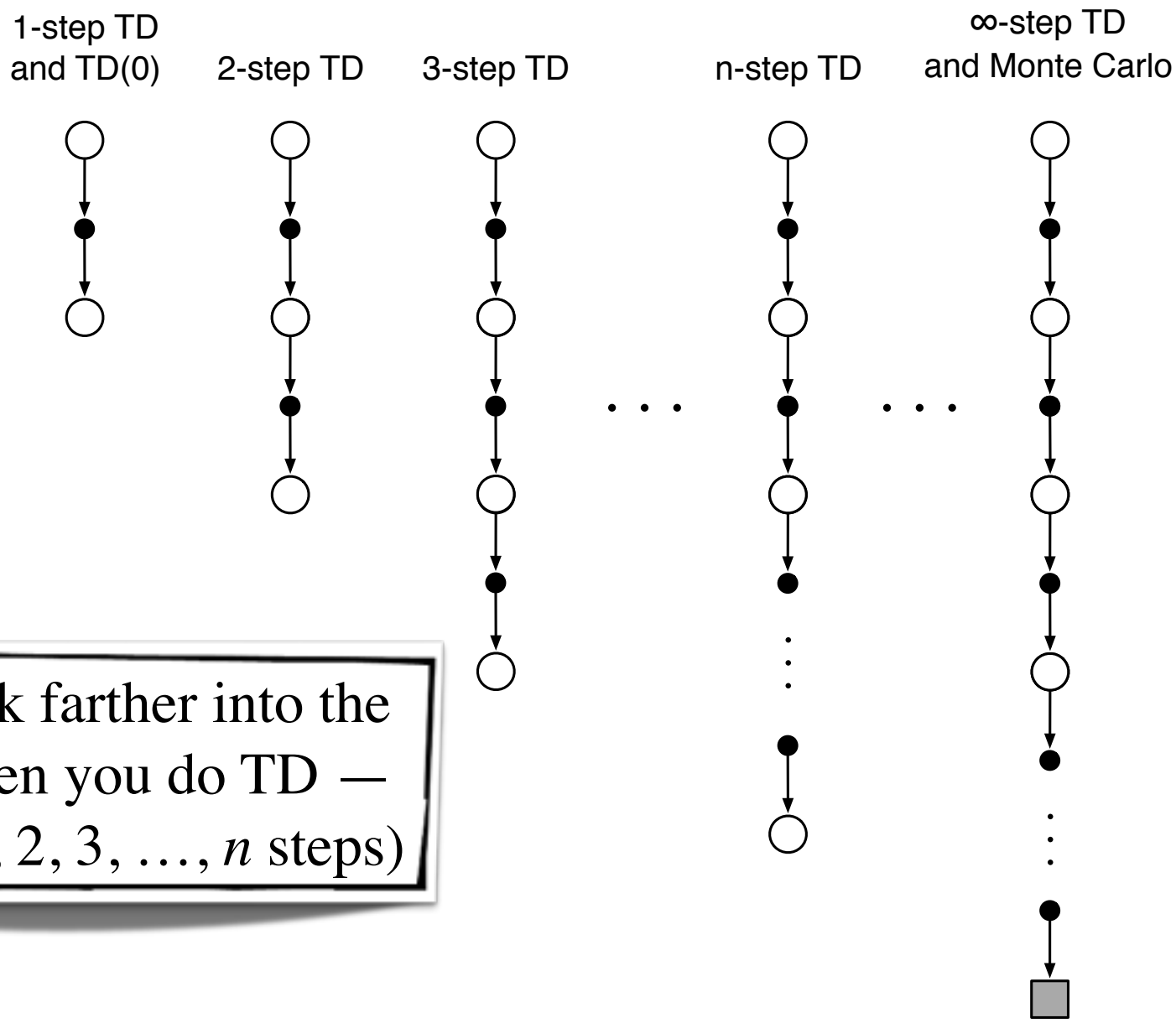
Chapter 7:

Multi-step Bootstrapping

Unifying Monte Carlo and TD

key algorithms: n-step TD, n-step Sarsa, Tree-backup, Q(σ)

n-step TD Prediction



Idea: Look farther into the future when you do TD — backup (1, 2, 3, …, *n* steps)

