10 在线策略控制近似

本章我们会回到控制问题，现在用行动-值函数 $\hat{q}(s,a,\mathbf{w}) \approx q_*(s,a)$ 的参数化近似，其中 $\mathbf{w} \in \mathbb{R}^d$ 是一个有限维的权重向量。我们继续把关注限制在在线策略情形，把离线策略方法留到章节11。本章的特点是半梯度 Sarsa 算法，半梯度TD(0) (上一章) 到行动值和在线策略控制的的自然延拓。在episodic情形中，这个延拓是直接的，但在连续情形下我们不得不退回几步，并再次考察我们是怎么使用衰减来定义一个最优策略的。惊讶的是，一旦我们用真正的函数近似，我们就不得不放弃衰减，换成使用一种新的"平均回报 (*average-reward*) "的控制问题形式，以及新的"差分 (*differential*) "值函数。

先从episodic情形开始，我们把上一章中函数近似的想法从状态值延拓到行动值。然后我们把它们延拓到遵循在线策略GPI的一般模式的控制，使用 ε 贪婪来选择行动。我们会展示对 n 步线性 Sarsa 在Mountain Car问题中的结果。然后我们转向连续情形，并且针对用差分值的平均回报的情形，重复这些想法的发展。

10.1 Episodic半梯度控制

章节9的半梯度预测方法延拓到行动值是直观明了的。在这种情形中，近似值函数 $\hat{q} \approx q_\pi$ ，是被表示成用权重向量 $\mathbf{w}$ 的参数化函数形式。鉴于我们之前考虑的随机训练样例是形如 $S_t \mapsto U_t$ ，现在我们考虑样例是形如 $S_t, A_t \mapsto U_t$. 更新目标 U_t 可以是任意 $q_\pi(S_t, A_t)$ 的近似，包括通常的 *backed-up* 值比如全MC回报 (G_t) 或任何 n 步 Sarsa 的回报 (7.4). 一般的对行动-值预测的梯度下降更新是

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\big[U_t - \hat{q}(S_t, A_t, \mathbf{w}_t)\big]\nabla\hat{q}(S_t, A_t, \mathbf{w}_t). \tag{10.1}$$

举个例子，对于单步 Sarsa 方法的更新是

$$\mathbf{w}_{t+1} \doteq w_t + \alpha\big[R_{t+1} + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t)\big]\nabla\hat{q}(S_t, A_t, \mathbf{w}). \tag{10.2}$$

我们把这方法称作 episodic 半梯度单步 Sarsa (*episodic semi-gradient one-step Sarsa*). 对于一个固定的策略，这个方法会以和 TD(0) 相同的方式收敛，且有同种的误差边界 (9.14).

为了形成控制方法我们需要把这些行动-值预测方法与策略提升和行动选择的技术相结合。能应用于连续的动作，或大型离散集里的动作的合适技术，是目前正在研究的主题，但还没有明确的解决方案。另一方面，如果行动集是离散的，或者不是特别大的那我们能使用已经在前几章开发的技术。也就是，对每个下个状态 S_{t+1} 可用的可能的行动 a ，我们可以计算 $\hat{q}(S_{t+1}, a, \mathbf{w}_t)$ 并找到贪婪行动 $A^*_{t+1} = argmax_a\hat{q}(S_{t+1}, a, \mathbf{w}_t)$. 策略提升然后可以完成 (在本章处理的在线策略情形中) 通过改变评估策略为一个贪婪策略的软近似，比如 ε 贪婪策略。行动按照这个相同策略来选择。完整算法的伪代码在下面方框中给出。

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    If S' is terminal:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
      Go to next episode
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

例10.1：登山车任务

考虑一个任务，在陡峭的山路上驾驶一辆动力不足的车，如图10.1的左上方图片所示。困难在于引力比车的引擎要强，即便全力踩油门车还是不能加速爬坡。唯一的解决方案是先远离目标向左边相反的斜坡上爬。然后，通过全力踩油门车能够获得足够的惯性来带它爬上斜坡，即便它全程都在减速。这是一个连续性任务的简单示例，其中事物必须要在某种意义上变坏(远离目标) 在它们能变好之前。很多控制方法学处理这类问题都很困难，除非被人工设计者明确地帮助。

