强化学习第二次汇报

强化学习元素

- policy
- reward function
- Value function

Returns

$$R_t = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$

State-value function：

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\}$$

Action-value function:

$$Q^{\pi}(s,a) = E_{\pi}\{R_t | s_t = s, a_t = a\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}$$

备份图（backup diagrams）

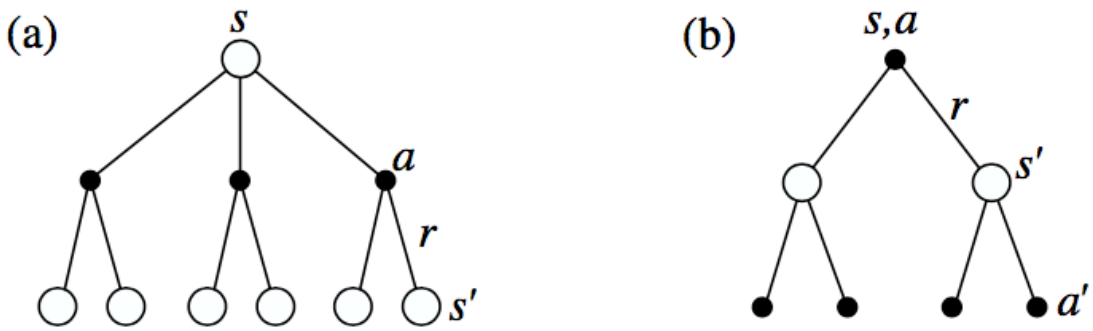


Figure 3.4 Backup diagrams for (a) V^{π} and (b) Q^{π}.

Dynamic Programm

- policy evaluation
- policy improvement
- policy iteration

一、Monte Carlo（MC）

DP算法需要事先知道$\mathcal{P}^a_{ss'}$和$\mathcal{R}^a_{ss'}$，通常是不实际的，而且有的时候，这两个量也是随时间变化的。而Monte Carlo(MC)方法只需要经验（experience）。

1、MC Policy Evaluation

估计方法是对状态s之后的returns求均值，当状态s不断地被访问到，returns的数量越多，则value function就会收敛到当前策略对应的value function。

根据求取均值的方式不同：

- **First-visit MC method**：只平均第一次访问状态s后的returns。
- **Every-visit MC method**：无论是否是第一次访问状态s，都进行平均。

两种方式随着s的访问次数趋于无穷，都会收敛于真值。

Initialize:
$\pi \leftarrow$ policy to be evaluated
$V \leftarrow$ an arbitrary state-value function
$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Repeat forever:
(a) Generate an episode using π
(b) For each state s appearing in the episode:
$R \leftarrow$ return following the first occurrence of s
Append R to $Returns(s)$
$V(s) \leftarrow$ average$(Returns(s))$

Figure 5.1 First-visit MC method for estimating V^π.

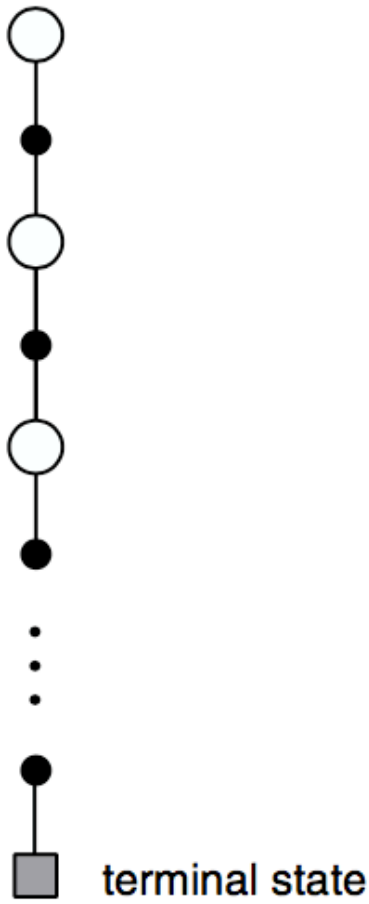


Figure 5.3 The backup diagram for Monte Carlo estimation of V^{π}.

对比DP的backup diagram可以看出，DP方法列出了所有可能的转换，而MC方法则是从可能的转换中进行采样。

On-Policy method attempt to evaluate or improve the policy that is used to make decisions.也就是说on-policy方法用于决策的和被提升的是同一个策略。

Off-Policy方法将behavior和estimation两部分功能分成两个策略，分别叫behavior policy和estimation policy。

2、On-Policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:
$Q(s,a) \leftarrow$ arbitrary
$Returns(s,a) \leftarrow$ empty list
$\pi \leftarrow$ an arbitrary ϵ-soft policy

Repeat forever:
(a) Generate an episode using π
(b) For each pair s, a appearing in the episode:
$R \leftarrow$ return following the first occurrence of s, a
Append R to $Returns(s,a)$
$Q(s,a) \leftarrow$ average($Returns(s,a)$)
(c) For each s in the episode:
$a^* \leftarrow \arg\max_a Q(s,a)$
For all $a \in \mathcal{A}(s)$:

$$\pi(s,a) \leftarrow \begin{cases} 1-\epsilon+\epsilon/|\mathcal{A}(s)| & \text{if } a = a^* \\ \epsilon/|\mathcal{A}(s)| & \text{if } a \neq a^* \end{cases}$$

Figure 5.6 An ϵ-soft on-policy Monte Carlo control algorithm.