Mathematics of *n*-step TD Returns/Targets

- Monte Carlo: $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$
- TD: $G_t^{(1)} \doteq R_{t+1} + \gamma V_t(S_{t+1})$
  - Use V_t to estimate remaining return
- *n*-step TD:
  - 2 step return: $G_t^{(2)} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+1})$
  - *n*-step return: $G_t^{(n)} \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$

    with $G_t^{(n)} \doteq G_t$ if $t + n \geq T$

n-step TD

- Recall the *n*-step return:

$$G_t^{(n)} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}), \quad n \geq 1, 0 \leq t < T-n$$

- Of course, this is not available until time *t+n*

- The natural algorithm is thus to wait until then:

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha \left[G_t^{(n)} - V_{t+n-1}(S_t) \right], \qquad 0 \leq t < T$$

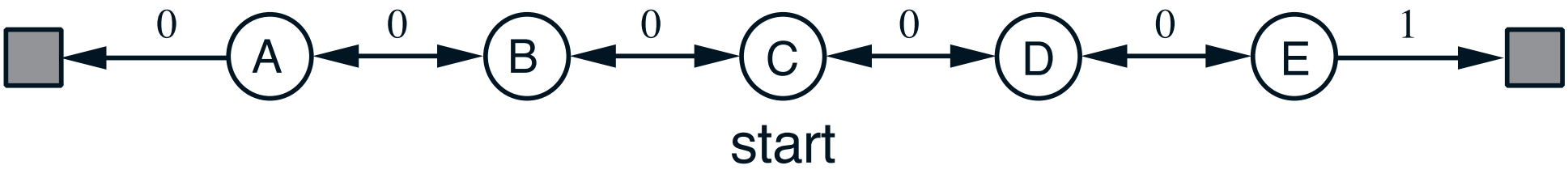
- This is called *n*-step TD

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

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

Repeat (for each episode):
    Initialize and store $S_0 \neq$ terminal
    $T \leftarrow \infty$
    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 $\tau \geq 0$:
    | | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
    | | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ ($G_\tau^{(n)}$)
    | | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
    Until $\tau = T - 1$