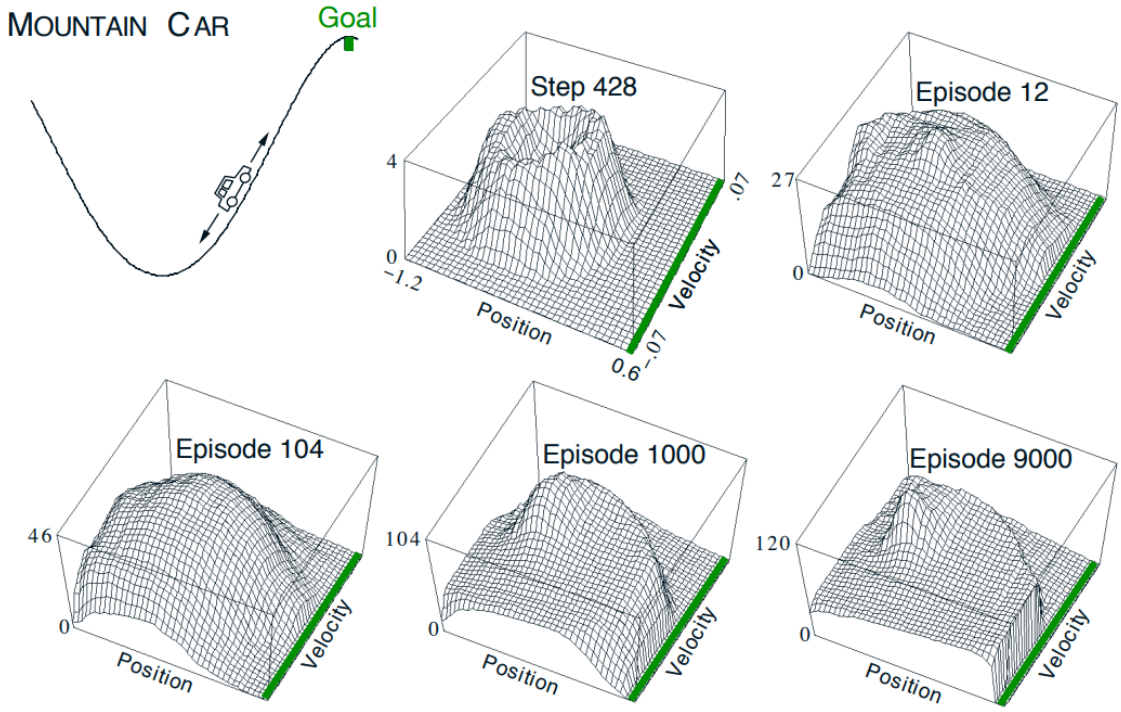

Figure 10.1: The Mountain Car task (upper left panel) and the cost-to-go function ($-\max_a \hat{q}(s, a, \mathbf{w})$) learned during one run.

在这个问题中回报是 -1 在全部的时间步上直到车移动到它在山顶的目标位置，这会结束episode. 有三个可能状态：全向前加速 (+1)，全向后加速 (-1)，零加速 (0). 车移动遵循一个简化的物理学。它的位置 x_t，和速度 $\dot{x}_t$，被更新通过

$$
\begin{aligned}
x_{t+1} &\doteq bound[x_t + \dot{x}_t] \\
\dot{x}_{t+1} &\doteq bound[\dot{x}_t + 0.001 A_t - 0.0025\cos(3x_t)],
\end{aligned}
$$

其中 $bound$ 操作使得 $-1.2 \leq x_{t+1} \leq 0.5$ 和 $-0.07 \leq \dot{x}_{t+1} \leq 0.07$. 此外，当 x_{t+1} 到达左边界时，$\dot{x}_{t+1}$ 会重置为 0. 当它到达右边界时，目标达到，episode会终止。每次episode从一个随机位置 $x_t \in [-0.6, -0.4)$，速度为0开始。为了把这两个连续性状态变量转变成二元特征，我们使用如图9.9中的网格平铺 (grid-tiling). 我们使用8个平铺，每个平铺片在每个维度上包含$1/8$ 的边界距离，并且非对称偏移设为如章节9.5.4讲述的那样。特征向量 $\mathbf{x}(s, a)$ 由平铺编码构建，然后和参数向量线性组合来近似行动-值函数：

$$
\hat{q}(s, a, \mathbf{w}) \doteq \mathbf{w}^T \mathbf{x}(s, a) = \sum_{i=1}^{d} w_i \cdot x_i(s, a), \tag{10.3}
$$

