Chapter 7:

Multi-step Bootstrapping

Unifying Monte Carlo and TD

key algorithms: n-step TD, n-step Sarsa, Tree-backup

What is bootstrapping

What is bootstrapping

- Updating your estimate of the value function in one state using your estimate of the value function in the next state

What is bootstrapping

- Updating your estimate of the value function in one state using your estimate of the value function in the next state
  - $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V_t(S_{t+1}) \mid S_t = s]$

What is bootstrapping

- Updating your estimate of the value function in one state using your estimate of the value function in the next state
  - $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V_t(S_{t+1}) \mid S_t = s]$
- Bootstrapping methods can often be more efficient than averaging sample returns

What is bootstrapping

- Updating your estimate of the value function in one state using your estimate of the value function in the next state
  - $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V_t(S_{t+1}) \mid S_t = s]$
- Bootstrapping methods can often be more efficient than averaging sample returns
- But sometimes $V_t(S_{t+1})$ may not be that accurate

What is bootstrapping

- Updating your estimate of the value function in one state using your estimate of the value function in the next state
  - $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V_t(S_{t+1}) \mid S_t = s]$
- Bootstrapping methods can often be more efficient than averaging sample returns
- But sometimes $V_t(S_{t+1})$ may not be that accurate
- Therefore the target, $R_{t+1} + \gamma V_t(S_{t+1})$, may not work well

Better bootstrapping?

Better bootstrapping?

- $V_{t+1}(s) \leftarrow \mathbb{E}_{\pi}[R_{t+1} + \gamma\, V_t(S_{t+1}) \mid S_t = s]$

Better bootstrapping?

- $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma\, V_t(S_{t+1}) \mid S_t = s]$

- But sometimes $V_t(S_{t+1})$ may not be that accurate

Better bootstrapping?

- $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V_t(S_{t+1}) \mid S_t = s]$

- But sometimes $V_t(S_{t+1})$ may not be that accurate
- Why?

Better bootstrapping?

- $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma\, V_t(S_{t+1}) \mid S_t = s]$

- But sometimes $V_t(S_{t+1})$ may not be that accurate
- Why?
  - rewards might be very sparse. Demo!

Better bootstrapping?

- $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V_t(S_{t+1}) \mid S_t = s]$

- But sometimes $V_t(S_{t+1})$ may not be that accurate
- Why?
  - rewards might be very sparse. Demo!
  - our initial estimates are poor: bad $V_1(S_{t+1})$

Better bootstrapping?

- $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V_t(S_{t+1}) \mid S_t = s]$

- But sometimes $V_t(S_{t+1})$ may not be that accurate
- Why?
  - rewards might be very sparse. Demo!
  - our initial estimates are poor: bad $V_1(S_{t+1})$
  - domain is not close to Markov

Better bootstrapping?

- $V_{t+1}(s) \leftarrow \mathbb{E}_\pi[R_{t+1} + \gamma V_t(S_{t+1}) \mid S_t = s]$

- But sometimes $V_t(S_{t+1})$ may not be that accurate
- Why?
  - rewards might be very sparse. Demo!
  - our initial estimates are poor: bad $V_1(S_{t+1})$
  - domain is not close to Markov
- In all these cases we might expect Monte Carlo methods to do better than TD(0)

Better bootstrapping?

Better bootstrapping?

- Turns out we just have to make the TD target, a little bit more like the Monte Carlo target

Better bootstrapping?

- Turns out we just have to make the TD target, a little bit more like the Monte Carlo target
  - $R_{t+1} + \gamma V_t(S_{t+1}) \Rightarrow R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+1} + \ldots$

Better bootstrapping?

- Turns out we just have to make the TD target, a little bit more like the Monte Carlo target
  - $R_{t+1} + \gamma V_t(S_{t+1}) \Rightarrow R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+1} + \ldots$
- This way we gain some of the robustness of MC methods

Better bootstrapping?

- Turns out we just have to make the TD target, a little bit more like the Monte Carlo target
  - $R_{t+1} + \gamma V_t(S_{t+1}) \Rightarrow R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+1} + \ldots$
- This way we gain some of the robustness of MC methods
- Without losing many of the benefits of TD

Better bootstrapping?

- Turns out we just have to make the TD target, a little bit more like the Monte Carlo target
  - $R_{t+1} + \gamma V_t(S_{t+1}) \Rightarrow R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+1} + \ldots$
- This way we gain some of the robustness of MC methods
- Without losing many of the benefits of TD
  - (1) not requiring terminations, (2) updating before the end of episodes

Better bootstrapping?

- Turns out we just have to make the TD target, a little bit more like the Monte Carlo target
  - $R_{t+1} + \gamma V_t(S_{t+1}) \Rightarrow R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+1} + \ldots$
- This way we gain some of the robustness of MC methods
- Without losing many of the benefits of TD
  - (1) not requiring terminations, (2) updating before the end of episodes
- And we unify TD(0) and Monte Carlo methods!

Better bootstrapping?

- Turns out we just have to make the TD target, a little bit more like the Monte Carlo target
  - $R_{t+1} + \gamma V_t(S_{t+1}) \Rightarrow R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+1} + \ldots$
- This way we gain some of the robustness of MC methods
- Without losing many of the benefits of TD
  - (1) not requiring terminations, (2) updating before the end of episodes
- And we unify TD(0) and Monte Carlo methods!
- This is todays topic!

Tyranny of the time step

Tyranny of the time step

- The world typically determines the one-time step dynamics of the tasks we wish to solve:

Tyranny of the time step

- The world typically determines the one-time step dynamics of the tasks we wish to solve:
  - how often we get new sensor readings

Tyranny of the time step

- The world typically determines the one-time step dynamics of the tasks we wish to solve:
  - how often we get new sensor readings
  - how often we select a new action

Tyranny of the time step

- The world typically determines the one-time step dynamics of the tasks we wish to solve:
  - how often we get new sensor readings
  - how often we select a new action
  - the time step

Tyranny of the time step

- The world typically determines the one-time step dynamics of the tasks we wish to solve:
  - how often we get new sensor readings
  - how often we select a new action
  - the time step
- In some domains we want a very small time step to allow quick reactions

Tyranny of the time step

- The world typically determines the one-time step dynamics of the tasks we wish to solve:
  - how often we get new sensor readings
  - how often we select a new action
  - the time step
- In some domains we want a very small time step to allow quick reactions
- But bootstrapping works best over a length of time where significant and recognizable change in state has occurred

Tyranny of the time step

- The world typically determines the one-time step dynamics of the tasks we wish to solve:
  - how often we get new sensor readings
  - how often we select a new action
  - the time step
- In some domains we want a very small time step to allow quick reactions
- But bootstrapping works best over a length of time where significant and recognizable change in state has occurred
  - if S_t looks very similar to S_{t+1}, bootstrapping won't be as effective

Tyranny of the time step

- The world typically determines the one-time step dynamics of the tasks we wish to solve:
  - how often we get new sensor readings
  - how often we select a new action
  - the time step
- In some domains we want a very small time step to allow quick reactions
- But bootstrapping works best over a length of time where significant and recognizable change in state has occurred
  - if S_t looks very similar to S_{t+1}, bootstrapping won't be as effective
- Multi-step methods enable bootstrapping over many small time steps

Consider a simple derivation for TD(0)

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$
  - $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$
  - $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$
  - $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - ...

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$
  - $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - …
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$
  - $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - …
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$
- Compared with

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$
  - $= \mathbb{E}_\pi[\sum_{k=0}\gamma^k R_{t+k+1} \mid S_t = s]$
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma\sum_{k=0}\gamma^k R_{t+k+1} \mid S_t = s]$
  - …
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$
- Compared with
  - $\mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + ... \mid S_t = s]$

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$
  - $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - …
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$
- Compared with
  - $\mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + ... \mid S_t = s]$
- $\gamma V_t(S_{t+1})$ is a stand in for the rest of the return: $\gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1}$

Consider a simple derivation for TD(0)

- $v_\pi(s) \doteq \mathbb{E}_\pi[G_t \mid S_t = s]$
  - $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
  - …
  - $= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s]$
- Compared with
  - $\mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + ... \mid S_t = s]$
- $\gamma V_t(S_{t+1})$ is a stand in for the rest of the return: $\gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1}$
- TD(0) uses a **1-step return:** $G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$

More than 1-step returns

More than 1-step returns

- $= \mathbb{E}_{\pi}[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$

More than 1-step returns

- $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$

More than 1-step returns

- $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- …

More than 1-step returns

- $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- …

  - We could have pulled more rewards out of the sum, say two of them: R_{t+1} and R_{t+2}

More than 1-step returns

- $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- …

- We could have pulled more rewards out of the sum, say two of them: R_{t+1} and R_{t+2}
- In that case $V_{t+1}(S_{t+2})$ would be a stand in for the rest of the return:

More than 1-step returns

- $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- …

  - We could have pulled more rewards out of the sum, say two of them: R_{t+1} and R_{t+2}
  - In that case $V_{t+1}(S_{t+2})$ would be a stand in for the rest of the return:
    - $R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + \ldots$

More than 1-step returns

- $= \mathbb{E}_\pi[\sum_{k=0}\gamma^k R_{t+k+1} \mid S_t = s]$
- $= \mathbb{E}_\pi[R_{t+1} + \gamma\sum_{k=0}\gamma^k R_{t+k+1} \mid S_t = s]$
- …

