Chapter 7

n-step Bootstrapping

2018/11/13

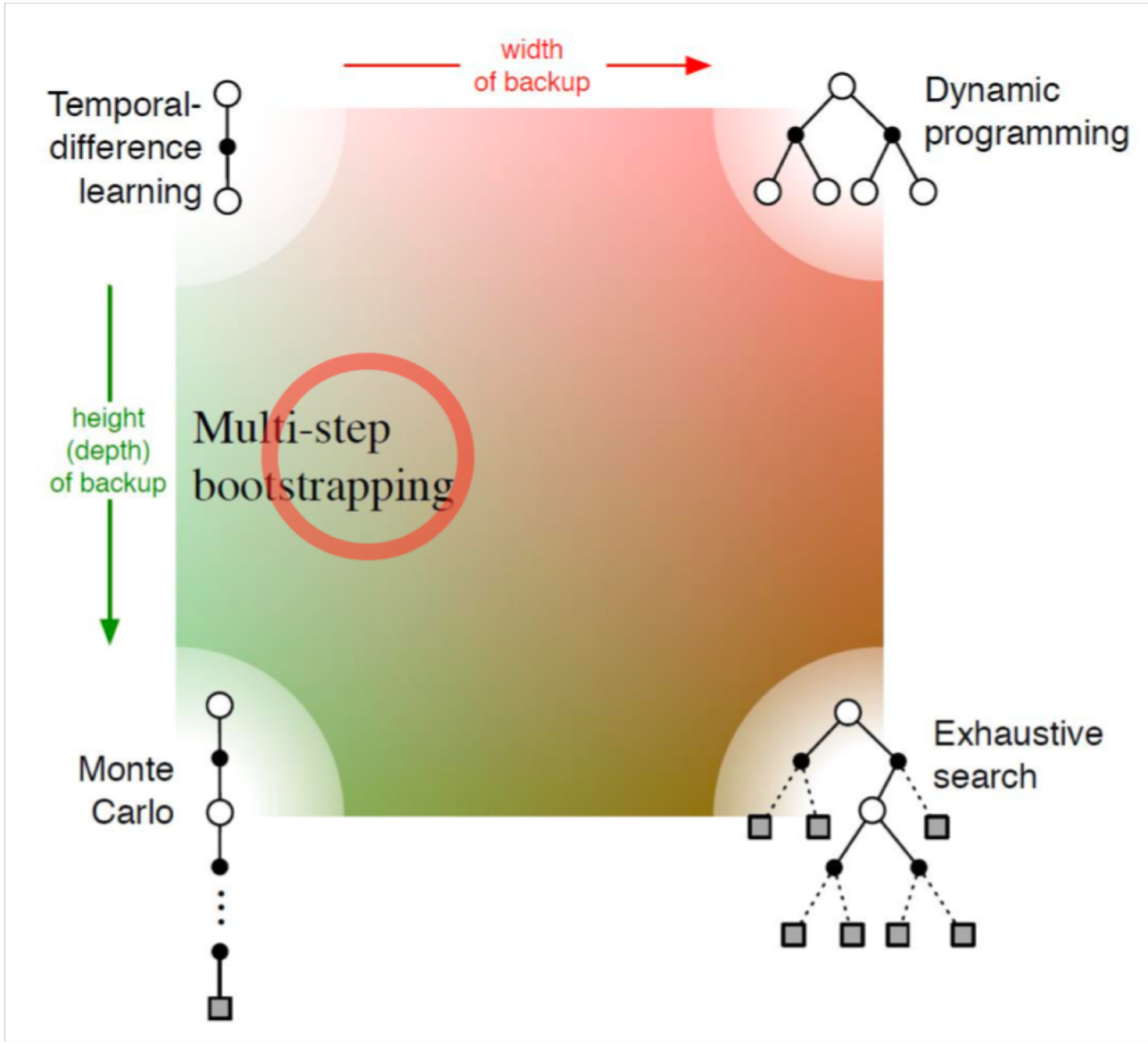
Dev3B 강준하

Index

Introduction

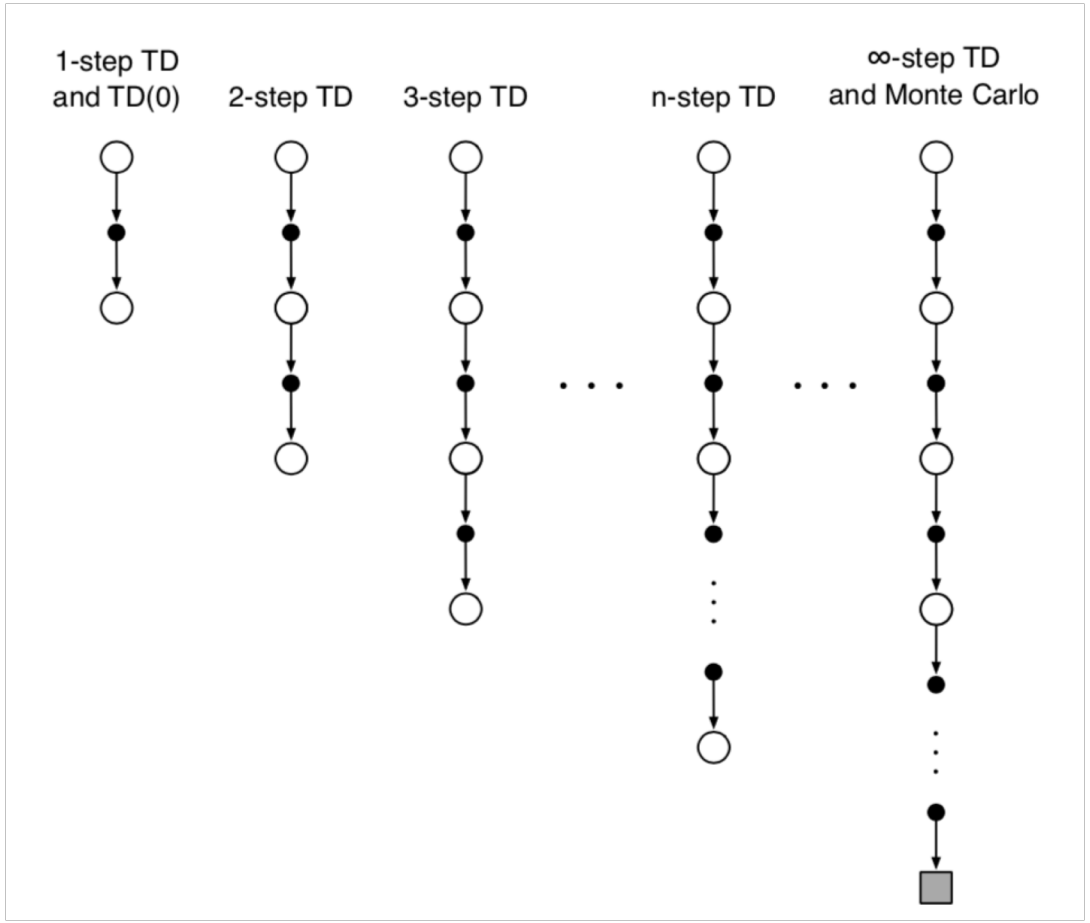
- MC + one-step TD = n-step TD
- Bootstrapping을 원하는 만큼만 하자!
- Chapter 12 Eligibility Traces의 introduction
- prediction problem → control method

Introduction



7.1 n-step TD Prediction

- Motivation : one-step은 너무 적고, MC는 너무 길다!



7.1 n-step TD Prediction

- Complete return : $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T,$
- One-step return : $G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1}),$
- Two-step return : $G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2}),$
- n-step return :

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}), \tag{7.1}$$

- n-step return update rule :

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\big[G_{t:t+n} - V_{t+n-1}(S_t)\big], \qquad 0 \le t < T, \tag{7.2}$$

7.1 n-step TD Prediction – Exercise 7.1

$$\delta_t = G_{t:t+n} - V(S_t) = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^{n} V(S_{t+n}) - V(S_t)$$

$$G_t - V(S_t) = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^{n} G_{t+n} - V(S_t) + \gamma^{n} V(S_{t+n}) - \gamma^{n} V(S_{t+n})$$

$$= \delta_t + \gamma^{n}(G_{t+n} - V(S_{t+n}))$$

$$= \delta_t + \gamma^{n}\delta_{t+n} + \gamma^{2n}\left(G_{t+2n} - V(S_{t+2n})\right)$$

(when $T = kn + r$)

$$= \sum_{i=0}^{k} \gamma^{in}\delta_{t+in} + \gamma^{k+1}\left(G_{t+(k+1)n} - V\left(S_{t+(k+1)n}\right)\right)$$

$$= \sum_{i=0}^{k} \gamma^{in}\delta_{t+in} \ (\because G_{t:t+n} = G_t \ if \ t + n \geq T)$$

7.1 n-step TD Prediction

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

Input: a policy π
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer n
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for S_t and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take an action according to $\pi(\cdot|S_t)$
  | 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 $T \leftarrow t+1$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose state's estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$ n-step reward
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ n-step return update $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

