第 5 章

时序差分学习

本章提要

时序差分学习是强化学习理论的核心内容之一，与动态规划方法和蒙特卡洛方法相比，时序差分学习的主要特点是采用自举法对代价函数进行估计。

本章的内容组织如下：在 5.1 节中，介绍时序差分学习的基本概念和在生物上的应用；在 5.2 节中，详细地介绍时序差分算法；在 5.3 节中，介绍 n 步回报的概念；在 5.4 节中，进一步介绍使用 λ 回报的时序差分算法。

5.1 时序差分学习基本概念

时序差分学习(Temporal Difference Learning，TDL)是一种无模型的强化学习方法，它采用自举法，通过估计当前的代价函数来学习[1]。时序差分(Temporal Difference，TD)算法最早由 Sutton 提出，他证明时序差分学习可以和有监督学习获得同样的结果而且占用更少的内存，并且具有更快的收敛速度[2]。TD 学习的主要特点是，代价函数是通过自举法(Bootstrapping)来更新的。在统计学上，自举法是一种通过对样本进行重采样得到的估计总体的方法。对于 TD 算法来说，就是当前代价函数本身产生代价函数的更新目标，这一点显然不同于有监督学习中的概念[3]。

作为对比，蒙特卡洛方法只在最终结果已知的情况下调整其估计，而 TD 方法则可以随时调整预测。具体来说，蒙特卡洛方法是模拟(或经历)一段序列，在序列结束后，根据序列上各个状态的回报值来估计系统状态代价。时序差分学习是模拟一段序列，每行动一步(或几步)，根据新状态的代价，然后估计执行前的状态代价。可以认为蒙特卡洛方法是最大步数的时序差分学习。

另一方面时序差分法与动物学习的时序差分模型有关。TD 算法在神经科学领域也受到关注。研究人员发现，腹侧被盖区(Ventral Tegmental Area，VTA)和黑质致密部(Substantia Nigra Pars Compacta，SNC)中多巴胺神经元的放电率似乎模拟了算法中的误

差函数[4]。误差函数报告任何给定状态或时间步骤的估计奖励与实际收到的奖励之间的差异。误差函数越大，预期奖励和实际奖励之间的差异越大。当它与准确地反映未来奖励的刺激配对时，错误可用于将刺激与未来奖励相关联。多巴胺细胞表现出类似的行为。在一个实验中，多巴胺细胞的测量是在训练猴子以将刺激与果汁的奖励相关联时进行的[5]。最初，当猴子接受果汁时，多巴胺细胞提高了发放率，表明预期和实际奖励的差异。随着时间的推移，这种反击的增加会传播到最早可靠的奖励刺激。一旦猴子接受了充分的训练，在达到预测奖励后，放电速率不会增加。当不产生期望的奖励时，多巴胺细胞放电速率持续下降至低于正常激活。这很好地模拟了 TD 中的误差函数如何用于强化学习。TD 方法最成功的应用是 IBM 的研究员 Gerald Tesauro 根据时序差分和神经网络编制的西洋双陆棋程序 TD-Gammon，棋力可以和最好的人类棋手相媲美。

TD 方法的优势在于可以根据不完整的轨迹学习，相较于蒙特卡洛方法，应用范围更广。TD 方法的不足之处也是显而易见的，TD 方法的更新目标是估计值，很难做到无偏估计。TD 目标的方差(Variance)比较低，也就是波动性小[6]。

