章节11 用近似的离线策略方法

本书已经从章节5开始把在线策略和离线策略当作两种替换的方式来处理从广义策略迭代的学习形式中固有的开发和探索之间的矛盾。前两章已经对在线策略情形用函数近似来处理，本章将对离线策略情形用函数近似来处理。事实证明，对于离线策略学习来说，到函数近似的延拓比起在线策略学习有显著的不同且更加困难。章节6和7中研发的列表性离线策略方法很很简便地拓展到半梯度算法，但这些算法收敛不如它们在在线策略训练下稳定。本章我们探索收敛性问题，用更近的视角观察线性函数近似理论，并引入学习能力的概念，然后讨论新的算法对于离线策略情形用更强的收敛保证性。在最后我们将会改进方法，但这理论结果将不会和在线策略学习一样强，实验结果也不会和在线策略学习一样令人满意。在这过程中，我们会获得对在强化学习中的近似更深的理解，对在线策略学习和离线策略学习。

回忆在离线策略学习中我们试图学习一个对于**目标策略** (*target policy*) π 的值函数，从一个不同的**行为策略** (*behavior policy*) b 中获取数据。在预测情形中，策略是静态的且固定的，我们试图学习的是状态值 $\hat{v} \approx v_\pi$ 或者行动值 $\hat{q} \approx q_\pi$. 在控制情形中，行动值被学习，且两种策略在学习中一般都会改变——π 是关于 $\hat{q}$ 的贪婪策略，b 是关于 $\hat{q}$ 的更加具有探索性的比如 ε-贪婪策略。

离线策略学习的挑战能被分为两个部分，一个是在列表性情形中出现的，另一个是仅用函数近似中出现的。第一部分的挑战与更新的目标 (不要和目标策略混淆) 有关，第二部分与更新的分布有关。在章节5和7中研发的关于重要性采样的技巧能处理第一部分；这些会增加方差但在所有成功的算法中，即列表性还是近似，都需要这么做。这些技巧到函数近似的延拓会在本章的第一小节中快速地被处理。

对于用函数近似的离线策略学习的第二部分的挑战还需要更多的东西，因为在离线情形中，更新的分布不是按照在线策略分布。在线策略分布对半梯度方法的稳定性是非常重要的。两种普遍的途径用来处理这点。一种是再次使用重要性采样方法，这次将更新分布变回在线策略分布，使得半梯度方法能保证收敛 (在线性情形中)。另一种是开发真正的梯度方法使得不再依赖于任何特殊的平稳分布。我们呈现基于两种途径的方法。这是一个尖端的研究领域，目前不明确哪种途径实际上更加有效。

11.1 半梯度方法

首先，我们会描述在前几章中针对离线策略情形开发的方法如何容易拓展到函数近似来作为半梯度方法。这些方法处理了离线策略学习的第一部分挑战 (改变更新目标) 但没有第二部分 (改变更新分布). 相应地，这些方法会在一些情形中发散，在这种意义下没有那么好，但仍然经常被成功使用。记住这些方法对列表性情形保证是平稳的且渐进无偏的，而列表性情形对应于函数近似的一种特殊情形。所以仍然可能将它们与特征选择方法以某种方式结合起来使得联合系统能保证是平稳的。任何事件中，这些方法都是简单的，因此是开始的好地方。

在章节7，我们描述了一类列表性离线策略算法。为了将它们转换成半梯度形式，我们简单地将对一个数组 (V或者Q) 的更新替换为对一个权重向量 ($\mathbf{w}$) 的更新，使用近似的值函数和它的梯度。大多数这些算法使用每步重要性采样比率：

$$\rho_t \doteq \rho_{t:t} = \frac{\pi(A_t|S_t)}{b(A_t|S_t)}. \tag{11.1}$$

例如，单步状态值算法是半梯度离线策略 TD(0)，就如对应的在线策略算法，除了增加的 ρ_t：

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\rho_t\delta_t\nabla\hat{v}(S_t, \mathbf{w}_t), \tag{11.2}$$

其中 δ_t 是恰当定义的取决于问题是episodic且折扣的，或者连续且无折扣的使用平均回报：

$$\delta_t \doteq R_{t+1} + \gamma\hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t), \tag{11.3}$$

或者

$$\delta_t \doteq R_{t+1} - \overline{R}_t + \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t), \tag{11.4}$$

对于行动值，单步算法是半梯度期望Sarsa：

$$\mathbf{w}_{t+1} \doteq \mathbf{w}_t + \alpha\delta_t \nabla \hat{q}(S_t, A_t, \mathbf{w}_t), \tag{11.5}$$

其中

$$\delta_t \doteq R_{t+1} + \gamma \sum_a \pi(a|S_{t+1})\hat{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t), \qquad \text{(episodic)}$$

或者

$$\delta_t \doteq R_{t+1} - \overline{R}_t + \sum_a \pi(a|S_{t+1})\hat{q}(S_{t+1}, a, \mathbf{w}_t) - \hat{q}(S_t, A_t, \mathbf{w}_t). \quad \text{(continuing)}$$

注意到这个算法没有使用重要性采样。在列表性情形中这显然是合适的，因为采样动作只有 A_t ，且学习它的值我们不需要考虑其它动作。用函数近似就没有那么显然，因为我们可能想要对不同的状态-行动对分别赋权一旦它们都有助于相同的整体近似。这个问题的正确解决需要对强化学习中的函数近似理论有更深入的理解。

在这些算法的多步推广中，状态值和行动值算法都包含重要性采样。半梯度Sarsa的 n 步版本是

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha\rho_{t+1}\cdots\rho_{t+n}[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})]\nabla\hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}) \tag{11.6}$$

其中

$$G_{t:t+n} \doteq R_{t+1} + \cdots + \gamma^{n-1}R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, \mathbf{w}_{t+n-1}), \qquad \text{(episodic)}$$

或者

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

其中这里我们在对待episodes的终止有点不正式。在第一个等式中，ρ_ks 对于 $k \geq T$ (其中 T 是episode的最后时间步) 应该被取成 1，且 $G_{t:t+n}$ 应该被取成 G_t 如果 $t + n \geq T$.

回忆我们在章节7展示过的一种离线策略算法，它完全不包含重要性采样: n 步后援树算法。这里是它的半梯度版本：

$$\mathbf{w}_{t+n} \doteq \mathbf{w}_{t+n-1} + \alpha[G_{t:t+n} - \hat{q}(S_t, A_t, \mathbf{w}_{t+n-1})]\nabla\hat{q}(S_t, A_t, \mathbf{w}_{t+n-1}), \tag{11.7}$$

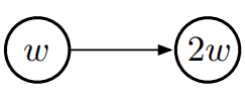
$$G_{t:t+n} \doteq \hat{q}(S_t, A_t, \mathbf{w}_{t-1}) + \sum_{k=t}^{t+n-1} \delta_k \prod_{i=t+1}^{k} \gamma\pi(A_i|S_i), \tag{11.8}$$

其中 δ_t 就如上方期望Sarsa所定义。我们也在章节7定义了另一种算法，它统一了所有行动值算法：n 步 $Q(\sigma)$. 我们留下这个算法的半梯度形式，以及 n 步状态值算法作为练习。

11.2 离线策略发离线策略学习发散的示例

在本节中我们开始讨论第二部分用函数近似的离线策略学习的挑战——更新分布不匹配于在线策略分布。我们会举一些具有指导意义的离线策略学习的反例——也就是半梯度和其它简单算法是不稳定和发散的情形。

为了建立直观，最好从考虑一个很简单的例子开始。假设，也许是作为一个大MDP中的一部分。有两个状态它们的估计值有函数形式 w 和 $2w$，其中参数向量 $\mathbf{w}$ 仅包含一个单一元素 w. 在线性函数近似下会发生这，如果两个状态都是简单数 (单一元向量)，在这个情形中是1和2。在第一个状态，仅有一个动作可使用，它会确定性地生成一个到第二个状态的迁移，且报酬为0：



其中两个圆中的表达式表示这两个状态的值。

假设初始 $w = 10$. 迁移将会从一个估计值为10的状态到一个估计值为20的状态。这看起来像一个好的迁移，w 会增大来提高第一个状态的估计值。如果 γ 接近于1，那么TD误差将接近10，如果 $\alpha = 0.1$，那么 w 会增大到大约11来试图减小TD误差。然而，第二个状态的估计值也会增加到接近于22。如果该迁移再次发生，那么它会从一个估计值约为11的状态到一个估计值约为22的状态，其中TD误差约为11——比之前变大了，而不是变小。它看起来更像是第一个状态被低估值了，它的值将会再次增大，这次会到约12.1。这看起来不好，实际上随着不断更新 w 会发散到无穷。

为了确定性地看待这我们需要在更新序列上观察得更仔细。在这两个状态间的迁移上的TD误差是

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t) = 0 + \gamma 2 w_t - w_t = (2\gamma - 1) w_t,$$

且离线策略半梯度TD(0) 更新 (从 (11.2)) 是

$$w_{t+1} = w_t + \alpha \rho + t \delta_t \nabla \hat{v}(S_t, w_t) = w_t + \alpha \cdot 1 \cdot (2\gamma - 1) w_t \cdot 1 = (1 + \alpha(2\gamma - 1)) w_t.$$

注意到重要性比率 ρ_t 是1在这次迁移上，因为从第一个状态只有一个动作可获得，所以它被取的概率在目标和行为策略中必须都是1。在上述最后的更新中，新的参数是旧的参数乘以一个标量常数 $1 + \alpha(2\gamma - 1)$. 如果这个常数是比1要大，那么系统将会不稳定，w 会达到正无穷或负无穷取决于它的初始值。这里这个常数大于1只要 $\gamma > 0.5$. 注意到稳定性不依赖于特定步长，只要 $\alpha > 0$. 小点或大点的步长只会影响 w 趋向无穷的速率，但不会影响它是否会达到无穷。

这个例子的关键是一个迁移被重复发生且没有在其它迁移中更新 w. 这在离线策略学习中是可能的因为行为策略会选择目标策略不会有的其它迁移。对于这些迁移，ρ_t 会是0，从而不会做任何更新。在在线策略学习下，ρ_t 则会一直为1。每个时间有一个从 w 状态到 $2w$ 状态的迁移，来增加 w，同时也必须有一个从 $2w$ 状态向外的迁移。那个迁移会减小 w，除非它是到一个值超过 $2w$ (因为 $\gamma < 1$)的状态，那么该状态必须紧跟着一个甚至更高值的状态，否则 w 将会减小。每个状态能支持这点只有通过创造一个更高的期望。最终这代价必须要被支付。在在线策略情形中必须遵守未来报酬的承诺和保持系统处于控制状态下。但在离线情形中，可以先做一个承诺然后在采取一个目标策略不会采取的行动后，忘记和免除它。

这个简单的例子讲述了很多为什么离线策略训练会导致发散的原因，但它不能完全令人信服因为它不是完整的——它只是一个完整MDP中的一部分。会存在一个有不稳定性的完整系统吗？一个简单的发散的完整示例是 **Baird反例** (*Baird's counterexample*). 考察图11.1展示的一个episodic 7状态，2动作的MDP。dashed行动会等概率地带系统到六个上方状态之一，相对地，solid行动会带系统到第7个状态。行为策略 b 分别以概率 $\frac{6}{7}$ 和 $\frac{1}{7}$ 选择行动 dashed 和 solid，从而到下一个状态的分布是一致的(对所有非终止状态)，这也是对每个episode的起始状态分布。目标策略 π 一直取 solid 行动，从而在线策略分布(对于 π) 集中在第七个状态。在所有迁移上的报酬都是0。折扣率是 $\gamma = 0.99$.