3、Evaluating One Policy While Following Another

如何利用某一个策略产生的Return来估计另一个策略，这是off-policy方法的基础。现在假设有两个策略π和π'，用$p_i(s)$和$p'_i(s)$表示状态序列出现的可能性

$$V(s) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} R_i(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}}$$

$$p_i(s) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}$$

$$\frac{p_i(s)}{p'_i(s)} = \frac{\prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}}{\prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) \mathcal{P}_{s_k s_{k+1}}^{a_k}} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}$$

4、Off-Policy MC Control

Initialize, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$:

$Q(s,a) \leftarrow$ arbitrary

$N(s,a) \leftarrow 0$; Numerator and

$D(s,a) \leftarrow 0$; Denominator of $Q(s,a)$

$\pi \leftarrow$ an arbitrary deterministic policy

Repeat forever:

(a) Select a policy π' and use it to generate an episode:

$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{T-1}, a_{T-1}, r_T, s_T$

(b) $\tau \leftarrow$ latest time at which $a_\tau \neq \pi(s_\tau)$

(c) For each pair s, a appearing in the episode at time τ or later:

$t \leftarrow$ the time of first occurrence of s, a such that $t \geq \tau$

$w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k,a_k)}$

$N(s,a) \leftarrow N(s,a) + wR_t$

$D(s,a) \leftarrow D(s,a) + w$

$Q(s,a) \leftarrow \frac{N(s,a)}{D(s,a)}$

(d) For each $s \in \mathcal{S}$:

$\pi(s) \leftarrow \arg\max_a Q(s,a)$

Figure 5.7 An off-policy Monte Carlo control algorithm.

二、Temporal-Difference Learning（TD）

MC和DP都是利用执行某一策略π之后的经验，来更新对真值V^π的估计V。

bootstrapping method：TD方法利用原来的估计来更新估计，与DP类似。

$$
\begin{aligned}
V^\pi(s) &= E_\pi\{R_t|s_t = s\} \\
&= E_\pi\{\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}|s_t = s\} \\
&= E_\pi\{\sum_{k=0}^{\infty} r_t + \gamma^k r_{t+k+2}|s_t = s\} \\
&= E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1})|s_t = s)\}
\end{aligned}
$$

MC方法估计的是第一行，DP方法估计的是最后一行。

Initialize $V(s)$ arbitrarily, π to the policy to be evaluated
Repeat (for each episode):
    Initialize s
    Repeat (for each step of episode):
        $a \leftarrow$ action given by π for s
        Take action a; observe reward, r, and next state, s'
        $V(s) \leftarrow V(s) + \alpha \left[r + \gamma V(s') - V(s)\right]$
        $s \leftarrow s'$
    until s is terminal

Figure 6.1 Tabular TD(0) for estimating V^{π}.

Figure 6.2 The backup diagram for TD(0).

1、Advantages of TD Prediction Methods

- 与DP相比不需要模型
- 与MC相比，天生的在线（on-line）原址（incremental）

2、Sarsa: On-Policy TD Control

control的问题都可以分成两类：on-policy和off-policy，On-Policy TD控制就是Sarsa。

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Choose a from s using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action a, observe r, s'
    Choose a' from s' using policy derived from Q (e.g., ϵ-greedy)
    $Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a) \right]$
    $s \leftarrow s'; a \leftarrow a';$
  until s is terminal

Figure 6.9 Sarsa: An on-policy TD control algorithm.

例子：windy gridworld

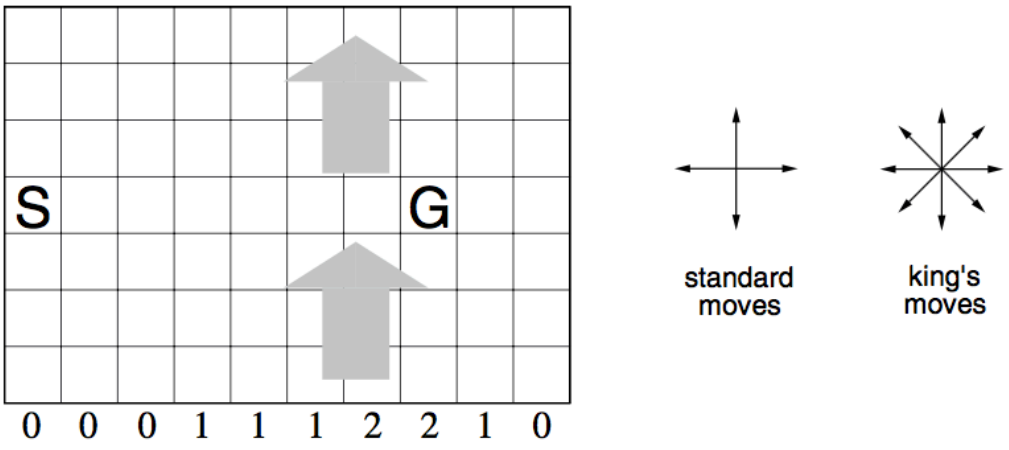


Figure 6.10 Gridworld in which movement is altered by a location-dependent, upward "wind."

