强化学习原理

第十三章读书笔记

姓名：石若川 学号：2111381 专业：智能科学与技术

13 策略梯度方法

此前学习的强化学习方法基本上是基于行为值函数的方法，而本章介绍的方法则直接学习参数化的策略。

- 基于动作价值函数的方法：先学习动作价值函数，然后根据估计的动作价值函数选择动作；如果没有动作价值函数的估计，策略也就不会存在。
- 直接学习参数化策略的方法：价值函数仍然可以用于学习策略的参数，但是动作选择不再直接依赖于价值函数。

引入记号 $\theta \in \mathbb{R}^{d'}$ 表示策略的参数向量。把在 t 时刻、状态 s 和参数 θ 下选择动作 a 的概率记为 $\pi(a|s,\theta) = \Pr\{A_t = a \mid S_t = s, \theta_t = \theta\}$。如果也使用估计的价值函数，那么价值函数的参数像 $\hat{v}(s,\mathbf{w})$ 中一样用 $\mathbf{w} \in \mathbb{R}^d$ 表示。

策略参数的学习方法都基于某种性能度量 $J(\theta)$ 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，所以它们的更新近似于 J 的梯度上升

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla J(\boldsymbol{\theta}_t)} \tag{1}$$

其中，$\widehat{\nabla J(\theta_t)} \in \mathbb{R}^{d'}$ 是一个随机估计，它的期望是性能指标对它的参数 θ_t 的梯度的近似。把所有符合这个框架的方法都称为策略梯度法。

学习策略和价值函数的方法一般被称为 Actor-Critic 方法，其中 actor 用于学习策略，critic 用于学习值函数。

13.1 策略近似及其优势

1. 策略参数化

在策略梯度方法中，策略可以用任意的方式参数化，只要 $\pi(\alpha|s,\theta)$ 对参数可导，即只要对于所有的 $s \in S, a \in \mathcal{A}(s)$ 和 $\theta \in \mathbb{R}^{d'}$, $\nabla\pi(a|s,\theta)$($\pi(a|s,\theta)$ 对参数 θ 的偏导组成的列向量) 存在且是有限的。在实际中，为了便于探索，一般会要求策略永远不会变成确定的 (即对于所有 s、a、θ, $\pi(a|s,\theta) \in (0,1)$)。

如果动作空间是离散的并且不是特别大，自然的参数化方法是对每一个“状态-动作”二元组估计一个参数化的数值偏好 $h(s,a,\theta) \in \mathbb{R}$。在每个状态下拥有最大偏好值的动作被选择的概率也最大，例如，可以根据一个指数柔性最大化 (softmax) 分布

$$\pi(a|s,\boldsymbol{\theta}) \doteq \frac{e^{h(s,a,\boldsymbol{\theta})}}{\sum_b e^{h(s,b,\boldsymbol{\theta})}} \tag{2}$$

这种形式的策略参数化称为动作偏好值的柔性最大化。

这些动作偏好值可以被任意地参数化。例如，它们可以用神经网络表示，θ 是网络连接权重的向量, 或者可以是特征的简单线性组合

$$h(s,a,\boldsymbol{\theta}) = \boldsymbol{\theta}^\top \mathbf{x}(s,a)$$

2. 优势

根据偏好柔性最大化分布选择动作的优势在于：

(1) 近似策略可以接近于一个确定策略。动作价值的估计值会收敛于对应的真实值，而这些真实值之间的差异是有限的。如果在柔性最大化分布中引入一个温度参数，然后这个温度参数可以随时间逐步减小接近一个确定值，但是在实际中如果没有比简单假设更多的关于真实动作价值的先验知识，则很难确定减小的流程，甚至初始温度都很难确定。

(2) 可以以任意的概率来选择动作。在有重要函数近似的问题中，最好的近似策略可能是一个随机策略。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。

(3) 策略可以用更简单的函数近似。策略和动作价值函数的复杂度因问题而异。对于一些情况，动作价值函数更简单，更容易近似。而对于其他一些情况，策略更简单。在后一种情况中，基于策略的方法一般学习得更快，并且得到更好的渐近策略。

(4) 策略参数化形式的选择有时是引入理想中的策略形式的先验知识的一个好方法。

13.2 策略梯度定理

考虑分幕式情况，将性能指标定义为幕初始状态的价值。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

$$J(\boldsymbol{\theta}) \doteq v_{\pi_{\boldsymbol{\theta}}}(s_0),$$

其中，$v_{\pi\rho}$ 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 (式1中所需要的), 其中没有涉及对状态分布的求导。在分幕式情况下策略梯度定理表达式如下