- We could have pulled more rewards out of the sum, say two of them: R_{t+1} and R_{t+2}
- In that case $V_{t+1}(S_{t+2})$ would be a stand in for the rest of the return:
  - $R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + ...$
  - $R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$

More than 1-step returns

- $= \mathbb{E}_\pi[\sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- $= \mathbb{E}_\pi[R_{t+1} + \gamma \sum_{k=0} \gamma^k R_{t+k+1} \mid S_t = s]$
- …

- We could have pulled more rewards out of the sum, say two of them: R_{t+1} and R_{t+2}
- In that case $V_{t+1}(S_{t+2})$ would be a stand in for the rest of the return:
  - $R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + \ldots.$
  - $R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$
- A 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 returns

n-step returns

- We could do 3-step, 4-step,….. n-step returns

n-step returns

- We could do 3-step, 4-step,….. n-step returns
  - $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$

n-step returns

- We could do 3-step, 4-step,….. n-step returns
  - $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- These still uses **bootstrapping**

n-step returns

- We could do 3-step, 4-step,….. n-step returns
  - $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- These still uses **bootstrapping**
  - they still uses $V_{t+n-1}(S_{t+n})$ inside the target

n-step returns

- We could do 3-step, 4-step,….. n-step returns
  - $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- These still uses **bootstrapping**
  - they still uses $V_{t+n-1}(S_{t+n})$ inside the target
- n-step returns approximate the **full return**, **truncated** after n-steps

n-step returns

- We could do 3-step, 4-step,….. n-step returns
  - $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- These still uses **bootstrapping**
  - they still uses $V_{t+n-1}(S_{t+n})$ inside the target
- n-step returns approximate the **full return**, **truncated** after n-steps
  - $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + \ldots + \gamma^{T-t-1} R_T$

n-step returns

- We could do 3-step, 4-step,….. n-step returns
  - $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- These still uses **bootstrapping**
  - they still uses $V_{t+n-1}(S_{t+n})$ inside the target
- n-step returns approximate the **full return**, **truncated** after n-steps
  - $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + \ldots + \gamma^{T-t-1} R_T$
- if $t+n \geq T$ (expands beyond termination), then we just use the full return: $G_{t:t+n} \doteq G_t$ if $t+n \geq T$

n-step returns

- We could do 3-step, 4-step,….. n-step returns
  - $G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$
- These still uses **bootstrapping**
  - they still uses $V_{t+n-1}(S_{t+n})$ inside the target
- n-step returns approximate the **full return**, **truncated** after n-steps
  - $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+1} + \ldots + \gamma^{T-t-1} R_T$
- if $t+n \geq T$ (expands beyond termination), then we just use the full return: $G_{t:t+n} \doteq G_t$ if $t+n \geq T$
- Monte Carlo methods are a special case being an ∞ step return

Mathematics of *n*-step TD Returns/Targets

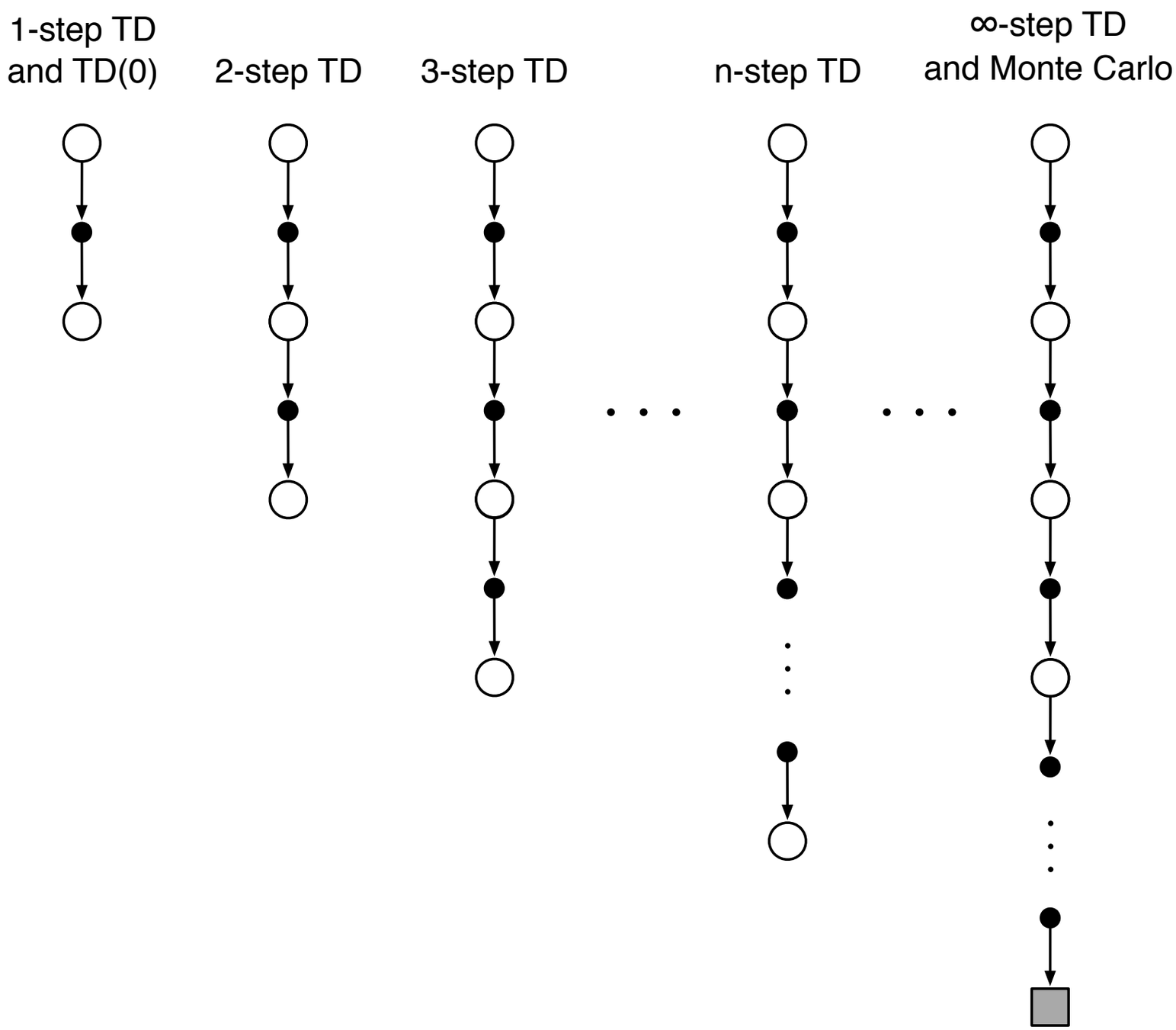
- Monte Carlo: $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-t-1} R_T$

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

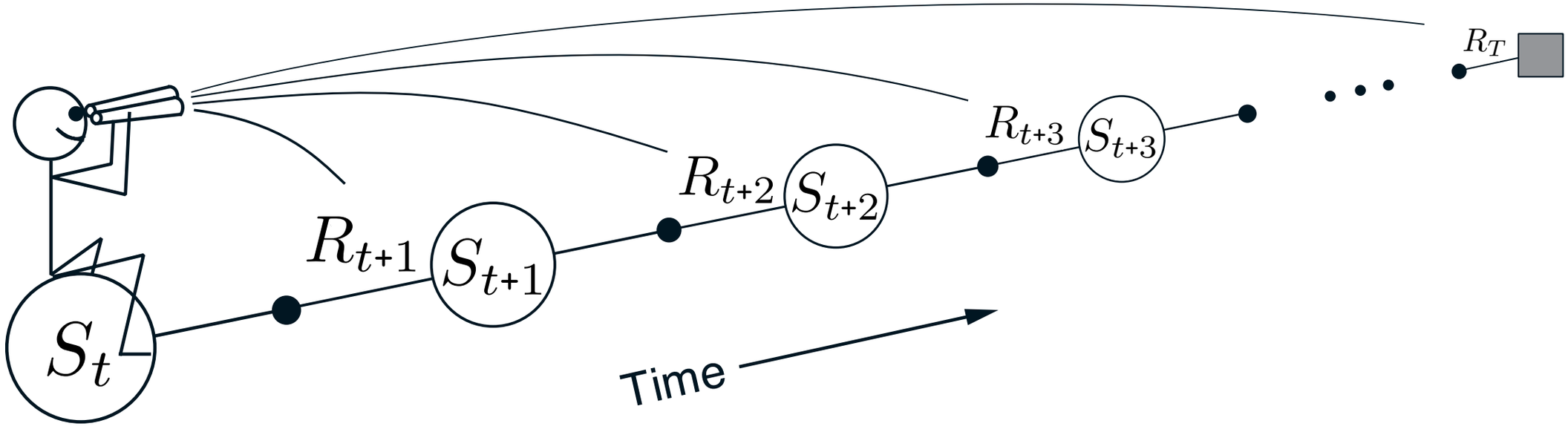
    with $G_{t:t+n} \doteq G_t$ if $t+n \geq T$)

Spectrum of returns



Forward View of TD(λ)