5.2 时序差分学习算法

我们从第 4 章的蒙特卡洛方法引出本章将要介绍的 TD 方法。将学习律设置为常数的蒙特卡洛方法的更新公式如下：

$$J(x_k) \leftarrow J(x_k) + \alpha(G_k - J(x_k)) \tag{5.1}$$

其中，G_k 是每条状态轨迹结束后在状态 x_k 获得的回报值，α 是学习律。蒙特卡洛方法更新公式的含义就是用实际回报值 G_k 作为代价函数 $J(x_k)$的估计值。具体做法是针对每个完整轨迹，考察实验中 x_k 的回报值 G_k 和当前代价函数 $J(x_k)$的偏差，并用该偏差值乘以学习律来更新得到 $J(x_k)$的新估值。从蒙特卡洛方法的更新公式中，我们注意到只有在整个状态轨迹进行完以后，才能得到状态轨迹上每个状态 x_k 对应的回报值 G_k。倘若一个状态轨迹非常长，那么就需要等待很长时间才能对 $J(x_k)$进行更新，这为蒙特卡洛方法带来了诸多不便。

现在我们对蒙特卡洛方法的更新公式做如下修改，把 G_k 换成 $R_{k+1}+\gamma J(x_{k+1})$，就得到了 TD 方法的代价函数更新公式：

$$J(x_k) \leftarrow J(x_k) + \alpha(R_{k+1} + \gamma J(x_{k+1}) - J(x_k)) \tag{5.2}$$

与蒙特卡洛方法用实际回报值 G_k 作为代价函数 $J(x_k)$的估计值相比，我们使用 $R_{k+1}+\gamma J(x_{k+1})$作为代价函数 $J(x_k)$的估计值。尽管只做了这一点修改，但是我们实际上已经解决了蒙特卡洛方法只能在轨迹结束后更新的缺点，即我们只需要知道 $(x_k, u_k, R_{k+1}, x_{k+1})$这组数据，就可以执行 TD 算法更新公式了，而这组数据是在做出控制 u_k，得到效用函数 R_{k+1}，观察到下个状态 x_{k+1} 后就可以获得的。

那么，为什么修改成这种形式呢？我们回忆一下代价函数的定义：

$$J_\pi(x) = E_\pi\{R_{k+1} + \gamma J_\pi(X_{k+1}) \mid X_k = x\} \tag{5.3}$$

容易发现期望符号里面就是 TD 更新公式的目标值。图 5-1 给出了 TD 算法更新公式的结构图。

蒙特卡洛和 TD 的更新公式有何不同呢？我们从以下几个方面分别说明。

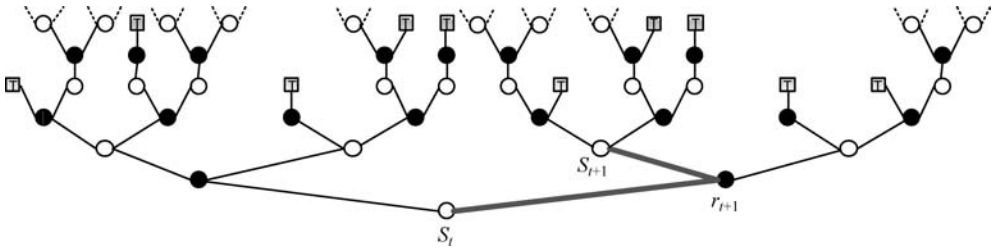

图 5-1　TD 算法的结构图

第一，从算法更新的时刻上，TD 方法在智能体运行过程中的每一时刻都更新状态 x_k 对应的代价函数 $J(x_k)$。而蒙特卡洛方法在智能体的整条轨迹结束之后才计算每个状态对应的回报值并做相应的更新。

第二，蒙特卡洛方法使用 G_k 作为对代价函数 $J(x_k)$更精确的估计，而 TD 方法使用 $R_{k+1}+\gamma J(x_{k+1})$作为代价函数 $J(x_k)$更精确的估计，二者在统计性质上表现出了诸多不同。对于蒙特卡洛方法来说，G_k 是 $J(x_k)$的无偏估计。然而，回报值 G_k 的计算涉及整个轨迹上的控制、转移状态和效用，如此多的随机因素使得将 G_k 作为 $J(x_k)$的估计带来了很大的方差。对于 TD 方法来说，由于目标值 $R_{k+1}+\gamma J(x_{k+1})$中的 $J(x_{k+1})$是不准确的，因此 $R_{k+1}+\gamma J(x_{k+1})$是 $J(x_k)$的有偏估计。然而，TD 目标 $R_{k+1}+\gamma J(x_{k+1})$只包含一个时刻的控制、转移状态和效用，随机因素较少，这使得将 $R_{k+1}+\gamma J(x_{k+1})$作为 $J(x_k)$的估计具有更小的方差。总的来说，蒙特卡洛是高方差、零偏差的方法，而 TD 是低方差、有偏差的方法。