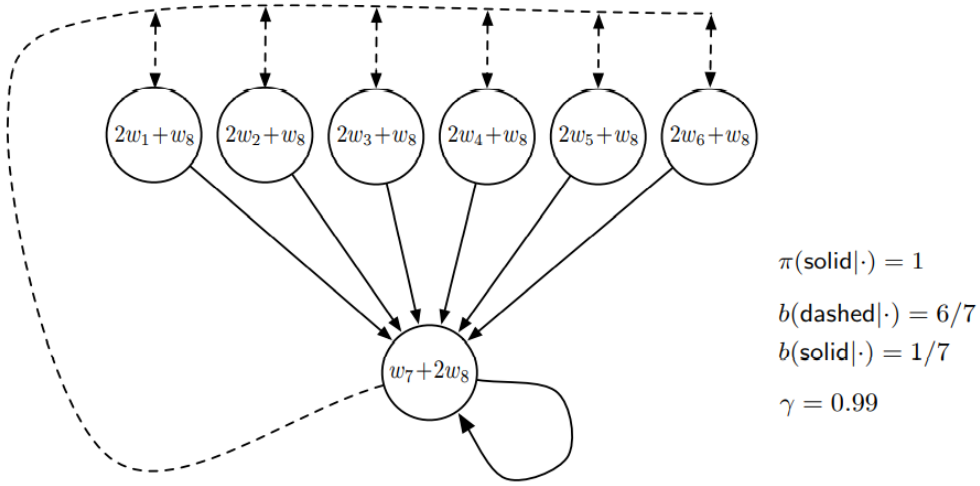


Figure 11.1: Baird's counterexample. The approximate state-value function for this Markov process is of the form shown by the linear expressions inside each state. The solid action usually results in the seventh state, and the dashed action usually results in one of the other six states, each with equal probability. The reward is always zero.

考虑按照每个状态圆中的表达式表示的线性参数来估计状态值。比如，最左边的状态的估计值是 $2w_1 + w_8$，其中下标对应于整体的权重向量 $\mathbf{w} \in \mathbb{R}^8$ 的分量；对于第一个状态对应于的特征向量是 $\mathbf{x}(1) = (2, 0, 0, 0, 0, 0, 0, 1)^T$. 所有迁移的报酬都为0，所以真正的值函数是 $v_\pi(s) = 0$，对所有 s，这能被精确近似如果 $\mathbf{w} = \mathbf{0}$. 事实上，有很多解，因为有比非终止状态 (7) 更多的权重向量的分量 (8). 另外，特征向量集 $\mathbf{x}(s) : s \in \mathcal{S}$，是一个线性无关集。所有的这些方面中，这个任务似乎都是线性函数近似所喜爱的情形。

如果我们对该问题采用半梯度TD(0)，那么权重会发散到无穷，如图11.2 (左) 所示。不稳定性会在任意正步长上发生，不管它有多小。实际上，它甚至会发生在如果期望更新是在动态规划 (DP) 下完成的，如图11.2 (右) 所示。即，如果权重向量 $\mathbf{w}_k$ 在相同的时间以半梯度的方式被所有状态更新，使用DP (基于期望) 目标:

$$\mathbf{w}_{k+1} \doteq \mathbf{w}_k + \frac{\alpha}{|\mathcal{S}|} \sum_s \Big(\mathbb{E}_\pi[R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_k) | S_t = s] - \hat{v}(s, \mathbf{w}_k) \Big) \nabla \hat{v}(s, \mathbf{w}_k). \quad (11.9)$$

在这个情形中，没有随机性和异步性，只是一个传统的DP更新。方法是传统的处理它用在半梯度函数近似 。但仍然系统是不稳定的。

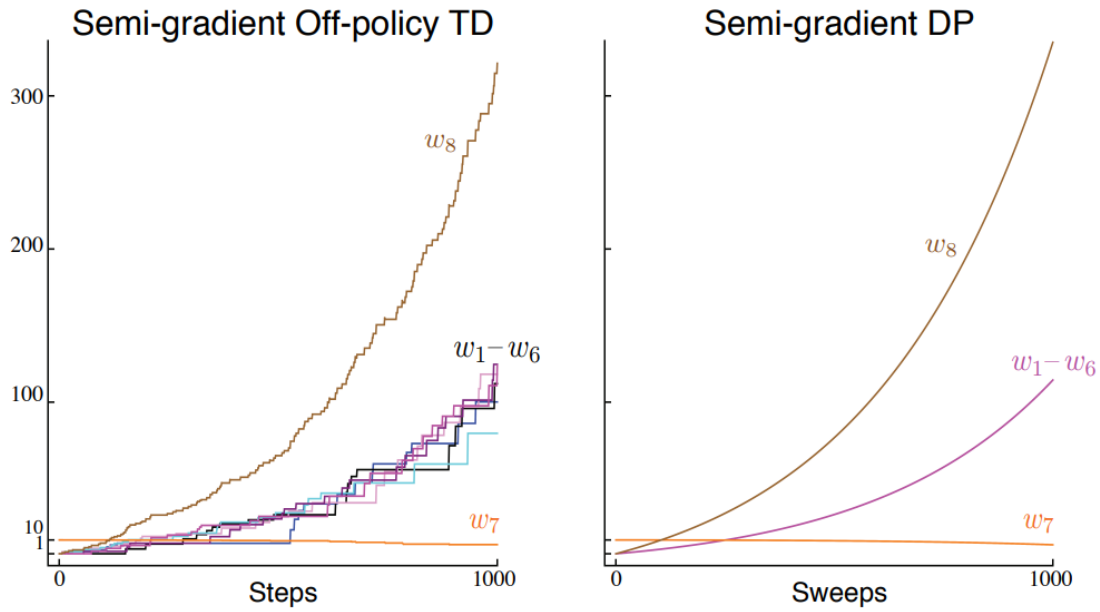
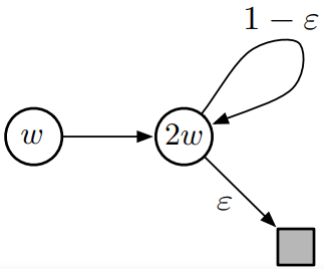


Figure 11.2: Demonstration of instability on Baird's counterexample. Shown are the evolution of the components of the parameter vector $\mathbf{w}$ of the two semi-gradient algorithms. The step size was $\alpha = 0.01$, and the initial weights were $\mathbf{w} = (1, 1, 1, 1, 1, 1, 10, 1)^\top$.

如果我们仅改变在 Baird反例中的DP更新分布，从一致分布到在线策略分布(一般需要异步更新)，那么收敛性会保证到误差边界为(9.14)的一个解上。这个例子令人惊讶因为使用的TD和DP方法按理上是最简单的且最好理解的拔靴法，且线性半梯度方法按理上是最简单的且最好理解的一类函数近似。这个例子证明了即便是最简单的拔靴和函数近似的组合也会不稳定如果更新没有按照在线策略分布完成。

也有很多类似于Baird的反例来证明Q-learning的发散性。这引起了担忧，因为否则Q-learning在所有的控制方法中有最好的收敛性保证。大量的工作已经投入到试图寻找一个这个问题的补救方法或得到一些更弱的但仍然能工作的保证。比如说，可能保证Q-learning的收敛性只要行为策略充分接近于目标策略，比如，当它是 ε-greedy策略。尽我们知识的最大努力，Q-learning在这种情形中从没有发现发散，但还没有理论证明。在这节的剩余部分我们会展示几个已经被探索过的想法。

假设不是在每次迭代中仅采取一步向期望单步返回，如在Baird反例中，我们实际上而是始终改变值函数为最佳的最小二乘近似。这能解决非稳定性问题吗？当然它会，如果特征向量 $\{\mathbf{x}(s) : s \in \mathcal{S}\}$，形成了一个线性无关集，如在Baird反例中所做的，因为那么在每次迭代上精确的近似是可能的且方法会退化为标准列表性DP。但是这里的重点是考虑不可能得到一个精确解的情形。在这种情形中稳定性不被保证即便在每次迭代建立最好的近似，如下面例中所示。

> 例11.1：*Tsitsiklis*和*VanRoy*反例
>
> 这个例子展示了用DP的线性函数近似将不会有效，即便在每一步都找到了最小二乘解。这个反例由一个拓展的有终止状态的 w 到 $2w$ 例子(本节先前的例子)建立，如图所示。如前，第一个状态的估计值是 w，第二个状态的估计值是 $2w$. 报酬在所有迁移上都是0，所以两个状态的真值都是0，可以用 $w = 0$ 精确表示。如果我们设置w_{k+1} 在每一步都是使得估计值和期望单步返回的 $\overline{VE}$ 最小化，那么我们有
>
> 
>
>
> $$\begin{aligned} w_{k+1} &= \arg\min_{w \in \mathbb{R}} \sum_{s \in \mathcal{S}} \left(\hat{v}(s, w) - \mathbb{E}_\pi[R_{t+1} + \gamma \hat{v}(S_{t+1}, w_k) | S_t = s] \right)^2 \\ &= \arg\min_{w \in \mathcal{R}} (w - \gamma 2 w_k)^2 + (2w - (1 - \varepsilon)\gamma 2 w_k)^2 \\ &= \frac{6 - 4\varepsilon}{5} \gamma w_k. \end{aligned} \tag{11.10}$$

> 这个序列$\{w_k\}$发散当$\gamma > \frac{5}{6-4\varepsilon}$且$w_0 \neq 0$.

另一个试图避免不稳定性的方式是对函数近似使用特殊的方法。特别地，对不从观察目标外推的函数近似方法能保证稳定性。这些方法称作**平均法** (averagers)，包含最近邻方法和局部加权回归，但没有流行的方法比如平铺编码和人工神经网络(ANNs).

11.3 死亡三项

我们到目前为止的讨论可以被总结为当我们将所有下列三个要素组合在一起时，会有不稳定性和发散性发生的危险，这组成的三项我们称为**死亡三项** (*the deadly triad*)：

> **函数近似** 从比内存和计算资源大得多的状态空间进行泛化的一个有力的，可拓展的方法 (例如，线性函数近似或ANNs)。
>
> **拔靴** 更新目标包含现有的估计 (如动态规划或TD方法) 而不是完全依赖于实际报酬和完整返回 (如MC方法)。
>
> **离线策略训练** 在一个不同于目标策略生成的迁移的分布上训练。遍历状态空间并一致更新所有状态，就如在动态规划中，这不遵守目标策略且是一个离线策略训练的示例。

特别地，注意到风险不是由于控制或广义策略迭代造成的。这些情形分析更加复杂，但非稳定性会出现在更简单的预测情形只要它包含所有死亡三项。这风险也不是由于学习或环境的不确定性造成的，因为它在计划方法中同等强地发生，比如动态规划，它的环境是完全已知的。

如果死亡三项的任意两项存在，但不是都存在时，不稳定性可以被避免。那么很自然地会去仔细研究这三项看看是否有任何一项可以放弃。

对于这三项，**函数近似**是最明显不能被放弃的。我们需要方法能拓展到大型问题和强大表达能力。我们至少需要有很多特征和参数的线性函数近似。状态类聚或非参数方法它们的复杂性会随数据增长，往往会很弱或者开支很大。最小二乘法比如LSTD有平方复杂性因此对大型问题开支过大。

不用**拔靴**来做是可能的，代价是计算和数据效率。也许最重要的是计算效率的损失。MC(非拔靴)方法需要空间来保存所有在做每个决策和得到最终返回之间发生的所有事物，直到得到最终返回所有它们的计算被完成。这些计算问题的代价在串行冯·诺伊曼计算机上并不明显，但在专用硬件上会很明显。用拔靴和资格轨迹(章节12)，数据能在它被生成的时间和地点被处理，然后就不需要再被使用。通过拔靴可以大量节省通信和内存。