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



n-step TD prediction

n-step TD prediction

- TD(0) updates toward the 1-step return:

n-step TD prediction

- TD(0) updates toward the 1-step return:
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [R_{t+1} + \gamma V_t(S_{t+1}) - V_t(s)]$

n-step TD prediction

- TD(0) updates toward the 1-step return:
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [R_{t+1} + \gamma V_t(S_{t+1}) - V_t(s)]$
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [G_{t:t+1} - V_t(s)]$

n-step TD prediction

- TD(0) updates toward the 1-step return:
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [R_{t+1} + \gamma V_t(S_{t+1}) - V_t(s)]$
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [G_{t:t+1} - V_t(s)]$
  - $G_{t:t+1} = R_{t+1} + \gamma V_t(S_{t+1})$ is the **update target**

n-step TD prediction

- TD(0) updates toward the 1-step return:
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [R_{t+1} + \gamma V_t(S_{t+1}) - V_t(s)]$
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [G_{t:t+1} - V_t(s)]$
    - $G_{t:t+1} = R_{t+1} + \gamma V_t(S_{t+1})$ is the **update target**
- To do n-step TD prediction, we simply update towards the n-step return:

n-step TD prediction

- TD(0) updates toward the 1-step return:
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [R_{t+1} + \gamma V_t(S_{t+1}) - V_t(s)]$
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [G_{t:t+1} - V_t(s)]$
    - $G_{t:t+1} = R_{t+1} + \gamma V_t(S_{t+1})$ is the **update target**
- To do n-step TD prediction, we simply update towards the n-step return:
  - $V_{t+n}(s) \leftarrow V_{t+n-1}(s) + \boldsymbol{\alpha}[G_{t:t+n} - V_{t+n-1}(s)]$

n-step TD prediction

- TD(0) updates toward the 1-step return:
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [R_{t+1} + \gamma V_t(S_{t+1}) - V_t(s)]$
  - $V_{t+1}(s) \leftarrow V_t(s) + \boldsymbol{\alpha} [G_{t:t+1} - V_t(s)]$
    - $G_{t:t+1} = R_{t+1} + \gamma V_t(S_{t+1})$ is the **update target**
- To do n-step TD prediction, we simply update towards the n-step return:
  - $V_{t+n}(s) \leftarrow V_{t+n-1}(s) + \boldsymbol{\alpha}[G_{t:t+n} - V_{t+n-1}(s)]$
  - $V_{t+n}(s) \leftarrow V_{t+n-1}(s) + \boldsymbol{\alpha}[R_{t+1} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n}) - V_{t+n-1}(s)]$

But, there is a catch

But, there is a catch

- The n-step return uses rewards in the future

But, there is a catch

- The n-step return uses rewards in the future
- In TD(0) we wait one step, see the reward and next state and make an update of the **previous** states' value

But, there is a catch

- The n-step return uses rewards in the future
- In TD(0) we wait one step, see the reward and next state and make an update of the **previous** states' value
  - we wait one step to update toward one-step returns

But, there is a catch

- The n-step return uses rewards in the future
- In TD(0) we wait one step, see the reward and next state and make an update of the **previous** states' value
  - we wait one step to update toward one-step returns
- In n-step TD methods we have to wait n-steps, before we can update

But, there is a catch

- The n-step return uses rewards in the future
- In TD(0) we wait one step, see the reward and next state and make an update of the **previous** states' value
  - we wait one step to update toward one-step returns
- In n-step TD methods we have to wait n-steps, before we can update
  - no updates to V during the first n-1 steps of the episode

But, there is a catch

- The n-step return uses rewards in the future
- In TD(0) we wait one step, see the reward and next state and make an update of the **previous** states' value
  - we wait one step to update toward one-step returns
- In n-step TD methods we have to wait n-steps, before we can update
  - no updates to V during the first n-1 steps of the episode
  - we have to perform n-1 updates after the end of the episode, before we begin the next episode

n-step TD for policy evaluation

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 $\tau \geq 0$:
    | | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
    | | 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 $\tau = T - 1$

Lets consider 3-step TD

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**
  - **S_{t+1} is terminal; SET: T = t+1 = 4**

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**
  - **S_{t+1} is terminal; SET: T = t+1 = 4**
- t = 4, t not less than T

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**
  - **S_{t+1} is terminal; SET: T = t+1 = 4**
- t = 4, t not less than T
  - τ = t-n+1 = 2; do learning update V(S_2)

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**
  - **S_{t+1} is terminal; SET: T = t+1 = 4**
- t = 4, t not less than T
  - τ = t-n+1 = 2; do learning update V(S_2)
- t = 5

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**
  - **S_{t+1} is terminal; SET: T = t+1 = 4**
- t = 4, t not less than T
  - τ = t-n+1 = 2; do learning update V(S_2)
- t = 5
  - τ = t-n+1 = 3; do learning update V(S_3)

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**
  - **S_{t+1} is terminal; SET: T = t+1 = 4**
- t = 4, t not less than T
  - τ = t-n+1 = 2; do learning update V(S_2)
- t = 5
  - τ = t-n+1 = 3; do learning update V(S_3)
- τ = T -1? is 3 == 3 **yes**

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**
  - **S_{t+1} is terminal; SET: T = t+1 = 4**
- t = 4, t not less than T
  - τ = t-n+1 = 2; do learning update V(S_2)
- t = 5
  - τ = t-n+1 = 3; do learning update V(S_3)
- τ = T -1? is 3 == 3 **yes**
  - **Start next episode**

Lets consider 3-step TD

- Imagine a sample short episode: S_0, S_1, S_2, S_3, S_T
- T = ∞
- t = 0 (S_t=S_0, S_{t+1}=S_1)
  - τ = t-n+1 = -2
  - **skip**, $\tau < 0$
- t=1 (S_t=S_1, S_{t+1}=S_2)
  - τ = t-n+1 = -1
  - **skip**, $\tau < 0$
- t = 2 (S_t=S_2, S_{t+1}=S_3)
  - τ = t-n+1 = 0; $\tau \geq 0$
  - do learning update V(S_τ), V(S_0)
    - **we waited n-1 = 2 steps to start updating**

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 $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
      $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

- t = 3 (S_t=S_3, S_{t+1}=S_T)
  - τ = t-n+1 = 1; do learning update V(S_1)
  - **last step of episode**
  - **S_{t+1} is terminal; SET: T = t+1 = 4**
- t = 4, t not less than T
  - τ = t-n+1 = 2; do learning update V(S_2)
- t = 5
  - τ = t-n+1 = 3; do learning update V(S_3)
- τ = T -1? is 3 == 3 **yes**
  - **Start next episode**
- **we updated n-1 = 2 steps past the end of the episode**

Computing the n-step return

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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $V(S_\tau) \leftarrow V(S_\tau) + \alpha\,[G - V(S_\tau)]$
  Until $\tau = T - 1$

Computing the n-step return

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 $\tau \geq 0$:
  | | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

- Inside the $\tau \geq 0$ conditional, we compute the n-step return:

Computing the n-step return

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 $\tau \geq 0$:
  | | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

- Inside the $\tau \geq 0$ conditional, we compute the n-step return:
  1. sum up the rewards (n), weighted by powers of γ

Computing the n-step return

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 $\tau \geq 0$:
  | | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$ $(G_{\tau:\tau+n})$
  | | $V(S_\tau) \leftarrow V(S_\tau) + \alpha [G - V(S_\tau)]$
  Until $\tau = T - 1$

- Inside the $\tau \geq 0$ conditional, we compute the n-step return:
  1. sum up the rewards (n), weighted by powers of γ
  2. AND add in appropriate state value if not beyond end of episode (τ+n<T)

Computing the n-step return

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 $n + 1$

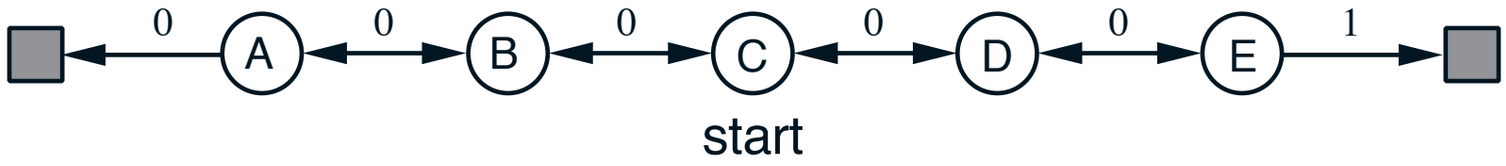
Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | | Take an action according to $\pi(\cdot|S_t)$
  | | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | | If S_{t+1} is terminal,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

- Inside the $\tau \geq 0$ conditional, we compute the n-step return:
  1. sum up the rewards (n), weighted by powers of γ
  2. AND add in appropriate state value if not beyond end of episode (τ+n<T)
- Try on your own to convince yourself the indexing is correct

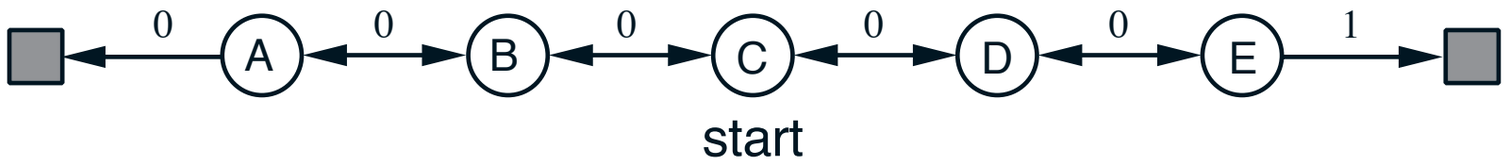
Convergence of n-step TD methods

- n-step TD prediction converges to the correct predictions
- We have a family of methods:
  - 1-step TD methods, TD(0) at one end
  - Monte Carlo methods at the other extreme
- Let’s do an experiment