第三，从算法的收敛性来看，蒙特卡洛方法具有较好的收敛性，且与代价函数的初始值无关。而 TD 方法虽然也具有收敛性，但是会受到代价函数初始值的影响。

第四，从算法的有效性来看，对于有限 MDP 问题，蒙特卡洛和 TD 都是收敛的。但是如果只有有限的经验，则蒙特卡洛和 TD 的有效性差距就显现出来了。我们用下面这个例子[7]来说明经验有限的情况下，蒙特卡洛和 TD 算法学习结果的不同。假设在一个 MDP 中，有两个状态 x_1，x_2，折扣因子 $\gamma=1$，我们有以下 8 段经验：

经验 1：x_1-0-x_2-0　　经验 2：x_2-1

经验 3：x_2-1　　经验 4：x_2-1

经验 5：x_2-1　　经验 6：x_2-1

经验 7：x_2-1　　经验 8：x_2-0

x_1-0-x_2-0 表示智能体开始于状态 x_1，从 x_1 转移到 x_2，效用函数为 0，从 x_2 转移到终止状态，效用函数为 0。x_2-1 表示智能体开始于状态 x_2，从 x_2 转移到终止状态，效用函数为 1。x_2-0 表示智能体开始于状态 x_2，从 x_2 转移到终止状态，效用函数为 0。

我们估计状态 x_2 的代价函数。状态 x_2 在这 8 段状态-效用轨迹（以下简称轨迹）中出现了 8 次，其中回报为 0 的是第 1 段轨迹和第 8 段轨迹，共两次；回报为 1 的是第 2 段轨迹到第 7 段轨迹。因此，我们有

$$J(x_2)=E\{G_k(x_2)\}=\frac{6}{8}=0.75 \tag{5.4}$$

现在我们使用蒙特卡洛方法估计状态 x_1 的代价函数。状态 x_1 只在第一段轨迹中出

现了一次，因此，我们有

$$J(x_1)=E\{G_k(x_1)\}=0 \tag{5.5}$$

现在使用 TD 方法估计状态 x_1 的代价函数，为了简便起见，设置 TD 更新公式中的 $\alpha=1$。根据 TD 的代价函数更新公式，我们有

$$J(x_1)\leftarrow J(x_1)+\alpha[0+\gamma J(x_2)-J(x_1)] \tag{5.6}$$

从而 $J(x_1)=0.75$。显然，使用蒙特卡洛方法和 TD 对状态 x_1 的代价函数进行估计，得到了不同的结果。

我们对这种不同的结果进一步分析。蒙特卡洛方法使用状态的回报值作为对代价函数的估计，因此，使用蒙特卡洛方法得到的代价函数的估计值是使平方误差 $\sum_{j=1}^{K}(G_k^j-J(x_k))^2$ 达到最小的估计值。而 TD 方法在有限轨迹上的收敛结果则是这些有限轨迹对应的最大似然马尔可夫模型的解。对上面这个例子来说，轨迹 1 中出现状态 x_1 转移到状态 x_2 的情况，且效用函数为 0。因此，在我们建立的最大似然马尔可夫模型中，从状态 x_1 到状态 x_2 的转移概率为 100%，且效用函数一定为 0。在这 8 段轨迹中，状态 x_2 的下一个状态都是终端状态，但是有两段轨迹得到的效用为 0，6 段轨迹得到的效用为 1，因此，在我们建立的最大似然马尔可夫模型中，从状态 x_2 到终端状态的转移概率为 100%，有 25%的概率获得效用 0，有 75%的概率获得效用 1，得到最大似然马尔可夫模型如图 5-2 所示。从图中可以看到，$J(x_1)=J(x_2)=0.75$。总的来说，蒙特卡洛方法和 TD 方法之所以在有限轨迹得到的结果不同，是因为蒙特卡洛方法在有限轨迹上的收敛结果是最小二乘误差的解，而 TD 方法在有限轨迹上的收敛结果则是这些有限轨迹对应的最大似然马尔可夫模型的解。