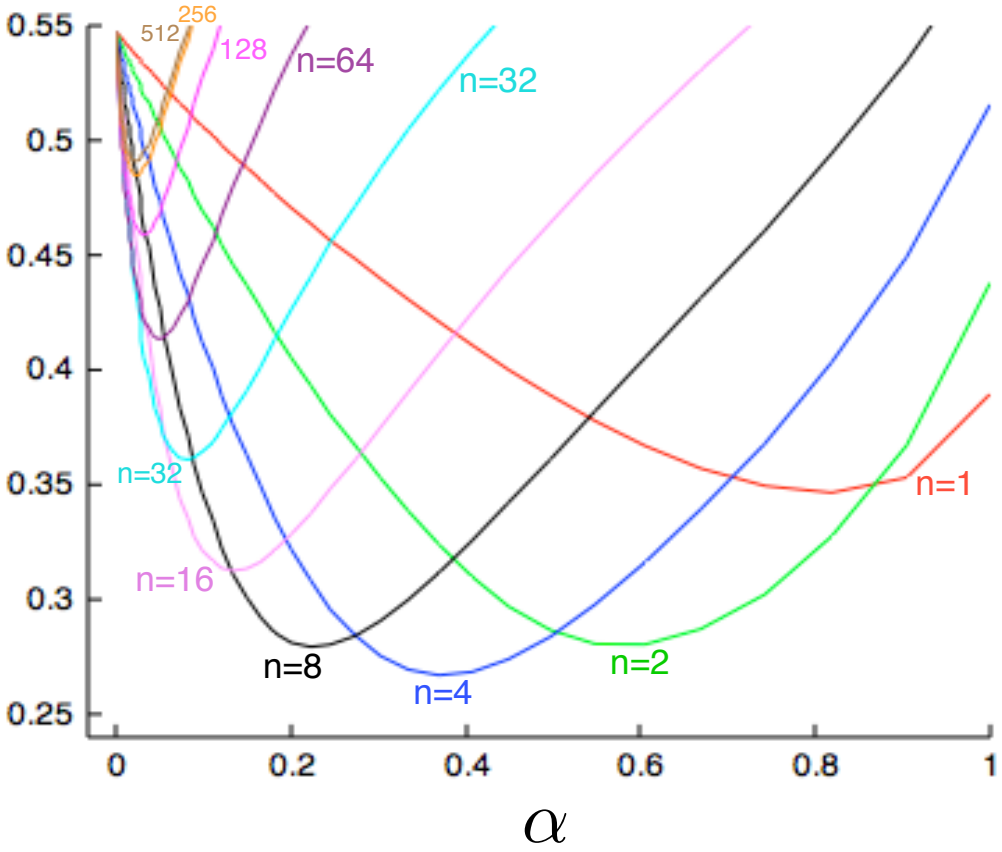
Random Walk Examples



- How does 2-step TD work here?
- How about 3-step TD?

A Larger Example – 19-state Random Walk

Average RMS error over 19 states and first 10 episodes



n-step TD results

- An intermediate α is best
- An intermediate n is best
- n>1 reduces variances in general

Conclusions Regarding *n*-step Methods (so far)

- Generalize Temporal-Difference and Monte Carlo learning methods, sliding from one to the other as n increases
  - $n = 1$ is TD as in Chapter 6
  - $n = \infty$ is MC as in Chapter 5
  - an intermediate n is often much better than either extreme
  - applicable to both continuing and episodic problems
- There is some cost in computation
  - need to remember the last n states
  - learning is delayed by n steps
  - per-step computation is small and uniform, like TD
- Everything generalizes nicely: error-reduction theory, Sarsa, off-policy by importance sampling, Expected Sarsa, Tree Backup
- The very general n-step Q(σ) algorithm includes everything!