放弃拔靴带来的数据效率的损失也是巨大的。我们已经重复见到这点，比如章节7 (图7.2) 和9 (图9.2)，其中一定程度的拔靴能比MC方法在随机游走预测任务种完成得出色得多，在章节10中的登山车控制任务 (图10.4) 中也看到同样的结果。许多其它问题展示了用拔靴能极大加速学习(比如，见图12.14). 拔靴经常产生更快的学习因为它允许学习能利用状态性质，即返回到状态时能识别它的能力。另一方面，拔靴会损害学习，在状态表示是差的且会导致差的泛化的问题上 (比如，似乎是在 Tetris 上的情形)。一个差的状态表示也会导致偏见；这是拔靴方法的渐进近似品质界更差的原因 (等式9.14)。总的来说，拔靴的能力被认为是极度有价值的。我们有时候选择不使用它通过选择长的n步更新(或者一个大的拔靴参数，$\lambda \approx 1$；见章节12) 但是通常拔靴会显著提高效率。这是一种我们会很想要留在工具箱中的能力。

最后，**离线策略学习**；我们能放弃这吗？在线策略方法通常是足够的。对于模型自由的强化学习，我们可以简单使用Sarsa而不是Q-learning. 离线策略方法可以从目标策略解放行为。这能被看作是一个诱人的便利但不是一个必需。但是，离线策略学习对于其它预期使用情形是**至关重要的**，尽管这本书中尚未提及这些情形，但对于创建一个强大的智能代理者的更大的目标可能是重要的。

在这些使用情形中，代理者学习的不仅仅是一个单一的值函数或者是单一的策略，而是并行学习大量的它们。有心理学证据表明人类和动物学习来预测很多不同的感官事件，而不仅是报酬。我们可以被不寻常事件惊讶到，然后校正对它们的预测，即使它们是中性性价(既不好也不坏)。这种预测可能是构成世界预测模型的基础，比如被用于计划。我们在眼球运动后会预测我们将会看到什么，要多久才能走到

家，完成一次篮球的跳投的概率，我们从接受一个新项目中将获得的满足感。在所有这些情形中，我们想预测的事件取决于我们以什么方式运行的。为了学习它们的全部，并行地，需要从一个经验流中学习。有很多目标策略，因此一个行为策略不能和所有它们相同。但并行学习还只是概念上可以，因为行动策略能够与很多目标策略有一部分重复。要充分利用这点这需要离线策略学习。

11.4 线性值函数几何

为了更好地理解离线策略学习的稳定性挑战，思考值函数近似更加抽象和独立于学习是怎么完成的会有帮助。我们可以想象所有可能的状态-值函数的空间——所有状态到实值的函数 $v : \mathcal{S} \to \mathbb{R}$. 这里的大多数值函数并不对应任何策略。对我们的目的更重要的是，大多数不能用函数近似器表示，其中函数近似器根据设计会有远少于状态数的参数。

给定一个状态空间的枚举 $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$，任何值函数 v 对应于一个按序列表每个状态值 $[v(s_1), v(s_2), \ldots, v(s_{|\mathcal{S}|})]^\top$ 的向量。这个值函数的向量表示有于状态数同样多的分量。在大多数情形中，我们想要使用函数近似，那就会有过多的分量来显式表示向量。但是，这个向量的思想在概念上是有用的。接下来，我们交替地对待一个值函数和它的向量表示。

为了激发直观，考虑有三个状态的情形 $\mathcal{S} = \{s_1, s_2, s_3\}$ 和两个参数 $\mathbf{w} = (w_1, w_2)^\top$. 我们然后能将所有值函数/向量视作在三维空间上的点。参数提供了一个额外的在二维子空间上的坐标系统。任何权重向量 $\mathbf{w} = (w_1, w_2)^\top$ 是二维子空间的一个点，且因此也是一个完整的对三个状态赋值的值函数 $v_\mathbf{w}$. 用一般函数近似，可表示的函数的全空间和子空间之间的关系会变得复杂，在在线性值函数近似情形中，子空间仅是一个简单的平面，如图11.3所暗示的。

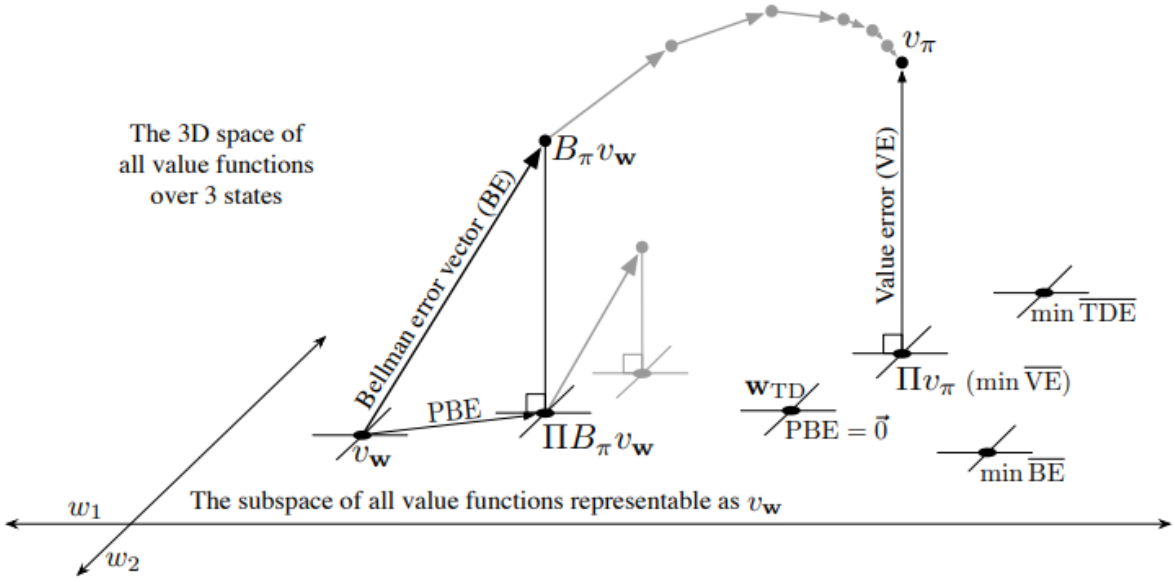


Figure 11.3: The geometry of linear value-function approximation. Shown is the three-dimensional space of all value functions over three states, while shown as a plane is the subspace of all value functions representable by a linear function approximator with parameter $\mathbf{w} = (w_1, w_2)^\top$. The true value function v_π is in the larger space and can be projected down (into the subspace, using a projection operator Π) to its best approximation in the value error (VE) sense. The best approximators in the Bellman error (BE), projected Bellman error (PBE), and temporal difference error (TDE) senses are all potentially different and are shown in the lower right. (VE, BE, and PBE are all treated as the corresponding vectors in this figure.) The Bellman operator takes a value function in the plane to one outside, which can then be projected back. If you iteratively applied the Bellman operator outside the space (shown in gray above) you would reach the true value function, as in conventional dynamic programming. If instead you kept projecting back into the subspace at each step, as in the lower step shown in gray, then the fixed point would be the point of vector-zero PBE.

现在考虑一个单一固定策略 π. 我们假设它的真值函数 v_π 过于复杂以致无法用一个近似精确表示。因此 v_π 不在子空间；在图中它被刻画为在可表示函数的平面子空间的上方。

如果 v_π 不能被精确表示，那么什么可表示值函数是最接近于它呢？这就变成了一个有许多答案的微妙的问题。为了开始，我们需要一个度量来衡量两个值函数之间的距离。给定两个值函数 v_1 和 v_2，我们能够讨论它们之间的差向量 $v = v_1 - v_2$. 如果 v 很小，那么这两个值函数互相很接近。但我们要怎么衡量这个差向量的大小？传统的欧几里得范数已经不再合适，因为如章节9.2中讨论的，一些状态会比其它状态更重要因为它们更频繁发生或者因为我们更关心它们(章节9.11). 在章节9.2中，让我们使用分布 $\mu : \mathcal{S} \to [0, 1]$ 来具体表述我们对不同状态要被精确赋值的关心程度(通常取作在线策略分布). 我们然后能够定义值函数之间的距离用这个范数

$$\|v\|_\mu^2 \doteq \sum_{s \in \mathcal{S}} \mu(s) v(s)^2. \tag{11.11}$$

注意到章节9.2中的$\overline{VE}$ 能够被用这个范数简单写成 $\overline{VE}(\mathbf{w}) = \|v_\mathbf{w} - v_\pi\|_\mu^2$. 对于任意值函数 v，寻找在可表示值函数的子空间中最接近它的值函数的运算是一个投射运算。我们定义一个投射运算 Π 能够由任意一个值函数得到在我们的范数下最接近的可表示函数：

$$\Pi v \doteq v_\mathbf{w} \quad where \quad \mathbf{w} = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \|v - v_\mathbf{w}\|_\mu^2. \tag{11.12}$$

最接近真值函数 v_π 的可表示值函数，即它的投射 Πv_π 如图11.3所暗示的。这是由MC方法找到的渐近解，尽管通常很慢。连城运算会在下面方框中更加全面地讨论。

> 投射矩阵
>
> ---
>
> 对于一个线性函数近似器，投射运算是线性的，这表明它可以被表示成一个 $|\mathcal{S}| \times |\mathcal{S}|$ 矩阵:
>
> $$\Pi \doteq \mathbf{X}(\mathbf{X}^\top \mathbf{D} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{D}, \tag{11.13}$$
>
> 其中，如章节9.4，$\mathbf{D}$ 表示 $|\mathcal{S}| \times |\mathcal{S}|$ 对角矩阵，对角元为 $\mu(s)$，$\mathbf{X}$ 表示 $|\mathcal{S}| \times d$ 矩阵其行向量是特征向量 $\mathbf{x}(s)^\top$，对于每个状态 s. 如果在 (11.13) 中的逆不存在的话，那么用广义逆来代替。使用这些矩阵，一个向量的平方范数可以写成
>
> $$\|v\|_\mu^2 = v^T \mathbf{D} v, \tag{11.14}$$
>
> 近似线性值函数可以写成
>
> $$v_\mathbf{w} = \mathbf{X}\mathbf{w}. \tag{11.15}$$
>
> 然后用线性代数知识
>
> $$\begin{aligned}
> & & \mathbf{v} - \Pi\mathbf{v} \perp \mathbf{v}_\mathbf{w},\ \forall \mathbf{w} \\
> Suppose\ \Pi\mathbf{v} = \mathbf{X}\mathbf{w}_\mathbf{v} & & \\
> \Rightarrow & & \mathbf{v} - \mathbf{X}\mathbf{w}_\mathbf{v} \perp \mathbf{X} \cdot \mathbf{w},\ \forall \mathbf{w} \\
> \Rightarrow & & \mathbf{w}^\top \mathbf{X}^\top \cdot \mathbf{D} \cdot (\mathbf{v} - \mathbf{X}\mathbf{w}_\mathbf{v}) = \mathbf{0},\ \forall \mathbf{w} \\
> \Rightarrow & & \mathbf{X}^\top \mathbf{D}(\mathbf{v} - \mathbf{X}\mathbf{w}_\mathbf{v}) = \mathbf{0} \\
> \Rightarrow & & \mathbf{X}^\top \mathbf{D}\mathbf{v} = \mathbf{X}^\top \mathbf{D}\mathbf{X}\mathbf{w}_\mathbf{v} \\
> \Rightarrow & & \mathbf{w}_\mathbf{v} = (\mathbf{X}^\top \mathbf{D}\mathbf{X})^{-1}\mathbf{X}^\top \mathbf{D}\mathbf{v} \\
> \Rightarrow & & \Pi\mathbf{v} = \mathbf{X}(\mathbf{X}^\top \mathbf{D}\mathbf{X})^{-1}\mathbf{X}^\top \mathbf{D}\mathbf{v} \\
> that\ is & & \Pi = \mathbf{X}(\mathbf{X}^\top \mathbf{D}\mathbf{X})^{-1}\mathbf{X}^\top \mathbf{D}
> \end{aligned}$$