7.1 n-step TD Prediction

- error reduction property
- $$\max_s \left| \mathbb{E}_\pi[G_{t:t+n} | S_t = s] - v_\pi(s) \right| \leq \gamma^n \max_s \left| V_{t+n-1}(s) - v_\pi(s) \right|, \qquad (7.3)$$

maximum error using n-step return

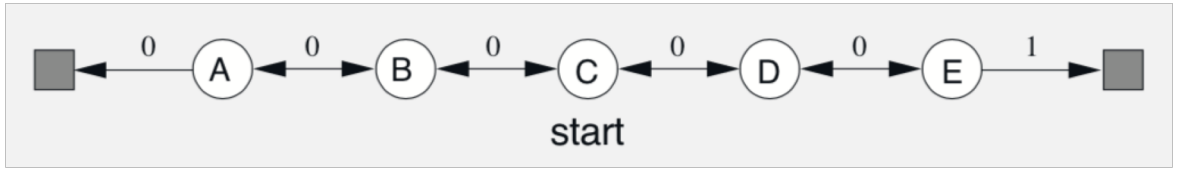
maximum error using V

- Proof : by DP's local value improvement theorem
  - If $|V(s) - v_*(s)| \leq M$ for all states s, then for any state s'

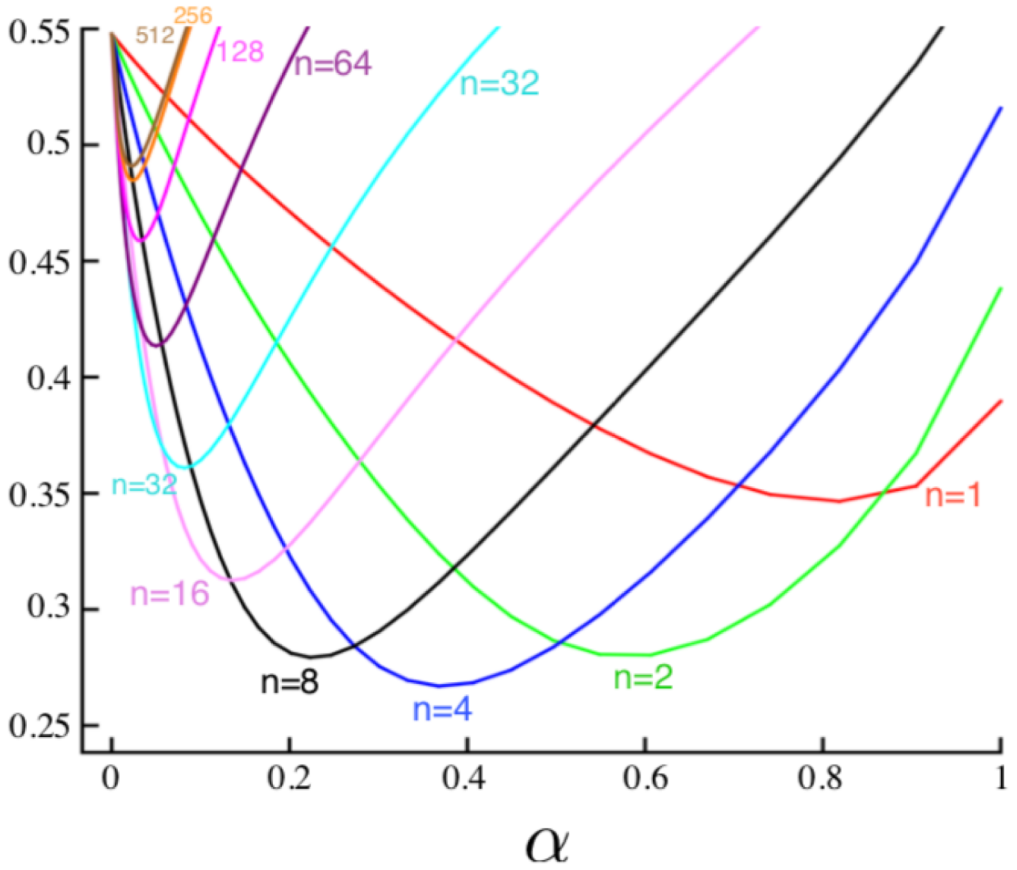
$$|V'(s') - v_*(s)| \leq \gamma M$$

  - where V' is updated (by DP) estimate of the value of s'.