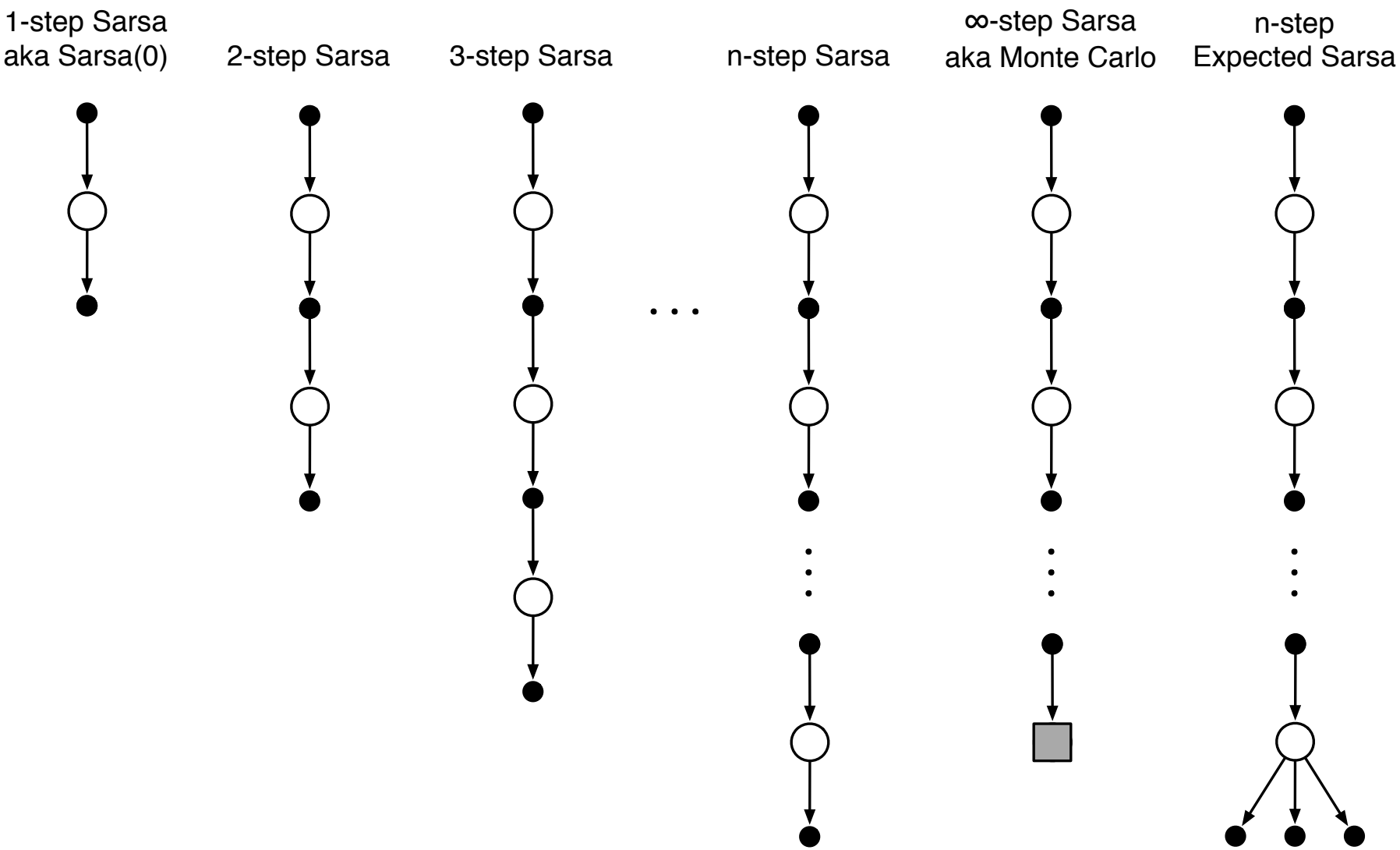
Error-reduction property

- Error reduction property of *n*-step returns

$$\underbrace{\max_s \left| \mathbb{E}_\pi \left[G_t^{(n)} \middle| S_t = s \right] - v_\pi(s) \right|}_{\text{Maximum error using } n\text{-step return}} \leq \underbrace{\gamma^n \max_s \left| V_t(s) - v_\pi(s) \right|}_{\text{Maximum error using V}}$$

- Using this, you can show that *n*-step methods converge

Multi-step backup for action values



On-policy *n*-step Action-value Methods

- Action-value form of *n*-step return

$$G_t^{(n)} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n})$$

- *n*-step Sarsa:

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \left[G_t^{(n)} - Q_{t+n-1}(S_t, A_t) \right]$$

- *n*-step Expected Sarsa is the same update with a slightly different *n*-step return:

$$G_t^{(n)} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \sum_a \pi(a|S_{t+n}) Q_{t+n-1}(S_{t+n}, a)$$

Off-policy *n*-step Methods by Importance Sampling

- The *importance-sampling ratio for n steps*:

$$\rho_t^{t+n} \doteq \prod_{k=t}^{\min(t+n-1,T-1)} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$$

- We get off-policy methods by weighting updates by this ratio
- Off-policy *n*-step TD:

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\rho_t^{t+n}\left[G_t^{(n)} - V_{t+n-1}(S_t)\right]$$

- Off-policy *n*-step Sarsa:

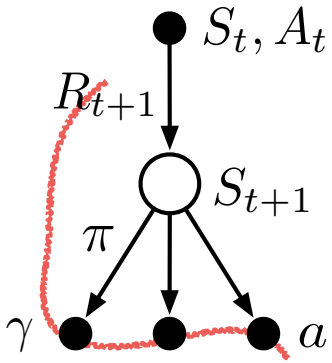
$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha\rho_{t+1}^{t+n}\left[G_t^{(n)} - Q_{t+n-1}(S_t, A_t)\right]$$

- Off-policy *n*-step Expected Sarsa:

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha\,\rho_{t+1}^{t+n-1}\left[G_t^{(n)} - Q_{t+n-1}(S_t, A_t)\right]$$