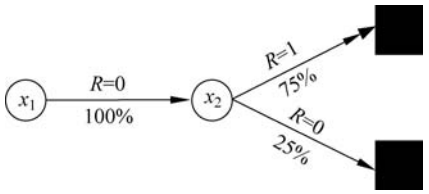

图 5-2　状态转移图

第五，蒙特卡洛方法不需要利用模型的马尔可夫性，而 TD 方法需要利用马尔可夫性。具体来说，蒙特卡洛方法不需要知道 x_k 后面是谁，而只需知道 R_k 就行了，并且蒙特卡洛方法在非马尔可夫环境下同样有效。然而，TD 方法需要知道 x_k 后面的 x_{k+1} 是谁，它在马尔可夫环境下更有效。

我们将 TD 算法的实施步骤总结在算法 5.1 中。

算法 5.1：时序差分学习算法

步骤 1：给定待估计的策略 π，初始状态分布 D，初始化代价函数 $J(x)$(e.g. $J(x)=0$, $\forall x\in X$)，给定学习律 α，令 $k=0$。

步骤 2：重复(对所有轨迹)：

如果 x_k 为初始状态，则初始化状态 $x_k\sim D$。

重复(对每步状态转移)：

由策略 π 得到控制 $u_k\sim\pi(x_k)$。

执行控制 u_k，观察效用 R_{k+1} 和下一个状态 x_{k+1}。

更新 $J(x_k)\leftarrow J(x_k)+\alpha[R_{k+1}+\gamma J(x_{k+1})-J(x_k)]$。

$k \leftarrow k+1$。
直到 x_k 为终端状态。
步骤 3：直到 $J(x)$收敛。
步骤 4：输出 $J(x)$。

5.3　n 步回报

我们将 TD 方法的学习目标 $G_k^{(1)}=R_{k+1}+\gamma J(x_{k+1})$视为 1 步回报，将蒙特卡洛方法的学习目标 $G_k=R_{k+1}+\gamma R_{k+2}+\cdots$视为无穷步回报。那么很自然地可以想到，是否可以将二者结合，在 TD 方法中增加回报的计算步数呢？

下面考虑以下几种 n 步回报。1 步回报为 $G_k^{(1)}=R_{k+1}+\gamma J(x_{k+1})$，2 步回报为 $G_k^{(2)}=R_{k+1}+\gamma R_{k+2}+\gamma^2 J(x_{k+2})$，无穷步回报为 $G_k^{(\infty)}=R_{k+1}+\gamma R_{k+2}+\cdots$。由此我们定义 n 步回报 $G_k^{(n)}=R_{k+1}+\gamma R_{k+2}+\cdots+\gamma^{n-1}R_{k+n}+\gamma^n J(x_{k+n})$。

在 5.2 节中的 TD 算法更新公式中的 1 步回报为 $G_k^{(1)}=R_{k+1}+\gamma J(x_{k+1})$改为 n 步回报，就得到了 n 步时序差分学习的更新公式：