TD方法找到的是不一样的解。为了理解它们的根本原理，回忆值函数 v_π 的贝尔曼方程是

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')], \quad \text{for all } s \in \mathcal{S}. \tag{11.16}$$

真值函数 v_π 是 (11.16) 的唯一精确解。如果一个近似值函数 $v_{\mathbf{w}}$ 被用来代替 v_π，那么修正后的等式左右两边的差值能被用来作为 $v_{\mathbf{w}}$ 距离 v_π 多远的一个衡量。我们把这称作在状态 s 上的**贝尔曼误差** (*Bellman error*):

$$\overline{\delta}_{\mathbf{w}}(s) \doteq \left(\sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_{\mathbf{w}}(s')] \right) - v_{\mathbf{w}}(s) \tag{11.17}$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma v_{\mathbf{w}}(S_{t+1}) - v_{\mathbf{w}}(S_t) | S_t = s, A_t \sim \pi], \tag{11.18}$$

这清晰地展示了贝尔曼误差和TD误差 (11.3) 之间的关系。贝尔曼误差是TD误差的期望。

由所有状态上的贝尔曼误差组成的向量 $\overline{\delta}_{\mathbf{w}} \in \mathbb{R}^{|\mathcal{S}|}$，被称作**贝尔曼误差向量** (*Bellman error vector*) (在图11.3中表示为 BE). 这个向量的整体大小，在范数下，是一个值函数误差的整体衡量，称作**均方贝尔曼误差** (*mean square Bellman error*):

$$\overline{BE}(\mathbf{w}) = \|\overline{\delta}_{\mathbf{w}}\|^2_\mu. \tag{11.19}$$

一般不可能能减少 $\overline{BE}$ 到 0 (即 $v_{\mathbf{w}} = v_\pi$)，但对于线性函数近似有一个 $\mathbf{w}$ 唯一值使得 $\overline{BE}$ 最小化。这个点在可表示函数子空间 (在图11.13中用 $\min \overline{BE}$ 标记) 一般是不同于最小化 $\overline{VE}$ (表示为 Πv_π) 的点。寻找最小化 $\overline{BE}$ 的方法在后面两节会讨论。

在图 11.3 中展示的贝尔曼误差向量是应用**贝尔曼运算** (*Bellman operator*) $B_\pi : \mathbb{R}^{|\mathcal{S}|} \to \mathbb{R}^{|\mathcal{S}|}$ 到近似值函数上的结果。贝尔曼运算被定义为

$$(B_\pi v)(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v(s')], \tag{11.20}$$

对所有$s \in \mathcal{S}, v : \mathcal{S} \to \mathbb{R}$. 对 $v_{\mathbf{w}}$ 的贝尔曼误差向量可以写作 $\overline{\delta}_{\mathbf{w}} = B_\pi v_{\mathbf{w}} - v_{\mathbf{w}}$.

如果贝尔曼运算应用到在可表示子空间的一个值函数上，那么，一般它会产生一个在子空间外的新的值函数，如图中所暗示的。在动态规划中(不用函数近似)，这个运算会重复应用在可表示空间外的点上，如图11.3的灰色箭头所暗示的。最终这个过程会收敛到真值函数 v_π，贝尔曼运算的唯一不动点，唯一一个值函数使得

$$v_\pi = B_\pi v_\pi, \tag{11.21}$$

这恰好是写对 π 的贝尔曼方程 (11.16) 的另一种方式。

用函数近似，然而，在子空间外面的中间值函数不能被表示。在图11.3上方部分的灰色箭头不能够被遵循是因为在第一个更新(黑线)后值函数必须要被投射回可表示的值函数。下次迭代会从子空间开始；然后值函数再次通过贝尔曼运算取到子空间外面，接着再次由投射运算映射回去，如图中更下面的灰色箭头和直线所暗示的。遵循这些箭头是一个用近似的拟DP过程。

在这种情形中我们关心贝尔曼误差向量回到可表示空间的投射。这是投射贝尔曼误差向量 $\Pi\overline{\delta}_{\mathbf{w}}$ 如图11.3中展示为PBE。这个向量的大小，范数意义下，是另一个近似值函数误差的度量。对任意一个近似值函数 $v_{\mathbf{w}}$，我们定义**均方投射贝尔曼误差** (*mean square Projected Bellman error*)，记作 $\overline{PBE}$，即

$$\overline{PBE}(\mathbf{w}) = \|\Pi\overline{\delta}_{\mathbf{w}}\|^2_\mu. \tag{11.22}$$

用函数近似一直存在一个近似值函数 (在子空间上) 有零 $\overline{PBE}$；即是TD不动点 $\mathbf{w}_{TD}$，在章节9.4中引入。如我们所见，该点在半梯度TD方法和离线策略训练中不是一直稳定的。如图中所示，这个值函数一般不同于最小化 $\overline{VE}$ 和 $\overline{BE}$ 的值函数。保证收敛到这个值函数的方法会在章节11.7和11.8中讨论。

11.5 贝尔曼误差的梯度下降

随着对值函数近似和它的各种目标的更深入理解，我们现在回到离线策略学习中稳定性的挑战。我们希望应用随机梯度下降方法(SGD，章节9.3)，其所做的更新期望上等于目标函数的负梯度。这些方法一直在向目标下山 (期望上) 因此一般是稳定的且有优秀的收敛性质。在这本书已经探索的算法中，只有MC方法是真正的SGD方法。这些方法在在线策略和离线策略训练下以及对一般非线性 (可微的) 函数近似器都稳定收敛，尽管它们经常比不是SGD方法的拔靴的半梯度方法要慢，半梯度方法可能会在离线策略训练中发散，就如本章前面所见的，还有在非函数近似的不自然情形中也会发散 (Tsitsiklis 和 Van Roy, 1997). 用一个真正的SGD方法将不可能有这样的发散性。

SGD的吸引力是如此之强以至大量研究投入来寻找一个实用的利用它到强化学习的方式。所有这些努力的开始位置是一个要优化的误差或目标函数的选择。在本节及下一节中我们会探索最流行建议的目标函数，即基于前节引入的**贝尔曼误差**，的起源和局限。尽管这已经是一种流行的且有影响力的方式，但我们这得出的结论是，它是一个错误且不会生成好的学习算法。另一方面，这种途径失败的方式是有趣的，它能提供什么可能是构成一个好的途径的观点。

为了开始，我们先不考虑贝尔曼误差，而是一些更直接原始的东西。时序差分学习是由TD误差驱动的。为什么不取最小化TD误差的平方期望作为目标呢？在一般函数近似情形中，带折扣的单步TD误差是

$$\delta_t = R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t).$$

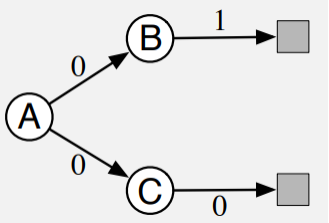
一个可能的目标函数就为我们可能称作**均方TD误差** (*mean square TD error*):

$$\begin{aligned}\overline{TDE}(\mathbf{w}) &= \sum_{s \in \mathcal{S}} \mu(s) \mathbb{E}[\delta_t^2 | S_t = s, A_t \sim \pi] \\ &= \sum_{s \in \mathcal{S}} \mathbb{E}[\rho_t \delta_t^2 | S_t = s, A_t \sim b] \\ &= \mathbb{E}_b[\rho_t \delta_t^2]. \qquad (\text{如果}\mu\text{是在b下发生的分布})\end{aligned}$$

最后一个等式是SGD需要的形式；它给出了一个作为期望的目标能够被经历所采样 (记住经历是由行为策略 b 生成的). 因此按照标准的SGD方式，我们可以推导出基于这个期望值的样例的每步更新:

$$\begin{aligned}\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2}\alpha \nabla (\rho_t \delta_t^2) \\ &= \mathbf{w}_t - \alpha \rho_t \delta_t \nabla \delta_t \\ &= \mathbf{w}_t + \alpha \rho_t \delta_t (\nabla \hat{v}(S_t, \mathbf{w}_t) - \gamma \nabla \hat{v}(S_{t+1}, \mathbf{w}_t)),\end{aligned} \qquad (11.23)$$

你会认出这和半梯度TD算法 (11.2) 是一样的除了增加的最后一项。这一项补全了梯度使得这是一个真正的SGD算法，有优秀的收敛性保证。让我们称这个算法是原生残差梯度 (*naive residual-gradient*) 算法 (在 Baird 后，1995). 尽管原生残差梯度算法稳定收敛，它并不一定收敛到一个想要的位置。

> 例11.2: A-split 例子，展示原生残差梯度算法的天真
>
> 考虑一个三状态的episodic MRP，如右展示。episodic从状态A开始，然后随机'分裂'，一半时间走向B (然后始终走向终止并有一个报酬1) 一半时间走向C (然后始终走向终止并有一个报酬0). 对于第一个转移，从A离开的，无论走哪条路径报酬始终是0。由于这是一个episodic问题，我们可取 γ 为1。我们也假设在线策略训练，所以 ρ_t 是始终为1，然后列表性函数近似，所以学习方法可以自由地对所有三个状态任意且独立赋值。因此，这应该是一个简单的问题。
>
> 
>
>
> 这些值应该是什么？从A，一半时间返回是1，一半时间返回是0；A的值应该是 $\frac{1}{2}$. 从B，返回始终是1，所以它的值应该是1，类似的从C返回始终是0，所以它的值应该是0。这些是真值且这是一个列表性问题，所有之前呈现的方法都应该精确收敛到它们。

> 然而，原生残差梯度算法找到了不同的值对B和C。它在B收敛到值 $\frac{3}{4}$，在C收敛到值 $\frac{1}{4}$ (A正确收敛到 $\frac{1}{2}$). 这些值实际上是最小化 $\overline{TDE}$.
>
> 让我们计算这些值的 $\overline{TDE}$. 每个episode的第一个转移要么是从A的 $\frac{1}{2}$ 增加到B的 $\frac{3}{4}$，一个 $\frac{1}{4}$ 的变化，或者从A的 $\frac{1}{2}$ 减少到C的 $\frac{1}{4}$，一个 $-\frac{1}{4}$ 的变化。因为在这些转移上报酬是0，以及 $\gamma = 1$，这些变化是TD误差，因此在第一个转移上TD误差的平方一直是 $\frac{1}{16}$. 第二个转移是相似的；它要么从B的 $\frac{3}{4}$ 增加到报酬 1 (以及一个值为0的终止状态)，或者从C的 $\frac{1}{4}$ 减少到报酬 0 (以及一个值为0的终止状态). 因此，TD误差始终是 $\pm\frac{1}{4}$，在第二步上有一个误差平方为 $\frac{1}{16}$. 因此，对于这组值，$\overline{TDE}$ 在两步上均是 $\frac{1}{16}$.
>
> 现在让我们计算对真值 (B是1，C是0，A是 $\frac{1}{2}$) 的 $\overline{TDE}$. 在这个情形中第一个转移要么是从 $\frac{1}{2}$ 增加到 1，到B，或者从 $\frac{1}{2}$ 到 0，到C；每种情形绝对误差都是 $\frac{1}{2}$ 且误差平方是 $\frac{1}{4}$. 第二个转移误差是0因为开始值，要么是1或者是0，取决于它是从B还是C转移，始终都准确地和立即报酬和返回匹配。因此平方TD误差在第一个转移是 $\frac{1}{4}$，在第二个转移是 0，对于在两个转移上的一个平均回报是 $\frac{1}{8}$. 由于 $\frac{1}{8}$ 比 $\frac{1}{16}$ 大，这个解依照 $\overline{TDE}$ 是更差的。在这个简单问题上，真值没有最小的 $\overline{TDE}$.