对每个状态 s 和 行动 a 对。

图10.1展示了通常发生的情况在这个形式的函数近似解决这个任务时。展示的是在一次运行中学习到的值函数的负值 (代价函数). 最初的行动值都是0，这是积极的 (在这次任务中所有真值都是负的)，导致发生了广泛的探索甚至探索参数 ε 是 0. 这可以在图中的中上方部分看到，标记了 "Step 428". 在这个时间甚至一个episode都没有被完成，但车已经在山谷中来回摇摆，沿着状态空间中的环形轨迹。被频繁访问的状态值都比未探索的状态差，因为实际回报已经比 (不切实际的) 期待要差了。这持续驱使代理者离开它曾在的地方，去探索新状态，直到一个解决方案被找到。

图10.2 展示了在这问题上半梯度 Sarsa 用不同步长的学习曲线。

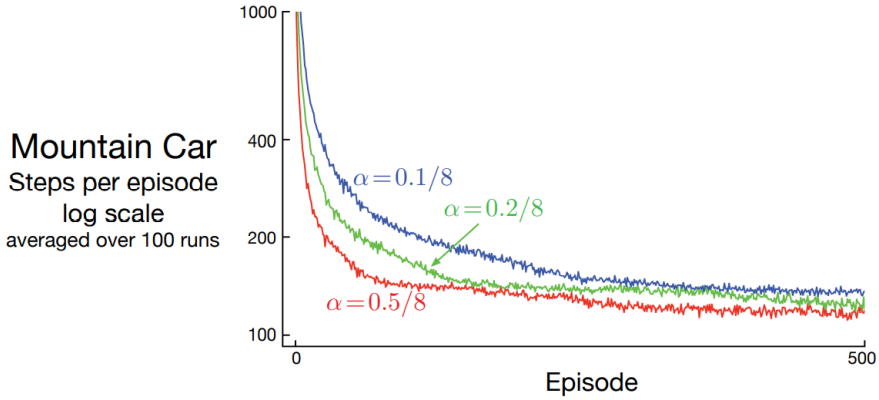

Figure 10.2: Mountain Car learning curves for the semi-gradient Sarsa method with tile-coding function approximation and ε-greedy action selection. ■

10.2 半梯度 n 步 Sarsa

我们能通过使用一个 n 步返回作为半梯度Sarsa更新方程 (10.1) 中的更新目标来得到一个n步 episodic 半梯度 Sarsa 版本。n步返回立即从它的列表性形式 (7.4) 泛化到了一个函数近似形式：

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \quad t+n < T, \qquad (10.4)$$

其中 $G_{t:t+n} \doteq G_t$ 如果 $t+n \geq T$，和之前一样。n 步更新方程是

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})] \nabla \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}), \quad 0 \leq t < T. \qquad (10.5)$$

完整的伪代码在下面给出。

Episodic semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_*$ or q_π

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$, a positive integer n
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$ or ε-greedy wrt $\hat{q}(S_0, \cdot, \mathbf{w})$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
    If $t < T$, then:
      Take action A_t
      Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
      If S_{t+1} is terminal, then:
        $T \leftarrow t+1$
      else:
        Select and store $A_{t+1} \sim \pi(\cdot|S_{t+1})$ or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
    $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    If $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ $(G_{\tau:\tau+n})$
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha \left[G - \hat{q}(S_\tau, A_\tau, \mathbf{w})\right] \nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
  Until $\tau = T - 1$

就如我们之前所见，使用在中间级别的拔靴，性能是最好的，相应的 n 应该大于1。图10.3展示了在登山车任务上算法在 $n = 8$ 时倾向于比在 $n = 1$ 时学得更快而且得到更好的渐进表现。图10.4展示了一个更详细的研究参数 α 和 n 对学习速率的影响的结构。