$$J(x_k) \leftarrow J(x_k)+\alpha(G_k^{(n)}-J(x_k)) \tag{5.7}$$

用不同步数计算的回报值更新代价函数，就得到不同的 TD 算法，如图 5-3 所示。

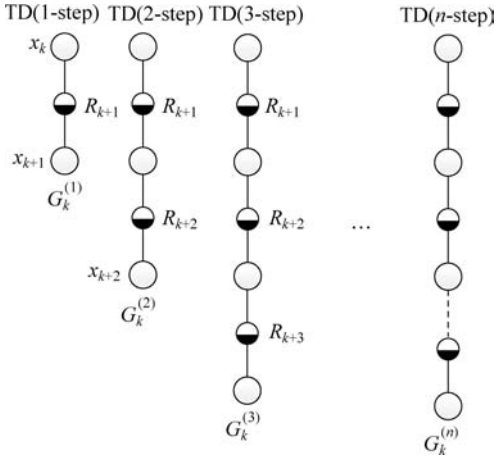

图 5-3　使用 n 步回报更新代价函数

那么，一个自然的问题就是，使用多少步计算的回报更新代价函数更好呢？下面将进行数学分析。首先，考虑 1 步回报和代价函数真值之间的误差。

$$\begin{aligned}\max_{x_k}\left|E\left[G_k^{(1)}\right]-J_\pi(x_k)\right| &= \max_{x_k}\left|E\left[R_{k+1}+\gamma J(x_{k+1})\right]-J_\pi(x_k)\right| \\ &= \max_{x_k}\left|\left[\mathcal{T}^\pi(V)\right](x_k)-J_\pi(x_k)\right| \\ &\leqslant \gamma\left\|J-J_\pi\right\|\end{aligned} \tag{5.8}$$

其中，$\mathcal{T}^{\pi}$是贝尔曼期望算子，$\mathcal{T}^{\pi}(J)=\mathcal{R}^{\pi}+\gamma\,\mathcal{P}^{\pi}J$。令$\underbrace{\mathcal{T}^{\pi}\circ\cdots\circ\mathcal{T}^{\pi}}_{n}(J)$表示 n 次映射，接下来我们考虑 n 步回报与代价函数真值的期望误差

$$\begin{aligned}
&\max_{x_k}\left|E\left[G_k^{(n)}\right]-J_{\pi}(x_k)\right|\\
=&\max_{x_k}\left|E\left[R_{k+1}+\gamma R_{k+2}+\cdots+\gamma^{n-1}J(x_{k+n})\right]-J_{\pi}(x_k)\right|\\
=&\max_{x_k}\left|E\left[R_{k+1}+\gamma E\left[R_{k+2}+\cdots+\gamma E\left[R_{k+n}+\gamma J(x_{k+n})\right]\cdots\right]\right]-J_{\pi}(x_k)\right|\\
=&\max_{x_k}\left|\left[\underbrace{\mathcal{T}^{\pi}\circ\cdots\circ\mathcal{T}^{\pi}}_{n}(J)\right](x_k)-J_{\pi}(x_k)\right|\\
\leqslant&\gamma^{n}\left\|J-J_{\pi}\right\| \qquad (5.9)
\end{aligned}$$

由于 $\gamma<1$，因此使用 n 步回报作为代价函数的估计值有助于提高学习的准确性。n 越大，回报与代价函数真值之间的偏差就越小。当 n 趋于无穷时，就变成了蒙特卡洛方法。但是，当 n 过大时，会带来与蒙特卡洛方法一样的问题，即估计值的方差很大。那么我们能否对不同的 n 步回报求均值，从而得到一种具有小偏差和小方差的估计值呢？这就引出了 TD(λ)算法。

5.4 TD(λ)算法

TD(λ)是 TD 的一种延伸的算法，它是由理查德·S. 萨顿(Richard S. Sutton)发明的一种学习算法，该算法是由阿瑟·塞缪尔(Arthur Samuel)在早期关于时序差分学习的研究中提出的。TD(λ)被杰拉德·泰索罗(Gerald Tesauro)用来创建 TD-Gammon，这是一个能够达到人类专家水平的游戏程序[8]。

