot \nabla_\theta \ln \pi(A|S; \theta) \right] \right] = 0.$$

定理的证明放到第 8.4 节, 对数学感兴趣的读者可以阅读。

定理中的策略梯度表示成了期望的形式, 我们对期望做蒙特卡洛近似。从环境中观

测到一个状态 s ，然后根据策略网络抽样得到 $a \sim \pi(\cdot|s; \theta)$ 。那么策略梯度 $\nabla_{\theta} J(\theta)$ 可以近似为下面的随机梯度：

$$g_b(s, a; \theta) = [Q_{\pi}(S, A) - b] \cdot \nabla_{\theta} \ln \pi(A|S; \theta).$$

虽然 b 的取值对 $\nabla_{\theta} J(\theta)$ 毫无影响。不论 b 的取值是 0 还是 $V_{\pi}(s)$ ，得到的随机梯度 $g_b(s, a; \theta)$ 都是 $\nabla_{\theta} J(\theta)$ 的无偏估计：

$$\text{Bias} = \mathbb{E}_{S,A} [g_b(s, a; \theta)] - \nabla_{\theta} J(\theta) = 0.$$

但是对随机梯度 $g_b(s, a; \theta)$ 是有影响的。用不同的 b ，得到的方差

$$\text{Var} = \mathbb{E}_{S,A} [\|g_b(s, a; \theta) - \nabla_{\theta} J(\theta)\|^2]$$

会有所不同。如果 b 很接近 $Q_{\pi}(s, a)$ 关于 a 的均值，那么方差会比较小。所以 $b = V_{\pi}(s)$ 是很好的基线。

8.1.2 基线的直观解释

策略梯度公式 (8.1) 期望中的 $Q_{\pi}(S, A) \cdot \nabla_{\theta} \ln \pi(A|S; \theta)$ 的意义是什么呢？以图 8.1 中的左图为例。给定状态 s_t ，动作空间是 $\mathcal{A} = \{\text{左}, \text{右}, \text{上}\}$ ，动作价值函数给每个动作打分：

$$Q_{\pi}(s_t, \text{左}) = 80, \quad Q_{\pi}(s_t, \text{右}) = -20, \quad Q_{\pi}(s_t, \text{上}) = 180.$$

这些分值会乘到梯度 $\nabla_{\theta} \ln \pi(A|S; \theta)$ 上，在做完梯度上升之后，新的策略会倾向于分值高的动作。

- 动作价值 $Q_{\pi}(s_t, \text{上}) = 180$ 很大，说明基于状态 s_t 选择动作“上”是很好的决策。让梯度 $\nabla_{\theta} \ln \pi(\text{上}|s_t; \theta)$ 乘以大的系数 $Q_{\pi}(s_t, \text{上}) = 180$ ，那么做梯度上升更新 θ 之后，会让 $\pi(\text{上}|s_t; \theta)$ 变大，在状态 s_t 的情况下更倾向于动作“上”。
- 相反， $Q_{\pi}(s_t, \text{右}) = -20$ 说明基于状态 s_t 选择动作“右”是糟糕的决策。让梯度 $\nabla_{\theta} \ln \pi(\text{右}|s_t; \theta)$ 乘以负的系数 $Q_{\pi}(s_t, \text{右}) = -20$ ，那么做梯度上升更新 θ 之后，会让 $\pi(\text{右}|s_t; \theta)$ 变小，在状态 s_t 的情况下选择动作“右”的概率更小。