在A-分裂示例中使用的是列表性表示，因此真正的状态值能够被精确表示，但原生残差梯度算法找到了不同的值，而这些值有比真值更低的 $\overline{TDE}$. 最小化 $\overline{TDE}$ 是天真的；通过惩罚所有TD误差它达到的更像是时序平滑而不是精确预测。

一个更好的想法似乎是最小化均方贝尔曼误差 ($\overline{BE}$). 如果学习的是精确值，那么贝尔曼误差都是零。因此，一个贝尔曼误差-最小化算法对于A-分裂示例应该没有问题。我们一般不能期待零贝尔曼误差，因为它蕴含找到真值函数，而这我们假定是在可表示值函数空间外的。但接近这一理想似乎是一个自然的目标。我们已经看到，贝尔曼误差与TD误差紧密相关。一个状态上的贝尔曼误差是该状态上的TD误差的期望。所以让我们重复上述期望TD误差的推导 (所有这里期望都隐含在 S_t 上的条件)：

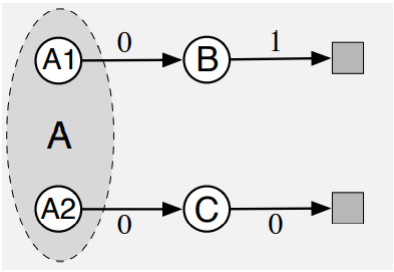
$$
\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2}\alpha\nabla(\mathbb{E}[\delta_t]^2) \\
&= \mathbf{w}_t - \frac{1}{2}\alpha\nabla(\mathbb{E}_b[\rho_t\delta_t]^2) \\
&= \mathbf{w}_t - \alpha\mathbb{E}_b[\rho_t\delta_t]\nabla\mathbb{E}_b[\rho_t\delta_t] \\
&= \mathbf{w}_t - \alpha\mathbb{E}_b[\rho_t(R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w}))]\mathbb{E}_b[\rho_t\nabla\delta_t] \\
&= \mathbf{w}_t + \alpha\big[\mathbb{E}_b[\rho_t(R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}))] - \hat{v}(S_t,\mathbf{w})\big]\big[\nabla\hat{v}(S_t,\mathbf{w}) - \gamma\mathbb{E}_b[\rho_t\nabla\hat{v}(S_{t+1},\mathbf{w})]\big].
\end{aligned}
$$

这种更新和各种采样它的方式被称作是**残差梯度算法** (*residual-gradient algorithm*). 如果你简单地在所有期望处使用样例值，那么上方的等式就几乎完全退化成 (11.23)，原生残差梯度算法。但这是原始的，因为上方等式包含了下一个状态 S_{t+1}，在两个期望中都出现了且相乘到一起。为了得到一个乘积的无偏样例，下一个状态的两个独立的样例是需要的，但是在一般的与外界环境交互中仅有可以得到一个。一个期望或另一个能被采样，但不能全是。

有两种方式来完成残差梯度算法的工作。一种是在确定性环境的情形中。如果到下一个状态的转移是确定的，那么两个样例将必须相同，那么原生算法是有效的。另一种方式是得到下一个状态 S_{t+1} 的两个独立样例，从 S_t，一个用作第一个期望，另一个用作第二个期望。在实际与环境交互时，这似乎是不可能的，但与一个模拟环境交互时，这是可能的。我们可以简单地回滚到先前状态然后得到一个替代的下一个状态，在从第一个下个状态前行之前。这两种情形残差梯度算法都保证收敛到一个 $\overline{BE}$ 的最小值处在通常的步长条件下。作为一个真正的SGD方法，这个收敛性是稳定的，无论是应用到线性还是非线性函数近似器上。在线性情形中，收敛性总是保证到唯一的最小化 $\overline{BE}$ 的 $\mathbf{w}$.

然而，残差梯度方法的收敛性至少存在三个方面是不令人满意的。第一是从经验上它收敛很慢，要比半梯度方法慢很多。事实上，这种方法的支持者已经建议，在初期将它和更快的半梯度方法结合来提高其速度，然后逐渐转换成残差梯度来保证收敛性 (Baird 和 Moore, 1999). 第二个残差梯度算法是不令人满意的方面是它仍然似乎收敛到错误的值上。它的确在所有列表性情形中始终得到正确值，比如A-分裂，对于那些贝尔曼方程的精确解是可能的情形。但如果我们如果用真正的函数近似来检查样例，那么

残差梯度算法，和 $\overline{BE}$ 目标，似乎会找到错误的值函数。最能说明这点的示例之一是A-分裂示例的一个变种，被称作是A-presplit示例，如下所示，残差梯度算法在其中找到了同原生版本一样差的解。这个例子展示了直观地展示了最小化 $\overline{BE}$ (残差梯度算法确实做的) 可能不是一个想要的目标。

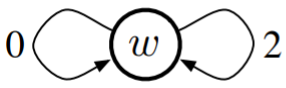
> 例11.3: A-presplit 例子，一个$\overline{BE}$ 的反例
>
> 
>
>
> 考虑一个三状态的episodic MRP如右图所示：Episodes等概率地开始于A1或者A2。这两个状态在函数近似器看来完全相同，像一个单一状态A它的特征表示是不同且无关于另外两个状态B和C的状态表示的，这两个状态表示之间也互不相同。特别地，函数近似器的参数有三个分量，一个给状态B赋值，一个给状态C赋值，一个给两个状态A1和A2赋值。除了初始状态的选择以外，系统是确定性的。如果它开始于A1，那么它会转移到B，报酬为0，然后到终止状态，报酬为1。如果它开始于A2，那么它会转移到C，然后到终止状态，两个报酬都是0。
>
> 对于一个学习算法，只用特征来看，这个系统看起来和A-split例子是完全相同的。这个系统似乎始终开始于A，然后等概率地后接B或C，然后终止并有报酬1或0确定性地取决于先前的状态。如A-split例子中，B和C的真值是1和0，且A1和A2最好的共享值是 $\frac{1}{2}$，根据对称性。
>
> 因为这个问题外部表现完全和A-split例子相同，我们已经知道算法会找到哪些值。半梯度TD会收敛到刚讲的理想值，原生残差梯度算法会收敛到对B和C值分别是 $\frac{3}{4}$ 和 $\frac{1}{4}$. 所有状态转移是确定性的，所以非原生残差梯度算法也应该收敛到这些值(在这个情形中它们是相同的算法)。因此它说明这个'原生'解也必须是最小化 $\overline{BE}$ 的，事实也确实如此。在一个确定性问题中，贝尔曼误差和TD误差全部是相同的，所以 $\overline{BE}$ 始终和 $\overline{TDE}$ 相同。在这个例子上最优化 $\overline{BE}$ 会导致和在A-split例子中原生残差梯度算法同样的错误模式。

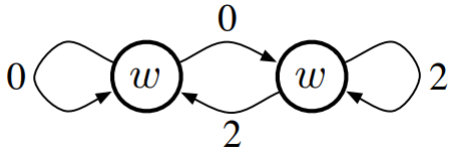
第三个残差梯度算法的收敛性的不满意的方面会在下节中解释。就像第二个方面，第三个方面也是用 $\overline{BE}$ 目标本身的问题而不是任何实现它的特定算法。

11.6 贝尔曼误差是不可学的

这节我们会引入可学能力的概念，它不同于机器学习中普遍使用的概念。在那里，一个猜想被称作"可学的"，如果它是**效率上**可学的，意味着它能够被样例数的多项式内而不是指数内学习。这里我们以一种更基本的方式来使用这个词语，意味着用任意数量的经历来说明可学性。可以证明，即使从无限数量的实验数据中，也不能学习到很多强化学习明显感兴趣的性质。这些性质是良定义的，且能够被给定的环境的内部结构的知识所计算，但不能由特征向量、行动和报酬的观察序列被计算或估计。我们称它们是**不可学的**。可以证明这两节所引入的贝尔曼误差目标 ($\overline{BE}$) 在这种意义下是不可学的。贝尔曼误差目标不能从观察数据中学习可能是不去寻找它的最有力原因。

为了使得可学性的概念更加清晰，让我们从一些简单例子开始。考虑两个Markov回报过程(MRPs) 如下图：





其中一个状态有两条边离开，每个迁移被认为是同等概率发生的，其上数字表示它收到的报酬。所有状态出现相同；它们都产生一个相同的单一分量的特征向量 $x = 1$ 且有近似值 w. 因此，数据轨迹唯一变化的部分是报酬序列。左边的MRP始终在相同的状态且随机发出0和2的无限流，其中每个概率为0.5. 右边的MRP，在每一步，要么呆在它当前的状态或者转换到另一个状态，等概率地。在这个MRP中报酬是确定性的，在一个状态一直是0，另一个状态一直是2，但由于每个状态都是每步等可能出现的，观察数据也将是一个随机的0和2的无限流，等价于左边MRP生成的。(我们可以假设右边的MRP等概率地从两

个状态中的随机某个状态开始。) 因此，即便给出无穷数量的数据，也不可能讲出是这两个MRP中的哪一个生成了它。特别地，我们不能讲出MRP有一个状态还是两个，是随机的还是确定性的。这些东西都是不可学的。

这对MRP也解释了 $\overline{VE}$ 目标 (9.1) 是不可学的。如果 $\gamma = 0$，那么这三个状态 (两个MRPs中) 的真值，从左到右是1，0，2. 假设 $w = 1$. 那么$\overline{VE}$ 对于左 MRP 是0，右 MRP 是1. 因为两个问题中的$\overline{VE}$ 是不同的，但数据生成有相同的分布，所以 $\overline{VE}$ 是不能被学习的。$\overline{VE}$ 不是数据分布的唯一函数。且如果它不能被学习，那么$\overline{VE}$ 怎么能成为对学习有用的目标呢?

如果一个目标不能被学习，它确实会使其效用受到质疑。然而在 $\overline{VE}$ 情形中却有一种可运作的方式。注意到相同的解 $w = 1$，是上述两个MRPs的最优值(假设 μ 对右边MRP中两个不可区分状态是相同的). 这是不是一个巧合，还是说它一般是对的，对于所有有相同数据分布的MDPs也有相同的最优参数向量? 如果这是对的——我们将之后证明这点——那么 $\overline{VE}$ 将仍是一个有用的目标。虽然 $\overline{VE}$ 不可学，但最优化的参数却是!