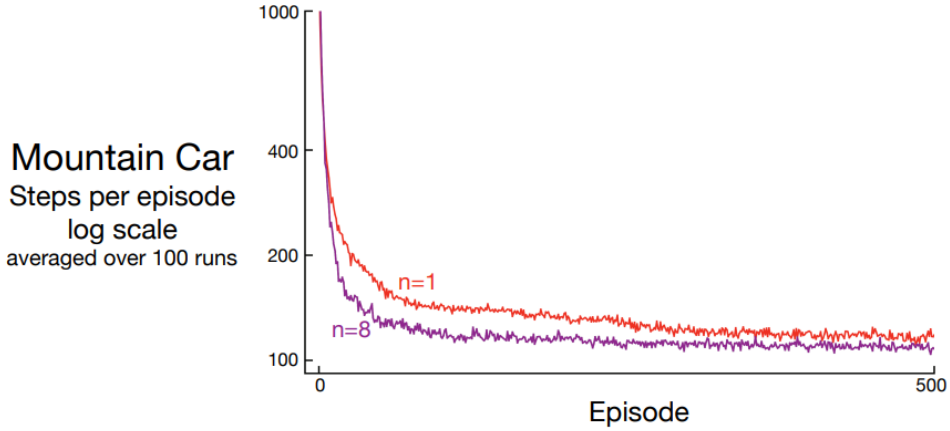

Figure 10.3: Performance of one-step vs 8-step semi-gradient Sarsa on the Mountain Car task. Good step sizes were used: $\alpha = 0.5/8$ for $n = 1$ and $\alpha = 0.3/8$ for $n = 8$.

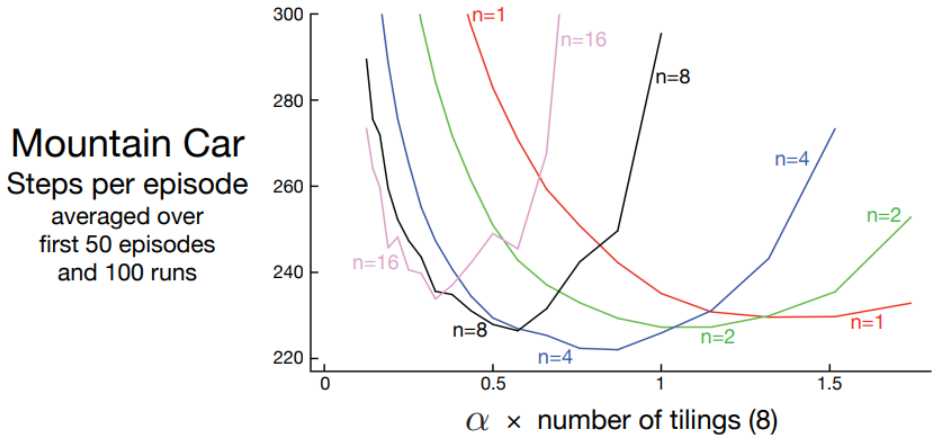

Figure 10.4: Effect of the α and n on early performance of n-step semi-gradient Sarsa and tile-coding function approximation on the Mountain Car task. As usual, an intermediate level of bootstrapping ($n = 4$) performed best. These results are for selected α values, on a log scale, and then connected by straight lines. The standard errors ranged from 0.5 (less than the line width) for $n = 1$ to about 4 for $n = 16$, so the main effects are all statistically significant.

例10.1

在这章我们没有具体地考虑或给出任何MC方法的伪代码。它们会是什么样子？为什么不给出它们的伪代码是合理的？它们会在爬山车任务中怎么表现？

例10.2

给出半梯度一步期望Sarsa (one-step Expected Sarsa) 用于控制。

例10.3

为什么图10.4显示的结果中大的 n 比小的 n 有更大的标准差?

10.3 平均回报：连续型任务设置的新问题

我们现在介绍第三种经典设置——同时和episodic和折扣的设置——来规划在马尔科夫决策问题(MDPs) 中目标。像折扣设置，**平均回报** (*average reward*) 设置也适用连续性问题，即代理者和环境之间的交互将会永远持续，没有终止或起始状态的问题。但是不同于折扣设置，它没有折扣——代理者对延时回报的关心程度就是与立即回报相同。平均回报设置是在动态规划中经典理论中普遍考虑的主要设置之一，但在强化学习中不太常见。如我们会在下节讨论的，折扣设置在函数近似中存在问题，因此需要用平均回报来取代它。