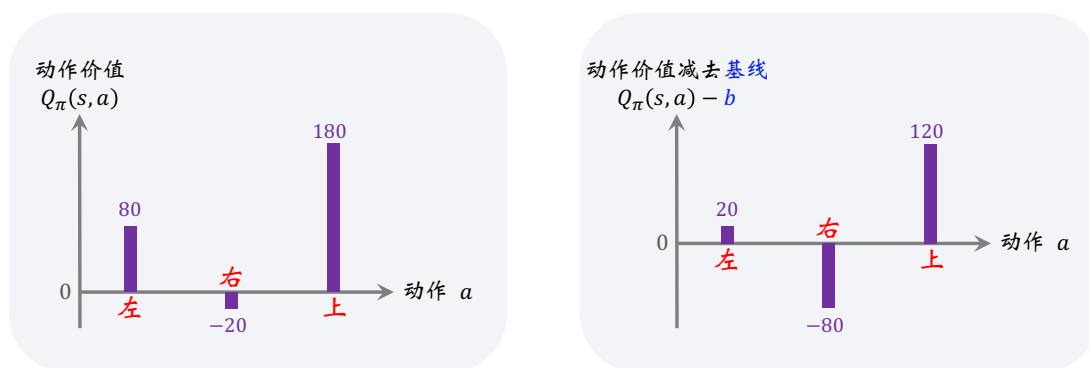


图 8.1: 动作空间是 $\mathcal{A} = \{\text{左}, \text{右}, \text{上}\}$ 。设状态 s_t 是给定的。左图纵轴表示动作价值 $Q_{\pi}(s_t, a_t)$ 。右图纵轴表示动作价值减去基线 $Q_{\pi}(s_t, a_t) - b$ ，其中基线 $b = 60$ 。

根据上述分析，我们在乎的是动作价值 $Q_\pi(s_t, \text{左})$ 、 $Q_\pi(s_t, \text{右})$ 、 $Q_\pi(s_t, \text{上})$ 三者的相对大小，而非绝对大小。如果给三者都减去 $b = 60$ ，那么三者的相对大小是不变的；动作“上”仍然是最好的，动作“右”仍然是最差的。见图 8.1 中的右图。因此

$$\left[Q_\pi(s_t, a_t) - b \right] \cdot \nabla_{\theta} \ln \pi(A | S; \theta)$$

依然能指导 θ 做调整，使得 $\pi(\text{上} | s_t; \theta)$ 变大，而 $\pi(\text{右} | s_t; \theta)$ 变小。

8.2 带基线的 REINFORCE 算法

上一节推导出了带基线的策略梯度，并且对策略梯度做了蒙特卡洛近似。本节中，我们使用状态价值 $V_\pi(s)$ 作基线，得到策略梯度的一个无偏估计：

$$g(s, a; \theta) = [Q_\pi(s, a) - V_\pi(s)] \cdot \nabla_\theta \ln \pi(a | s; \theta).$$

我们在第 7.4 节中学过 REINFORCE，它使用实际观测的回报 u 来代替动作价值 $Q_\pi(s, a)$ 。此处我们同样用 u 代替 $Q_\pi(s, a)$ 。此外，我们还用一个神经网络 $v(s; \mathbf{w})$ 近似状态价值函数 $V_\pi(s)$ 。这样一来， $g(s, a; \theta)$ 就被近似成了：

$$\tilde{g}(s, a; \theta) = [u - v(s; \mathbf{w})] \cdot \nabla_\theta \ln \pi(a | s; \theta).$$

可以用 $\tilde{g}(s, a; \theta)$ 作为策略梯度 $\nabla_\theta J(\theta)$ 的近似，更新策略网络参数：

$$\theta \leftarrow \theta + \beta \cdot \tilde{g}(s, a; \theta)$$

8.2.1 策略网络和价值网络

带基线的 REINFORCE 需要两个神经网络：策略网络 $\pi(a|s; \theta)$ 和价值网络 $v(s; \mathbf{w})$ ；神经网络结构如图 8.2 和 8.3 所示。策略网络与之前章节的完全一样：输入是状态 s ，输出是一个向量，每个元素表示一个动作的概率。