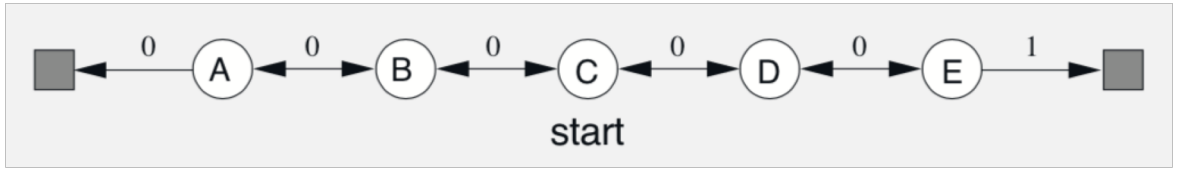
7.1 n-step TD Prediction – Example 7.1



Average RMS error over 19 states and first 10 episodes



7.1 n-step TD Prediction – Exercise 7.3



- Exercise 7.3 : 이 예제에서는 왜 state를 늘렸을까(5에서 19로)? 적은 state에서는 n을 변화시키는 것에서 이득이 있을까? 왼쪽 말단의 reward를 0 대신 -1로 설정하면 최적 n값이 바뀔까?
  - 이 예제에서 n-step을 적용함으로써 얻을 수 있는 이득은 1 step만 보고 update가 이루어지므로 말단 state의 value function만 변화가 이루어지는 문제를 해결할 수 있다는 점이다.
  - 그러므로 state가 넓을수록 n-step 방식이 멀리 보고 update를 한다는 장점을 알아보기가 쉽다.
  - 이는 곧 적은 state에서는 n 값의 변화가 치명적이지 않음을 의미한다.
  - 왼쪽 reward를 -1로 변형하면 왼쪽으로 가는 episode도 반영되기 때문에 수렴은 정확하고 빨라지겠지만, 최적 n값이 변경되지는 않을 것이다.

7.2 n-step Sarsa

- Control method for n-step TD
- Using also state-action value function instead of state value function
- $$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^{n} Q_{t+n-1}(S_{t+n}, A_{t+n}), \quad n \geq 1, 0 \leq t < T - n, \tag{7.4}$$
- update rule
- $$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t) \right], \qquad 0 \leq t < T, \tag{7.5}$$

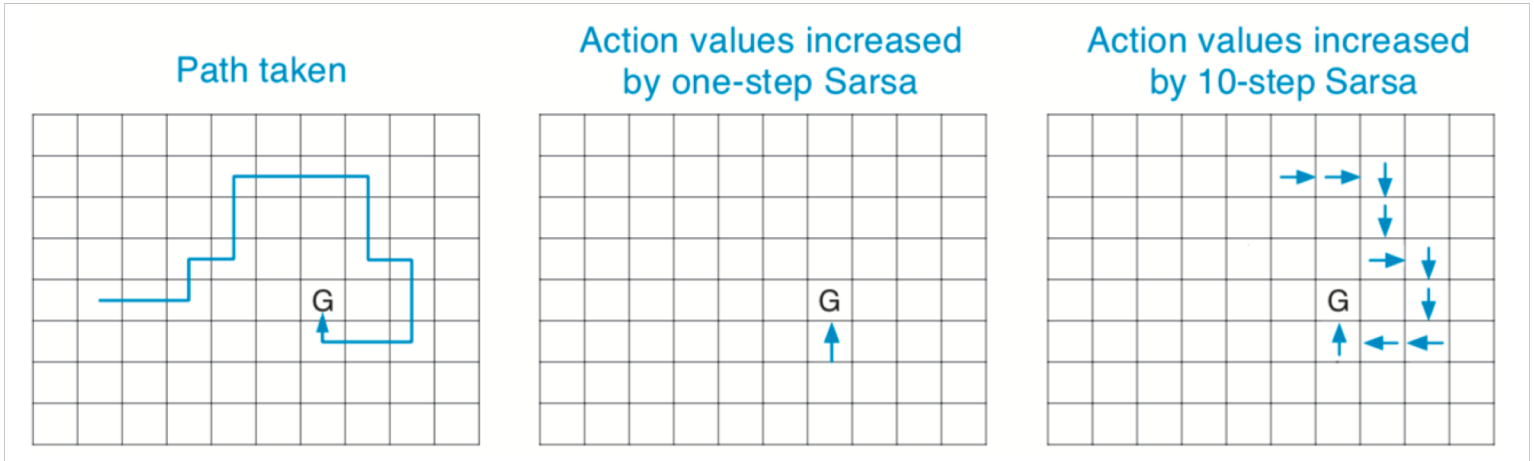
7.2 n-step Sarsa

n-step Sarsa for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or to a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n
All store and access operations (for 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 $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t-n+1$ (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 $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T-1$

7.2 n-step Sarsa – Figure 7.4



- one-step Sarsa에서는 한 episode를 통해 update를 시행하면 goal 직전의 칸만 value가 update된다.
- 하지만 10-step Sarsa에서는 한 episode로 goal 전 10개의 칸을 update하므로 수렴이 빨라진다.