Off-policy Learning w/o Importance Sampling: The *n*-step Tree Backup Algorithm

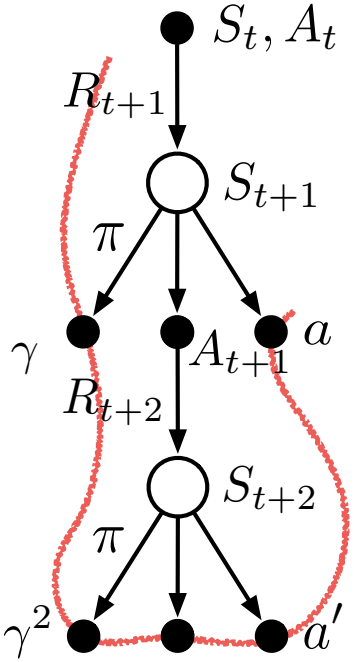
Expected Sarsa
and 1-step Tree Backup



$$R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

Target

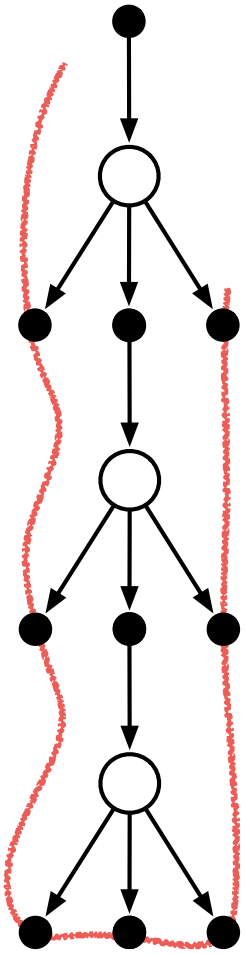
2-step Tree Backup



$$R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

$$+\gamma\, \pi(A_{t+1}|S_{t+1}) \left(R_{t+2} + \gamma \sum_{a'} \pi(a'|S_{t+2}) Q(S_{t+2}, a') \right)$$

3-step TB



n-step Tree Backup

- Expected Action Value

$$V_t \doteq \sum_a \pi(a|S_t) Q_{t-1}(S_t, a).$$

- New form of TD error

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

- n-step return

$$\begin{aligned} G_t^{(1)} &\doteq R_{t+1} + \gamma V_{t+1} \\ &= Q_{t-1}(S_t, A_t) + \delta_t, \end{aligned}$$

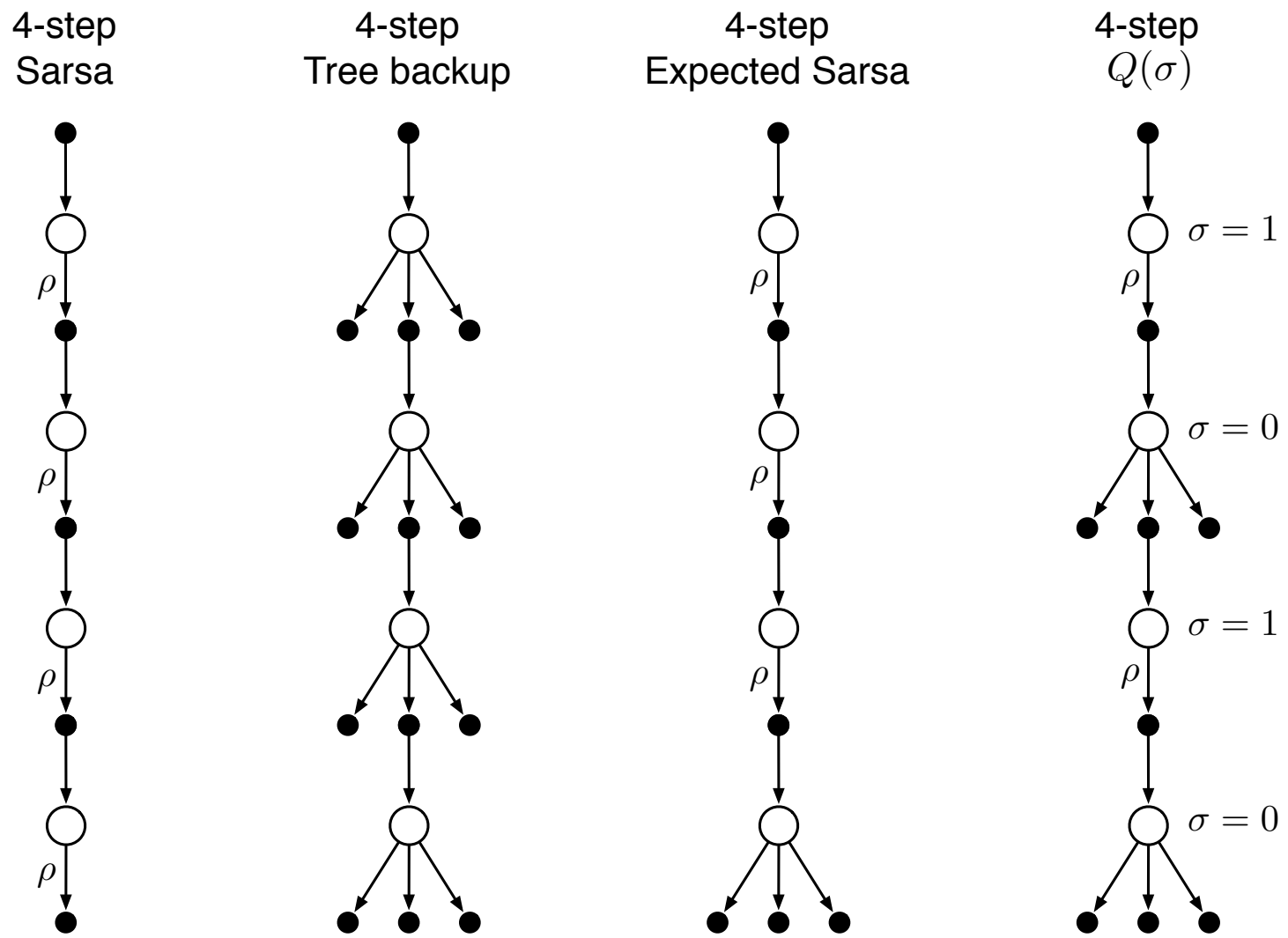
$$G_t^{(n)} \doteq Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n-1, T-1)} \delta_k \prod_{i=t+1}^{k} \gamma \pi(A_i|S_i),$$

