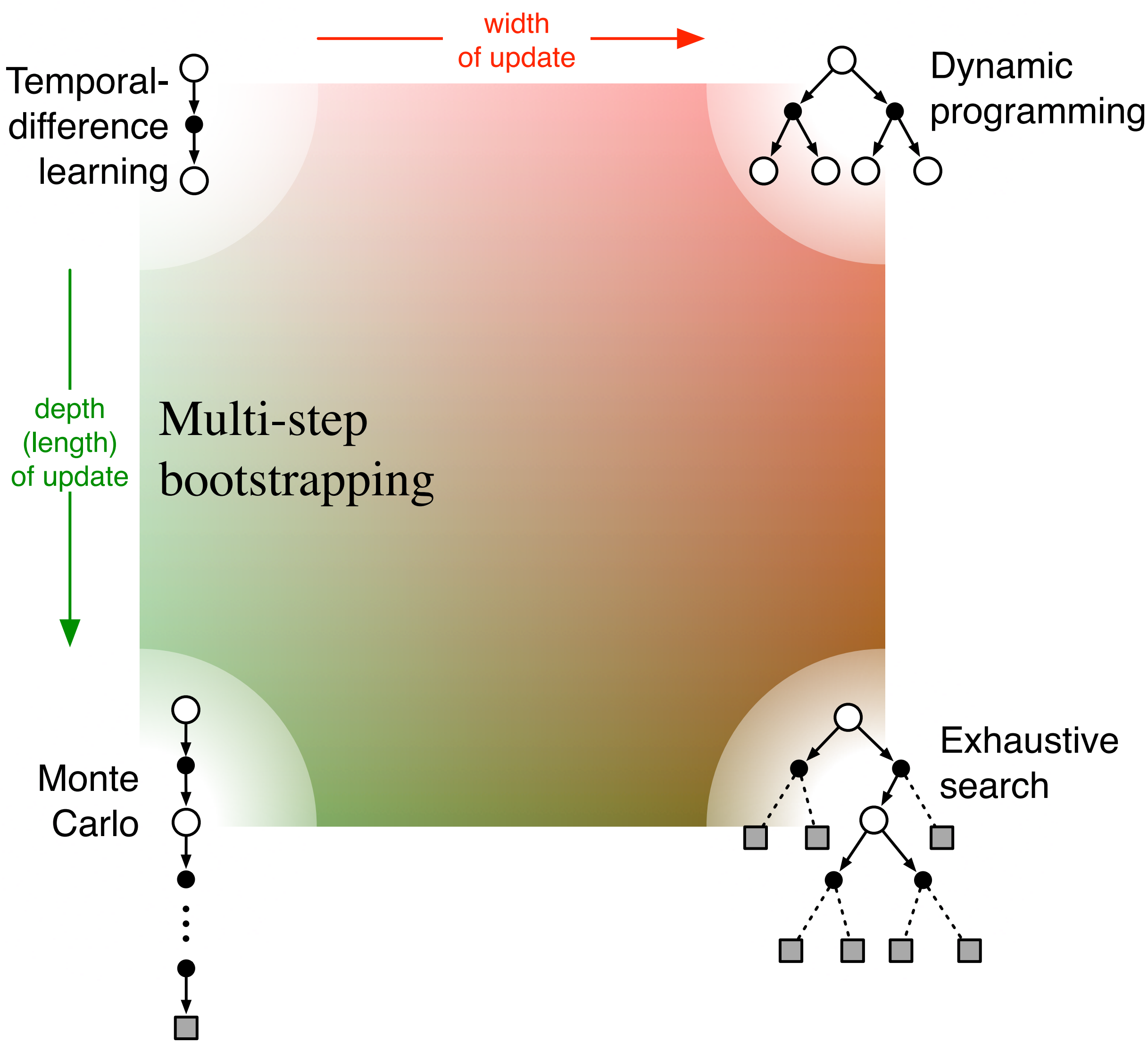
Chapter 7:

Multi-step Bootstrapping

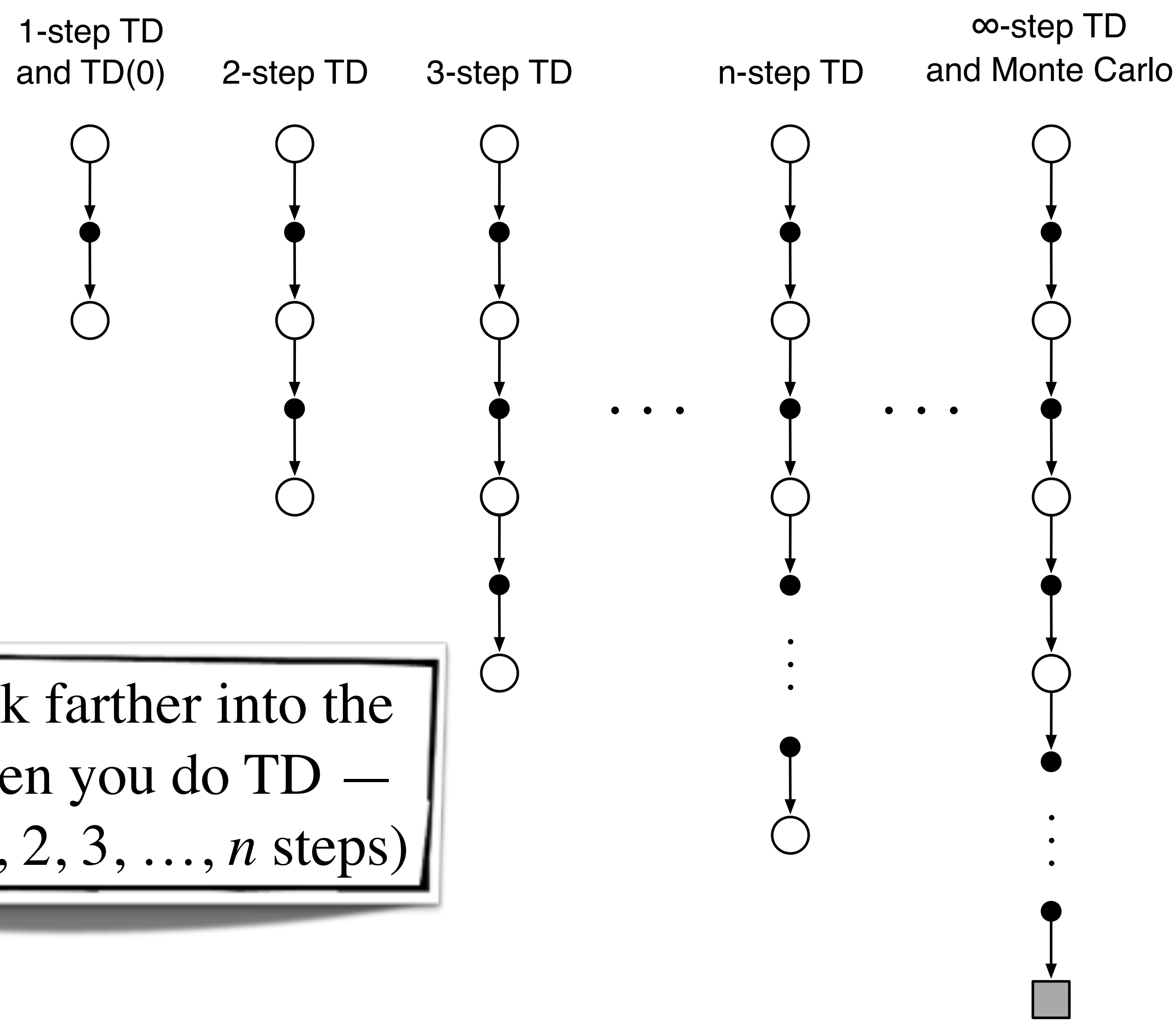
Unifying Monte Carlo and TD

key algorithms: n-step TD, n-step Sarsa, Tree-backup, $Q(\sigma)$

Unified View



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: $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$