The Random Walk

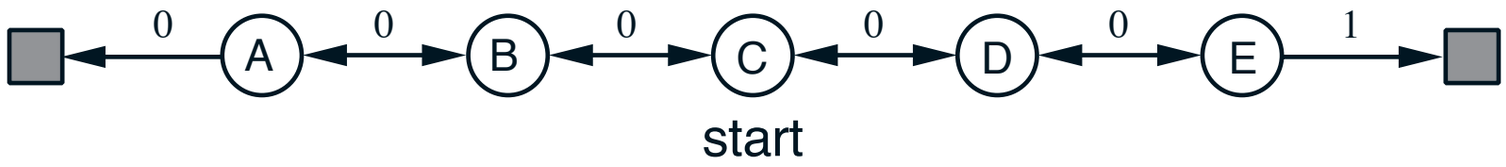


The Random Walk



- Imagine: C->D->E->Termination, $G_t = 1$

The Random Walk



- Imagine: C->D->E->Termination, $G_t = 1$
- Initialization: V(s) = 0.5

The Random Walk

- Imagine: C->D->E->Termination, $G_t = 1$
- Initialization: V(s) = 0.5
- TD(0) would only change V(E), toward 1.0

The Random Walk

- Imagine: C->D->E->Termination, $G_t = 1$
- Initialization: V(s) = 0.5
- TD(0) would only change V(E), toward 1.0
- How would TD(0) change V(D)?

The Random Walk

- Imagine: C->D->E->Termination, $G_t = 1$
- Initialization: V(s) = 0.5
- TD(0) would only change V(E), toward 1.0
- How would TD(0) change V(D)?
  - consider the update rule

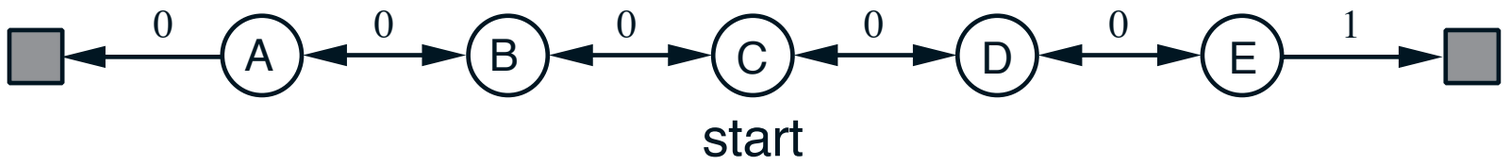
The Random Walk

- Imagine: C->D->E->Termination, $G_t = 1$
- Initialization: V(s) = 0.5
- TD(0) would only change V(E), toward 1.0
- How would TD(0) change V(D)?
  - consider the update rule
  - $V(D) = V(D) + \boldsymbol{\alpha}[R_{t+1} + \gamma V(E) - V(D)]$

The Random Walk

- Imagine: C->D->E->Termination, $G_t = 1$
- Initialization: V(s) = 0.5
- TD(0) would only change V(E), toward 1.0
- How would TD(0) change V(D)?
  - consider the update rule
  - $V(D) = V(D) + \alpha[R_{t+1} + \gamma V(E) - V(D)]$
  - $V(D) = 0.5 + \alpha[0 + (1)0.5 - 0.5]$; V(D) unchanged

The Random Walk with n-step updates



The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1
  - TD update:

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1
  - TD update:
  - V(D) = V(D) + $\boldsymbol{\alpha}$[1 - V(D)] = change of $\boldsymbol{\alpha}$(0.5)

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1
  - TD update:
  - V(D) = V(D) + α[1 - V(D)] = change of α(0.5)
- What would be the 2-step return from C?

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1
  - TD update:
  - V(D) = V(D) + $\boldsymbol{\alpha}$[1 - V(D)] = change of $\boldsymbol{\alpha}$(0.5)
- What would be the 2-step return from C?
  - $R_1 + \gamma R_2 + \gamma^2 V(E)$

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1
  - TD update:
  - V(D) = V(D) + $\boldsymbol{\alpha}$[1 - V(D)] = change of $\boldsymbol{\alpha}$(0.5)
- What would be the 2-step return from C?
  - $R_1 + \gamma R_2 + \gamma^2 V(E)$
  - 0 + 0 + (1)0.5

The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1
  - TD update:
  - V(D) = V(D) + $\boldsymbol{\alpha}$[1 - V(D)] = change of $\boldsymbol{\alpha}$(0.5)
- What would be the 2-step return from C?
  - $R_1 + \gamma R_2 + \gamma^2 V(E)$
  - 0 + 0 + (1)0.5
  - TD update:

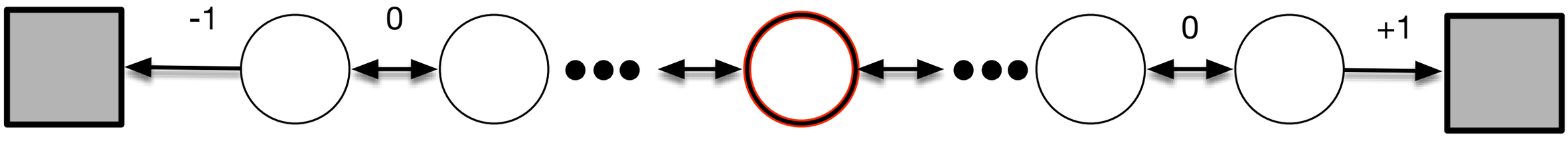
The Random Walk with n-step updates

- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1
  - TD update:
  - V(D) = V(D) + $\boldsymbol{\alpha}$[1 - V(D)] = change of $\boldsymbol{\alpha}$(0.5)
- What would be the 2-step return from C?
  - $R_1 + \gamma R_2 + \gamma^2 V(E)$
  - 0 + 0 + (1)0.5
  - TD update:
  - V(C) = V(C) + $\boldsymbol{\alpha}$[0 - V(C)] = V(C); no change

The Random Walk with n-step updates

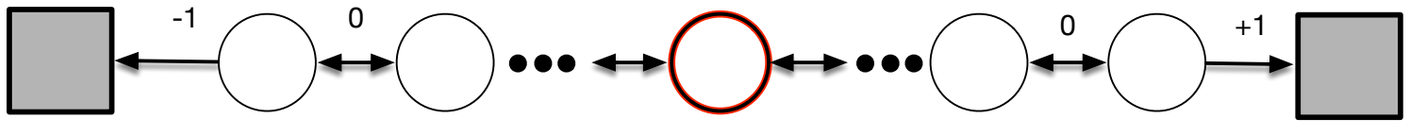
- Imagine: C->D->E->Termination, $G_t = 1$ and V(s) = 0.5, non-terminal
- What would be the 2-step return from D?
  - $R_2 + \gamma R_3 + \gamma^2 V(T)$
  - 0 + 1
  - TD update:
  - V(D) = V(D) + $\boldsymbol{\alpha}$[1 - V(D)] = change of $\boldsymbol{\alpha}$(0.5)
- What would be the 2-step return from C?
  - $R_1 + \gamma R_2 + \gamma^2 V(E)$
  - 0 + 0 + (1)0.5
  - TD update:
  - V(C) = V(C) + $\boldsymbol{\alpha}$[0 - V(C)] = V(C); no change
- Any n-step TD method with n>2 would update all three visited states!