在介绍 TD(λ)算法之前，我们首先给出 λ 回报 $G_k^{(\lambda)}$ 的概念。$G_k^{(\lambda)}$ 是所有 n 步回报的加权和。参数 λ 是跟踪衰减参数，范围是 0 到 1 的闭区间。定义：$G_k^{(n)}=R_{k+1}+rR_{k+2}+\cdots+r^{n-1}R_{k+n}+r^{h}V(S_{t+n})$对于无穷长的轨迹，对不同的 n 步回报施加权重$(1-\lambda)\lambda^{n-1}$，我们得到 G_k^{λ} 的计算公式

$$G_k^{\lambda}=(1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_k^{(n)} \qquad (5.10)$$

对于有限长度的轨迹，设终止时刻为 T，则 G_k^{λ} 的计算公式为

$$G_k^{\lambda}=(1-\lambda)\sum_{j=1}^{T-k-1}\lambda^{j-1}G_k^{(j)}+\lambda^{T-k-1}G_k \qquad (5.11)$$

我们使用 G_k^{λ} 作为对代价函数的估计，既利用了长步数回报的精度，又降低了高方差的影响。当 $\lambda=1$ 时，就相当于蒙特卡洛强化学习算法。当 $\lambda=0$ 时，就相当于前面介绍的 TD 算法，因此，前面介绍的 TD 算法又称为 TD(0)算法。

利用 G_k^{λ} 来更新当前状态的代价函数的方法称为 TD(λ)方法。TD(λ)算法有两种形式，一种是前向 TD(λ)，一种是后向 TD(λ)。

前向 TD(λ)通过状态 x_k 的 λ 回报 G_k^{λ} 来估计该状态的代价函数，更新公式为

$$J(x_k)\leftarrow J(x_k)+\alpha(G_k^{\lambda}-J(x_k)) \qquad (5.12)$$

其中 $G_k^{\lambda}=(1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_k^{(n)}$，而 $G_k^{(n)}=R_{k+1}+\gamma R_{k+2}+\cdots+\gamma^{n-1}R_{k+n}+\gamma^{n}J(x_{k+n})$。

前向 TD(λ)的执行方式与蒙特卡洛算法是类似的，只有当整个轨迹终止时，才能计算轨迹上状态的 λ 回报 G_k^λ，它和蒙特卡洛算法一样是一种离线学习的方式。不同的是，蒙特卡洛算法使用 G_k 作为代价函数的估计值，而前向 TD(λ)使用 G_k^λ 作为代价函数的估计值。

图 5-4 是对前向 TD(λ)的形象解释，当获取了由状态 x_k 到终端状态的整条轨迹后，我们就计算轨迹上每个状态对应的回报值，再计算每个状态对应的 λ 回报，然后根据前向 TD(λ)的更新公式更新这些状态的代价函数。

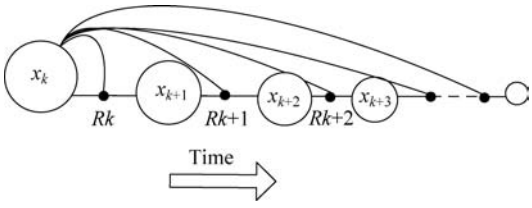

图 5-4　前向 TD(λ)

从上面的叙述中可以看到，利用前向 TD(λ)估计代价函数时，$G_k^{(\lambda)}$ 的计算需要知道未来时刻的观测量，因此需要等到整个轨迹结束后才能计算 $G_k^{(\lambda)}$。那么 TD(λ)能否像 TD(0)算法一样，实现在线更新呢？答案是可以的，下面我们就来介绍后向 TD(λ)。