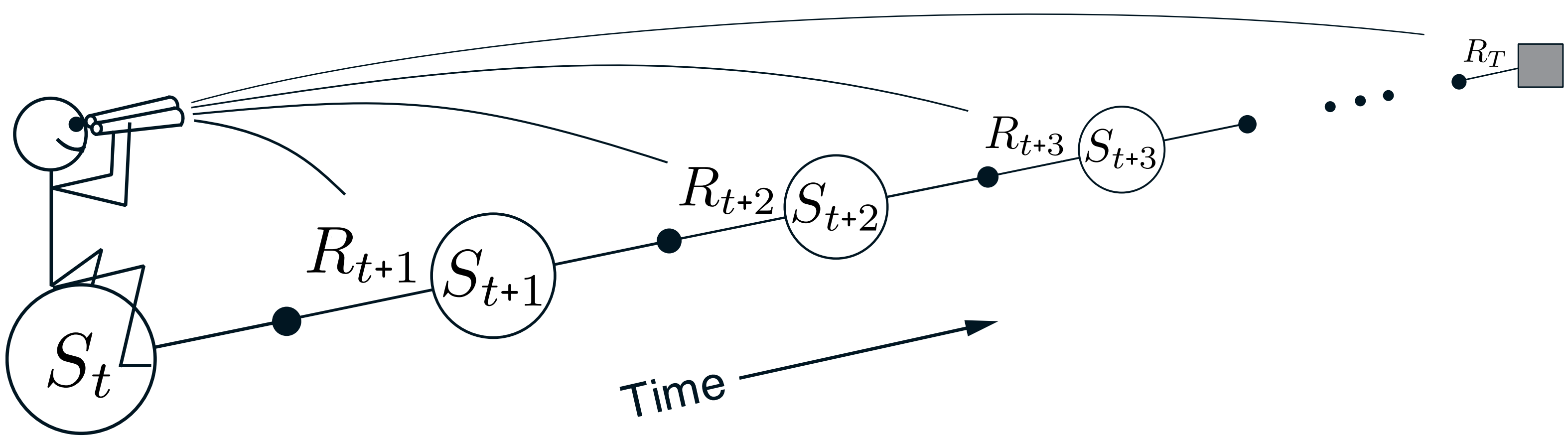
- TD: $G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$
  - Use V_t to estimate remaining return

- n-step TD:
  - 2 step return: $G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$
  - n-step return: $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$

    with $G_{t:t+n} \doteq G_t$ if $t+n \geq T$)

Forward View of TD(λ)

- Look forward from each state to determine update from future states and rewards:



***n*-step TD**

- Recall the *n*-step return:

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

- Of course, this is not available until time *t+n*
- The natural algorithm is thus to wait until then:

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

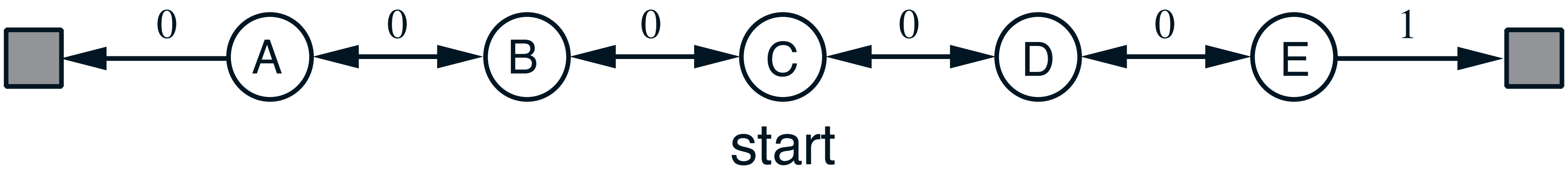
- This is called *n*-step TD

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

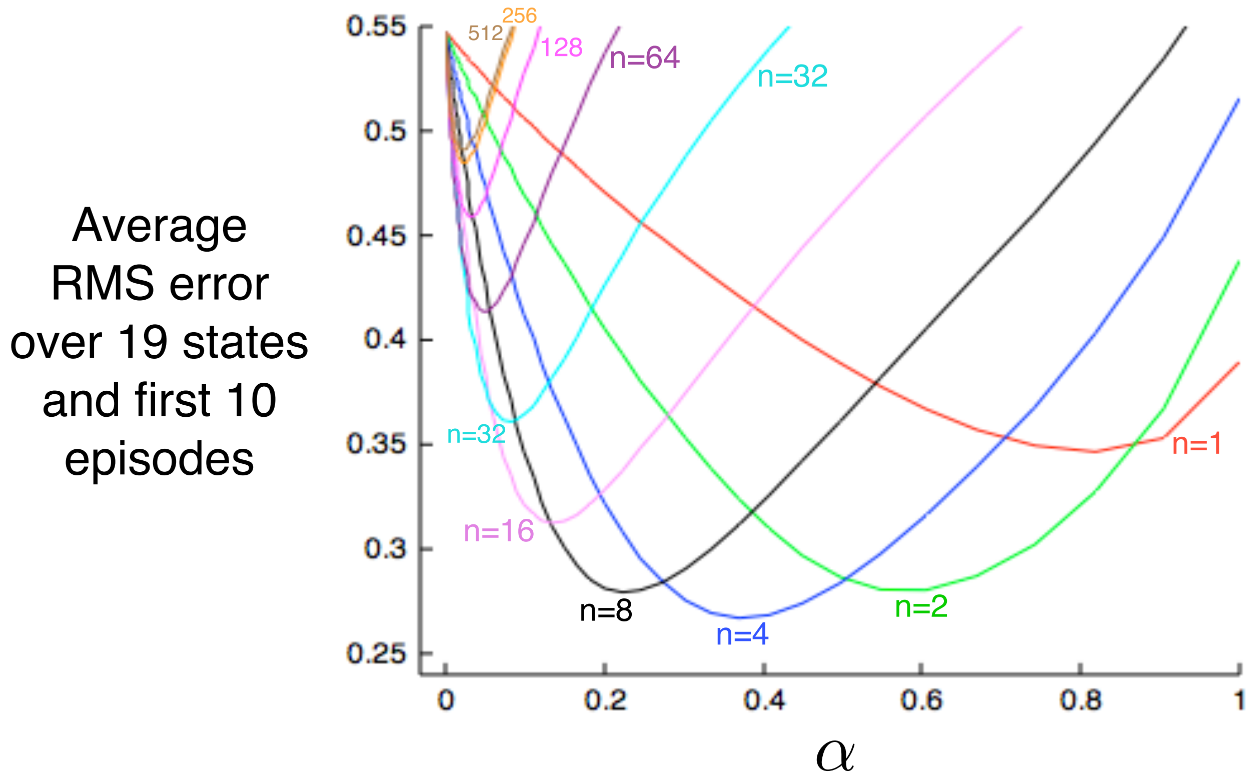
Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take an action according to $\pi(\cdot|S_t)$
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, 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: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha \left[G - V(S_\tau)\right]$
  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



n-step TD results

- An intermediate α is best
- An intermediate n is best
- Do you think there is an optimal n? for every task?

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!