The Random Walk with n-step updates

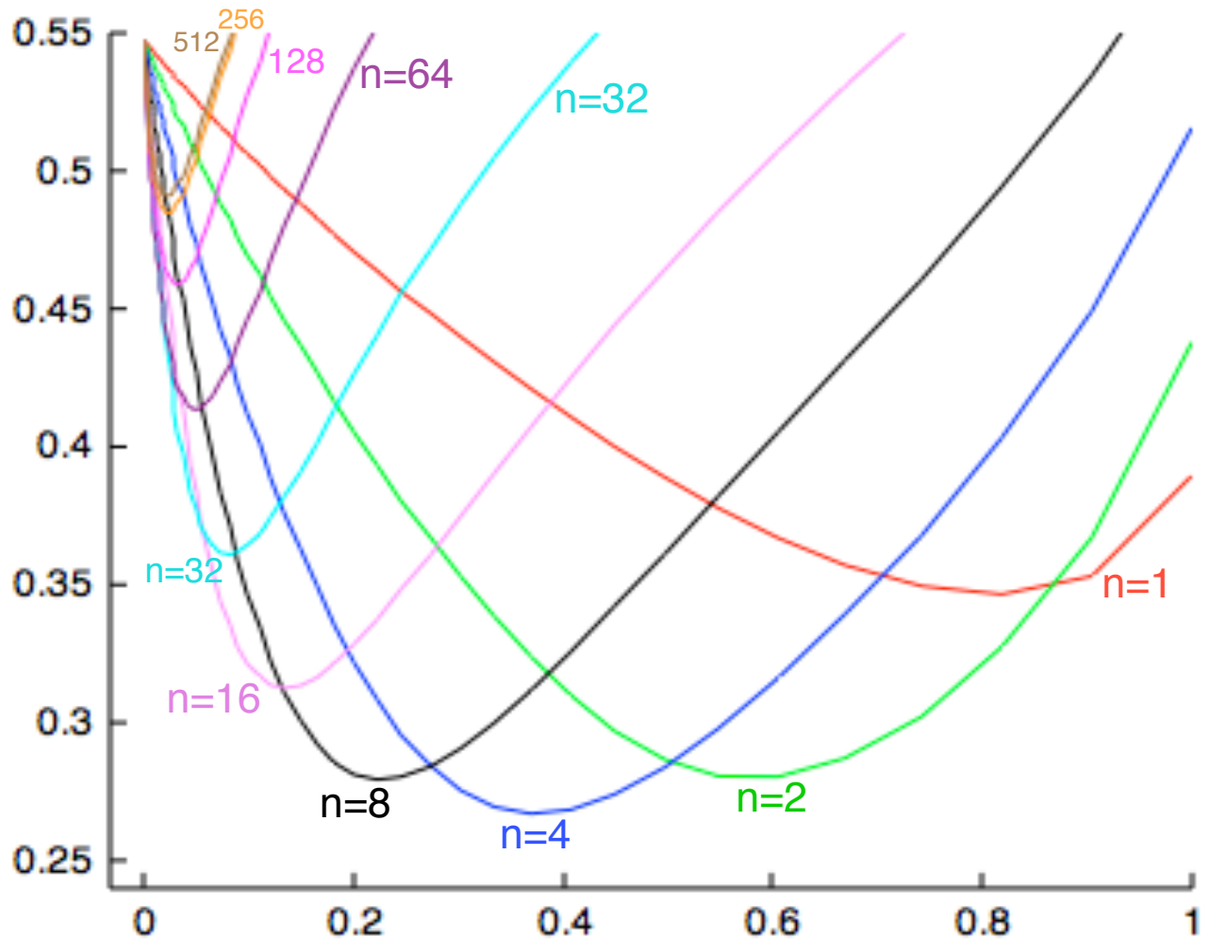


- V initialized to vec{0}, π = uniform random

The Random Walk with n-step updates



Average RMS error over 19 states and first 10 episodes



- Red line is TD(0), larger n are near Monte Carlo
- Methods in between perform better

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

n-step Sarsa

- Previous chapter we saw:
  - one-step Sarsa or Sarsa(0)
- n-step Sarsa:
  - learn Q functions, instead of V
  - generalize the definition of n-step return to state-action values

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

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

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

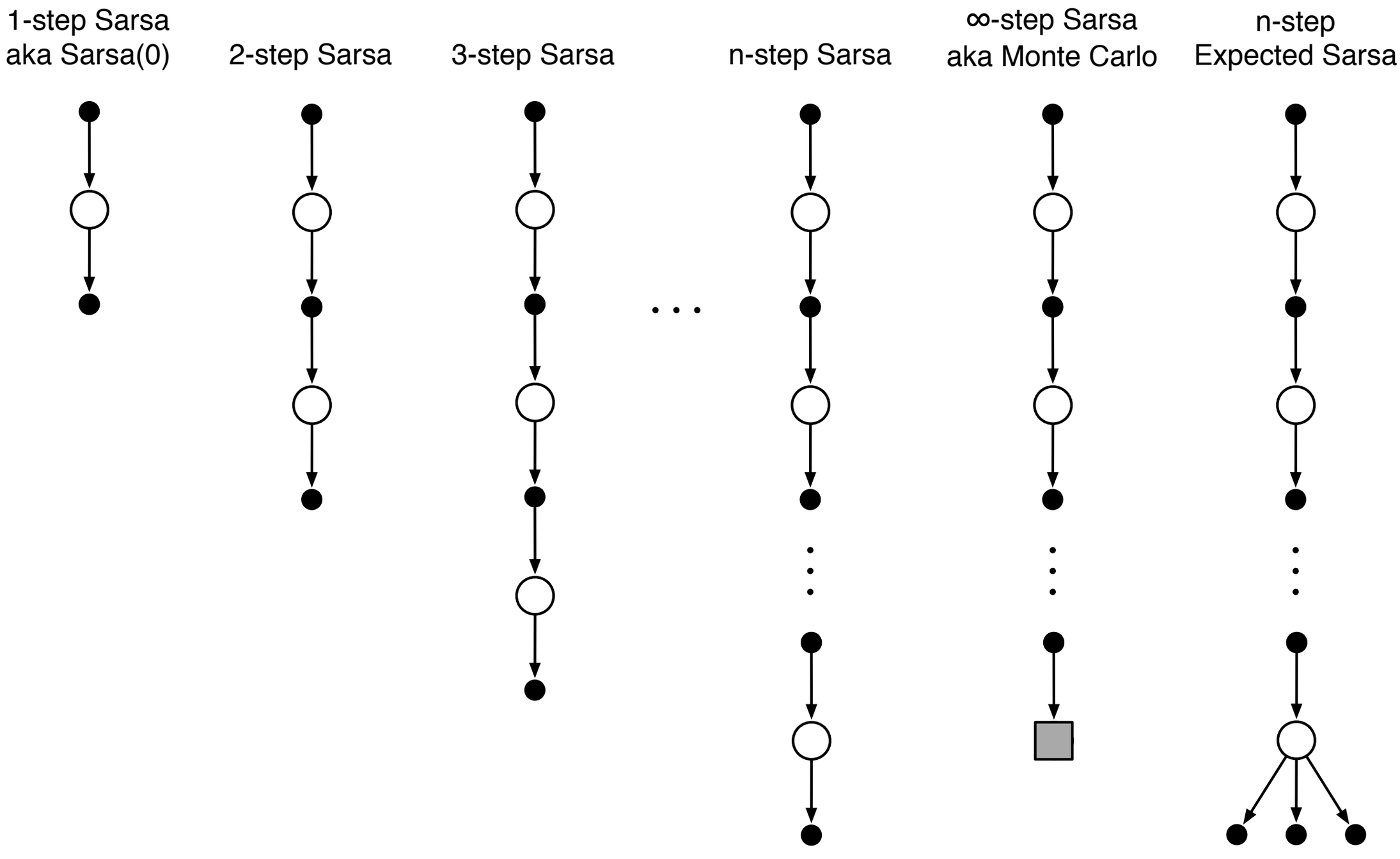
- *n*-step Sarsa:

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

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

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

It's much the same for action values



n-step Sarsa

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

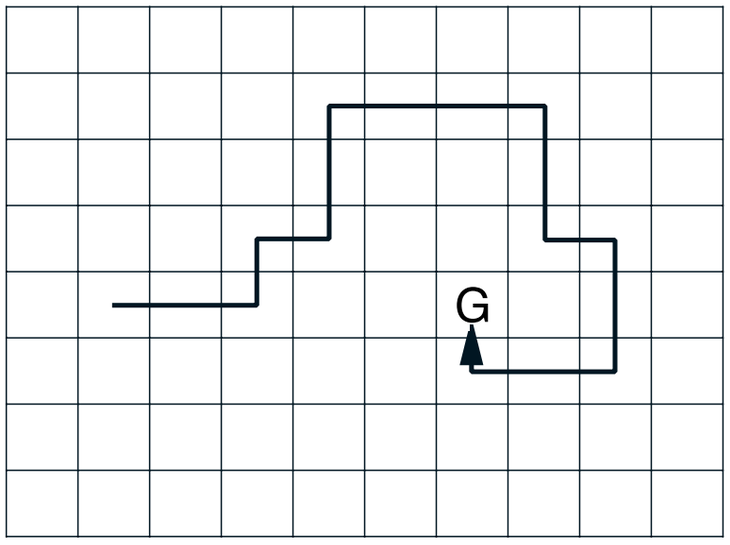
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize π to be ε-greedy with respect to Q, or to a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n
All store and access operations (for S_t, A_t, and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

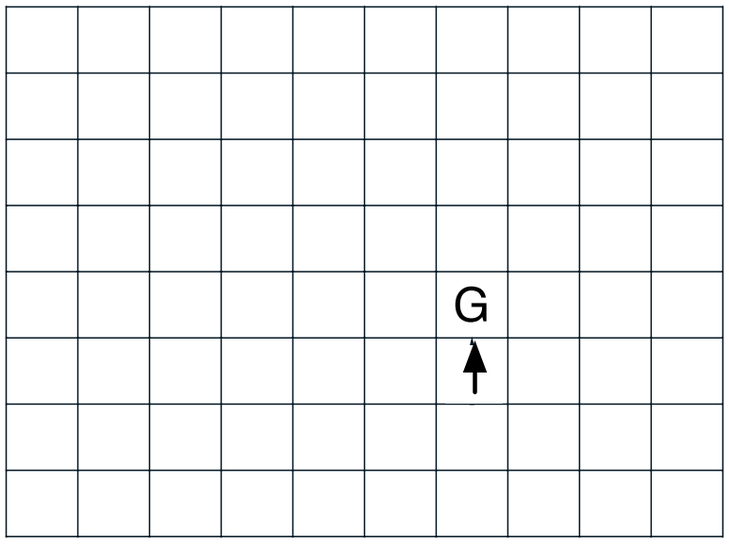
Efficiency of n-step updating

- A multi-step update algorithm like n-step Sarsa can propagate sparse reward information
  - given a single episode, information can be used to update the Q values for previously observed <s,a>
  - especially useful in sparse or **delayed reward** problems