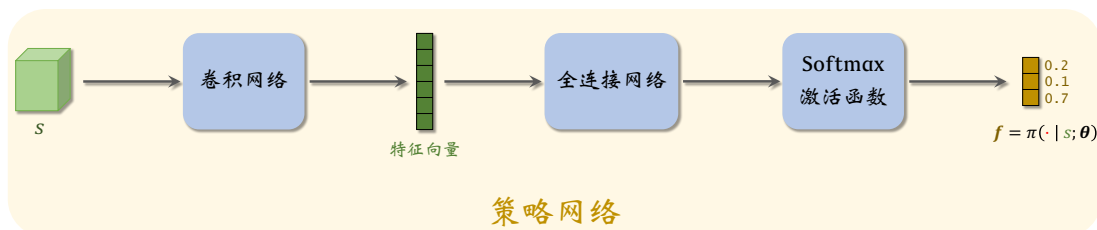


图 8.2: 策略网络 $\pi(a|s; \theta)$ 的神经网络结构。输入是状态 s ，输出是动作空间中每个动作的概率值。举个例子，动作空间是 $\mathcal{A} = \{\text{左}, \text{右}, \text{上}\}$ ，策略网络的输出是三个概率值： $\pi(\text{左}|s; \theta) = 0.2$ ， $\pi(\text{右}|s; \theta) = 0.1$ ， $\pi(\text{上}|s; \theta) = 0.7$ 。

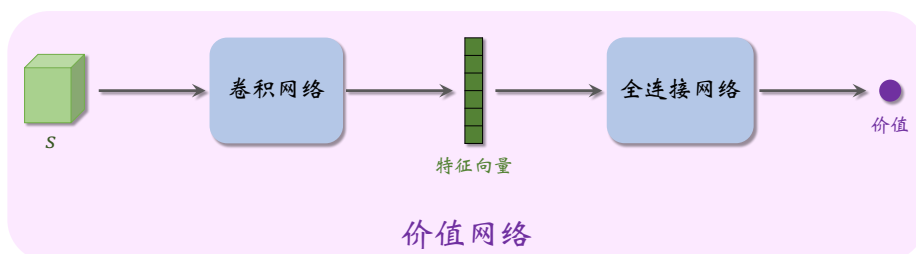


图 8.3: 价值网络 $v(s; \mathbf{w})$ 的结构。输入是状态 s ；输出是状态的价值。

价值网络 $v(s; \mathbf{w})$ 与之前使用的价值网络 $q(s, a; \mathbf{w})$ 区别较大。此处的 $v(s; \mathbf{w})$ 是对状态价值 V_π 的近似，而非对动作价值 Q_π 的近似。 $v(s; \mathbf{w})$ 的输入是状态 s ，输出是一个

实数，作为基线。策略网络和价值网络的输入都是状态 s ，因此可以让两个神经网络共享卷积网络的参数，这是编程实现中常用的技巧。

虽然带基线的 REINFORCE 有一个策略网络和一个价值网络，但是这种方法不是 Actor-Critic。价值网络没有起到“评委”的作用；真正帮助策略网络（演员）改进参数 θ （演员的演技）的不是价值网络，而是实际观测到的回报 u 。价值网络只是作为基线而已。

8.2.2 算法的推导

训练策略网络的方法是近似的策略梯度上升。从 t 时刻开始，智能体完成一局游戏，观测到全部奖励 r_t, r_{t+1}, \dots, r_n ，然后计算回报 $u_t = \sum_{k=t}^n \gamma^{k-t} \cdot r_k$ 。让价值网络做出预测： $\hat{v}_t = v(s_t; \mathbf{w})$ ，作为基线。这样就得到了带基线的策略梯度：