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta}) \tag{3}$$

其中，这个梯度是关于参数向量 θ 每个元素的偏导组成的列向量，π 表示参数向量 θ 对应的策略。符号 $\propto$ 表示 “正比于”。在分幕式情况下，这个比例常量是幕的平均长度；在持续性的情况下，这个常量是 1, 这种情况下应该是等式。这里的分布 μ 是在策略 π 下的同轨策略分布。下面给出分幕式情况下的策略梯度定理证明。

$$
\begin{aligned}
\nabla v_\pi(s) &= \nabla \left[\sum_a \pi(a|s) q_\pi(s,a) \right], \text{ 对所有} s \in \mathcal{S} \\
&= \sum_a \left[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla q_\pi(s,a) \right] \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \nabla \sum_{s',r} p(s',r|s,a) \big(r + v_\pi(s') \big) \Big] \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \nabla v_\pi(s') \Big] \\
&= \sum_a \Big[\nabla \pi(a|s) q_\pi(s,a) + \pi(a|s) \sum_{s'} p(s'|s,a) \sum_{a'} \big[\nabla \pi(a'|s') q_\pi(s',a') + \pi(a'|s') \sum_{s''} p(s''|s',a') \nabla v_\pi(s'') \big] \big] \\
&= \sum_{x \in \mathcal{S}} \sum_{k=0}^{\infty} \Pr(s \to x, k, \pi) \sum_a \nabla \pi(a|x) q_\pi(x,a)
\end{aligned}
$$

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &= \nabla v_\pi(s_0) \\
&= \sum_s \left(\sum_{k=0}^{\infty} \Pr(s_0 \to s, k, \pi) \right) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_s \eta(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \frac{\eta(s)}{\sum_{s'} \eta(s')} \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&= \sum_{s'} \eta(s') \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a) \\
&\propto \sum_s \mu(s) \sum_a \nabla \pi(a|s) q_\pi(s,a)
\end{aligned}
$$

13.3 REINFORCE：蒙特卡洛策略梯度

式3右边是将目标策略 π 下每个状态出现的频率作为加权系数的求和项，如果按策略 π 执行，则状态将按这个比例出现。因此

$$
\begin{aligned}
\nabla J(\boldsymbol{\theta}) &\propto \sum_s \mu(s) \sum_a q_\pi(s,a) \nabla \pi(a|s,\boldsymbol{\theta}) \\
&= \mathbb{E}_\pi \left[\sum_a q_\pi(S_t,a) \nabla \pi(a|S_t,\boldsymbol{\theta}) \right] \qquad (4)
\end{aligned}
$$

将随机梯度上升算法 (式1) 实例化为

$$
\theta_{t+1} \doteq \theta_t + \alpha \sum_a \hat{q}(S_t,a,\mathbf{w}) \nabla \pi(a|S_t,\boldsymbol{\theta}) \qquad (5)
$$

这里 $\hat{q}$ 是由学习得到的 q_π 的近似。这个算法被称为全部动作算法，因为它的更新涉及了所有可能的动作。

从4继续推导可以得到

$$\begin{aligned}\nabla J(\boldsymbol{\theta}) &= \mathbb{E}_\pi\left[\sum_a \pi(a|S_t,\boldsymbol{\theta}) q_\pi(S_t,a)\frac{\nabla\pi(a|S_t,\boldsymbol{\theta})}{\pi(a|S_t,\boldsymbol{\theta})}\right] \\ &= \mathbb{E}_\pi\left[q_\pi(S_t,A_t)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right] \\ &= \mathbb{E}_\pi\left[G_t\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta})}{\pi(A_t|S_t,\boldsymbol{\theta})}\right]\end{aligned}$$

将其代入式1可以得到

$$\theta_{t+1} \doteq \theta_t + \alpha G_t \frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)} \tag{6}$$

这个算法被称为 REINFORCE 。每一个增量更新都正比于回报 G_t 和一个向量的乘积，这个向量是选取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。

具体算法如下框所示，注意伪代码与式6略有不同。式6中向量 $\frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$ 的一种更紧凑的表达方式 $\nabla \ln\pi(A_t|S_t,\theta_t)$。另外，伪代码中引入了带折扣的情况。