为了理解这点，引入另一个自然目标函数是有用的，这次它清晰地是可学的。始终可观察的一个误差是，每次的值估计和该次的返回之间的。**均方返回误差** (*mean square return error*)，记作 $\overline{RE}$，是在 μ 下这个误差的平方的期望。在在线策略情形中 $\overline{RE}$ 可以被写成

$$\begin{aligned}\overline{RE} &= \mathbb{E}\big[(G_t - \hat{v}(S_t, \mathbf{w}))^2\big] \\ &= \overline{VE}(\mathbf{w}) + \mathbb{E}\big[(G_t - v_\pi(S_t))^2\big]. \end{aligned} \tag{11.24}$$

因此，两个目标是相同的除了一个不依赖于参数向量的方差项。两个目标因此必须有相同的最优参数值 $\mathbf{w}^*$. 全部的关系在图11.4中有总结。

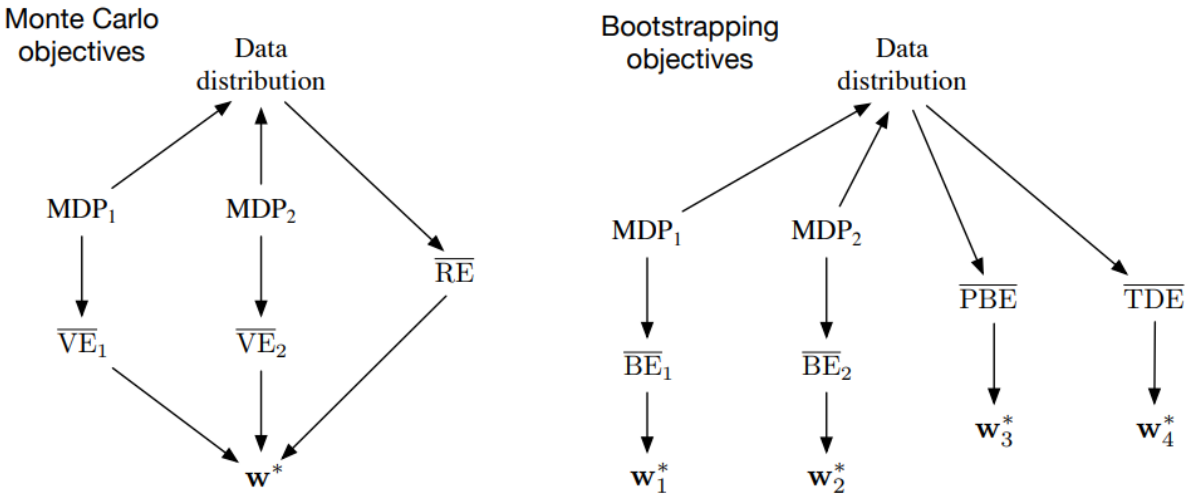
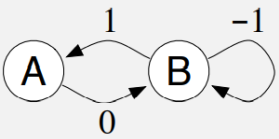
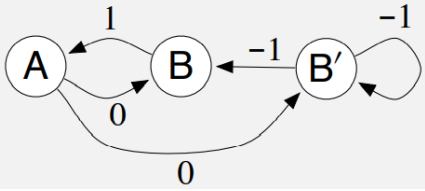


Figure 11.4: Causal relationships among the data distribution, MDPs, and various objectives. **Left, Monte Carlo objectives:** Two different MDPs can produce the same data distribution yet also produce different $\overline{\text{VE}}$s, proving that the $\overline{\text{VE}}$ objective cannot be determined from data and is not learnable. However, all such $\overline{\text{VE}}$s must have the same optimal parameter vector, $\mathbf{w}^*$! Moreover, this same $\mathbf{w}^*$ can be determined from another objective, the $\overline{\text{RE}}$, which *is* uniquely determined from the data distribution. Thus $\mathbf{w}^*$ and the $\overline{\text{RE}}$ are learnable even though the $\overline{\text{VE}}$s are not. **Right, Bootstrapping objectives:** Two different MDPs can produce the same data distribution yet also produce different $\overline{\text{BE}}$s *and* have different minimizing parameter vectors; these are not learnable from the data distribution. The $\overline{\text{PBE}}$ and $\overline{\text{TDE}}$ objectives and their (different) minima can be directly determined from data and thus are learnable.

> 例11.4: 贝尔曼的可学性的反例
>
> 为了展示全范围的可能性，我们需要一个比之前考虑的稍微更复杂的Markov回报过程(MRPs). 考虑以下两个MRPs:





其中当有两条边离开一个状态时，两个转移都假设是等概率发生的，上面的数字表示收到的报酬。左边的MRP有两个不同表示的状态。右边的MRP有三个状态，其中两个B和B'，出现相同，应该被给相同的近似值。特别地，$\mathbf{w}$ 有两个分量，第一个分量对状态A赋值，第二个分量对状态B和B'赋值。第二个MRP已经被设计成在所有三个状态上耗费时间相同，因此我们可取 $\mu(s) = \frac{1}{3}$，对所有 s.

注意到两个MRPs的可观察数据分布是完全相同的。在两个情形中，代理者会看到一个单一出现的A接着0，然后一些数目的明显的Bs(在右边MRP中视B'为B)，每个后面都接着 -1 除了最后一个接着是1，然后又再次开始于一个单一A和0，以此类推。所有统计细节也都是相同的；在两个MRPs中，一个长度为 k 的Bs字符串的概率是 $\frac{1}{2}(\frac{2}{3})^k$.

现在假设 $\mathbf{w} = \mathbf{0}$. 在第一个MRP中，这是一个精确解，且 $\overline{BE}$ 是0。在第二个MRP中，这个解会在B和B'中产生一个平方误差1，使得 $\overline{BE} = \mu(B)1 + \mu(B')1 = \frac{2}{3}$. 这两个MRPs，生成相同的数据分布，但有不同的 $\overline{BE}$；所以$\overline{BE}$ 是不可学的。

更多地 (不同于之前对 $\overline{VE}$ 的例子) 最小化的 $\mathbf{w}$ 的值在两个MRP中是不同的。对于第一个MRP，$\mathbf{w} = \mathbf{0}$ 最小化 $\overline{BE}$ 对于任意 γ. 对于第二个MRP，最小化的 $\mathbf{w}$ 是一个关于 γ 的复杂函数，但在极限情况下，即 $\gamma \to 1$，它是 $(-\frac{1}{3}, \frac{1}{6})^\top$. 因此最小化 $\overline{BE}$ 的解不能单独从数据中估计；超出数据所能揭露的MRP的知识是需要的。在这个意义上，原则上是不能追求 $\overline{BE}$ 作为一个学习目标的。

可能是令人惊讶的，在第二个MRP中 $\overline{BE}$ 最小化的A值是远离零的。回忆A有一个专用的权重因此它的值应该不受函数近似限制的。A接着的是一个报酬0以及转移到一个值接近为0的状态 (这里应该理解为两个状态B真值的平均)，这暗示着 $v_{\mathbf{w}}(A)$ 应该为零；为什么它的最优值实质上是负的而不是0？答案是令 $v_{\mathbf{w}}(A)$ 为负会减小从B回到A的误差。在这个确定性转移上的报酬是1，表明B应该有一个比A大1的值。因为总体的均值是近似于零的，A的值就会往-1走。$\overline{BE}$ 最小化的A值是约大于 $-\frac{1}{3}$，这是一个减少离开A和进入A的误差的妥协。

下面给出右边MRP的 $\overline{BE}$ 最小化时的 $\mathbf{w}$.

$$
\begin{aligned}
&BE(A) = \gamma w_2 - w_1, BE(B) = 1 + \gamma w_1 - w_2, BE(B') = -1 + \gamma w_2 - w_2 \\
&\overline{BE} = \frac{1}{3}[(\gamma w_2 - w_1)^2 + (1 + \gamma w_1 - w_2)^2 + (-1 + \gamma w_2 - w_2)^2] \\
&\quad = \frac{1}{3}[(1+\gamma^2)w_1^2 + 2(\gamma^2 - \gamma + 1)w_2^2 - 4\gamma w_1 w_2 + 2\gamma w_1 - 2\gamma w_2 + 2] \\
&\quad \begin{cases} \frac{\partial \overline{BE}}{\partial w_1} = \frac{1}{3}(2(1+\gamma^2)w_1 - 4\gamma w_2 + 2\gamma) \triangleq 0 \\ \frac{\partial \overline{BE}}{\partial w_2} = \frac{1}{3}(2(1+\gamma^2)w_1 - 4\gamma w_2 + 2\gamma) \triangleq 0 \end{cases} \\
&\Rightarrow \begin{cases} w_1 = \frac{2\gamma w_2 - \gamma}{1+\gamma^2} \\ w_2 = \frac{2\gamma w_1 + \gamma}{2(1-\gamma+\gamma^2)} \end{cases} \\
&\Rightarrow \begin{cases} w_1 = -\frac{\gamma}{1+\gamma+\gamma^2} \\ w_2 = \frac{\gamma}{2(1+\gamma+\gamma^2)} \end{cases}
\end{aligned}
$$

我们可以看到 $\mu(A)w_1 + \mu(B)w_2 + \mu(B')w_2 = 0$，也就是整体MRP期望返回为0，这与左MRP相同。

现在让我们回到 $\overline{BE}$. $\overline{BE}$ 是像 $\overline{VE}$ 的，因为它们都能从MDP的知识中计算但不能从数据中学习。但是不同于 $\overline{VE}$ 它的最小解是不可学的。下面的方框会展示一个反例——两个MRPs生成相同的数据分布但它们的最小化参数向量是不同的，证明了最优化参数向量不是数据的一个函数，因此不能从数据中学习。另外我们已经考虑过的拔靴目标 $\overline{PBE}$ 和 $\overline{TDE}$，能够从数据确定 (是可学的)，也能确定最优解，它们和 $\overline{BE}$ 的最小解一般互不相同。一般情况在图11.4中右半部分总结。

因此，$\overline{BE}$ 是不可学的；它不能够被特征向量和其它可观察数据估计。这使得 $\overline{BE}$ 局限于基于模型的设置。没有算法能不访问超出特征向量的内在的MDP状态来最小化 $\overline{BE}$. 残差梯度算法能够最小化 $\overline{BE}$ 因为它允许从相同的状态双重采样——不是一个相同特征向量的状态，而是一个保证是相同的内在状态。我们可以看到现在没有关于这的办法。最小化 $\overline{BE}$ 需要这种对象征性的内在的MDP的访问。这是 $\overline{BE}$ 一个重要的限制超出在前面的A-presplit示例中识别的限制。所有这些都使得对 $\overline{PBE}$ 有更多关注。

11.7 梯度TD方法

我们现在考虑SGD方法来最小化 $\overline{PBE}$. 作为真正的SGD的方法，这些**梯度TD方法** (*Gradient-TD methods*) 有稳定的收敛性质即便在离线策略训练和非线性函数近似下。记住在线性情形中始终有一个精确解，TD不动点 $\mathbf{w}_{TD}$，在该点 $\overline{PBE}$ 是零。这个解可以通过最小二乘法 (章节9.8) 被找到，但该方法有复杂度 $O(d^2)$ 关于参数的数量。我们转而寻找SGD方法，它只有 $O(d)$ 且有稳定收敛性质。梯度TD方法接近于实现这些目标，代价是几乎双倍计算复杂度。

为了推导一个对 $\overline{PBE}$ (假设线性函数近似) 的SGD方法我们从拓展和重写目标 (11.22) 为矩阵形式开始:

$$\begin{aligned}
\overline{PBE}(\mathbf{w}) &= \|\Pi\bar{\delta}_{\mathbf{w}}\|_\mu^2 \\
&= (\Pi\bar{\delta}_{\mathbf{w}})^\top \mathbf{D}\Pi\bar{\delta}_{\mathbf{w}} && (\text{from } (11.14)) \\
&= \bar{\delta}_{\mathbf{w}}^T \Pi^\top \mathbf{D}\Pi\bar{\delta}_{\mathbf{w}} \\
&= \bar{\delta}_{\mathbf{w}}\mathbf{D}\mathbf{X}(\mathbf{X}^\top\mathbf{D}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{D}\bar{\delta}_{\mathbf{w}} && (11.25) \\
(\text{使用}(11.13)\text{以及恒等 } \Pi^T\mathbf{D}\Pi &= \mathbf{D}\mathbf{X}(\mathbf{X}^\top\mathbf{D}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{D}) \\
&= (\mathbf{X}^\top\mathbf{D}\bar{\delta}_{\mathbf{w}})^\top(\mathbf{X}^\top\mathbf{D}\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{D}\bar{\delta}_{\mathbf{w}}). && (11.26)
\end{aligned}$$

关于 $\mathbf{w}$ 的梯度是

$$\nabla\overline{PBE}(\mathbf{w}) = 2\nabla[\mathbf{X}^\top\mathbf{D}\bar{\delta}_{\mathbf{w}}]^\top(\mathbf{X}^\top\mathbf{D}\mathbf{X})^{-1}(\mathbf{X}^\top\mathbf{D}\bar{\delta}_{\mathbf{w}}).$$

为了将这转变成一个SGD方法，我们必须在每个时间步上采样有这个性质的一些东西来作为它的期望值。取 μ 作为在行为策略下访问状态的分布。所有上述三个因子能被写成在这个分布下期望的形式。比如，最后的因子可以写成

$$\mathbf{X}^\top\mathbf{D}\bar{\delta}_{\mathbf{w}} = \sum_s \mu(s)\mathbf{x}(s)\bar{\delta}_{\mathbf{w}}(s) = \mathbb{E}[\rho_t\delta_t\mathbf{x}_t],$$