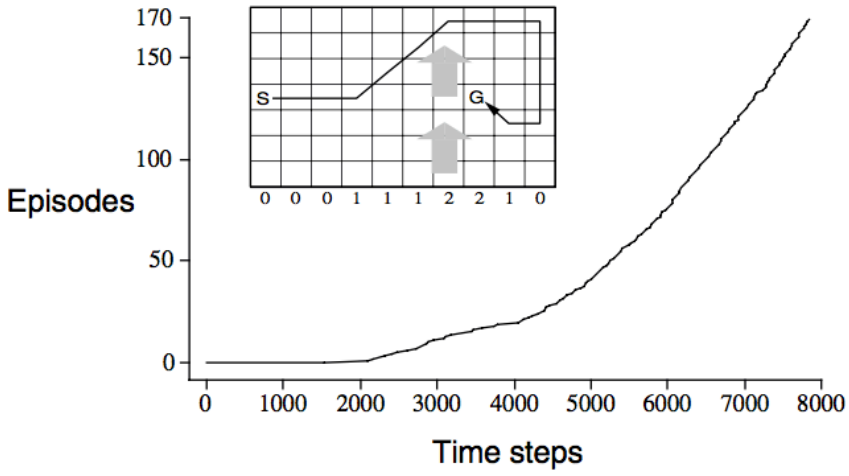


Figure 6.11 Results of Sarsa applied to the windy gridworld.

3、Q-Learning: Off-policy TD Control

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)]$$

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    Take action a, observe r, s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
    $s \leftarrow s'$;
  until s is terminal

Figure 6.12 Q-learning: An off-policy TD control algorithm.

三、Eligibility Traces（资格迹）

Eligibility trace（资格迹）是RL的一个基本机制。几乎所有的TD方法都可以和eligibility trace结合。

可以从两个角度对资格迹进行理解：

- Theoretical view（forward）
- mechanistic（backward）

1、n-Step TD Prediction

n-step return

在MC和TD方法之间，存在一簇方法，叫做*n-step TD methods*。

MC方法对整个episode的Returns进行了备份:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$$

$$R_t^{(1)} = r_{t+1} + \gamma V_t(s_{t+1})$$

$$R_t^{(2)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 V(s_{t+2})$$

$$R_t^{(n)} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + +\gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n})$$

我称$R_t^{(n)}$为t时刻的*n-step return*

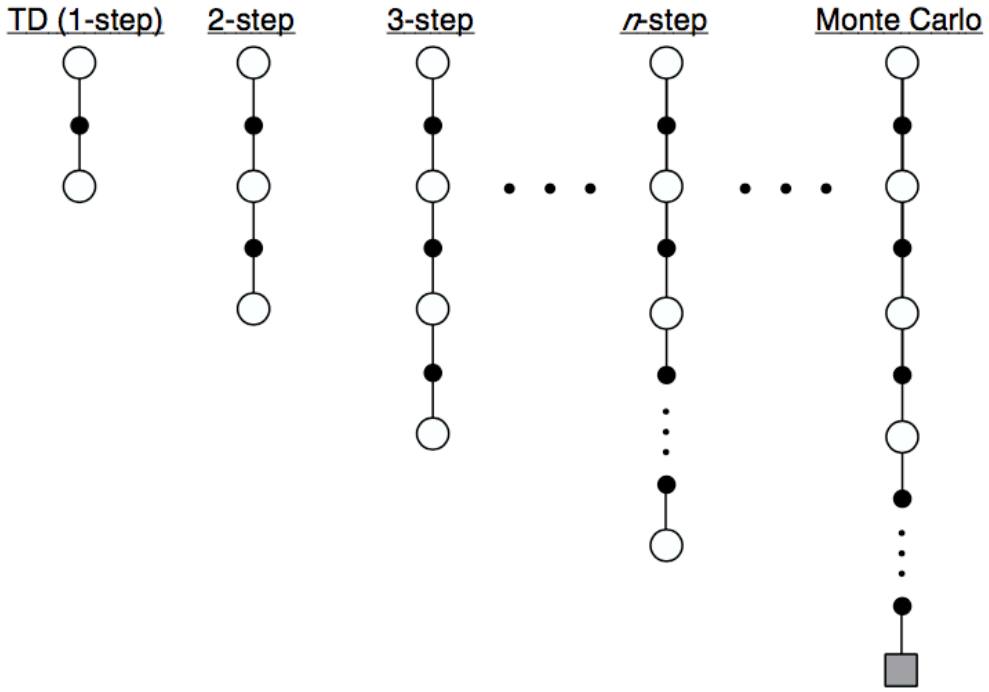


Figure 7.1 The spectrum ranging from the one-step backups of simple TD methods to the up-until-termination backups of Monte Carlo methods. In between are the n-step backups, based on n steps of real rewards and the estimated value of the n^{th} next state, all appropriately discounted.