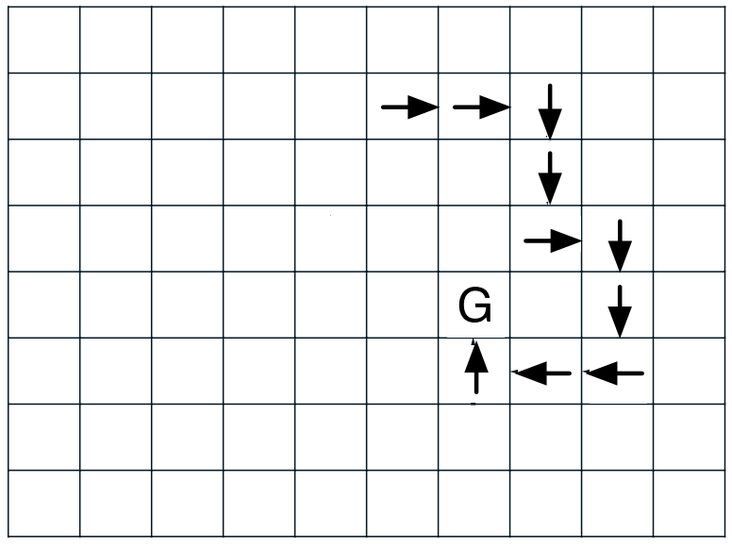
Path taken



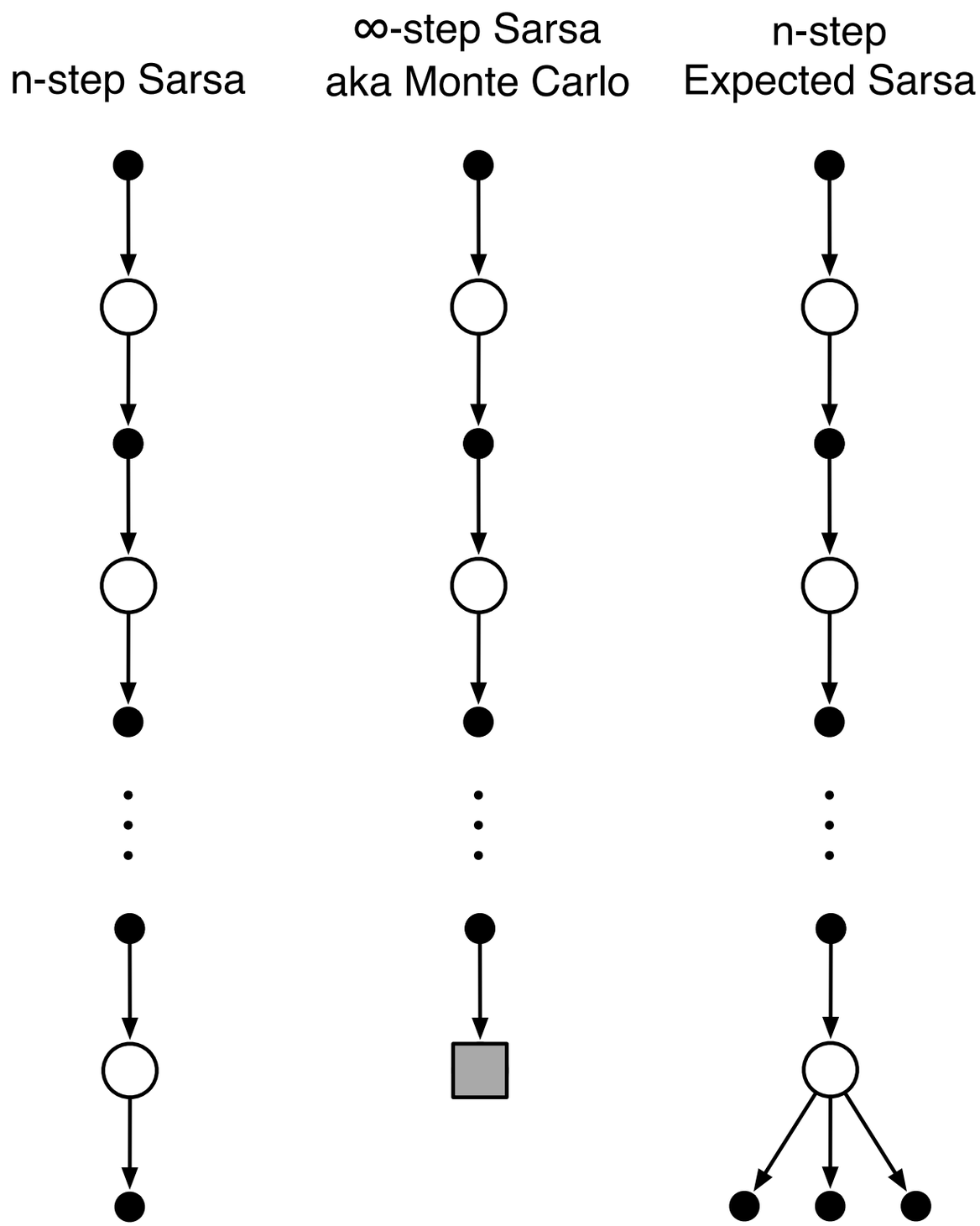
Action values increased by one-step Sarsa



Action values increased by 10-step Sarsa



n-step expected Sarsa



n-step expected Sarsa

- Same equations as n-step Sarsa, with a slightly different n-step return

$$G_{t:t+n} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \bar{V}_{t+n-1}(S_{t+n})$$

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

Other n-step methods

1. n-Step off-policy control method for estimating the optimal policy, using importance sampling
2. n-step Off-policy control without importance sampling: **Tree Backup method**
   - from early 2000's
   - renewed interest under new name: Retrace algorithm
3. n-step Q(σ): unifying n-step Sarsa, n-step Expected Sarsa, and Tree Backup algorithms

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:

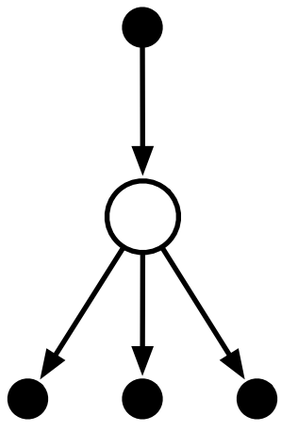
$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1:t+n} \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)\right]$$

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

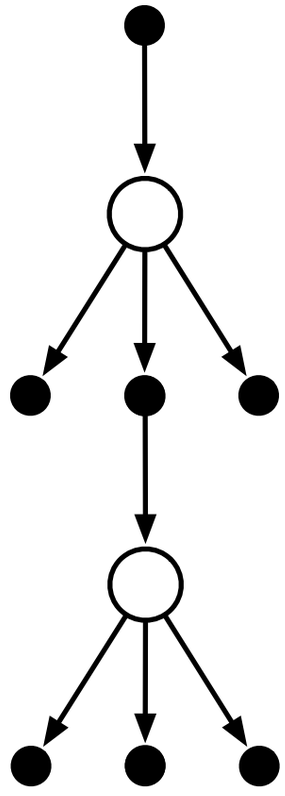
Just like above except expected *n*-step return and ρ goes to $t+n-2$