在平均回报设置中，策略 π 的值被定义为平均回报率 (average rate of reward)，或简化为平均回报(average reward)，在遵循该策略时下，我们将其标记为 $r(\pi)$:

$$
\begin{aligned}
r(\pi) &\doteq \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t|S_0, A_{0:t-1} \sim \pi] && (10.6)\\
&= \lim_{t\to\infty} \mathbb{E}[R_t|S_0, A_{0:t-1} \sim \pi], && (10.7)\\
&= \sum_s \mu_\pi(s) \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) r,
\end{aligned}
$$

其中期望是在起始状态 S_0，和按照 π 取的后续行动 $A_0, A_1, \ldots, A_{t-1}$为条件的。第二个和第三个等式成立，若平稳状态分布 $\mu_\pi(s) \doteq \lim_{t\to\infty} Pr\{S_t = s|A_{0:t-1} \sim \pi\}$，存在且独立于 S_0 ，换句话说，如果MDP是**遍历的** (*ergodic*). 在一个遍历的MDP，起始状态和任何代理者所做的早期决策只能够有一个短暂的影响；在长期运行中在一个状态上的期望会取决于策略和MDP转移概率。遍历性是充分的但不是必要的对保证在 (10.6) 中极限的存在。

在无折扣连续性情形中，不同种类的最优性之间存在微妙的区别。但是，对大多数实际目的，可以足够简化成按照它们的每个时间步的平均回报来分级策略，换句话说，按照它们的 $r(\pi)$. 这个值基本上是在 π 下的平均回报，如 (10.7) 所暗示的，或者回报率 (*reward rate*). 特别地，我们认为所有能得到 $r(\pi)$ 的最大值的策略是最优的。

注意到平稳状态分布 μ_π 是一种特殊分布，在这种分布下，如果你按照 π 选择行动，你会保持在相同的分布。也即，

$$
\sum_s \mu_\pi(s) \sum_a \pi(a|s) p(s'|s, a) = \mu_\pi(s'). \tag{10.8}
$$

在平均回报设置中，返回是根据回报和平均回报之间的差值定义的：

$$
G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \cdots. \tag{10.9}
$$

这也被称为差分返回，对应的值函数也被称为差分值函数。差分值函数是根据新的返回定义的，就如传统的值函数是根据折扣返回定义的；因此我们将对差分值函数使用相同的记号，$v_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]$ 和 $q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$ (v_* 和 q_* 也相似). 差分值函数也有贝尔曼方程，只是和之前所见的略有不同。我们简单地去掉所有 γ 并把所有回报用回报和真的平均回报的差值来取代：

$$
\begin{aligned}
v_\pi(s) &= \sum_a \pi(a|s) \sum_{r,s'} p(s', r|s, a) \Big[r - r(\pi) + v_\pi(s')\Big],\\
q_\pi(s, a) &= \sum_{r',s} p(s', r|s, a) \Big[r - r(\pi) + \sum_{a'} \pi(a'|s') q_\pi(s', a')\Big],\\
v_*(s) &= \max_a \sum_{r,s'} p(s', r|s, a) \Big[r - \max_\pi r(\pi) + v_*(s')\Big],\\
q_*(s, a) &= \sum_{r,s'} p(s', r|s, a) \Big[r - \max_\pi r(\pi) + \max_{a'} q_*(s', a')\Big]
\end{aligned}
$$

(参考 (3.14)，练习3.17，(3.19) 和 (3.20)).

也有两个TD误差的差分形式：

$$
\delta_t \doteq R_{t+1} - \bar{R}_t + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t), \tag{10.10}
$$

和

$$
\delta_t \doteq R_{t+1} - \bar{R}_t + \hat{q}(S_{t+1}, A_{t+1}, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t), \tag{10.11}
$$

其中 $\bar{R}_t$ 是时间 t 上的平均回报 $r(\pi)$ 的估计。有了这些替换的定义，我们大多数算法中和很多理论结果能够不做改变完成平均回报设置。

比如，半梯度Sarsa的一个平均回报版本就可以如在 (10.2) 定义的那样，除了TD误差的微分形式。即，用

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\delta_t \nabla \hat{q}(S_t, A_t, \mathbf{w}_t), \tag{10.12}$$