N-step backup

根据备份的值进行值更新

$$\Delta V_t(s_t) = \alpha[R_t^{(n)} - V_t(s_t)]$$

当然对于不等于s_t的状态，$\Delta V_t(s) = 0$。

$$V_{t+1}(s) = V_t(s) + \Delta V_t(s)$$

- On-line updating
- Off-line updating

Error reduction property

经过参数更新后，最差情况下，对值函数的估计误差不会超过上一轮估计的最大误差的γ^n倍。

例子：随机游走

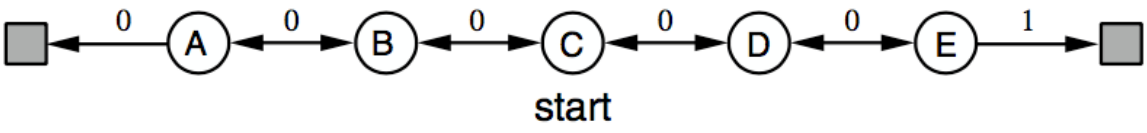


Figure 6.5 A small Markov process for generating random walks.

向左向右的概率各为二分之一，最左边reward为1，其他状态reward为0。

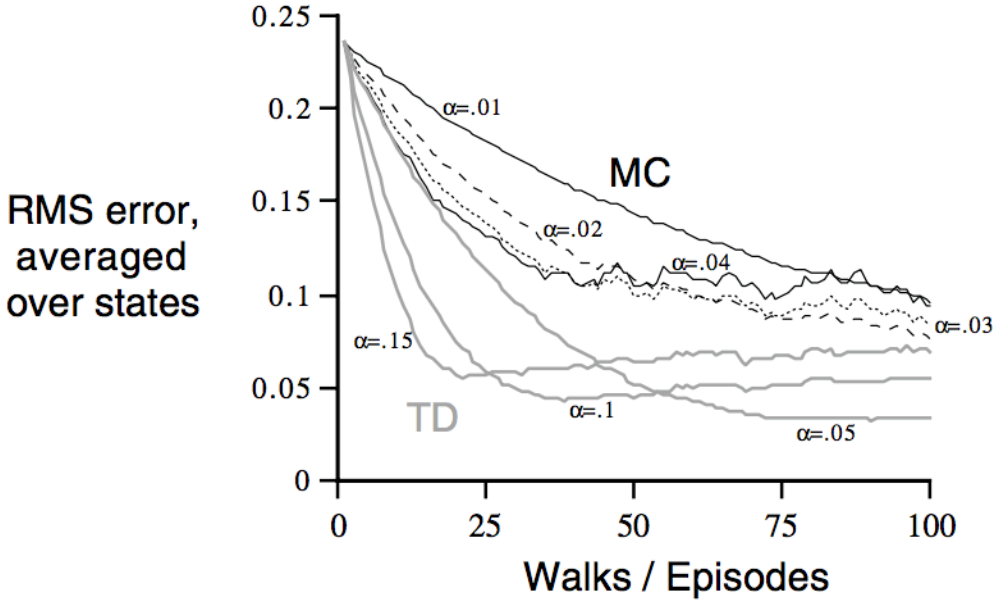


Figure 6.7 Learning curves for TD(0) and constant-α MC methods, for various values of α, on the prediction problem for the random walk. The performance measure shown is the root mean-squared (RMS) error between the value function learned and the true value function, averaged over the five states. These data are averages over 100 different sequences of episodes.