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

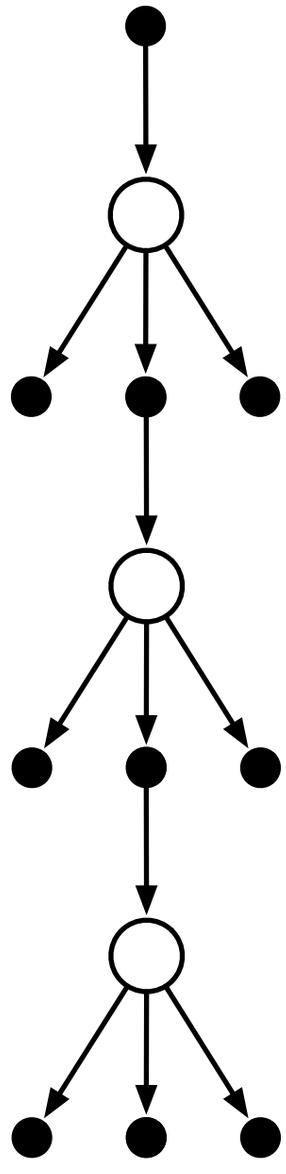
Expected Sarsa
and 1-step Tree Backup



2-step Tree Backup

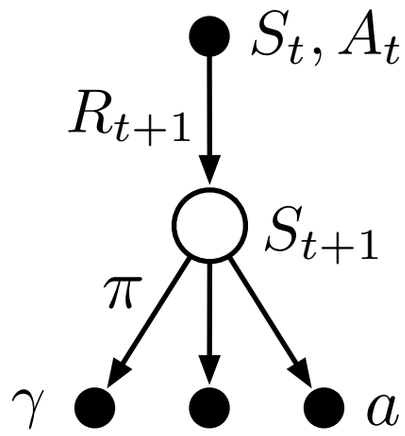


3-step TB

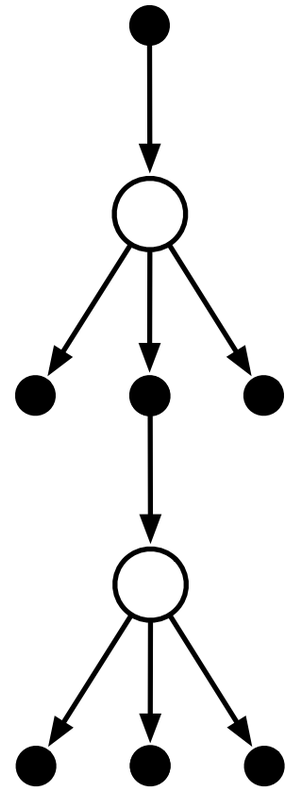


Off-policy Learning w/o Importance Sampling: The *n*-step Tree Backup Algorithm

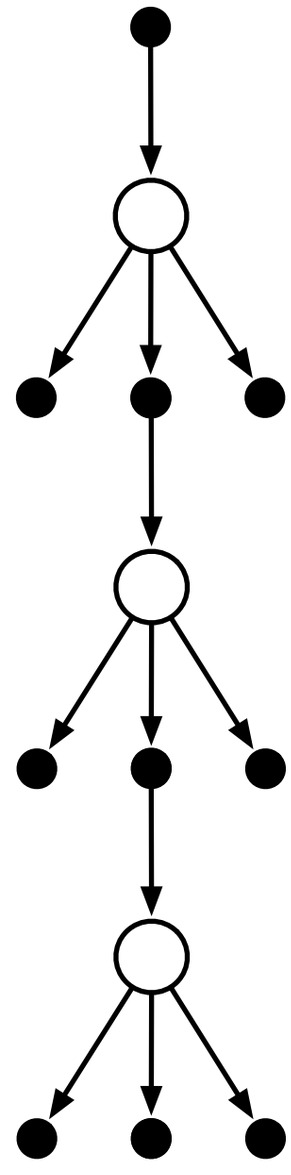
Expected Sarsa
and 1-step Tree Backup



2-step Tree Backup

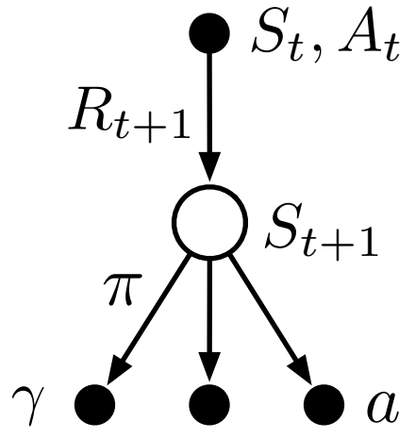


3-step TB



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

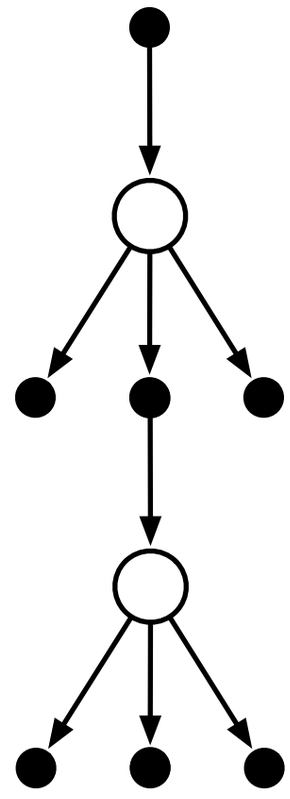
Expected Sarsa
and 1-step Tree Backup



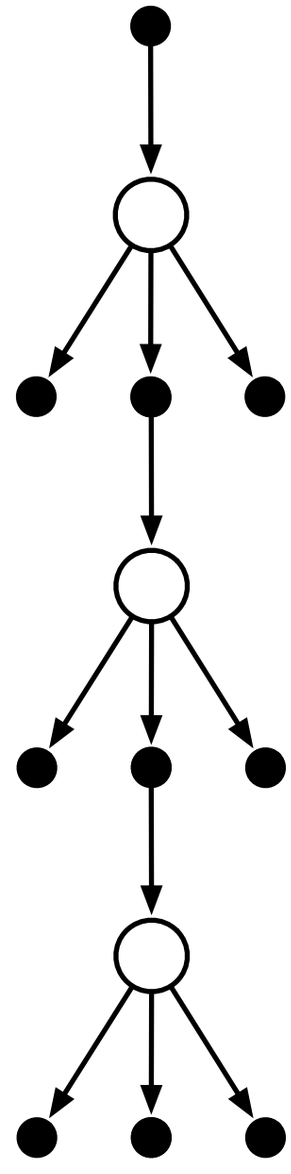
$$R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

Target

2-step Tree Backup

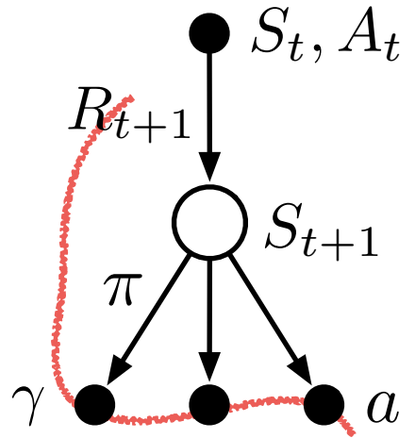


3-step TB



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

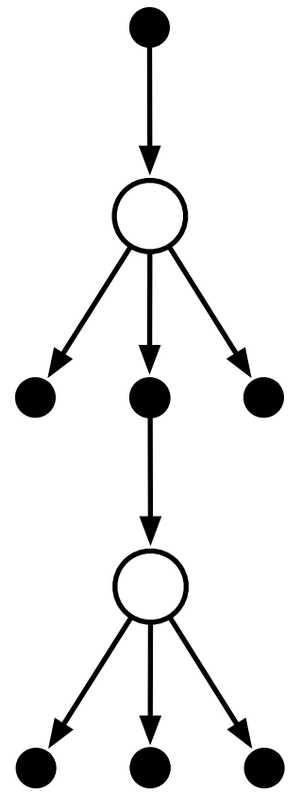
Expected Sarsa
and 1-step Tree Backup



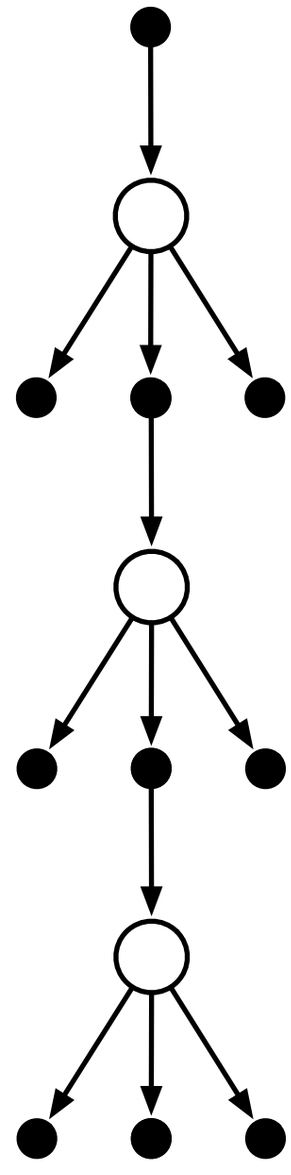
$$R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

Target

2-step Tree Backup

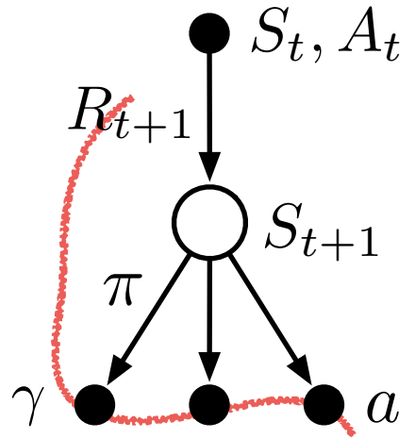


3-step TB



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

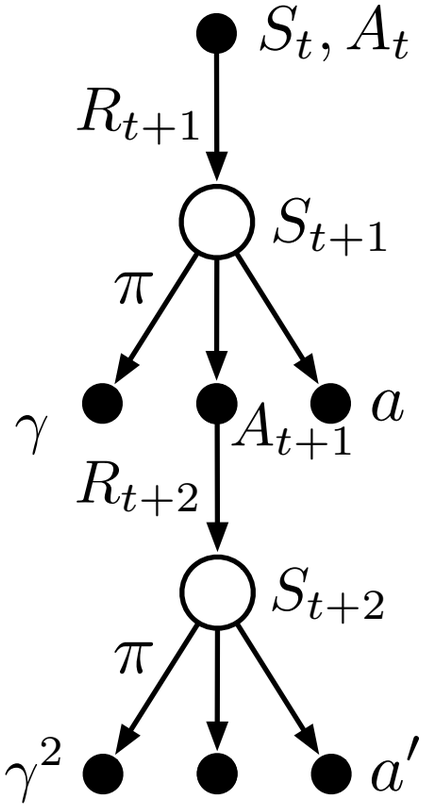
Expected Sarsa
and 1-step Tree Backup



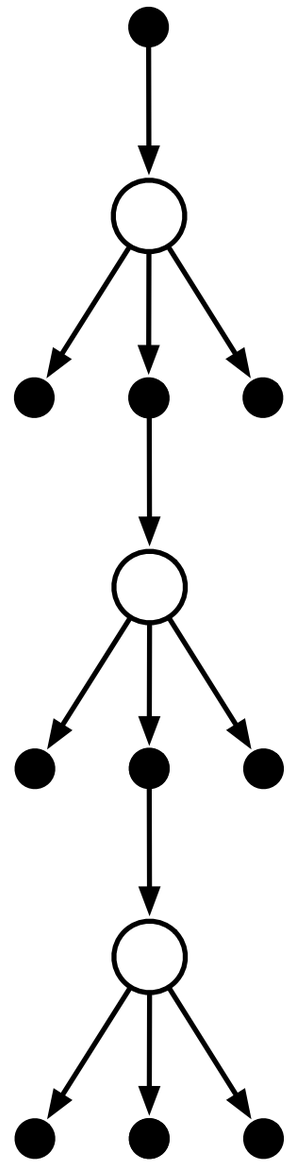
$$R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