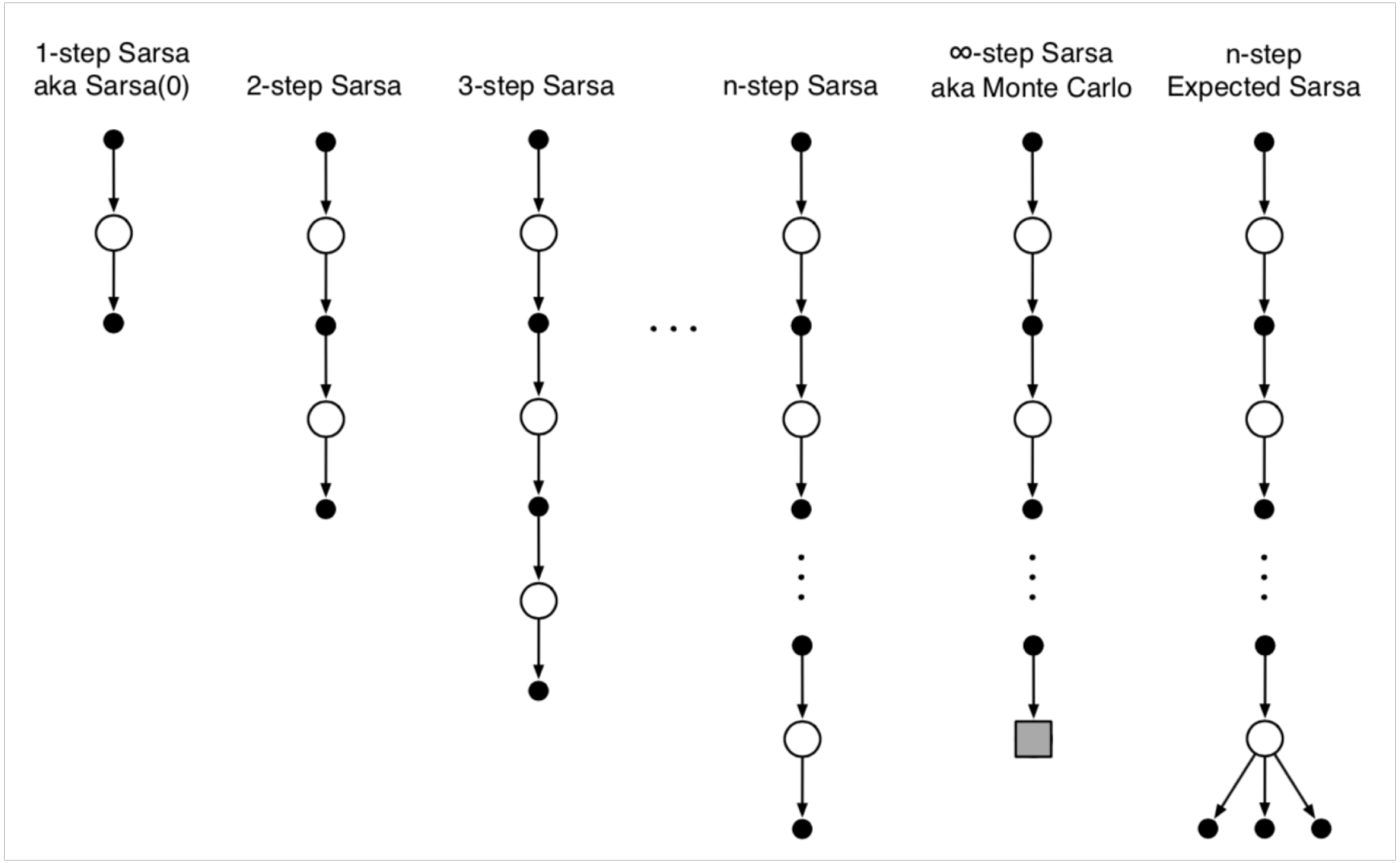
7.2 n-step Sarsa – n-step expected Sarsa

- n-step째의 value function을 expected value로 구함

- $$G_{t:t+n} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^{n} \bar{V}_{t+n-1}(S_{t+n}), \qquad t+n < T, \tag{7.7}$$

- $$\bar{V}_t(s) \doteq \sum_a \pi(a|s) Q_t(s,a), \qquad \text{for all } s \in \mathcal{S}. \tag{7.8}$$

7.2 n-step Sarsa



7.3 n-step Off-policy Learning

- Using importance sampling ratio concept

- $$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha \rho_{t:t+n-1} \left[G_{t:t+n} - V_{t+n-1}(S_t) \right], \qquad 0 \leq t < T, \tag{7.9}$$

- $$\rho_{t:h} \doteq \prod_{k=t}^{\min(h,T-1)} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}. \tag{7.10}$$

- Sarsa를 위해 수정 (state value function → state-action value function)

- $$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1:t+n} \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t) \right], \tag{7.11}$$

- 자세한 내용은 MC때와 동일...

