Reinforcement Learning: An Introduction notebook

黎雷蕾

2017 年 12 月 1 日

目录

Chapter 7

Multi-step Bootstrapping

7.1 n-step TD Prediction

蒙特卡洛方法就是一个 n 步的 TD 算法。one-step return 可以写为:

$$G_{t:t+1} = R_{t+1} + \gamma V_t(S_{t+1}) \tag{7.1}$$

two-step return 可以写为:

$$G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2}) \tag{7.2}$$

那么 n-step return:

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) \tag{7.3}$$

n-step TD for estimating $V \approx v_\pi$

1. 随机地初始化 $V(s)$;
2. 准备两个参数：步长参数 $\alpha \in (0, 1]$, 一个正整数 n;
3. 对于每次估计：
   (a) 初始化 S_0 并使其不处于中断状态；
   (b) $T \leftarrow \infty$;
   (c) **for** $t = 0, 1, 2, \cdots \;\&\; \tau \neq T-1$ **do**
       if t<T **then**
           根据 $\pi(\cdot|\pi)$;
           观察并记录 R_{t+1} 和 S_{t+1};
           if S_{t+1} 为中断状态 (terminal) **then**
               $T \leftarrow t+1$
           end if
       end if
       $\tau = t - n + 1$; 表示将要更新的阶段的下标；
       if $\tau \geq 0$ **then**
           $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$;
           if $\tau + n < T$ **then**
               $G \leftarrow G + \gamma^n V(S_{\tau+n})$;
           end if
           $V(S_\tau) \leftarrow V(S_\tau) + \alpha[G - V(S_\tau)]$
       end if
   end for

7.2 n-step Sarsa

我们首先可以定义 n 步动作函数 Q 的 return：

$$G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}) \tag{7.4}$$

Q 值可以被定义为：

$$Q_{t+n}(S_t, A_t) = Q_{t+n-1}(S_t, A_t) + \alpha[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)] \tag{7.5}$$

那么 n-step Sarsa 可以被定义成：

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

1. 随机地初始化 $Q(s,a)$;
2. 根据 Q 采用 ϵ-贪心算法初始化 π，或者根据实际需求初始化策略 π;
3. 参数: 步长参数 $\alpha \in (0,1]$，$\epsilon > 0$，一个正整数 n;
4. 对于每次模拟，均进行如下操作：
   (a) 初始化 S_0 使其不处于中断状态；
   (b) 选择动作 $A_0 \sim \pi(\cdot|S_0)$;
   (c) $T \leftarrow \infty$;
   (d) **for** $t = 0,1,2,\cdots \&\&\ \tau \neq 0$ **do**
        if $t < T$ **then**
            选择动作 A_t;
            根据动作 A_t 观察得到下一个 R_{t+1}, S_{t+1};
            if S_{t+1} 中断状态 **then**
                $T \leftarrow t+1$;
            else
                选择动作 $A_{t+1} \sim \pi(\cdot|S_{t+1})$;
            end if
        end if
        当前时序下标:$\tau \leftarrow t-n+1$;
        if $\tau \geq 0$ **then**
            $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$;
            if $\tau + n < T$ **then**
                $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$;
            end if
            $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$;
            if 策略 π 已经被学习到 **then**
                确保 $\pi(\cdot|S_\tau)$ 是关于 Q 的 ϵ-贪心法；
            end if
        end if
       end for

我们可以将 return 定义成：

$$G_{t:t+n} = R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \sum_a \pi(a|S_{t+n}) Q_{t+n-1}(S_{t+n}, a) \quad (7.6)$$

这样就可以把上述算法改写成 Expected Sarsa 算法了。

7.3 n-step Off-policy Learning by Importance Sampling

在异策略 TD(n) 算法中引入重要性采样，并设定权重 $\rho_{t:t+n-1}$，那么 return 可以重新定义成：

$$V_{t+n}(S_t) = V_{t+n-1}(S_t) + \alpha \rho_{t:t+n-1} [G_{t:t+n} - V_{t+n-1}(S_t)] \quad (7.7)$$