为了实现 TD(λ)的在线更新，我们引入一个变量，名为资格迹。我们为 MDP 中的每个状态都定义一个资格迹，轨迹开始时，资格迹为 0，即 $E_0(x)=0, \forall x \in X$。每个时刻、每个状态的资格迹都按 $\gamma\lambda$ 的比例衰减，同时对观测到的状态的资格迹加 1。也就是说，资格迹的更新公式如下

$$E_k(x)=\gamma\lambda E_{k-1}(x)+1(X_k=x) \tag{5.13}$$

其中，$1(X_k=x)$是一个条件判断表达式。资格迹反映了状态被观测的次数和频率。假设当前时刻为 k，智能体所处的状态为 x_{k+1}，得到的 TD 误差为 δ_k。此时，我们要对每个状态进行更新，但是更新的程度是不同的，x_{k-1} 的代价函数更新应该乘以 $\gamma\lambda$，x_{k-2} 的代价函数更新应乘以 $\gamma\lambda^2$。这种更新程度的不同，就蕴含在资格迹变量中。

由此，我们就得到了后向 TD(λ)的更新公式：

$$\delta_k=R_{k+1}+\gamma J(x_{k+1})-J(x_k) \tag{5.14}$$

$$J(x)\leftarrow J(x)+\alpha\delta_k E_k(x) \tag{5.15}$$

当 $\lambda=0$ 时，只有当前状态得到更新，等同于 TD(0)算法。

图 5-5 为后向 TD(λ)的示意图，当处于状态 x_{k+1} 时，得到 TD 误差 δ_k，根据这个 TD 误差，我们对之前所有状态的代价函数进行更新，但是更新的程度要由资格迹确定。可以看到，更新的方向与时间方向是相反的，因此称为后向 TD(λ)。

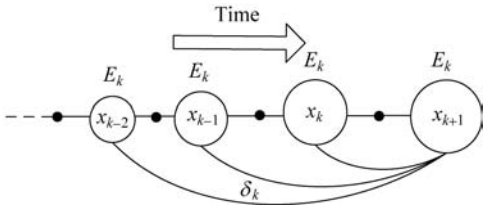

图 5-5　后向 TD(λ)

我们将后向 TD(λ)算法的实施步骤总结在算法 5.2 中。

算法 5.2：后向 TD(λ)算法

步骤 1：首先计算当前状态的 TD 偏差：$\delta_k = R_{k+1} + \gamma J(x_{k+1}) - J(x_k)$。

步骤 2：更新资格迹：

$$E_k(x) = \begin{cases} \gamma\lambda E_{k-1}, & \text{if } x \neq x_k \\ \gamma\lambda E_{k-1} + 1, & \text{if } x = x_k \end{cases}$$

步骤 3：对于状态空间中的每个状态 x，更新代价函数：$J(x) \leftarrow J(x) + \alpha\delta_k E_k(x)$。

从对轨迹完整性的要求来说，前向 TD(λ)需要使用完整的轨迹，而后向 TD(λ)则是在每一个时间步上都更新代价函数。但是，当使用离线更新时，前向 TD(λ)和后向 TD(λ)在同一条轨迹上的更新量是相同的，即

$$\sum_{j=0}^{T} \Delta V_j^{TD}(X) = \sum_{j=0}^{T} \Delta V_j^{\lambda}(x_j) 1_{X=x_j}, \quad \forall X \in \mathcal{X} \tag{5.16}$$

这是一个非常重要的结论，这个结论阐述了前向 TD(λ)和后向 TD(λ)在更新量上的等价性。$\Delta V_j^{TD}(S)$代表后向 TD(λ)在 j 时刻对状态 X 的更新量，$\Delta V_j^{\lambda}(x_j)$代表前向 TD(λ)在 j 时刻对观测量 x_j 的更新。我们有