7.3 n-step Off-policy Learning

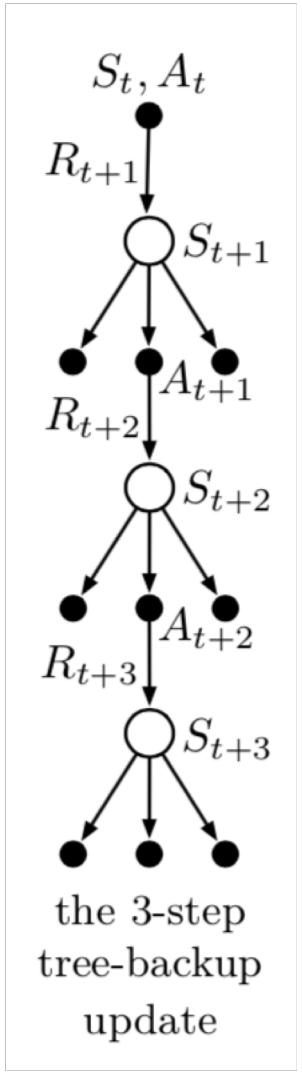
Off-policy n-step Sarsa for estimating $Q \approx q_*$ or q_π

Input: an arbitrary behavior policy b such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations (for 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 $A_0 \sim b(\cdot|S_0)$
  $T \leftarrow \infty$
  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 an action $A_{t+1} \sim b(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | | $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ $(\rho_{\tau+1:t+n-1})$
  | | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
  | | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\left[G - Q(S_\tau, A_\tau)\right]$
  | | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
  Until $\tau = T - 1$

7.5 The n-step Tree Backup Algorithm

- Importance sampling 없이 off-policy learning을 할 수 있음
- 행동에 의한 reward를 선택될 확률을 곱해서 받음
- n-step Expected Sarsa?

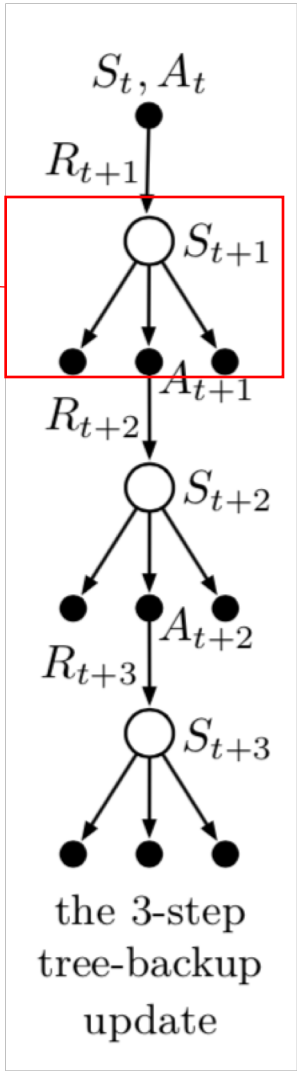


the 3-step tree-backup update

7.5 The n-step Tree Backup Algorithm

- One-step return (same as expected sarsa),

$$G_{t:t+1} \doteq R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a), \qquad (7.15)$$



the 3-step tree-backup update

7.5 The n-step Tree Backup Algorithm

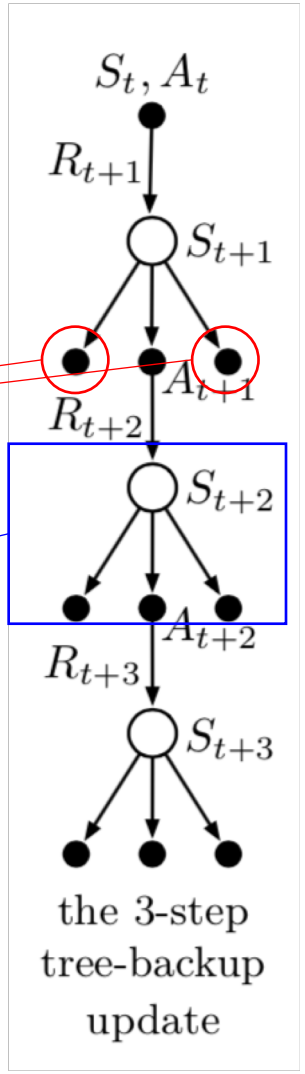
- One-step return (same as expected sarsa),

$$G_{t:t+1} \doteq R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_t(S_{t+1}, a), \qquad (7.15)$$

- Two-step return

$$G_{t:t+2} \doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) \left(R_{t+2} + \gamma \sum_a \pi(a|S_{t+2}) Q_{t+1}(S_{t+2}, a) \right)$$

$$= R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+2},$$



the 3-step tree-backup update

7.5 The n-step Tree Backup Algorithm

- n-step return

$$G_{t:t+n} \doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+n-1}(S_{t+1}, a) \; + \; \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+n}, \quad (7.16)$$

- n-step return written as a sum of expectation-based TD errors

$$G_{t:t+n} = Q(S_t, A_t) + \sum_{k=t}^{\min(t+n-1, T-1)} \delta_k \prod_{i=t+1}^{k} \gamma \pi(A_i|S_i),$$

where $\delta_t \doteq R_{t+1} + \gamma \bar{V}_t(S_{t+1}) - Q(S_t, A_t)$ and $\bar{V}_t$ is given by (7.8).