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

它是策略梯度 $\nabla_{\theta} J(\theta)$ 的近似。最后做梯度上升更新 θ ：

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

这样可以让目标函数 $J(\theta)$ 逐渐增大。

训练价值网络的方法是回归 (Regression)。回忆一下，状态价值是回报的期望：

$$V_{\pi}(s_t) = \mathbb{E}[U_t | S_t = s_t],$$

期望消掉了动作 A_t, A_{t+1}, \dots, A_n 和状态 S_{t+1}, \dots, S_n 。训练价值网络的目的是让 $v(s_t; \mathbf{w})$ 接近 u_t 的期望。定义损失函数：

$$L(\mathbf{w}) = \frac{1}{2} [v(s_t; \mathbf{w}) - u_t]^2.$$

设 $\hat{v}_t = v(s_t; \mathbf{w})$ 。损失函数的梯度是：

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = (\hat{v}_t - u_t) \cdot \nabla_{\mathbf{w}} v(s_t; \mathbf{w}).$$

做一次梯度下降更新 \mathbf{w} ：

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

8.2.3 训练流程

当前策略网络的参数是 θ_{now} ，价值网络的参数是 \mathbf{w}_{now} 。执行下面的步骤，对参数做一轮更新。

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. 让价值网络做预测：

$$\hat{v}_t = v(s_t; \mathbf{w}_{\text{now}}), \quad \forall t = 1, \dots, n.$$

4. 计算误差 $\delta_t = \hat{v}_t - u_t$, $\forall t = 1, \dots, n$ 。

5. 用 $\{s_t\}_{t=1}^n$ 作为数据, 做反向传播计算:

$$\nabla_{\mathbf{w}} v(s_t; \mathbf{w}_{\text{now}}), \quad \forall t = 1, \dots, n.$$

6. 更新价值网络参数:

$$\mathbf{w}_{\text{new}} \leftarrow \mathbf{w}_{\text{now}} - \alpha \cdot \sum_{t=1}^n \delta_t \cdot \nabla_{\mathbf{w}} v(s_t; \mathbf{w}_{\text{now}}).$$

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

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

8. 做随机梯度上升更新策略网络参数:

$$\boldsymbol{\theta}_{\text{new}} \leftarrow \boldsymbol{\theta}_{\text{now}} + \beta \cdot \sum_{t=1}^n \gamma^{t-1} \cdot \underbrace{\delta_t \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t; \boldsymbol{\theta}_{\text{now}})}_{\text{即近似梯度 } \hat{\mathbf{g}}(s_t, a_t; \boldsymbol{\theta}_{\text{now}})}.$$

8.3 Advantage Actor-Critic (A2C)

之前我们推导出了带基线的策略梯度，并且对策略梯度做了蒙特卡洛近似，得到得到策略梯度的一个无偏估计：

$$\mathbf{g}(s, a; \boldsymbol{\theta}) = \left[\underbrace{Q_{\pi}(s, a) - V_{\pi}(s)}_{\text{优势函数}} \right] \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a | s; \boldsymbol{\theta}). \quad (8.2)$$

公式中的 $Q_{\pi} - V_{\pi}$ 被称作优势函数 (Advantage Function)。因此，基于上面公式得到的 Actor-Critic 方法被称为 Advantage Actor-Critic，缩写 A2C。

A2C 属于 Actor-Critic 方法。有一个策略网络 $\pi(a|s; \boldsymbol{\theta})$ ，相当于演员，用于控制智能体运动。还有一个价值网络 $v(s; \mathbf{w})$ ，相当于评委，他的评分可以帮助策略网络（演员）改进技术。两个神经网络的结构与上一节中的完全相同，但是本节和上一节用不同的方法训练两个神经网络。

8.3.1 算法推导

训练价值网络： 训练价值网络 $v(s; \mathbf{w})$ 的算法是从贝尔曼公式来的：

$$V_{\pi}(s_t) = \mathbb{E}_{A_t \sim \pi(\cdot | s_t; \boldsymbol{\theta})} \left[\mathbb{E}_{S_{t+1} \sim p(\cdot | s_t, A_t)} \left[R_t + \gamma \cdot V_{\pi}(S_{t+1}) \right] \right].$$