REINFORCE：π_* 的模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 $\theta \in \mathbb{R}^{d'}$(如初始化为 0)
无限循环 (对于每一幕):
  根据 $\pi(\cdot|\cdot,\theta)$, 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，$t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\theta \leftarrow \theta + \alpha\gamma^t G \nabla \ln\pi(A_t|S_t,\theta)$

13.4 带有基线的 REINFORCE

可以将策略梯度定理 (式 3) 进行推广，在其中加入任意一个与动作价值函数进行对比的基线 $b(s)$

$$\nabla J(\boldsymbol{\theta}) \propto \sum_s \mu(s) \sum_a \Big(q_\pi(s,a) - b(s)\Big)\nabla\pi(a|s,\boldsymbol{\theta}) \tag{7}$$

这个基线可以是任意函数，只要不随动作 a 变化，上述等式仍然成立，这是因为有

$$\sum_a b(s)\nabla\pi(a|s,\boldsymbol{\theta}) = b(s)\nabla\sum_a \pi(a|s,\boldsymbol{\theta}) = b(s)\nabla 1 = 0.$$

可以使用式7所示的策略梯度定理推导出包含基线的新的 REINFORCE 版本

$$\boldsymbol{\theta}_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big(G_t - b(S_t)\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \tag{8}$$

一般，加入这个基线不会使更新值的期望发生变化，但是可以减小方差。状态价值函数 $\hat{v}(S_t, \mathbf{w})$ 就是一个比较自然地能想到的基线，其中，$\mathrm{w} \in \mathbb{R}^{\mathrm{d}}$ 是权重向量。REINFORCE 使用蒙特卡洛方法学习策略参数 θ，很自然地也可以使用蒙特卡洛方法学习状态价值函数的权重 w。用学习到的状态价值函数作为基线的 REINFORCE 算法的完整伪代码在下面框中给出。

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 $\pi(a|s, \theta)$
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 $\alpha^\theta > 0$, $\alpha^{\ddot{W}} > 0$
初始化策略参数 $\theta \in \mathbb{R}^{d'}$ 和状态价值函数的权重 $\mathrm{w} \in \mathbb{R}^d$（如初始化为 O）
无限循环 (对于每一幕):
  根据 $\pi(\cdot|\cdot, \theta)$, 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，$t = 0, 1, \ldots, T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k$
    $\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathrm{w}} \delta \nabla \hat{v}(S_t, \mathbf{w})$
    $\theta \leftarrow \theta + \alpha^\theta \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$

这个算法有两个步长，记为 α^θ 和 α^{w}(其中 α^θ 是式 8中的 α)。价值函数的步长 α^{w} 相对比较容易设置。但是对如何设置策略参数更新的步长 α^θ 还不清楚。它取决于收益变化的范围以及策略参数化形式。

13.5 Actor-Critic 方法

1. 区分带基线的强化学习方法与 Actor-Critic 方法

尽管带基线的强化学习方法既学习了一个策略函数也学习了一个状态价值函数，也不认为它是一种 “Actor-Critic” 方法，因为它的状态价值函数仅被用作基线，而不是作为一个 “Critic”。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基线。只有采用自举法时，才会出现依赖于函数逼近质量的偏差和渐近性收敛。通过自举法引入的偏差以及状态表示上的依赖经常是很有用的，因为它们降低了方差并加快了学习。带基线的强化学习方法是无偏差的，并且会渐近地收敛至局部最小值，但是和所有的蒙特卡洛方法一样，它的学习比较缓慢 (产生高方差估计)，并且不便于在线实现，或者不便于应用于持续性问题。使用时序差分可以消除这些不便，并且通过使用多步方法，可以灵活地选择自举操作的程度。

2. 单步 Actor-Critic 方法

首先考虑单步 “Actor-Critic” 方法，它和时序差分法很类似，如 TD(0)、Sarsa(0) 和 Q-learning。单步方法的主要优势在于它们是完全在线和增量式的，同时也避免了使用资格迹的复杂性。单步 “Actor-

Critic" 方法使用单步回报来代替 REINFORCE 算法 (式8) 中的整个回报，具体如下

$$\theta_{t+1} \doteq \boldsymbol{\theta}_t + \alpha\Big(G_{t:t+1} - \hat{v}(S_t, \mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \tag{9}$$

$$= \boldsymbol{\theta}_t + \alpha\Big(R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})\Big)\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \tag{10}$$

$$= \boldsymbol{\theta}_t + \alpha\delta_t\frac{\nabla\pi(A_t|S_t,\boldsymbol{\theta}_t)}{\pi(A_t|S_t,\boldsymbol{\theta}_t)} \tag{11}$$