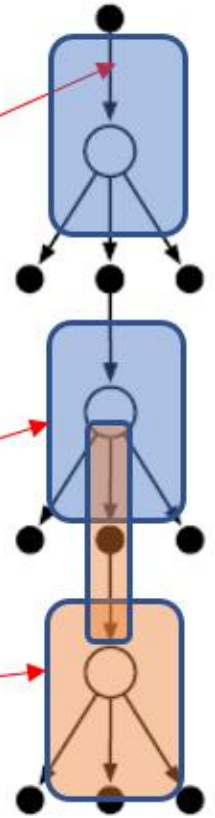
7.5 The n-step Tree Backup Algorithm

Second, we define a form of TD error:

$$\delta_t \doteq R_{t+1} + \gamma V_{t+1} - Q_{t-1}(S_t, A_t). \tag{7.11}$$

Using these we can define the n-step returns of the tree-backup algorithm as:

$$\begin{aligned}
G_t^{(1)} &\doteq R_{t+1} + \gamma V_{t+1} \quad \text{기대값} \quad \text{(same as the target of Expected Sarsa (6.9))} \\
&= Q_{t-1}(S_t, A_t) + \delta_t, \\
G_t^{(2)} &\doteq R_{t+1} + \gamma V_{t+1} - \gamma\pi(A_{t+1}|S_{t+1})Q_t(S_{t+1}, A_{t+1}) + \gamma\pi(A_{t+1}|S_{t+1})\big[R_{t+2} + \gamma V_{t+2}\big] \\
&= R_{t+1} + \gamma V_{t+1} + \gamma\pi(A_{t+1}|S_{t+1})\big[R_{t+2} + \gamma V_{t+2} - Q_t(S_{t+1}, A_{t+1})\big] \\
&= R_{t+1} + \gamma V_{t+1} + \gamma\pi(A_{t+1}|S_{t+1})\delta_{t+1} \\
&= Q_{t-1}(S_t, A_t) + \delta_t + \gamma\pi(A_{t+1}|S_{t+1})\delta_{t+1}, \\
G_t^{(3)} &\doteq Q_{t-1}(S_t, A_t) + \delta_t + \gamma\pi(A_{t+1}|S_{t+1})\delta_{t+1} + \gamma^2\pi(A_{t+1}|S_{t+1})\pi(A_{t+2}|S_{t+2})\delta_{t+2}, \\
G_t^{(n)} &\doteq Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n-1,T-1)} \delta_k \prod_{i=t+1}^{k} \gamma\pi(A_i|S_i),
\end{aligned} \tag{7.12}$$



3-step tree backup

7.5 The n-step Tree Backup Algorithm

n-step Tree Backup for estimating $Q \approx q_*$ or q_π

Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer n
All store and access operations can take their index mod $n+1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    Choose an action A_0 arbitrarily as a function of S_0; Store A_0
    $T \leftarrow \infty$
    Loop for $t = 0, 1, 2, \ldots$:
    | If $t < T$:
    | Take action A_t; observe and store the next reward and state as R_{t+1}, S_{t+1}
    | If S_{t+1} is terminal:
    | $T \leftarrow t+1$
    | else:
    | Choose an action A_{t+1} arbitrarily as a function of S_{t+1}; Store A_{t+1}
    | $\tau \leftarrow t+1-n$ (τ is the time whose estimate is being updated)
    | If $\tau \geq 0$:
    | If $t+1 \geq T$:
    | $G \leftarrow R_T$
    | else
    | $G \leftarrow R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$
    | Loop for $k = \min(t, T-1)$ down through $\tau+1$:
    | $G \leftarrow R_k + \gamma \sum_{a \neq A_k} \pi(a|S_k) Q(S_k, a) + \gamma \pi(A_k|S_k) G$
    | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha \left[G - Q(S_\tau, A_\tau) \right]$
    | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
    Until $\tau = T-1$

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

- n-step TD + MC + Tree Backup Algorithm
- 각 단계의 σ 값에 따라서 expectation(tree)으로 update할 지 sampling(TD)으로 update할 지 결정
  - $\sigma = 0$: expectation (Tree Backup Algorithm)
  - $\sigma = 1$: sampling (TD)