我们对贝尔曼方程左右两边做近似：

- 方程左边的 $V_{\pi}(s_t)$ 可以近似成 $v(s_t; \mathbf{w})$ 。 $v(s_t; \mathbf{w})$ 是价值网络在 t 时刻对 $V_{\pi}(s_t)$ 做出的估计。
- 方程右边的期望是关于当前时刻动作 A_t 与下一时刻状态 S_{t+1} 求的。给定当前状态 s_t ，智能体执行动作 a_t ，环境会给出奖励 r_t 和新的状态 s_{t+1} 。用观测到的 r_t 、 s_{t+1} 对期望做蒙特卡洛近似，得到：

$$r_t + \gamma \cdot V_{\pi}(s_{t+1}). \quad (8.3)$$

- 进一步把公式 (8.3) 中的 $V_{\pi}(s_{t+1})$ 近似成 $v(s_{t+1}; \mathbf{w})$ ，得到

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

把它称作 TD 目标。它是价值网络在 $t+1$ 时刻对 $V_{\pi}(s_t)$ 做出的估计。

$v(s_t; \mathbf{w})$ 和 \hat{y}_t 都是对动作价值 $V_{\pi}(s_t)$ 的估计。由于 \hat{y}_t 部分基于真实观测到的奖励 r_t ，我们认为 \hat{y}_t 比 $v(s_t; \mathbf{w})$ 更可靠。所以把 \hat{y}_t 固定住，鼓励 $v(s_t; \mathbf{w})$ 更接近 \hat{y}_t 。

具体这样更新价值网络参数 \mathbf{w} 。定义损失函数

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

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

$$\nabla_{\mathbf{w}} L(\mathbf{w}) = \underbrace{(\hat{v}_t - \hat{y}_t)}_{\text{TD 误差 } \delta_t} \cdot \nabla_{\mathbf{w}} v(s_t; \mathbf{w}).$$

定义 TD 误差为 $\delta_t \triangleq \hat{v}_t - \hat{y}_t$ 。做一轮梯度下降更新 \mathbf{w} :

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \cdot \delta_t \cdot \nabla_{\mathbf{w}} v(s_t; \mathbf{w}).$$

这样可以让价值网络的预测 $v(s_t; \mathbf{w})$ 更接近 \hat{y}_t 。

训练策略网络: A2C 从公式 (8.2) 出发, 对 $g(s, a; \theta)$ 做近似, 记作 \tilde{g} , 然后用 \tilde{g} 更新策略网络参数 θ 。下面我们做数学推导。回忆一下贝尔曼公式:

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

把近似策略梯度 $g(s_t, a_t; \theta)$ 中的 $Q_{\pi}(s_t, a_t)$ 替换成上面的期望, 得到:

$$\begin{aligned} g(s_t, a_t; \theta) &= [Q_{\pi}(s_t, a_t) - V_{\pi}(s_t)] \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta) \\ &= [\mathbb{E}_{S_{t+1}} [R_t + \gamma \cdot V_{\pi}(S_{t+1})] - V_{\pi}(s_t)] \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta). \end{aligned}$$

当智能体执行动作 a_t 之后, 环境给出新的状态 s_{t+1} 和奖励 r_t ; 利用 s_{t+1} 和 r_t 对上面的期望做蒙特卡洛近似, 得到:

$$g(s_t, a_t; \theta) \approx [r_t + \gamma \cdot V_{\pi}(s_{t+1}) - V_{\pi}(s_t)] \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta).$$

进一步把状态价值函数 $V_{\pi}(s)$ 替换成价值网络 $v(s; \mathbf{w})$, 得到:

$$\tilde{g}(s_t, a_t; \theta) \triangleq \underbrace{[r_t + \gamma \cdot v(s_{t+1}; \mathbf{w}) - v(s_t; \mathbf{w})]}_{\text{TD 目标 } \hat{y}_t} \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta).$$

前面定义了 TD 目标和 TD 误差:

$$\hat{y}_t \triangleq r_t + \gamma \cdot v(s_{t+1}; \mathbf{w}) \quad \text{和} \quad \delta_t \triangleq v(s_{t+1}; \mathbf{w}) - \hat{y}_t.$$