其中 δ_t 由 (10.11) 给出。完整算法的伪代码在下面给出。这个算法的一个局限是它不收敛到差分值，二四收敛到差分值加一个任意偏移。注意到如果所有的值如果都偏移同一个量，上面给的贝尔曼方程和TD误差是不会受影响的。因此，实际上这个偏移可能不重要。这个算法如何能够改成去除掉偏移是一个未来研究的有趣的问题。

> 例10.2 一个访问控制队列任务
>
> 这是一个决策任务，涉及对一组10台服务器的访问控制。有四个不同优先级的用户会在单一队列中到达。如果一台服务器提供访问，用户会对服务器支付一个报酬 1, 2, 4, 或 8 ，取决于它们的优先级，优先级越高用户支付越多。在每个时间步，在队列头部的用户要么被接受 (分配给一个服务器)，或者拒绝 (从队列中移除，报酬为0). 无论哪种情形，在下一个时间步在队列中的下一个用户会被考虑。队列不会空，队列中用户的优先级是统一随机分布。当然一个用户不能被服务当没有空闲的服务器；在这种情况下用户会始终被拒绝。每个忙碌的服务器在每个时间步以概率 $p = 0.06$ 变得空闲。尽管我们刚刚描述它们的明确性，但让我们假设到达和离开的统计数据是未知的。这个任务是决定每个时间步是接受还是拒绝下一个用户，基于它的优先级和空闲的服务器数量，从而使得无折扣下实现最大化长期回报。
>
> 在这个例子中我们考虑这个问题的一个列表性解法。尽管状态之间没有泛化，我们仍能在一般函数近似设置中考虑它，因为这个设置泛化了列表性设置。因此我们有一个差分行动-值估计对每一个状态 (空闲服务器的数量和在队列头部的用户优先级) 和行动 (接受或拒绝) 对。图10.5展示了由差分半梯度Sarsa搜寻到的解法，其中参数是 $\alpha = 0.01, \beta = 0.01, \varepsilon = 0.1$. 初始行动值和 $\overline{R}$ 是零。

10.4 弃用折扣设置

连续折扣问题表述在列表性情形中已经非常有用，其中每个状态的返回都可以被分别辨识和平均。但在近似情形中是否使用这个问题表述是存疑的。

为了发现原因，考虑一个无限的返回序列，没有开始和结束，也没有明确的辨识状态。状态可能仅用特征向量来表示，使得很难区分相互之间的状态。比如特殊地，所有特征向量都可能是相同的。因此我们可能只有报酬序列 (和动作)，而性能必须单纯靠这些来评估。怎么才能做到这？一种方式是通过在一段很长的区间上平均回报——也就是平均回报设置的想法。折扣要怎样被使用？对于每一个时间步我们能衡量折扣返回。一些返回将是小的而一些将是大的，所以我们必须再次在一个充分大的时间区间上将它们平均。在连续配置中没有起始和结束，也没有特殊的时间步，所以没有能够额外做的事情。但是，如果你做了这，那么折扣平均会正比于平均回报。事实上，对于策略 π，折扣返回的平均一直都是 $r(\pi)/(1-\gamma)$，也就是，本质上是平均回报 $r(\pi)$. 特别地，在平均折扣返回设置中所有策略的**排序** (ordering) 将完全和在平均回报设置中的一样。折扣比率 γ 因此在问题描述中没有效果。实际上它可以是**零**且分级将不会改变。

这个惊讶的事实在下面的方框中会被证明，但基本思想能够通过对称性论证看到。每个时间步和每个其他的都完全相同。有了折扣，每个报酬将会在某些返回中的每个位置中完全只出现一次。第 t 次报酬将出现在第 $t-1$ 次返回中无折扣，在 $t-2$ 次返回中折扣一次，在第 $t-1000$ 次返回中折扣 999 倍。在第 t 次报酬上的权重因此是 $1+\gamma+\gamma^2+\gamma^3+\cdots = 1/(1-\gamma)$. 因为所有状态是相同的，它们全都被赋这个权重，因此返回的平均将会是平均回报的这个倍数，即 $r(\pi)/(1-\gamma)$.