可以采用半梯度方法 TD(0) 来学习状态价值函数。完整算法的伪代码在下框中给出。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使用，之后都不会再次使用。

单步 "Actor-Critic" 方法 (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 $\alpha^{\boldsymbol{\theta}} > 0$，$\alpha^{\boldsymbol{W}} > 0$
初始化策略参数 $\theta \in \mathbb{R}^{d'}$ 和状态价值函数的权重 $\mathbf{w} \in \mathbb{R}^d$ (如初始化为 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 $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla\hat{v}(S,\mathbf{w})$
    $I \leftarrow \gamma I$
    $S + S'$

3. 带资格迹的 Actor-Critic 方法

单步的 Actor-Critic 方法很容易能够推广到 n-step 方法的前向视图中，并推广到 $\gamma-return$ 回报算法，将单步回报替换为 $G_{t:t+n}$ 或 G_t^λ 即可。下面给出伪代码。

带资格迹的“Actor-Critic”方法 (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 $\hat{v}(s,\mathbf{w})$
参数：迹衰减率 $\lambda^\theta \in [0,1]$, $\lambda^{\mathrm{w}} \in [0,1]$; 步长 $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathrm{w}} > 0$
初始化策略参数 $\theta \in \mathbb{R}^{d'}$ 和状态价值函数的权重 $\mathbf{w} \in \mathbb{R}^d$ (如初始化为0)
无限循环 (对于每一幕)：
  初始化 S(幕的第一个状态)
  $\mathbf{z}^\theta \leftarrow \mathbf{0}$($d'$ 维资格迹向量)
  $\mathbf{z}^{\mathrm{w}} \leftarrow \mathbf{0}$ (d 维资格迹向量)
  $I \leftarrow 1$
  当 S 不是终止状态时，循环：
    $A \sim \pi(\cdot|S,\theta)$
    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$\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{z}^{\mathrm{w}} \leftarrow \gamma\lambda^{\mathrm{w}}\mathbf{z}^{\mathrm{w}} + I\nabla\hat{v}(S,\mathbf{w})$
    $\mathbf{z}^\theta \leftarrow \gamma\lambda^\theta\mathbf{z}^\theta + I\nabla \ln\pi(A|S,\boldsymbol{\theta})$
    $\mathrm{w} \leftarrow \mathrm{w} + \alpha^{\mathrm{w}}\delta\mathrm{z}^{\mathrm{w}}$
    $\theta \leftarrow \theta + \alpha^\theta\delta\mathbf{z}^\theta$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

13.6 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，需要根据每个时刻上的平均收益来定义性能

$$\begin{aligned} J(\boldsymbol{\theta}) \doteq r(\pi) &\doteq \lim_{h\to\infty}\frac{1}{h}\sum_{t=1}^{h}\mathbb{E}[R_t \mid S_0, A_{0:t-1}\sim\pi] \\ &= \lim_{t\to\infty}\mathbb{E}[R_t \mid S_0, A_{0:t-1}\sim\pi] \\ &= \sum_s \mu(s)\sum_a \pi(a|s)\sum_{s',r} p(s',r\mid s,a)r \end{aligned} \tag{12}$$

其中，μ 是策略 π 下的稳定状态的分布，$\mu(s) \doteq \lim_{t\to\infty}\Pr\{S_t = s|A_{0:t}\sim\pi\}$, 并假设它一定存在并独立于 S_0 (一种遍历性假设)。这是一个很特殊的状态分布，如果一直根据策略 π 选择动作，则这个分布会保持不变：

$$\sum_s \mu(s)\sum_a \pi(a|s,\boldsymbol{\theta})p(s'|s,a) = \mu(s'), \text{对所有} s' \in \mathcal{S} \tag{13}$$