因此, 可以把 \tilde{g} 写成:

$$\tilde{g}(s_t, a_t; \theta) \triangleq -\delta_t \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta).$$

\tilde{g} 是 g 的近似, 所以也是策略梯度 $\nabla_{\theta} J(\theta)$ 的近似。用 \tilde{g} 更新策略网络参数 θ :

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

这样可以让目标函数 $J(\theta)$ 变大。

策略网络与价值网络的关系: A2C 中策略网络 (演员) 和价值网络 (评委) 的关系如图 8.4 所示。智能体由策略网络 π 控制, 与环境交互, 并收集状态、动作、奖励。策略网络 (演员) 基于状态 s_t 做出动作 a_t 。价值网络 (评委) 基于 s_t 、 s_{t+1} 、 r_t 算出 TD 误差 δ_t 。策略网络 (演员) 依靠 δ_t 来判断自己动作的好坏, 从而改进自己的演技 (即参数 θ)。

读者可能会有疑问: 价值网络 v 只知道两个状态 s_t 、 s_{t+1} , 而并不知道动作 a_t , 那么价值网络为什么能评价 a_t 的好坏呢? 价值网络 v 告诉策略网络 π 的唯一信息是 δ_t 。回顾一下 δ_t 的定义:

$$-\delta_t = \underbrace{r_t + \gamma \cdot v(s_{t+1}; \mathbf{w})}_{\text{TD 目标 } \hat{y}_t} - \underbrace{v(s_t; \mathbf{w})}_{\text{基线}}.$$

基线 $v(s_t; \mathbf{w})$ 是价值网络在 t 时刻对 $\mathbb{E}[U_t]$ 的估计; 此时智能体尚未执行动作 a_t 。而 TD 目标 \hat{y}_t 是价值网络在 $t+1$ 时刻对 $\mathbb{E}[U_t]$ 的估计; 此时智能体已经执行动作 a_t 。

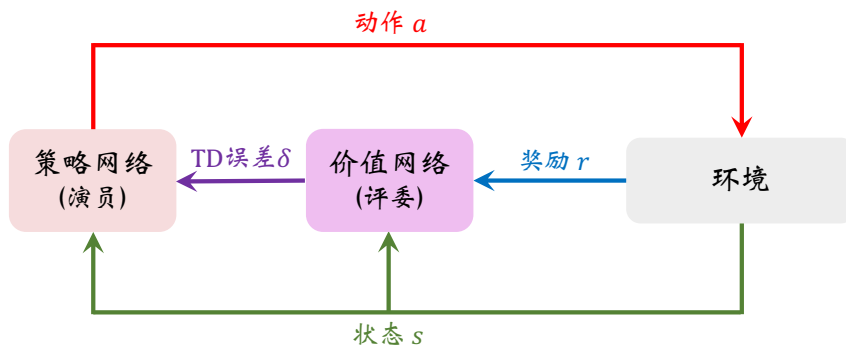


图 8.4: A2C 中策略网络（演员）和价值网络（评委）的关系图。

- 如果 $\hat{y}_t > v(s_t; \mathbf{w})$, 说明动作 a_t 很好, 使得奖励 r_t 超出预期, 或者新的状态 s_{t+1} 比预期好; 这种情况下应该更新 θ , 使得 $\pi(a_t|s_t; \theta)$ 变大。
- 如果 $\hat{y}_t < v(s_t; \mathbf{w})$, 说明动作 a_t 不好, 导致奖励 r_t 不及预期, 或者新的状态 s_{t+1} 比预期差; 这种情况下应该更新 θ , 使得 $\pi(a_t|s_t; \theta)$ 减小。