其中权重 ρ 可以定义为：

$$\rho_{t:h} = \prod_{k=t}^{\min(h,T-1)} \frac{\pi(A_k|S_k)}{b(A_k|S_k)} \quad (7.8)$$

同理 7.2 章的 n-step Sarsa 的更新公式也可以写成：

$$Q_{t+n}(S_t, A_t) = Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1:t+n-1} [G_{t:t+n} - Q_{t+n-1}(S_t, A_t)] \quad (7.9)$$

那么 off-policy n-step Sarsa 算法可以写作：

Off-policy n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

1. **输入**: 一个随机的策略 b，只需保证 $b(a|s) > 0$ 即可；
2. 随机地初始化 $Q(s, a)$;
3. 根据 Q 采用 ϵ-贪心算法初始化 π，或者根据实际需求初始化策略 π;
4. 参数: 步长参数 $\alpha \in (0, 1]$，$\epsilon > 0$，一个正整数 n;
5. 对于每次模拟，均进行如下操作：

(a) 初始化 S_0 使其不处于中断状态;

(b) 选择动作 $A_0 \sim \pi(\cdot|S_0)$;

(c) $T \leftarrow \infty$;

(d) **for** $t = 0, 1, 2, \cdots \&\& \ \tau \neq 0$ **do**

  if $t < T$ **then**

    选择动作 A_t;

    观察并记录 R_{t+1} 和 S_{t+1};

    if S_{t+1} 是中断状态 **then**

      $T \leftarrow t + 1$;

    else

      选择动作 $A_{t+1} \sim \pi(\cdot|S_{t+1})$;

    end if

  end if

  当前时序下标:$\tau \leftarrow t - n + 1$;

  if $\tau \geq 0$ **then**

    $\rho_{\tau+1:t+n-1} \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1,T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$;

    $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$;

    if $\tau + n < T$ **then**

      $G_{\tau:\tau+n} \leftarrow G_{\tau:\tau+n} + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$;

    end if

    if 策略 π 已经被学习到 **then**

      确保 $\pi(\cdot|S_\tau)$ 是关于 Q 的 ϵ-贪心法;

    end if

  end if

end for

7.4 Per-reward Off-policy Methods

前三节提到的 n-step 算法结构简单，但是可能效果并不是最好的。比较好的方法是采取 Per-reward 算法，它的 return 可以被定义为：

$$G_{t:h} = R_{t+1} + \gamma G_{t+1:h} \tag{7.10}$$

同样地，我们定义权重 ρ:

$$\rho_t = \frac{\pi(A_t|S_t)}{b(A_t|S_t)} \tag{7.11}$$

off-policy 的 return 可以写为：

$$G_{t:h} = \rho_t(R_{t+1} + \gamma G_{t+1:h}) + (1-\rho_t)V(S_t) \tag{7.12}$$

改成 Q 值：

$$\begin{aligned} G_{t:h} &= R_{t+1} + \gamma(\rho_{t+1}G_{t+1:h} + (1-\rho_{t-1})\overline{Q}_{t+1}) \\ \overline{Q}_{t+1} &= \sum_a \pi(a|S_t)Q_{t-1}(S_t, a) \end{aligned} \tag{7.13}$$

采用 off-policy 要注意：

1. 容易出现高方差；
2. 如果目标策略 π 和行为策略 b 的差异很大，那么这个方法收敛不太好。

7.5 Off-policy Learning Without Importance Sampling: The n-step Tree Backup Algorithm

n-step $tree-backup$ 算法示意图如下：

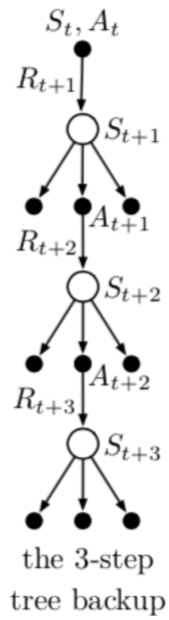

图 7.1: tree backup 算法示意图

一般来说，结点权重如下定义:

1. 第一级动作 a 对应的权重为 $\pi(a|S_{t+1})$;
2. 第二级中，总体的权重为 $\pi(A_{t+1}|S_{t+1})$, 其中未被选中的结点动作 a' 对应的权重为 $\pi(A_{t+1}|S_{t+1})\pi(a'|S_{t+2})$;
3. 同理第三级总权重为 $\pi(A_{t+1}|S_{t+1})\pi(A_{t+2}|S_{t+2})$, 其中未被选中的结点动作对应的权重为 $\pi(A_{t+1}|S_{t+1})\pi(A_{t+2}|S_{t+2})\pi(a''|S_{t+3})$;

假定在 target policy 中期望的价值函数:

$$V_t = \sum_a \pi(a|S_t)Q_{t-1}(S_t, a) \tag{7.14}$$

TD error 可以被定义成:

$$\delta_t = R_{t+1} + \gamma V_{t+1} - Q_{t-1}(S_t, A_t) \tag{7.15}$$

return 可以被定义成：

$$
\begin{aligned}
&G_{t:t+1} = R_{t+1} + \gamma V_{t+1} = Q_{t-1}(A_t, S_t) + \delta_t \\
&G_{t:t+n} = Q_{t-1}(A_t, S_t) + \sum_{k=t}^{m} in(t+n-1, T-1)\delta_k \prod_{t=t+1}^{k} \gamma\pi(A_i|S_i)
\end{aligned}
\tag{7.16}
$$

n-step Sarsa 的 Q 值可以定义为：

$$
Q_{t+n}(S_t, A_t) = Q_{t+n-1}(S_t, A_t) + \alpha[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)] \tag{7.17}
$$

n-step Tree Backup for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

1. 随机地初始化 $Q(s, a)$;
2. 根据 Q 采用 ϵ-贪心算法初始化 π，或者根据实际需求初始化策略 π;
3. 参数: 步长参数 $\alpha \in (0, 1]$，$\epsilon > 0$，一个正整数 n;
4. 对于每次模拟，均进行如下操作：
   (a) 初始化 S_0 使其不在中断状态；
   (b) 选择策略 $A_0 \sim \pi(\cdot|S_0)$;
   (c) $Q_0 = Q(S_0, A_0)$;
   (d) $T \leftarrow \infty$;
   (e)
   for $t = 0, 1, 2, \cdots \&\& \ \tau \neq 0$ **do**
       选择策略 A_0;
       观察获得下一个 reward R 和 S_{t+1};
       if S_{t+1} 是中断状态 **then**
           $T \leftarrow t + 1$;
           $\delta_t = R - Q_t$;
       else
           $\delta_t = R + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q_t$;
           随机地选择一个动作 A_{t+1};
           $Q_{t+1} = Q(S_{t+1}, A_{t+1})$;

$\pi_{t+1} = \pi(S_{t+1}, A_{t+1})$;
end if
当前时序下标:$\tau \leftarrow t - n + 1$;
if $\tau \geq 0$ **then**
$E \leftarrow 1$;
$G \leftarrow Q_\tau$;
for $k = \tau, \cdots, \min(\tau + n - 1, T - 1)$ **do**;
$G \leftarrow G + E\delta_k$;
$E \leftarrow \gamma E_{\pi_{k+1}}$;
end for
$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha[G - Q(S_\tau, A_\tau)]$;
if 策略 π 已经被学习到 **then**
确保 $\pi(\cdot|S_\tau)$ 是关于 Q 的 ϵ-贪心法;
end if
end if
end for

7.6 A Unifying Algorithm: n-step $Q(\sigma)$

为了比 ϵ-贪心算法进一步增加随机性，我们可以考虑探索 (exploration) 和期望 (expectation) 之间的连续变化，定义采样程度 $\sigma_t \in [0, 1]$:

$$\sigma_t = \begin{cases} 1, & \text{只进行采样} \\ 0, & \text{不采样，输出期望} \end{cases} \tag{7.18}$$