规模更大更复杂的随机游走，用n-step TD进行求解：

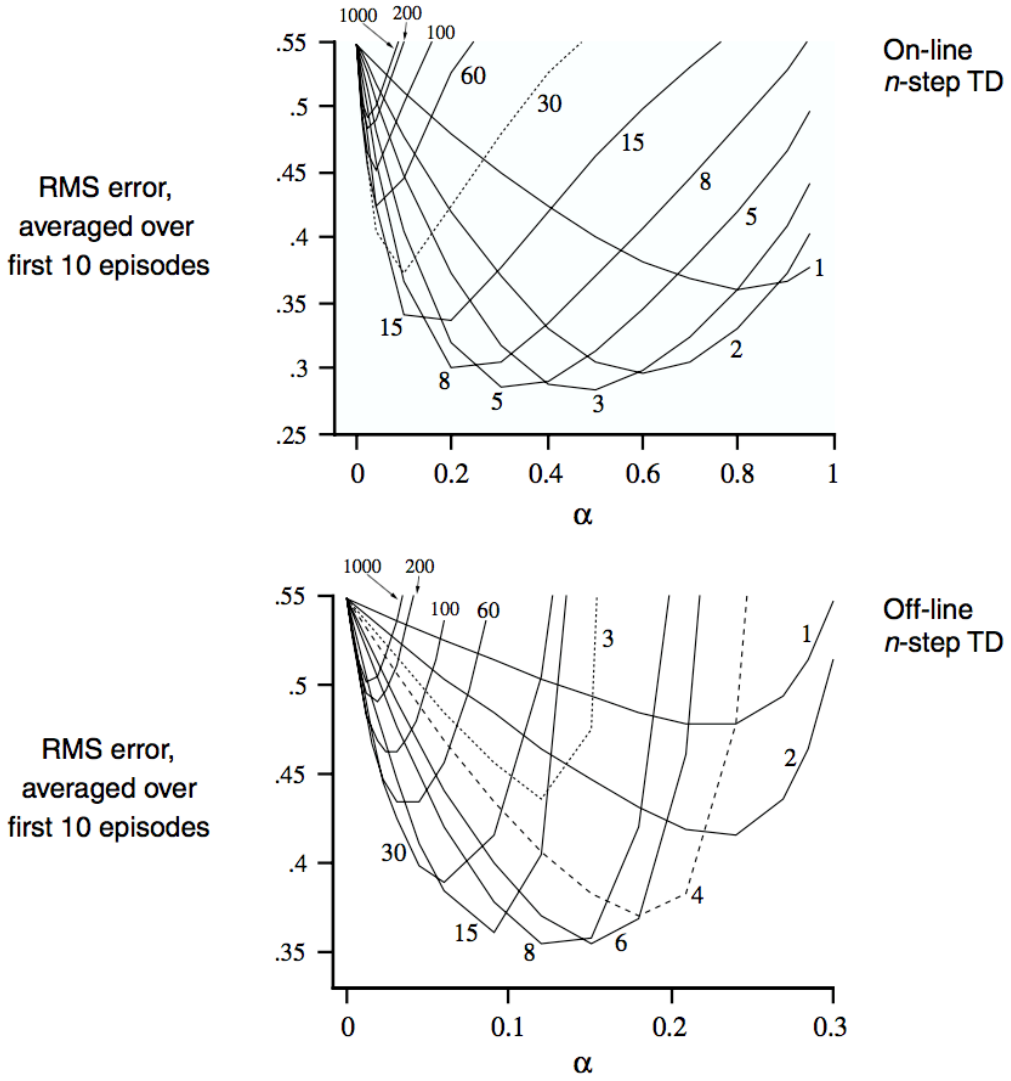
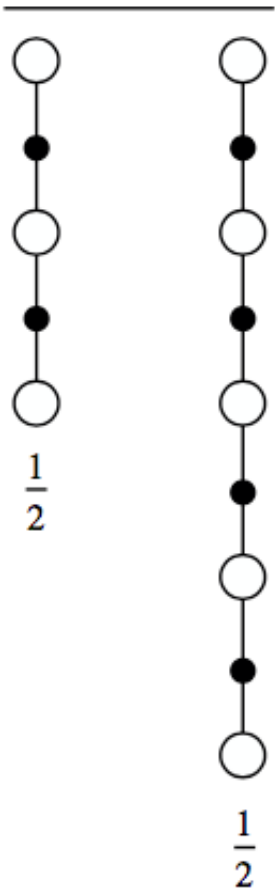


Figure 7.2 Performance of n-step TD methods as a function of α, for various values of n, on a 19-state random walk task. The performance measure shown is the root mean-squared (RMS) error between the true values of states and the values found by the learning methods, averaged over the 19 states, over the first 10 trials, and over 100 different sequences of walks.

2、The Forward View of TD(λ)

前面我都只是利用某一种backup对value进行修正，现在我们尝试混合不同长度的backup来作为Returns。

比如可以将2-step return和4-step return各取二分之一然后相加。



$\lambda - return$

更一般地，我们可以通过一定的比例将各种长度的backup叠加起来，由simpler component组成complex backup。TD(λ)算法就是利用某种特殊的比例，将不同长度的backup进行组合，称为$\lambda - return$。

$$R_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}R_t^{(n)}, \ 0 \le \lambda \le 1$$

可以将$\lambda - return$分解来看

$$R_t^\lambda = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}R_t^{(n)} + \lambda^{T-t-1}R_t$$