这就是半梯度TD(0)更新 (11.2) 的期望。第一个因子是这个更新的梯度的转置:

$$\begin{aligned}
\nabla\mathbf{E}[\rho_t\delta_t\mathbf{x}_t]^\top &= \mathbb{E}[\rho_t\nabla\delta_t^\top\mathbf{x}_t^\top] \\
&= \mathbf{E}[\rho_t\nabla(R_{t+1} + \gamma\mathbf{w}^\top\mathbf{x}_{t+1} - \mathbf{w}^\top\mathbf{x}_t)^\top\mathbf{x}_t^\top] \quad (\text{使用episodic } \delta_t) \\
&= \mathbb{E}[\rho_t(\gamma\mathbf{x}_{t+1} - \mathbf{x}_t)\mathbf{x}_t^\top].
\end{aligned}$$

最后，中间项是特征向量的外积矩阵的期望的逆:

$$\mathbf{X}^\top\mathbf{D}\mathbf{X} = \sum_s \mu(s)\mathbf{x}(s)\mathbf{x}(s)^\top = \mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top].$$

将 $\overline{PBE}$ 梯度中的表达式的三项替换成期望，我们得到

$$\frac{1}{2}\nabla\overline{PBE}(\mathbf{w}) = \mathbb{E}[\rho_t(\gamma\mathbf{x}_{t+1} - \mathbf{x}_t)\mathbf{x}_t^\top]\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbb{E}[\rho_t\delta_t\mathbf{x}_t] \qquad (11.27)$$

$$\nabla \overline{PBE}(\mathbf{w}) = 2\mathbb{E}[\rho_t(\gamma \mathbf{x}_{t+1} - \mathbf{x}_t)\mathbf{x}_t^\top]\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbb{E}[\rho_t\delta_t\mathbf{x}_t]. \tag{11.27}$$

可能把梯度写成这个形式不能明显看出我们所做的任何进展。它是一个三个表达式的积，且第一项和最后一项不是独立的。它们都依赖于下一个特征向量 $\mathbf{x}_{t+1}$; 我们不能简单地采样这两个期望然后将它们相乘。这会带来一个梯度的有偏估计就如在残差梯度算法中。

另一个想法是分别估计这三个期望然后将它们联合起来生成一个梯度的无偏估计。这确实能工作，但会需要大量计算资源，尤其是储存最开始的两个期望，它们都是 $d \times d$ 矩阵，而且需要计算第二项的逆。这个想法可以被改进。如果三个期望中的两个被估计和储存，那么第三个可以被采样然后联合两个存储的量一起使用。比如，你可以存储第二项的两个量的估计值(使用在章节 9.8中使用的递增逆更新技巧)，然后采样第一项。不幸的是，整体算法将仍然会有平方复杂度 (阶为 $O(d^2)$).

分别存储一些估计值然后将它们用采样联合起来是一个好的想法，它也在梯度TD方法中被使用。梯度TD方法估计然后存储在 (11.27) 的后两项。这些因子是一个 $d \times d$ 矩阵和一个 d维向量，所以它们的积仅仅是一个 d维向量，如 $\mathbf{w}$ 本身。我们记这第二个要学习的向量为 $\mathbf{v}$:

$$\mathbf{v} \approx \mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbb{E}[\rho_t\delta_t\mathbf{x}_t]. \tag{11.28}$$

这个形式被线性监督学习的学者所熟知。它是一个 试图从特征中近似 $\rho_t\delta_t$ 的线性最小二乘问题的解。标准SGD方法自增寻找向量 $\mathbf{v}$ 来最小化期望方差 $\left(\mathbf{v}^T\mathbf{x}_t - \rho_t\delta_t\right)^2$ 被称作**最小二乘均方 (LMS) 准则** (这里增加了一个重要性采样比率):

$$\mathbf{v}_{t+1} \doteq \mathbf{v}_t + \beta(\rho_t\delta_t - \mathbf{v}_t^\top\mathbf{x}_t)\mathbf{x}_t,$$

其中 $\beta > 0$ 是另一个步长参数。我们可以用这个方法高效实现 (11.28) 用 $O(d)$ 存储和每步计算量。

给定一个储存的估计值 $\mathbf{v}_t$ 来近似 (11.28)，我们可以使用基于 (11.27) 的SGD方法更新主要的参数向量 $\mathbf{w}_t$. 最简单的准则是

$$\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{2}\alpha\nabla\overline{PBE}(\mathbf{w}_t) && (\text{一般SGD准则})\\
&= \mathbf{w}_t - \frac{1}{2}\alpha 2\mathbb{E}[\rho_t(\gamma\mathbf{x}_{t+1} - \mathbf{x}_t)\mathbf{x}_t^\top]\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbb{E}[\rho_t\delta_t\mathbf{x}_t] && (\text{从 (11.27)})\\
&= \mathbf{w}_t + \alpha\mathbb{E}[\rho_t(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})]\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbb{E}[\rho_t\delta_t\mathbf{x}_t] && (11.29)\\
&\approx \mathbf{w}_t + \alpha\mathbb{E}[\rho_t(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})\mathbf{x}_t^\top]\mathbf{v}_t && (\text{基于 (11.28)})\\
&\approx \mathbf{w}_t + \alpha\rho_t(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})\mathbf{x}_t^\top\mathbf{v}_t. && (\text{采样})
\end{aligned}$$

这个算法被称为 $GTD2$. 注意到如果最终内积 $\mathbf{x}_t^T\mathbf{v}_t$ 被先完成，那么整体算法的复杂度是 $O(d)$.

一个稍微更好的算法能够通过在替换 $\mathbf{v}_t$ 之前做更多的分析步骤。接着 (11.29):

$$\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha\mathbb{E}[\rho_t(\mathbf{x}_t - \gamma\mathbf{x}_{t+1})]\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbb{E}[\rho_t\delta_t\mathbf{x}_t]\\
&= \mathbf{w}_t + \alpha(\mathbb{E}[\rho_t\mathbf{x}_t\mathbf{x}_t^\top] - \gamma\mathbb{E}[\rho_t\mathbf{x}_{t+1}\mathbf{x}^\top])\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbf{E}[\rho_t\delta_t\mathbf{x}_t]\\
&= \mathbf{w}_t + \alpha(\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top] - \gamma\mathbb{E}[\rho_t\mathbf{x}_{t+1}\mathbf{x}_t^\top])\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbb{E}[\rho_t\delta_t\mathbf{x}_t]\\
&= \mathbf{w}_t + \alpha\left(\mathbb{E}[\rho_\mathbf{t}\delta_\mathbf{t}\mathbf{x_t}] - \gamma\mathbb{E}[\rho_t\mathbf{x}_{t+1}\mathbf{x}_t^\top]\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^\top]^{-1}\mathbb{E}[\rho_t\delta_t\mathbf{x}_t]\right)\\
&\approx \mathbf{w}_t + \alpha\big(\mathbb{E}[\rho_t\delta_t\mathbf{x}_t] - \gamma\mathbb{E}[\rho_t\mathbf{x}_{t+1}\mathbf{x}_t^\top]\mathbf{v}_t\big) && (\text{基于 (11.28)})\\
&\approx \mathbf{w}_t + \alpha\rho_t(\delta_t\mathbf{x}_t - \gamma\mathbf{x}_{t+1}\mathbf{x}_t^\top\mathbf{v}_t), && (\text{采样})
\end{aligned}$$

其中如果最后乘积 ($\mathbf{x}_t^\top\mathbf{v}_t$) 被先完成，那么复杂度还是 $O(d)$. 这个算法被称作**带梯度校正的TD(0)** (*TD(0) with gradient correction, TDC*)，或者是 *GTD(0)*.

图11.5展示了一个样例和 TDC 在 Baird 反例上的期望性能。如预期的，$\overline{PBE}$ 会落到零，但注意到参数向量的单独分量并没有达到零。实际上，这些值仍然离最优解很远 $\hat{v}(s) = 0$，对所有 s，对于 $\mathbf{w}$ 必须与 $(1, 1, 1, 1, 1, 1, 4, -2)^\top$. 在1000次迭代之后我们仍离最优解很远，正如我们从 $\overline{VE}$ 中看到的，它还停留在2左右。这个系统实际上收敛到最优解，但这个过程极度缓慢因为 $\overline{PBE}$ 已经很接近0。

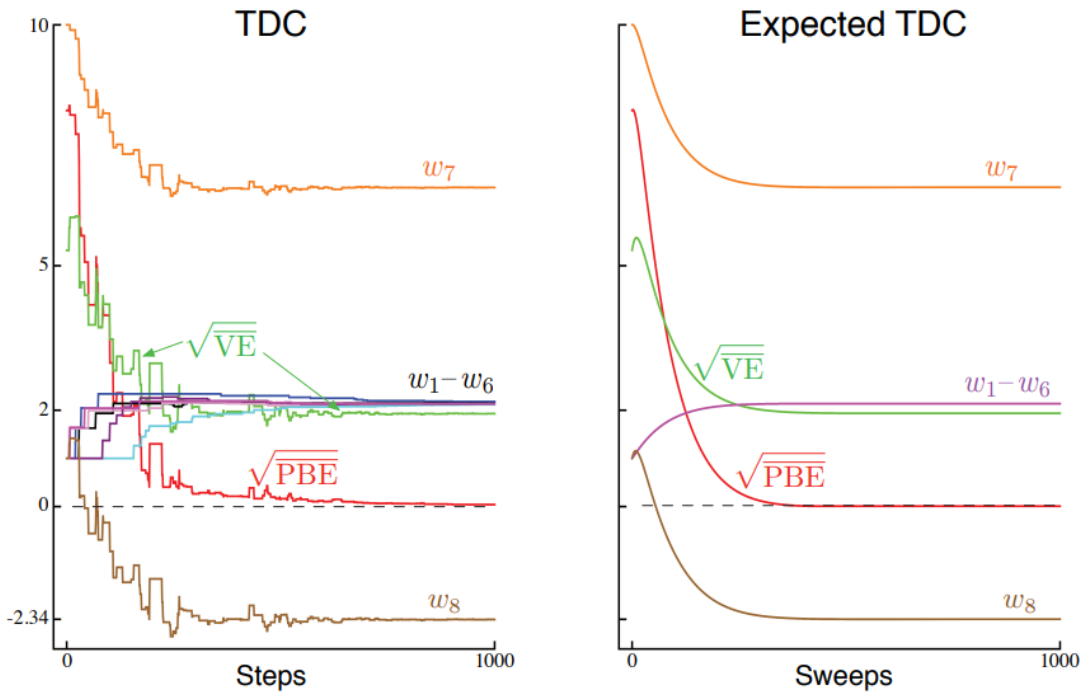


Figure 11.5: The behavior of the TDC algorithm on Baird's counterexample. On the left is shown a typical single run, and on the right is shown the expected behavior of this algorithm if the updates are done synchronously (analogous to (11.9), except for the two TDC parameter vectors). The step sizes were $\alpha = 0.005$ and $\beta = 0.05$.

GTD2 和 TDC 都包含两个学习过程，一个初级的对 $\mathbf{w}$，一个二级的对 $\mathbf{v}$. 初级学习过程的逻辑依赖于二级学习过程已经完成，至少近似地，当二级学习过程不再被一级的影响时。我们称这种不对成的依赖性为**级联**(*cascade*). 在级联中我们经常假设二级学习过程会进行得更快，因此始终在它的渐进值上，准备好并精确地帮助初级学习过程。这些方法的收敛性证明经常会明确做这个假设。这些被称作**双时间尺度** (two-time-scale) 证明。快的时间尺度是二级学习过程，慢的时间尺度是初级学习过程。如果 α 是初级学习过程的步长，β 是二级学习过程的步长，那么这些收敛性保证一般需要在极限情况下 $\beta \to 0$ 和 $\frac{\alpha}{\beta} \to 0$.