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

Initialize $Q(s,a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or as a fixed given policy
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

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  Store $Q(S_0, A_0)$ as Q_0
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If $t < T$:
  | | Take action A_t
  | | Observe the next reward R; observe and store the next state as S_{t+1}
  | | If S_{t+1} is terminal:
  | | | $T \leftarrow t+1$
  | | | Store $R - Q_t$ as δ_t
  | | else:
  | | | Store $R + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q_t$ as δ_t
  | | | Select arbitrarily and store an action as A_{t+1}
  | | | Store $Q(S_{t+1}, A_{t+1})$ as Q_{t+1}
  | | | Store $\pi(A_{t+1}|S_{t+1})$ as π_{t+1}
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | | $E \leftarrow 1$
  | | $G \leftarrow Q_\tau$
  | | For $k = \tau, \ldots, \min(\tau + n - 1, T - 1)$:
  | | | $G \leftarrow G + E\delta_k$
  | | | $E \leftarrow \gamma E \pi_{k+1}$
  | | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | | If π is being learned, then ensure that $\pi(a|S_\tau)$ is ε-greedy wrt $Q(S_\tau, \cdot)$
  Until $\tau = T - 1$

A Unifying Algorithm: *n*-step Q(σ)



Choose whether to sample or take the expectation *on each step* with $\sigma(s)$

Return in *n*-step Q(σ)

- TD error

$$\delta_t \doteq R_{t+1} + \gamma\big[\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) + (1 - \sigma_{t+1}) V_{t+1}\big] - Q_{t-1}(S_t, A_t).$$