当$\lambda = 1$时，就退化成MC的Return；当$\lambda = 0$，就退化成TD(0)的Return。

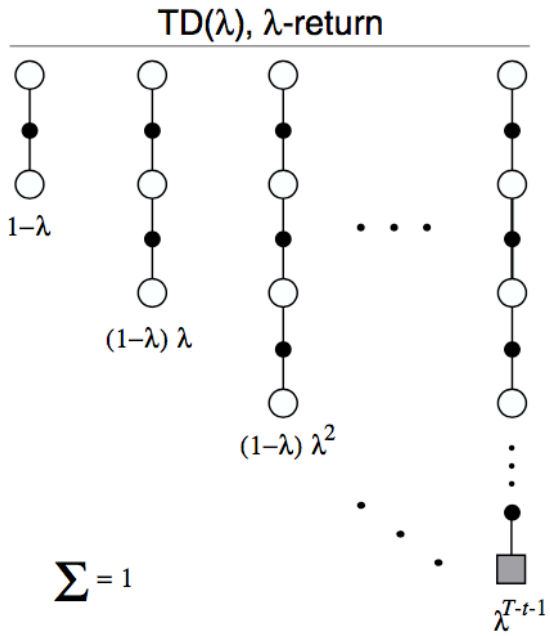


Figure 7.3 The backup digram for TD(λ). If $\lambda = 0$, then the overall backup reduces to its first component, the one-step TD backup, whereas if $\lambda = 1$, then the overall backup reduces to its last component, the Monte Carlo backup.

下图绘制了不同step backup的比例，求和为1。

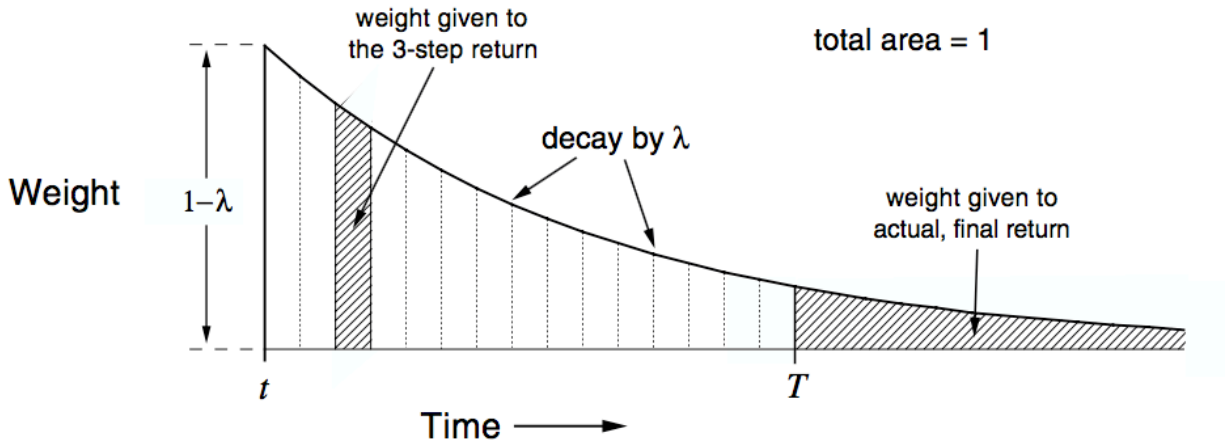


Figure 7.4 Weighting given in the λ-return to each of the *n*-step returns.

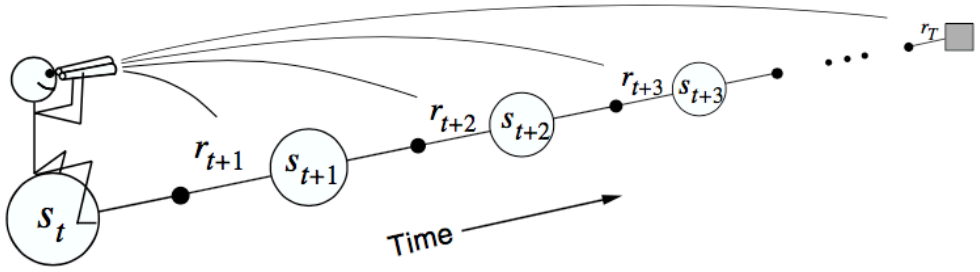


Figure 7.5 The forward or theoretical view. We decide how to update each state by looking forward to future rewards and states.

根据$\lambda - return$，value函数的更新就变成

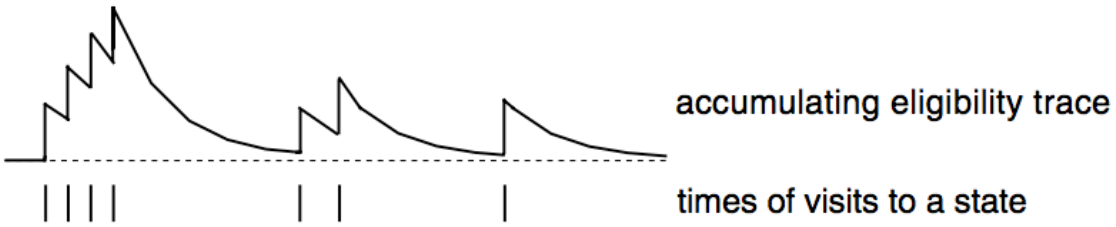
$$\Delta V_t(s_t) = \alpha[R_t^\lambda - V_t(s_t)]$$