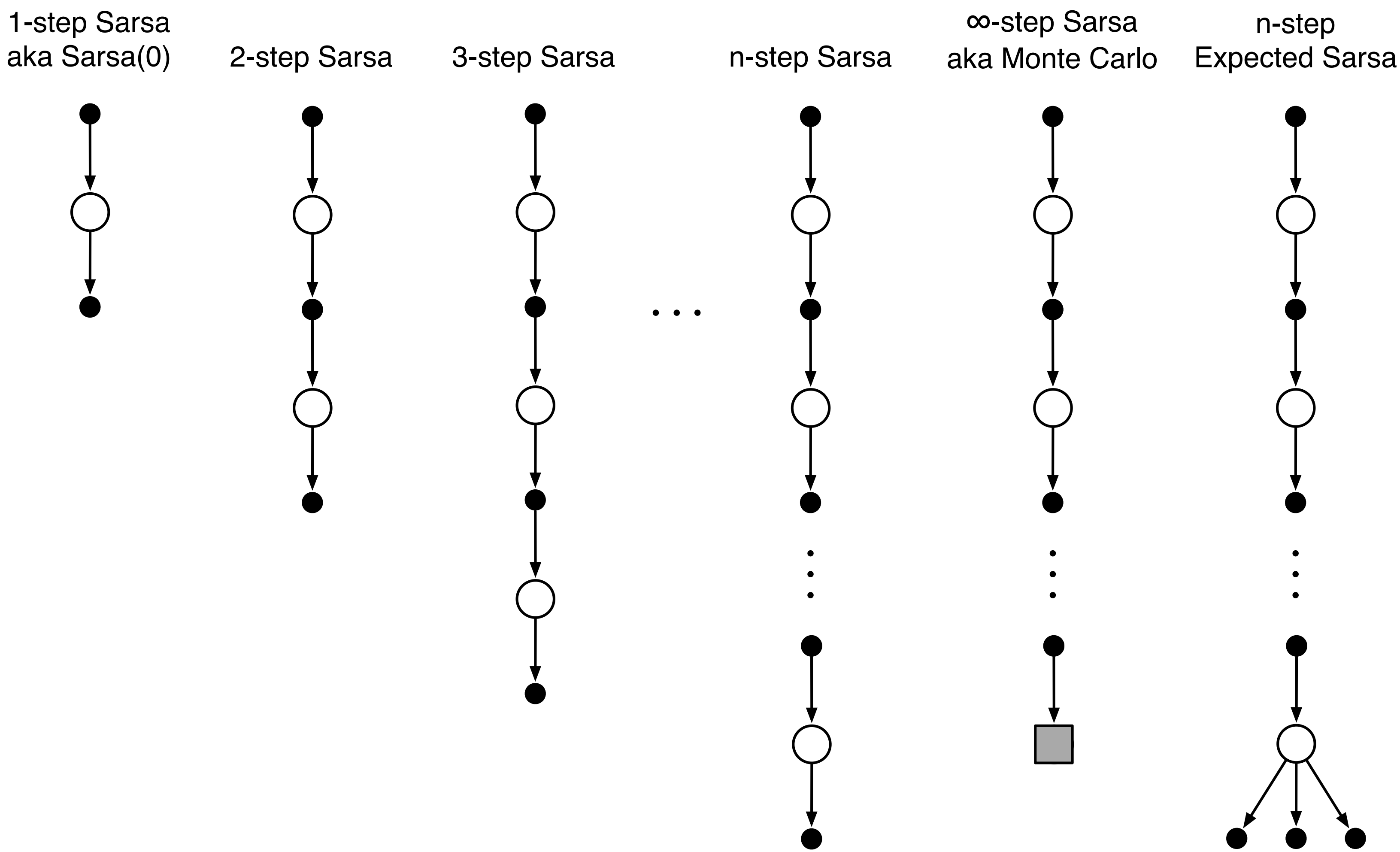
Error-reduction property

- Error reduction property of *n*-step returns

$$\underbrace{\max_s \Big| \mathbb{E}_\pi[G_{t:t+n} | S_t = s] - v_\pi(s) \Big|}_{\text{Maximum error using } n\text{-step return}} \leq \gamma^n \underbrace{\max_s \Big| V_{t+n-1}(s) - v_\pi(s) \Big|}_{\text{Maximum error using V}}$$