梯度TD方法是当前最好理解且应用最广的稳定离线策略方法。有到值函数和控制的拓展 (GQ, Maei et al., 2010) 以及到资格迹的拓展 (GTD(λ) 和 GQ(λ), Maei, 2011; Maei and Sutton, 2010)，还有到非线性函数近似的拓展 (Maei et al., 2009). 也有提出介于半梯度TD和梯度TD的自适应算法 (Hackman, 2012; White and White, 2016). 自适应TD算法在目标和行为策略非常不同的状态上表现得像梯度TD算法，在目标和行为策略相同的状态上表现得像半梯度算法。最后，梯度TD的思想已经和近端方法和控制变量的思想结合来生成更高效的方法 (Mahadevan et al., 2014; Du et al., 2017).

11.8 重要性TD方法

我们现在转向第二个主要策略，它已经被广泛探索，为了得到一个廉价且高效的用函数近似的离线策略学习方法。回忆线性半梯度TD方法当我在在线策略分布上训练时是高效且稳定的，且我们在章节9.4中展示的这必须工作在矩阵 $\mathbf{A}$ <!-在离线策略情形中，矩阵 $\mathbf{A}$ 一般定义成 $\mathbb{E}_{s\sim b}[\mathbf{x}(s)\mathbb{E}[\mathbf{x}(S_{t+1})^\top | S_t = s, A_t \sim \pi]]$.--->是正定的且在目标策略下在线策略状态分布 μ_π 和状态转移概率 $p(s|s,a)$ 之间是匹配的。在离线策略学习中，我们使用重要性采样重新对状态转移赋权重使得他们能够变得适合于学习目标策略，但状态分布仍然是行为策略。存在一个不匹配。一个自然的想法是也对状态进行再赋权，强调其中一些，抑制其它的，使得返回更新分布为在线策略分布。然后会有一个匹配，稳定性和收敛性将会遵循已存在的结果。这是**重要性TD方法** (*Emphatic-TD method*) 的思想，在章节9.11中首次引入为了在线策略训练。

实际上，"在线策略分布"的记号不是很准确，因为有很多在线策略分布，且任意其中一种都充分保证稳定性。考虑一个无折扣的episodic问题。episodes终止的方式完全取决于转移概率，但episodes的开始可以有几种不同的方式。然而episodes开始，如果所有状态转移都按照目标策略，那么状态分布结果是一个在线策略分布。你可以从靠近终止状态开始然后只访问高概率在episode结束前的少量状态。或者你可以从远离终止状态开始然后在终止前通过很多状态。两种都是在线策略分布，而且用线性半梯度方法来训练两个都会保证是稳定的。但是当过程开始后，只要更新所有遇到的状态直到终止前就能得到在线策略分布。

如果有折扣，那么它可以被看作是部分或概率性终止为了这些目的。如果 $\gamma = 0.9$，那么我们可以认为在每个时间步上有0.1的概率过程终止，然后立即从它转移到的状态上重新开始。一个折扣问题是一个在每一步上持续终止和以 $1-\gamma$ 的概率重启。这种思考折扣的方式是一种更一般的**伪终止** (*pseudo termination*) 概念——终止不影响状态转移的序列，但会影响学习过程和学习的量。这种伪终止对离线策略学习是重要的因为重启是可选的——记住我们可以以我们想要的任何方式开始——终止解放了将遇到的状态保持在在线策略分布中的需要。即，我们如果不考虑新的状态作为重启，那么折扣会很快给我们一个有限的在线策略分布。

用来学习episodic状态值的单步强调TD算法被定义成:

$$
\begin{aligned}
\delta_t &= R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}_t) - \hat{v}(S_t, \mathbf{w}_t), \\
\mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha M_t \rho_t \delta_t \nabla \hat{v}(S_t, \mathbf{w}_t), \\
M_t &= \gamma \rho_{t-1} M_{t-1} + I_t,
\end{aligned}
$$

其中 I_t 是**兴趣**，是任意的，M_t 是重要度，被初始化为$M_{-1} = 0$. 这个算法在Baird反例中表现得怎样？图11.6展示了参数分量期望的轨迹 (这里$I_t = 1$，对所有的 t). 有一些振荡但最终所有都会收敛且 $\overline{VE}$ 会收敛到0. 这些轨迹通过迭代计算参数向量估计的期望得到的，没有任何转移和报酬的采样方差。我们不显示直接运用重要性TD算法，因为它在Baird反例中的方差非常大以致它几乎不可能在计算实验中得到一致的结果。算法理论上在这个问题上收敛到最优解，但实际上它没有。我们会在下一节讨论减小所有这些算法的方差的主题。

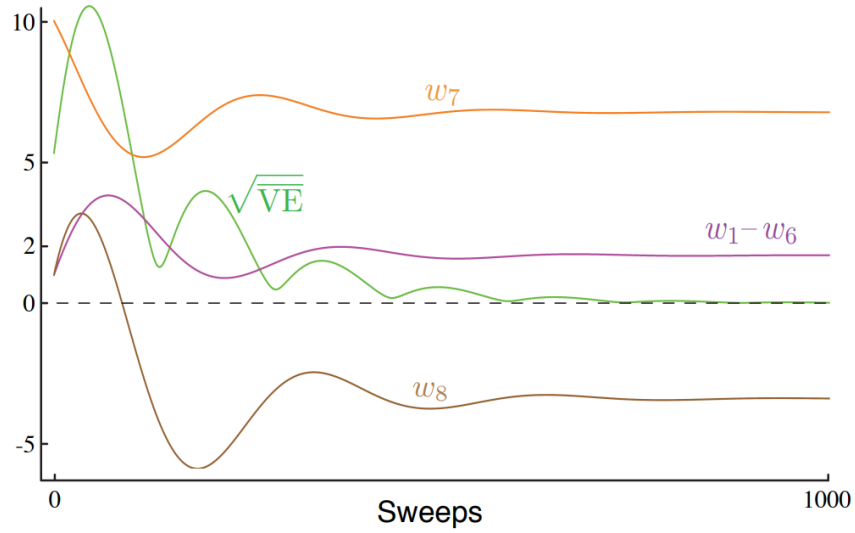


Figure 11.6: The behavior of the one-step Emphatic-TD algorithm in expectation on Baird's counterexample. The step size was $\alpha = 0.03$.

11.9 减小方差

离线策略学习固有地比在线策略学习有更大的方差。这并不令人惊讶；如果你得到的数据与一个策略相关性越小，你应该期待对策略的值学到得越少。在极端情形中，我们可能什么都没学到。比如你不能期待从烧晚饭中学习如何开车。只有目标和行为策略是相关的，如果它们访问相似的状态，采取相似的行动，才能够在离线策略训练中得到有意义的成果。

另一方面，任何策略都有很多邻策略，很多相似的策略在状态访问和行动选择上有大量的重叠，但这些策略并不完全相同。离线策略学习存在的理由是能够归纳这类大量的相关但不相同的策略。存在的问题是如何最好地利用经验。既然我们有在期望值上稳定的方法 (如果步长正确设置)，注意力自然转向了减少估计值的方差。有很多可能的想法，我们在这个导论书中只能接触其中一小些。

为什么控制方差在基于重要性采样的离线策略方法中尤其重要？如我们已经所见，重要性采样经常包含策略比率的乘积。这些乘积总是期望上为1 (5.13)，但它们的实际值会非常高或者低到0。相继的比率是无关的，所以它们的乘积期望上也一直是1，但它们可以有非常高的方差。回忆在SGD方法中那些乘以步长的比率，过高的方差意味着采取的步长是变化得非常大的。这对SGD是有问题的，因为会有偶尔非常大的步。它们必须不能变得如此大，否则会把参数带到非常不同梯度的空间的一部分中去。SGD方法依赖于在很多步上平均来得到一个良意义的梯度，如果它们从单个样例中做大的移动，它们就会变得不可靠。如果步长参数被设置得足够小来避免这，那么期望步长也会变得非常小，导致学习非常缓慢。Polyak-Ruppert平均 (Polyak, 1990; Ruppert, 1988; Polyak and Juditsky, 1992) 的动量的概念 (Derthick, 1984)，或者这些想法的更远的拓展可能会有重要的帮助。为参数向量的不同分量分别自适应地设置步长的方法也是恰当的 (例如，Jacobs, 1988; Sutton, 1992b, c), 以及 Karampatziakis 和 Langford (2010) 的"重要性权重感知"(importance weight aware) 更新。

在章节5中我们看到了加权重要性采样是如何显著表现得更好，且有更低的方差更新，相较于一般的重要性采样。但是，将加权重要性采样适应于函数近似是具有挑战性的，它很可能只能在复杂度为 $O(d)$ 下近似完成 (Mahmood and Sutton, 2015).

后援树算法 (章节7.5) 展示了可能实现一些不使用重要性采样的离线策略学习。这个想法已经被拓展到离线策略情形来得到稳定且更高效的方法 (Munos, Stepleton, Harutyunyan, and Bellemare, 2016 and Mahmood, Yu and Sutton, 2017).

另外，互补策划来允许目标策略部分被行为策略决定，以一种目标策略不会和行为策略非常不同而产生大的重要性比率的方式。比如，目标策略可以通过参考行为策略来定义，如 Precup et al. 提出的"识别器"。

11.10 小结

离线策略学习是一个诱人的挑战，考验我们在设计稳定和有效学习算法的才智。列表性Q-learning 使得离线策略学习似乎很简单，而且它有自然的泛化到期望Sarsa和后援树算法。但如我们这章所见，这些想法到重要的函数近似的拓展，甚至是线性函数近似，包含新的挑战并且迫使我们加深对强化学习算法的理解。

为什么要在这上费工夫？一个原因是寻找离线策略算法能够为处理探索和开发之间的平衡提供灵活性。另一个原因是从学习中解放行为，并且避免目标策略的专制。TD学习似乎支持并行学习多个事物的可能，即用一个经验流同步解决很多任务。我们确实能在特殊情形中做到这，但不是我们想要的每种情形或我们想要的那么高效。

在这章中我们将离线策略学习的挑战分为两个部分。第一部分，校正从行为策略学习的目标，可以直接明了地使用之前在列表性情形中设计的技巧来处理，尽管代价是增加更新的方差且因此会减慢学习。高方差很可能会一直作为离线策略学习的一个挑战。

离线策略学习的第二部分的挑战是以包含拔靴的半梯度TD方法的不稳定性显露的。我们寻找强力的函数近似，离线策略学习，和拔靴TD方法的效率及灵活性，但将这死亡三项的所有三个方面在一个算法中结合起来且不引入潜在的不稳定性是具有挑战的。有几种尝试。最流行的是寻找实现真正的随机梯度下降 (SGD) 在贝尔曼误差上 (亦称贝尔曼残差). 然而，我们的分析推断出在很多情形下这不是一个有吸引力的目标，且无论如何它不可能用一个学习算法实现——$\overline{BE}$ 的梯度从仅由特征向量而没有内在状态所揭露的经验中是不可学的。另一个方法，梯度TD方法，实现在**投射**贝尔曼误差中的SGD。$\overline{PBE}$ 的梯度是用复杂度 $O(d)$ 可学的，但是代价是一个有第二个步长的第二个参数向量。最新的方法族，重要性TD算法，改进对重新对更新赋权的旧想法，强调一些并抑制另外的。在这种方式中它们恢复了用计算上简单的半梯度方法使在线策略学习变得稳定的特殊性质。

整个离线策略学习的领域是相对新的且未解决的。哪种方法是最好的甚至是恰当的都还不清楚。在这章最后介绍的新的方法的复杂度是否真的需要？他们当中哪种能够有效地结合方差减少方法？离线策略学习的潜能仍然是撩人的，实现它的最好方式仍然是一个谜。