3、The Backward View of TD(λ)

其实可以看出，前向的TD(λ)是不可实现的（非因果）。本节就提供一种后向机制（backward mechanism），来实现前向的想法。

定义eligibility trace（资格迹）函数：

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & if s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1 & if s = s_t \end{cases}$$



TD算法中的TD-error

$$\delta_t = r_{t+1} + \gamma V_t(s_{t+1}) - V_t(s_t)$$

TD(λ)算法将δ_t按比例反馈更新所有状态

$$\Delta V_t(s) = \alpha\delta_t e_t(s),\ for\ all\ s \in S$$

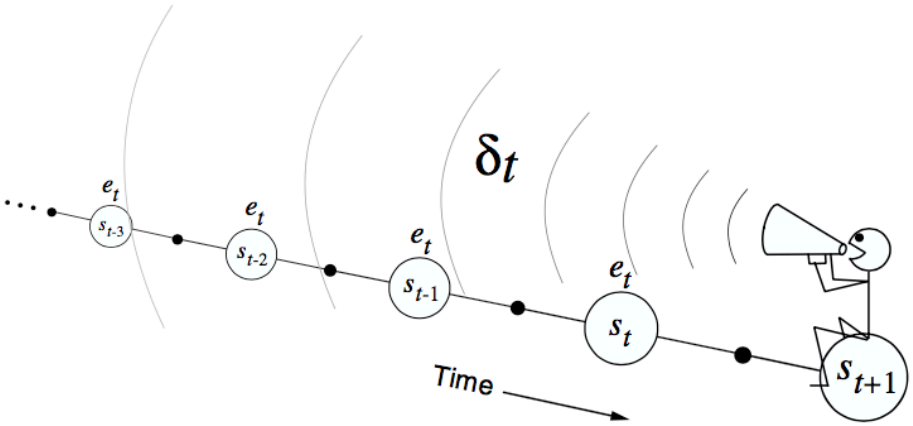


Figure 7.8 The backward or mechanistic view. Each update depends on the current TD error combined with traces of past events.

Initialize $V(s)$ arbitrarily and $e(s) = 0$, for all $s \in \mathcal{S}$
Repeat (for each episode):
  Initialize s
  Repeat (for each step of episode):
    $a \leftarrow$ action given by π for s
    Take action a, observe reward, r, and next state, s'
    $\delta \leftarrow r + \gamma V(s') - V(s)$
    $e(s) \leftarrow e(s) + 1$
    For all s:
      $V(s) \leftarrow V(s) + \alpha\delta e(s)$
      $e(s) \leftarrow \gamma\lambda e(s)$
    $s \leftarrow s'$
  until s is terminal

Figure 7.7 On-line tabular TD(λ).

MC的另一种实现方式

TD(1)是实现Monte Carlo的更通用的方式。不像MC方法，只适用于episodic任务，TD(1)可以用于discounted continuing tasks。

4、Sarsa(λ) and Q(λ)

前面讨论了TD(λ)的prediction，现在讨论control问题。

仿照state-value的更新，定义一个接受s,a两个参数的eligiblity trace。

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha\delta_t e_t(s,a), for\ all\ s,a$$

$$\delta_t = r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)$$

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) + 1 & if s = s_t, and\ a = a_t \\ \gamma\lambda e_{t-1}(s,a) & otherwise \end{cases}$$

Initialize $Q(s,a)$ arbitrarily and $e(s,a)=0$, for all s,a
Repeat (for each episode):
  Initialize s,a
  Repeat (for each step of episode):
    Take action a, observe r,s'
    Choose a' from s' using policy derived from Q (e.g., ϵ-greedy)
    $\delta \leftarrow r+\gamma Q(s',a')-Q(s,a)$
    $e(s,a) \leftarrow e(s,a)+1$
    For all s,a:
      $Q(s,a) \leftarrow Q(s,a)+\alpha\delta e(s,a)$
      $e(s,a) \leftarrow \gamma\lambda e(s,a)$
    $s \leftarrow s'; a \leftarrow a'$
  until s is terminal

Figure 7.11 Tabular Sarsa(λ).