这个例子和方框中更多的一般论点证明了如果我们优化了在在线策略分布上的折扣值，那么效果将等价于优化无折扣平均报酬；γ 的实际值将没有效果。这强力地表明折扣在有函数近似的控制问题的定义中没有作用。但是我们仍能在解决方法中使用折扣。折扣参数 γ 将从一个问题参数变为一个解决方法参数! 不幸的是，用函数近似的折扣算法不能在在线策略分布上优化折扣值，因此不能保证优化平均回报。

> 连续问题中的折扣无益
>
> 也许折扣能通过选择一个目标被保存，该目标将在以策略下出现的状态的分布上的折扣值相加：
>
> $$\begin{aligned} J(\pi) &= \sum_s \mu_\pi(s) v_\pi^\gamma(s) && \text{(其中 } v_\pi^\gamma \text{ 是折扣值函数)} \\ &= \sum\nolimits_{\_} \mu_\pi(s) \sum\nolimits_{\_} \pi(a|s) \sum\nolimits_{\_}\sum\nolimits_{\_} p(s', r|s, a)[r + \gamma v_\pi^\gamma(s')] && \text{(贝尔曼等式)} \end{aligned}$$

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

这建议的折扣目标评级策略等价于无折扣 (平均回报) 目标。折扣比率 γ 不会影响评级!

用折扣控制设置的困难的根本原因是，用函数近似我们已经失去了策略提升理论 (章节4.2). 如果我们改变策略来提升一个状态的折扣值，那么我们能保证在任何有用的意义上提升总体的策略，这将不再是事实。那个保证是我们强化学习控制方法的理论关键。用函数近似我们会失去它。

实际上，缺少策略提升理论也是一个对全episodic和平均回报设置的理论缺陷。一旦我们引入函数近似我们就不再能保证任何设置的提升。在章节13我们将引入一个替换的强化学习算法类，它基于参数化策略，并且我们有一个理论保证称作"策略梯度理论"，它和策略提升理论有着类似的作用。但是对于学习动作值的方法，我们似乎现在缺少一个局部提升保证 (可能由 Perkins 和 Precup (2003) 采取的方法会提供一部分答案). 我们确实知道 $\varepsilon - greedification$ 有时候会导致一个劣策略，由于策略会在好策略之间打转而不是收敛 (Gordon, 1996a). 这是一个由多重开放性理论问题的领域。

10.5 差分半梯度 n 步 Sarsa

为了一般化到 n 步拔靴，我们需要一个 n 步版本的TD误差。我们开始通过泛化 n 步返回 (7.4) 到它的差分形式，用函数近似：

$$
G_{t:t+n} \doteq R_{t+1} - \overline{R}_{t+n-1} + \cdots + R_{t+n} - \overline{R}_{t+n-1} + \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \tag{10.14}
$$

其中 $\overline{R}$ 是一个 $r(\pi)$ 的估值，$n \geq 1$，且 $t+n < T$. 如果$t+n \geq T$，那么我们如之前那样定义 $G_{t:t+n} \doteq G_t$. n 步TD误差就变成

$$
\delta_t \doteq G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}), \tag{10.15}
$$

之后我们就可以用这到我们的一般半梯度Sarsa更新 (10.12). 完整算法的伪代码在方框中给出。

Differential semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_\pi$ or q_*

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$, a policy π
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average-reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)
Algorithm parameters: step sizes $\alpha, \beta > 0$, small $\varepsilon > 0$, a positive integer n
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Initialize and store S_0 and A_0
Loop for each step, $t = 0, 1, 2, \ldots$:
  Take action A_t
  Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$, or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
  $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  If $\tau \geq 0$:
    $\delta \leftarrow \sum_{i=\tau+1}^{\tau+n} (R_i - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(S_\tau, A_\tau, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S_\tau, A_\tau, \mathbf{w})$

10.6 小结

在本章我们已经拓展了上章引入的参数化函数近似和半梯度下降的思想，到控制上。对于 episodic情形这拓展是立即的，但是对连续性情形我们不得不引入一个全新的基于每时间步最大化**平均回报设置** (*average reward setting*) 的问题描述。惊奇地，折扣描述不能延拓到控制中因为近似的存在。在近似情形中大多数策略不能被一个值函数表示。要保持的任意策略需要被评级，而标量平均报酬 $r(\pi)$ 提供了一个有效的方式来做到这点。

平均回报描述包含了新的的**差分** (*differential*) 版本对于值函数、贝尔曼方程和TD误差，但所有这些都和旧的是平行的，而且概念上的改变是小的。也有一种对平均回报情形的新的差分算法的平行设置。