用于持续性问题的 Actor-Critic 算法的伪代码在下框中给出。

带资格迹的 "Actor-Critic" 方法 (持续的), 用于估计 $pi_\theta \approx \pi_*$

输入：一个可微的参数化策略 $\pi(a|s,\theta)$
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：$\lambda^{\mathrm{w}} \in [0,1], \quad \lambda^{\theta} \in [0,1], \quad \alpha^{\mathrm{w}} > 0, \quad \alpha^{\theta} > 0, \quad \alpha^{\bar{R}} > 0$
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 $\mathrm{w} \in \mathbb{R}^{\mathrm{d}}$ 和策略参数 $\theta \in \mathbb{R}^{d'}$ (如初始化为 0)
初始化 $S \in \mathcal{S}$ (如初始化为 s_0)
$\mathbf{z}^{\mathrm{w}} \leftarrow 0$ (d 维资格迹向量)
$\mathbf{z}^{\theta} \leftarrow 0$ (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
  $\delta \leftarrow R - \bar{R} + \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \alpha^{R}\delta$
  $\mathbf{z}^{\mathrm{w}} \leftarrow \lambda^{\mathrm{w}}\mathbf{z}^{\mathrm{w}} + \nabla\hat{v}(S,\mathbf{w})$
  $\mathrm{z}^{\theta} \leftarrow \lambda^{\theta}\mathrm{z}^{\theta} + \nabla \ln \pi(A|S,\boldsymbol{\theta})$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathrm{w}}\delta\mathbf{z}^{\mathrm{w}}$
  $\theta \leftarrow \theta + \alpha^{\theta}\delta\mathbf{z}^{\theta}$
  $S \leftarrow S'$

在持续性问题中，用差分回报定义价值函数，$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]$ 以及 $q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$, 其中

$$G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots \tag{14}$$

13.7 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。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。例如，动作集可能是一个实数集，可以根据正态分布来选择动作。

正态分布的概率密度函数一般可以写为

$$p(x) \doteq \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{15}$$

其中，μ 和 σ 这里代表正态分布的均值和标准差。

为了得到一个参数化的策略函数，可以将策略定义为关于实数型的标量动作的正态概率密度，其中均值和标准差由状态的参数化函数近似给出。具体如下

$$\pi(a|s,\boldsymbol{\theta}) \doteq \frac{1}{\sigma(s,\boldsymbol{\theta})\sqrt{2\pi}} \exp\left(-\frac{(a-\mu(s,\boldsymbol{\theta}))^2}{2\sigma(s,\boldsymbol{\theta})^2}\right) \tag{16}$$

其中，$\mu : S \times \mathbb{R}^{d'} \to \mathbb{R}$ 和 $\sigma : S \times \mathbb{R}^{d'} \to \mathbb{R}^+$ 是两个参数化的近似函数。将策略的参数向量划分为两个部分，$\theta = [\theta_\mu, \theta_\sigma]^\top$, 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函数

来逼近。标准差必须为正数，因而使用线性函数的指数形式比较好。因此

$$\mu(s,\boldsymbol{\theta}) \doteq \boldsymbol{\theta}_{\mu}^{\top}\mathbf{x}_{\mu}(s) \quad \text{和} \quad \sigma(s,\boldsymbol{\theta}) \doteq \exp\left(\boldsymbol{\theta}_{\sigma}^{\top}\mathbf{x}_{\sigma}(s)\right) \tag{17}$$

其中，$\mathrm{x}_{\mu}(\mathrm{s})$ 和 $\mathrm{x}_{\sigma}(\mathrm{s})$ 是状态特征问量。

13.8 本章总结

本章讨论了学习一个参数化的策略来保证无须根据动作价值函数的估计值来选择动作，虽然可能仍然需要估计动作价值函数，并用其更新策略参数。特别地，本章介绍了策略梯度法，即在每一步更新中，朝着性能指标对策略参数的梯度的估计值的方向进行更新。

学习和储存策略参数的方法拥有很多优点：

- 能够学习选择动作的特定的概率
- 能够实现合理程度的试探并逐步接近确定性的策略
- 能够自然地处理连续状态空间

一般来说，这些事情对于基于策略的方法都十分简单，而对于 $\varepsilon - greedy$ 法和动作价值函数方法来说都是几乎不可能的。另外，在某些问题中，参数化的策略表示比使用价值函数更加简单，这些问题更适合使用参数化策略方法。

参数化策略的方法也因策略梯度定理相较于动作价值函数方法拥有一个重要的理论优势。策略梯度定理给出了一个明确的公式来表明在不涉及状态分布导数的情况下，性能指标是如何被策略参数影响的。策略梯度定理为所有的策略梯度法提供了理论依据。

REINFORCE 法直接来自于策略梯度定理。增加一个状态价值函数作为基线降低了 REINFORCE 法的方差，同时也没引入偏差。使用状态价值函数进行自举会引入偏差，虽然如此，它还是可以被接受的，因为基于自举法的时序差分一般好于蒙特卡洛法 (大幅度降低方差)。状态价值函数也可以用来给策略的动作选择进行评估打分。相应地，前者称为 Actor，后者称为 Critic，整个方法经常称之为“Actor-Critic”法。