- Return

$$\begin{aligned}
G_t^{(1)} &\doteq R_{t+1} + \gamma\big[\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) + (1 - \sigma_{t+1}) V_{t+1}\big] \\
&= \delta_t + Q_{t-1}(S_t, A_t), \\
G_t^{(2)} &\doteq R_{t+1} + \gamma\big[\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) + (1 - \sigma_{t+1}) V_{t+1}\big] \\
&\quad - \gamma(1 - \sigma_{t+1})\pi(A_{t+1}|S_{t+1}) Q_t(S_{t+1}, A_{t+1}) \\
&\quad + \gamma(1 - \sigma_{t+1})\pi(A_{t+1}|S_{t+1})\big[R_{t+2} + \gamma\big[\sigma_{t+2} Q_t(S_{t+2}, A_{t+2}) + (1 - \sigma_{t+2}) V_{t+2}\big]\big] \\
&\quad - \gamma\sigma_{t+1} Q_t(S_{t+1}, A_{t+1}) \\
&\quad + \gamma\sigma_{t+1}\big[R_{t+2} + \gamma\big[\sigma_{t+2} Q_t(S_{t+2}, A_{t+2}) + (1 - \sigma_{t+2}) V_{t+2}\big]\big] \\
&= Q_{t-1}(S_t, A_t) + \delta_t \\
&\quad + \gamma(1 - \sigma_{t+1})\pi(A_{t+1}|S_{t+1})\delta_{t+1} \\
&\quad + \gamma\sigma_{t+1}\delta_{t+1} \\
&= Q_{t-1}(S_t, A_t) + \delta_t + \gamma\big[(1 - \sigma_{t+1})\pi(A_{t+1}|S_{t+1}) + \sigma_{t+1}\big]\delta_{t+1} \\
G_t^{(n)} &\doteq Q_{t-1}(S_t, A_t) + \sum_{k=t}^{\min(t+n-1, T-1)} \delta_k \prod_{i=t+1}^{k} \gamma\big[(1 - \sigma_i)\pi(A_i|S_i) + \sigma_i\big]. \qquad (7.14)
\end{aligned}$$

n-step Q(σ) Algorithm

- With importance sampling

$$\rho_t^{t+n} \doteq \prod_{k=t}^{\min(t+n-1,T-1)} \left(\sigma_k \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)} + 1 - \sigma_k\right).$$

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

Input: an arbitrary behavior policy μ such that $\mu(a|s) > 0, \forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or as a fixed given policy
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

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \mu(\cdot|S_0)$
  Store $Q(S_0, A_0)$ as Q_0
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
    If $t < T$:
      Take action A_t
      Observe the next reward R; observe and store the next state as S_{t+1}
      If S_{t+1} is terminal:
        $T \leftarrow t+1$
        Store $\delta_t \leftarrow R - Q_t$
      else:
        Select and store an action $A_{t+1} \sim \mu(\cdot|S_{t+1})$
        Select and store σ_{t+1}
        Store $Q(S_{t+1}, A_{t+1})$ as Q_{t+1}
        Store $R + \gamma\sigma_{t+1}Q_{t+1} + \gamma(1-\sigma_{t+1})\sum_a \pi(a|S_{t+1})Q(S_{t+1},a) - Q_t$ as δ_t
        Store $\pi(A_{t+1}|S_{t+1})$ as π_{t+1}
        Store $\frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})}$ as ρ_{t+1}
    $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    If $\tau \geq 0$:
      $\rho \leftarrow 1$
      $E \leftarrow 1$
      $G \leftarrow Q_\tau$
      For $k = \tau, \ldots, \min(\tau + n - 1, T-1)$:
        $G \leftarrow G + E\delta_k$
        $E \leftarrow \gamma E\left[(1-\sigma_{k+1})\pi_{k+1} + \sigma_{k+1}\right]$
        $\rho \leftarrow \rho(1 - \sigma_k + \sigma_k\rho_k)$
      $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\left[G - Q(S_\tau, A_\tau)\right]$
      If π is being learned, then ensure that $\pi(a|S_\tau)$ is ε-greedy wrt $Q(S_\tau, \cdot)$
  Until $\tau = T - 1$

Conclusions Regarding *n*-step Methods

- Generalize Temporal-Difference and Monte Carlo learning methods, sliding from one to the other as n increases
  - $n = 1$ is TD as in Chapter 6
  - $n = \infty$ is MC as in Chapter 5
  - an intermediate n is often much better than either extreme
  - applicable to both continuing and episodic problems
- There is some cost in computation
  - need to remember the last n states
  - learning is delayed by n steps
  - per-step computation is small and uniform, like TD
- Everything generalizes nicely: error-reduction theory, Sarsa, off-policy by importance sampling, Expected Sarsa, Tree Backup
- The very general n-step Q(σ) algorithm includes everything!