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

It's much the same for action values



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

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

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

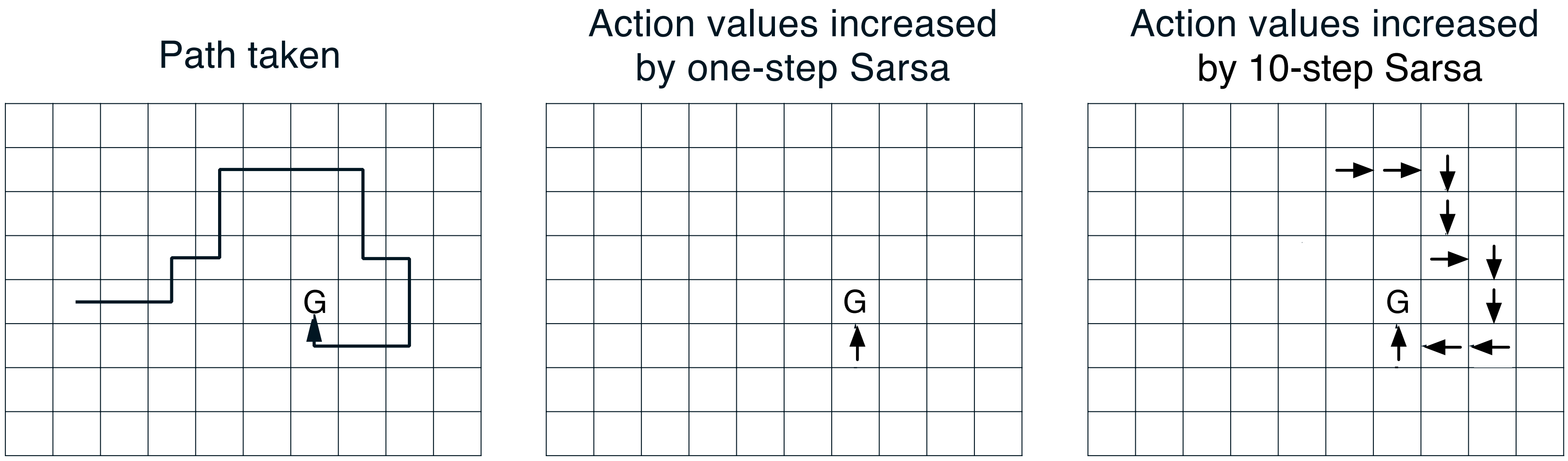
- *n*-step Sarsa:

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

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

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

n-step Methods Can Accelerate Learning



Off-policy n-step Methods by Importance Sampling

- Recall the *importance-sampling ratio*:

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

- We get off-policy methods by weighting updates by this ratio

- Off-policy n-step TD:

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

- Off-policy n-step Sarsa:

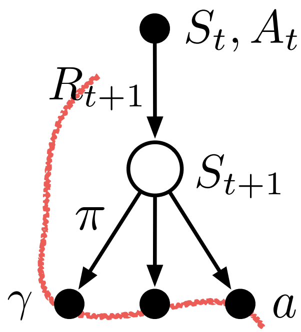
$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha\, \rho_{t+1:t+n-1} \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t) \right]$$

- Off-policy n-step Expected Sarsa:

Just like above except expected n-step return and ρ goes to $t + n - 2$

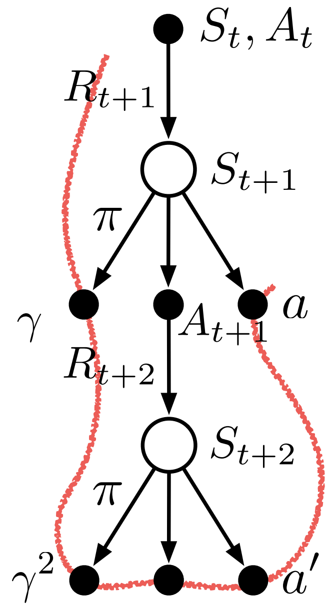
Off-policy Learning w/o Importance Sampling: The *n*-step Tree Backup Algorithm