综上所述, δ_t 中虽然不包含动作 a_t , 但是 δ_t 可以间接反映出动作 a_t 的好坏, 可以帮助策略网络（演员）改进演技。

8.3.2 训练流程

下面概括 A2C 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. 让价值网络打分:

$$\hat{v}_t = v(s_t; \mathbf{w}_{\text{now}}) \quad \text{和} \quad \hat{v}_{t+1} = v(s_{t+1}; \mathbf{w}_{\text{now}})$$

4. 计算 TD 目标和 TD 误差:

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

5. 更新价值网络:

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

6. 更新策略网络:

$$\theta_{\text{new}} \leftarrow \theta_{\text{now}} - \beta \cdot \delta_t \cdot \nabla_{\theta} \ln \pi(a_t | s_t; \theta_{\text{now}}).$$

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。

8.3.3 用目标网络改进训练

上述训练价值网络的算法存在自举问题——即用价值网络自己的估值 \hat{v}_{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 $v(s; \mathbf{w}^-)$ ，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 A2C 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： $a_t \sim \pi(\cdot | s_t; \boldsymbol{\theta}_{\text{now}})$ ，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. 让价值网络给 s_t 打分：

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

4. 让目标网络给 s_{t+1} 打分：

$$\hat{v}_{t+1}^- = v(s_{t+1}; \mathbf{w}_{\text{now}}^-).$$

5. 计算 TD 目标和 TD 误差：

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

6. 更新价值网络：

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

7. 更新策略网络：

$$\boldsymbol{\theta}_{\text{new}} \leftarrow \boldsymbol{\theta}_{\text{now}} - \beta \cdot \delta_t \cdot \nabla_{\boldsymbol{\theta}} \ln \pi(a_t | s_t; \boldsymbol{\theta}_{\text{now}}).$$

8. 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：

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

8.4 证明带基线的策略梯度定理

本节证明带基线的策略梯度定理 8.1。将定理 7.1 与引理 8.2 相结合，即可证得定理 8.1。

引理 8.2

设 b 是任意函数，但是不能依赖于 A 。那么对于任意的 s ,

$$\mathbb{E}_{A \sim \pi(\cdot|s;\theta)} \left[b \cdot \frac{\partial \ln \pi(A|s;\theta)}{\partial \theta} \right] = 0.$$



证明 由于基线 b 不依赖于动作 A ，可以把 b 提取到期望外面：

$$\begin{aligned} \mathbb{E}_{A \sim \pi(\cdot|s;\theta)} \left[b \cdot \frac{\partial \ln \pi(A|s;\theta)}{\partial \theta} \right] &= b \cdot \mathbb{E}_{A \sim \pi(\cdot|s;\theta)} \left[\frac{\partial \ln \pi(A|s;\theta)}{\partial \theta} \right] \\ &= b \cdot \sum_{a \in \mathcal{A}} \pi(a|s;\theta) \cdot \frac{\partial \ln \pi(a|s;\theta)}{\partial \theta} \\ &= b \cdot \sum_{a \in \mathcal{A}} \pi(a|s;\theta) \cdot \frac{1}{\pi(a|s;\theta)} \cdot \frac{\partial \pi(a|s;\theta)}{\partial \theta} \\ &= b \cdot \sum_{a \in \mathcal{A}} \frac{\partial \pi(a|s;\theta)}{\partial \theta}. \end{aligned}$$

上式最右边的连加是关于 a 求的，而偏导是关于 θ 求的，因此可以把连加放入偏导内部：

$$\mathbb{E}_{A \sim \pi(\cdot|s;\theta)} \left[b \cdot \frac{\partial \ln \pi(A|s;\theta)}{\partial \theta} \right] = b \cdot \frac{\partial}{\partial \theta} \underbrace{\sum_{a \in \mathcal{A}} \pi(a|s;\theta)}_{\text{恒等于 1}}.$$

因此

$$\mathbb{E}_{A \sim \pi(\cdot|s;\theta)} \left[b \cdot \frac{\partial \ln \pi(A|s;\theta)}{\partial \theta} \right] = b \cdot \frac{\partial 1}{\partial \theta} = 0.$$

□