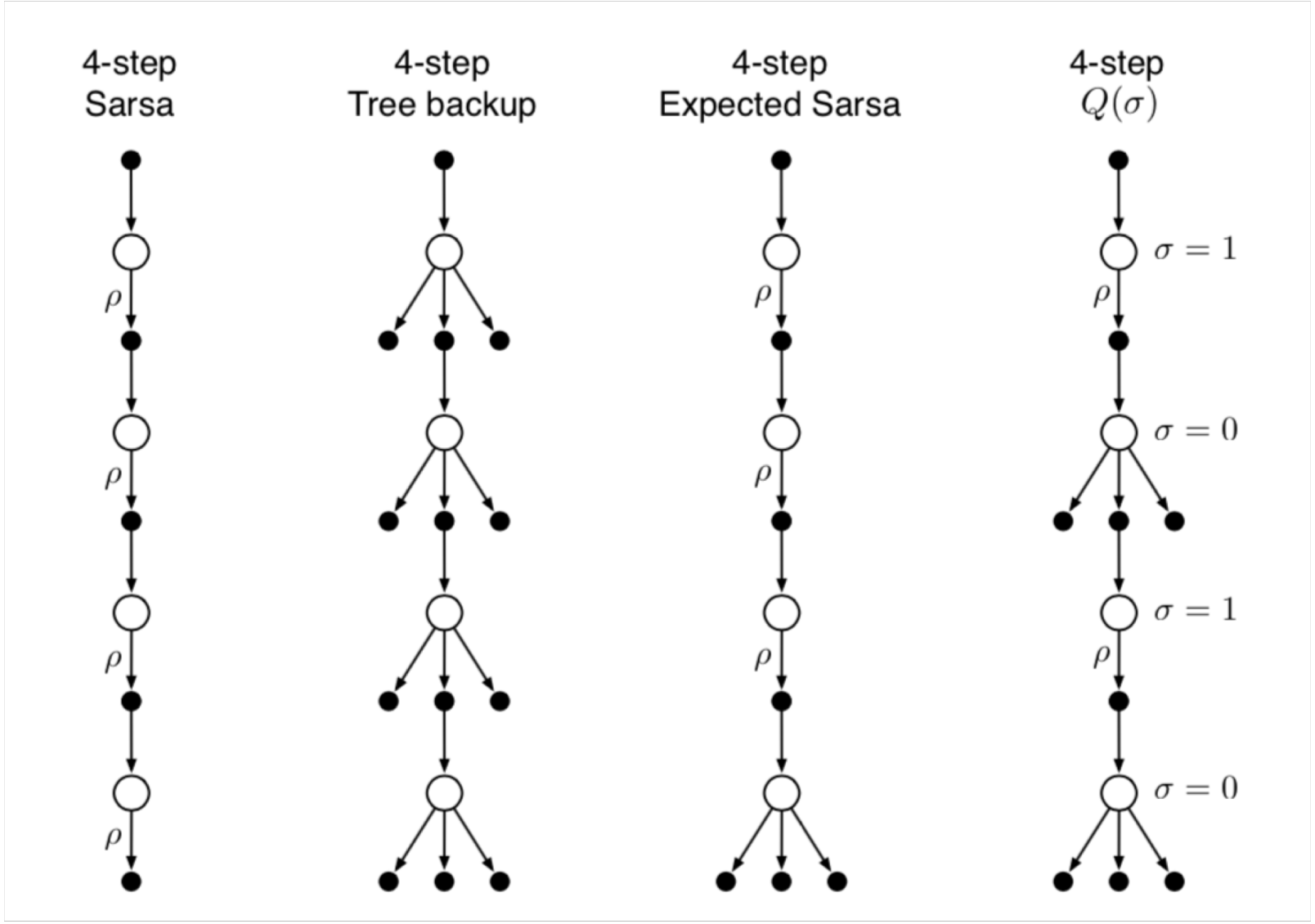
7.6 A Unifying Algorithm: n-step $Q(\sigma)$

Off-policy n-step $Q(\sigma)$ for estimating $Q \approx q_*$ or q_π

Input: an arbitrary behavior policy b such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n
All store and access operations can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Choose and store an action $A_0 \sim b(\cdot|S_0)$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
    If $t < T$:
      Take action A_t; observe and store the next reward and state as R_{t+1}, S_{t+1}
      If S_{t+1} is terminal:
        $T \leftarrow t+1$
      else:
        Choose and store an action $A_{t+1} \sim b(\cdot|S_{t+1})$
        Select and store σ_{t+1}
        Store $\frac{\pi(A_{t+1}|S_{t+1})}{b(A_{t+1}|S_{t+1})}$ as ρ_{t+1}
    $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    If $\tau \geq 0$:
      $G \leftarrow 0$:
      Loop for $k = \min(t+1, T)$ down through $\tau + 1$:
        if $k = T$:
          $G \leftarrow R_T$
        else:
          $\bar{V} \leftarrow \sum_a \pi(a|S_k) Q(S_k, a)$
          $G \leftarrow R_k + \gamma\big(\sigma_k \rho_k + (1-\sigma_k)\pi(A_k|S_k)\big)\big(G - Q(S_k, A_k)\big) + \gamma \bar{V}$
      $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[G - Q(S_\tau, A_\tau)]$
      If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt Q
  Until $\tau = T - 1$

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



7.7 Summary

- one-step TD + MC = n-step TD
- n-step Sarsa
- n-step Expected Sarsa
- n-step Off-policy Learning with Importance Sampling
- n-step Tree Backup Algorithm
- n-step $Q(\sigma)$

Thank you for listening!