$$\begin{aligned} G_k^{\lambda} - J(x_k) =& -J(x_k) + (1-\lambda)\lambda^0(R_{k+1} + \gamma J(x_{k+1})) + \\ & (1-\lambda)\lambda^1(R_{k+1} + \gamma R_{k+2} + \gamma^2 J(x_{k+2})) + \\ & (1-\lambda)\lambda^2(R_{k+1} + \gamma R_{k+2} + \gamma^2 R_{k+3} + \gamma^3 J(x_{k+3})) + \cdots \\ =& -J(x_k) + (\gamma\lambda)^0(R_{k+1} + \gamma J(x_{k+1}) - \gamma\lambda J(x_{k+1})) + \\ & (\gamma\lambda)^1(R_{k+2} + \gamma J(x_{k+2}) - \gamma\lambda J(x_{k+2})) + \\ & (\gamma\lambda)^2(R_{k+3} + \gamma J(x_{k+3}) - \gamma\lambda J(x_{k+3})) + \cdots \\ =& (\gamma\lambda)^0(R_{k+1} + \gamma J(x_{k+1}) - J(x_k)) + \\ & (\gamma\lambda)^1(R_{k+2} + \gamma J(x_{k+2}) - J(x_{k+1})) + \\ & (\gamma\lambda)^2(R_{k+3} + \gamma J(x_{k+3}) - J(x_{k+2})) + \cdots \\ =& \delta_k + \gamma\lambda\delta_{k+1} + (\gamma\lambda)^2\delta_{k+2} + \cdots \end{aligned} \tag{5.17}$$

因此，我们就得到前向 TD(λ)和后向 TD(λ)的更新总量是等价的，但在实际应用中，我们常常使用在线的后向 TD(λ)方法。

走近学者

Richard S. Sutton

理查德·萨顿(Richard S. Sutton)是加拿大计算机科学家。目前他是阿尔伯塔大学计算机科学教授和 iCORE 主席。Sutton 被认为是现代计算强化学习的创始人之一，他的几个重要贡献包括时间差异学习、策略梯度方法、Dyna 架构。

1978 年，Sutton 获得斯坦福大学心理学学士学位。于 1980 年与 1984 年分别获得马萨诸塞大学阿默斯特分校的计算机科学专业硕士和博士学位，师从 Andrew Barto。他的博士论文题

目为“强化学习中的时间信用分配”,介绍了执行网-评价网架构和“时间信用分配”。

1984年开始,Sutton在马萨诸塞大学阿默斯特分校攻读博士后。1985—1994年,他是GTE计算机和智能系统实验室技术人员的主要成员。1995年,他回到马萨诸塞大学阿默斯特分校担任资深研究科学家。1998年加入美国电话电报公司香农实验室并担任人工智能部首席技术人员。自2003年以来,他一直是阿尔伯塔大学计算科学系的教授和iCORE主席,他领导着强化学习和人工智能实验室(RLAI)。2017年6月,德米斯哈萨比斯宣布Sutton将领导一个新的阿尔伯塔Deepmind实验室,同时保留他在阿尔伯塔大学的教授职位。

Sutton是人工智能促进协会(AAAI)的成员。2003年,他获得了国际神经网络协会颁发的主席奖,并于2013年获得了马萨诸塞大学阿默斯特分校颁发的杰出研究成果奖。Sutton作为AAAI研究员的提名如下:“对机器学习中的许多方面做出重大贡献,包括强化学习、时序差分技术和神经网络。”

参考文献

[1] 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, SMC-13(5): 834-846.

[2] Sutton R S. Learning to predict by the methods of temporal differences[J]. Machine learning, 1988, 3(1): 9-44.

[3] Sutton R S, Barto A G. Reinforcement learning: An introduction[M]. USA, Cambridge: MIT Press, 1998.

[4] Schultz W, Dayan P, Montague P R. A neural substrate of prediction and reward[J]. Science, 1997, 275(5306): 1593-1599.

[5] Schultz W. Predictive reward signal of dopamine neurons[J]. Journal of neurophysiology, 1998, 80(1): 1-27.

[6] Bertsekas D P. A counterexample to temporal differences learning[J]. Neural Computation, 1995, 7(2): 270-279.

[7] Silver D. Reinforcement Learning Lecture 4: Model-Free Prediction[OL]. http://www.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/MC-TD.pdf.

[8] Tesauro G. Temporal difference learning and TD-Gammon[J]. Communications of the ACM, 1995, 38(3): 58-68.