单纯的采样造成的 $TD\ error$:

$$G_{t:t+n} = Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n-1,T-1)} \gamma^{k-t}[R_{k+1} + \gamma Q_k(S_{k+1}, A_{k+1}) - Q_{k-1}(S_k, A_k)] \tag{7.19}$$

我们把采样和期望的 TD error 通过 σ_t 进行概括：

$$\begin{aligned}\sigma_t &= R_{t+1} + \gamma[\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) + (1-\sigma_{t+1})\overline{Q}_{t+1}] - Q_{t-1}(S_t, A_t) \\ \overline{Q}_t &= \sum_a \pi(a|S_t) Q_{t-1}(S_t, a)\end{aligned} \tag{7.20}$$

所以我们可以定义 n-step $Q(\sigma)$ 的 return 了：

$$\begin{aligned}G_{t:t+1} &= \sigma_t + Q_{t-1}(S_t, A_t), \\ G_{t:t+n} &= Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n-1, T-1)} \sigma_k \prod_{i=t+1}^{k} \gamma[(1-\sigma_i)\pi(A_i|S_i) + \sigma_i]\end{aligned} \tag{7.21}$$

同时 importance sampling ratio 可以定义为：

$$\rho_{t:t+n} = \prod_{k=t}^{\min(t+n-1, T-1)} \left(\sigma_k \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)} + 1 - \sigma_k\right) \tag{7.22}$$

算法如下所示：

Off-policy n-step $Q(\sigma)$ for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given π

1. **输入**: 一个随机的策略 b，只需保证 $b(a|s) > 0$ 即可；
2. 随机地初始化 $Q(s, a)$;
3. 根据 Q 采用 ϵ-贪心算法初始化 π，或者根据实际需求初始化策略 π;
4. 参数: 步长参数 $\alpha \in (0, 1]$，$\epsilon > 0$，一个正整数 n;
5. 对于每次模拟，均进行如下操作：
6. (a) 初始化 S_0 使其不在中断状态；
   (b) 选择策略 $A_0 \sim b(\cdot|S_0)$;
   (c) $Q_0 = Q(S_0, A_0)$;
   (d) $T \leftarrow \infty$;

(e)

for $t = 0, 1, 2, \cdots \&\& \ \tau \neq T - 1$ **do**
  if $t < T$ **then**
    选择动作 A_t;
    观察获得下一个 reward R 和 S_{t+1};
    if S_{t+1} 是中断状态 **then**
      $T \leftarrow t + 1$;
      $\delta_t = R - Q_t$;
    else
      随机地选择一个动作 $A_{t+1} \sim b(\cdot|S_{t+1})$;
      随机选择一个 σ_t
      $Q_{t+1} = Q(S_{t+1}, A_{t+1})$;
      $\sigma_t = R + \gamma\sigma_{t+1}Q_{t+1} + \gamma(1 - \sigma_{t+1})\sum_a \pi(a|S_{t+1})Q(S_{t+1}|a) - Q_t$;
      $\pi_{t+1} = \pi(S_{t+1}, A_{t+1})$;
      $\rho_{t+1} = \frac{\pi(A_{t+1}|S_{t+1})}{b(A_{t+1}|S_{t+1})}$;
    end if
  end if
  当前时序下标:$\tau \leftarrow t - n + 1$;
  if $\tau \geq 0$ **then**
    $\rho \leftarrow 1$;
    $e \leftarrow 1$;
    $G \leftarrow Q_\tau$;
    for $k = \tau, \cdots, \min(\tau + n - 1, T - 1)$ **do**
      $G \leftarrow G + e\delta_k$;
      $e \leftarrow \gamma e[(1 - \sigma_{k+1})\pi_{k+1} + \sigma_{k+1}]$;
      $\rho \leftarrow \rho(1 - \sigma_k + \sigma_k\rho_k)$;
    end for
    $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho[G - Q(S_\tau, A_\tau)]$;
    if 策略 π 已经被学习到 **then**

        确保 $\pi(\cdot|S_\tau)$ 是关于 Q 的 ϵ-贪心法;
    end if
  end if
end for