Expected Sarsa
and 1-step Tree Backup

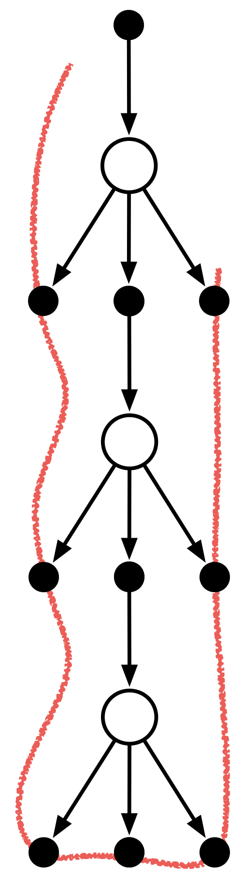


$$R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

2-step Tree Backup



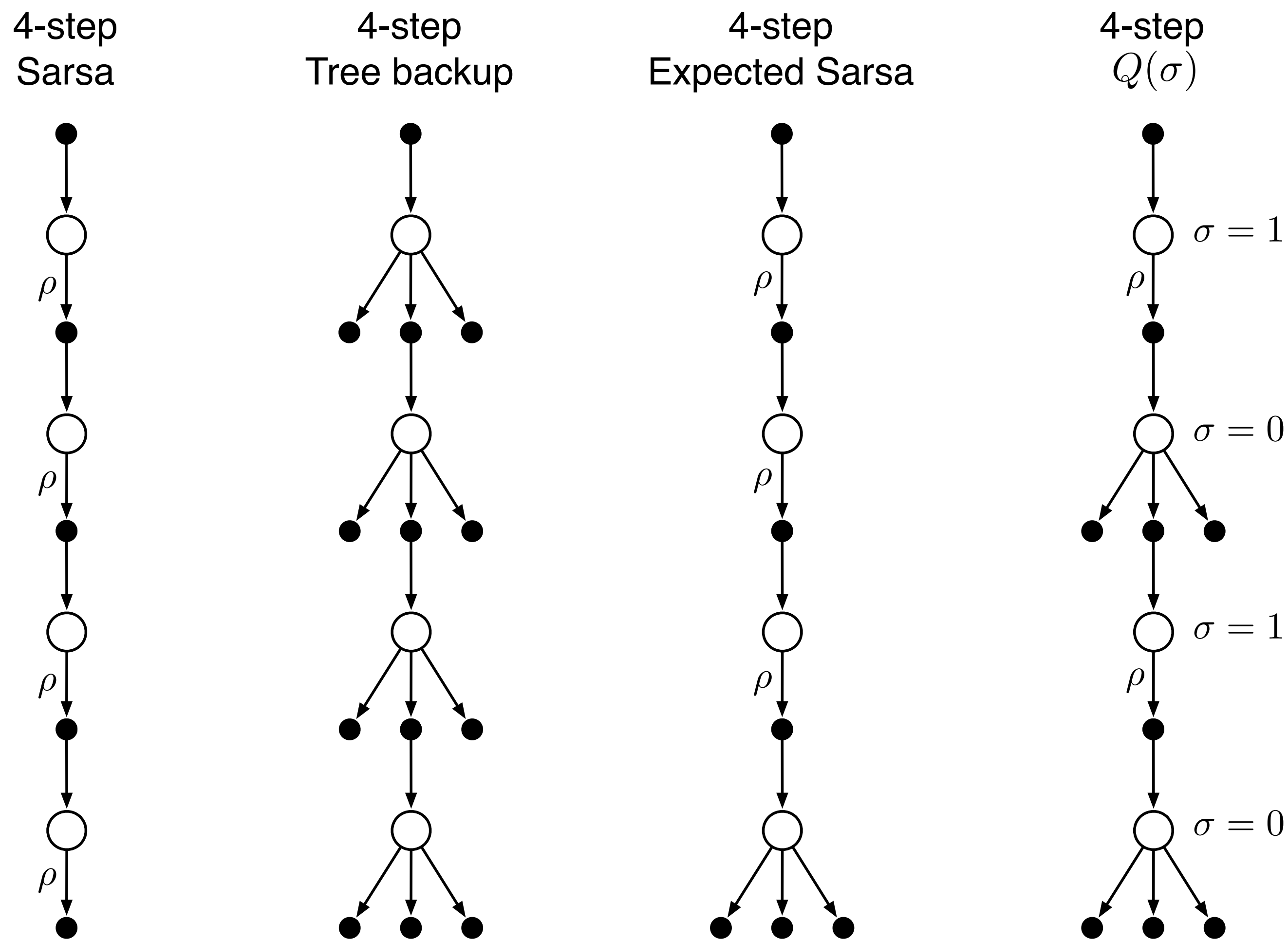
3-step TB



Target

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

A Unifying Algorithm: *n*-step Q(σ)



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

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!