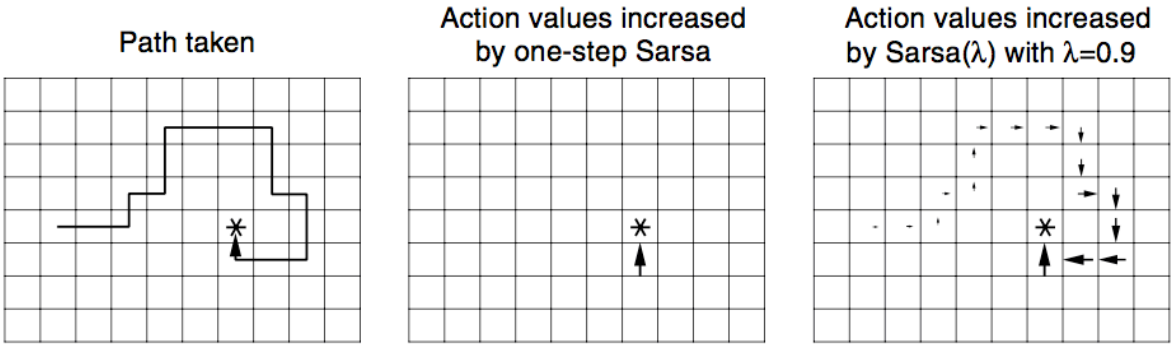
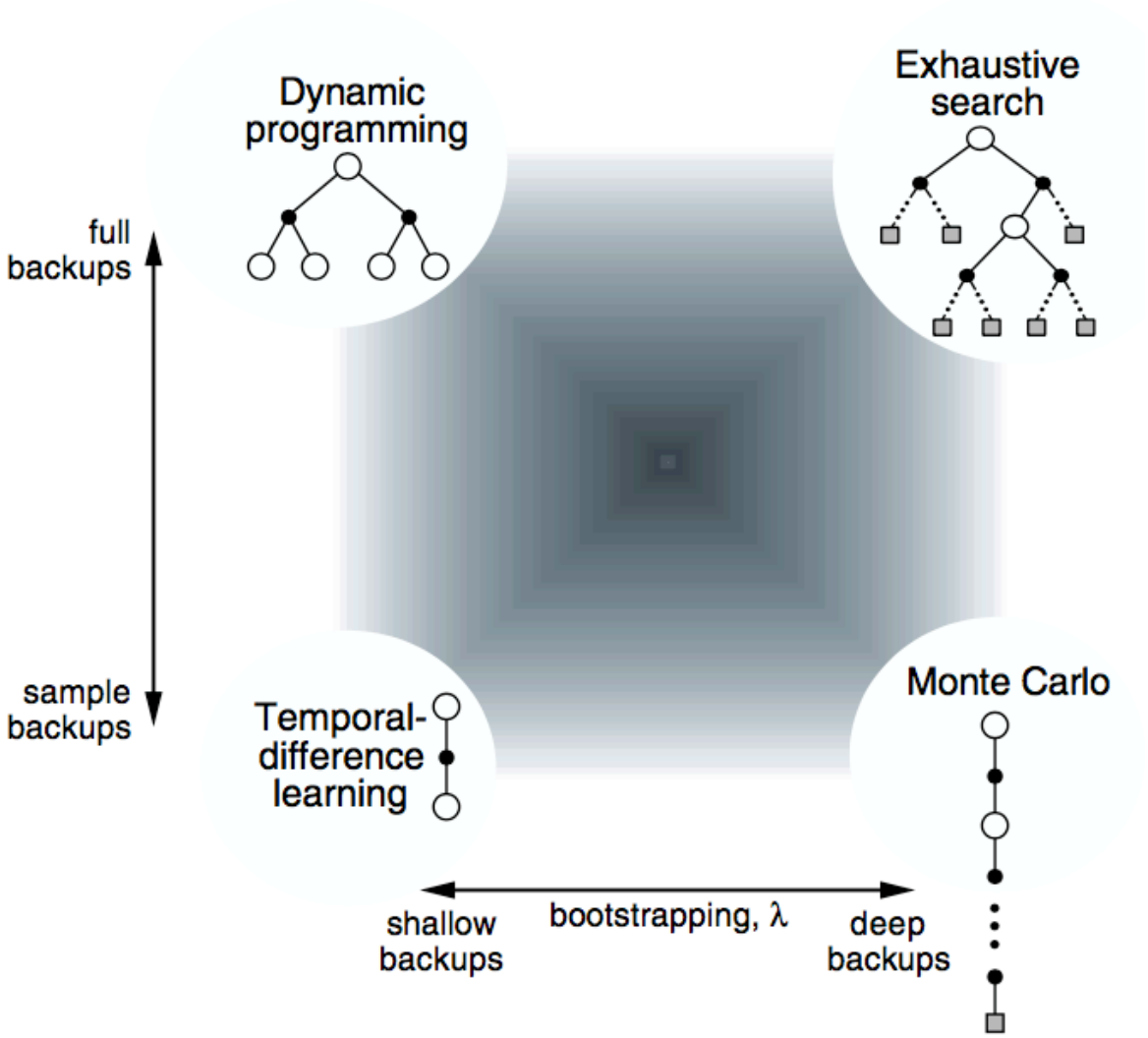


Figure 7.12 Gridworld example of the speedup of policy learning due to the use of eligibility traces.

Q(λ)有两种实现方式，比较复杂，就先不讲。

四、Overview


Dynamic
programming
Exhaustive
search
full
backups
sample
backups
Temporal-
difference
learning
Monte Carlo
shallow
backups
bootstrapping, λ
deep
backups