Target

2-step Tree Backup

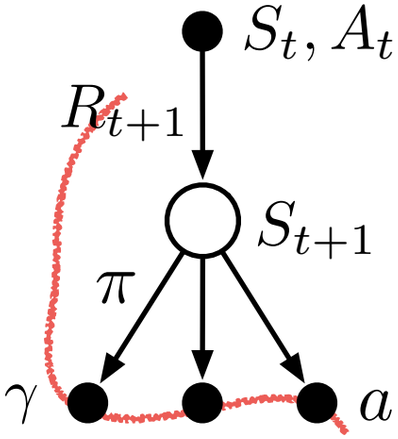


3-step TB



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

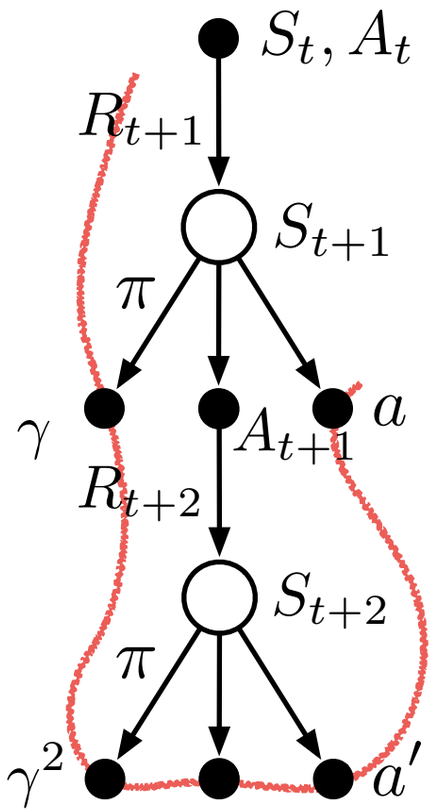
Expected Sarsa
and 1-step Tree Backup



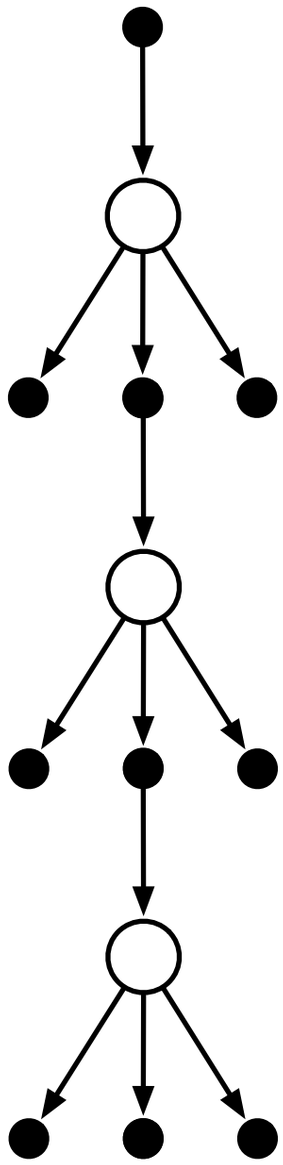
$$R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

Target

2-step Tree Backup

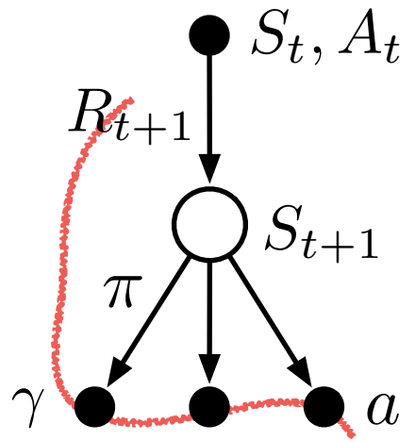


3-step TB



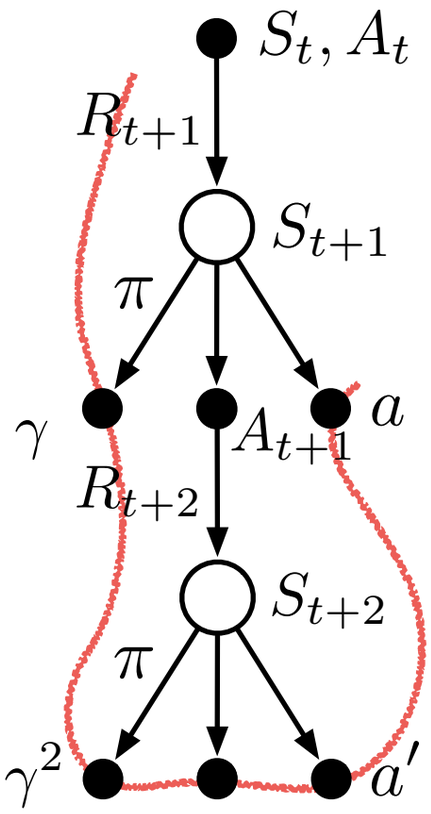
Off-policy Learning w/o Importance Sampling: The *n*-step Tree Backup Algorithm

Expected Sarsa
and 1-step Tree Backup

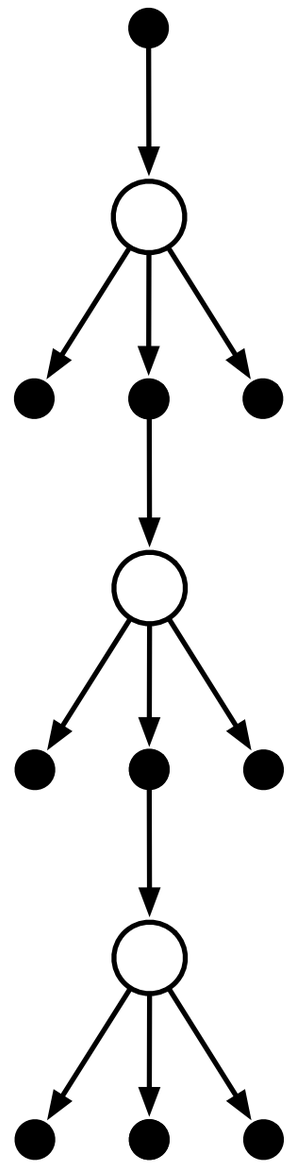


$$R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

2-step Tree Backup



3-step TB



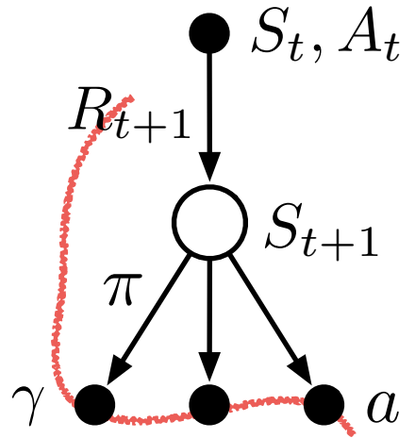
Target

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

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

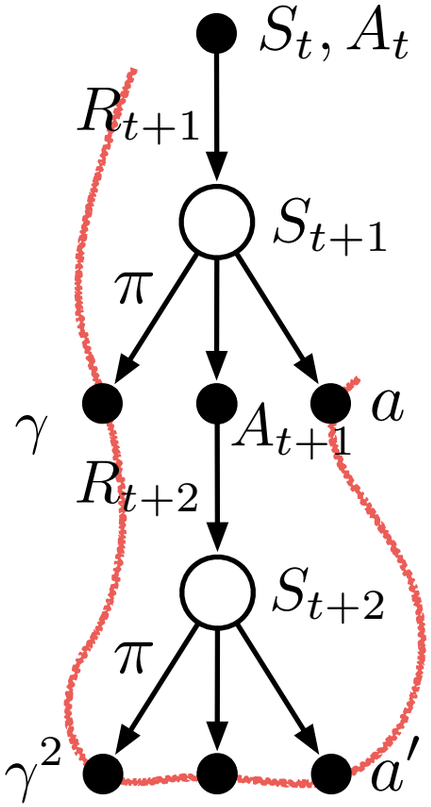
Off-policy Learning w/o Importance Sampling: The *n*-step Tree Backup Algorithm

Expected Sarsa
and 1-step Tree Backup

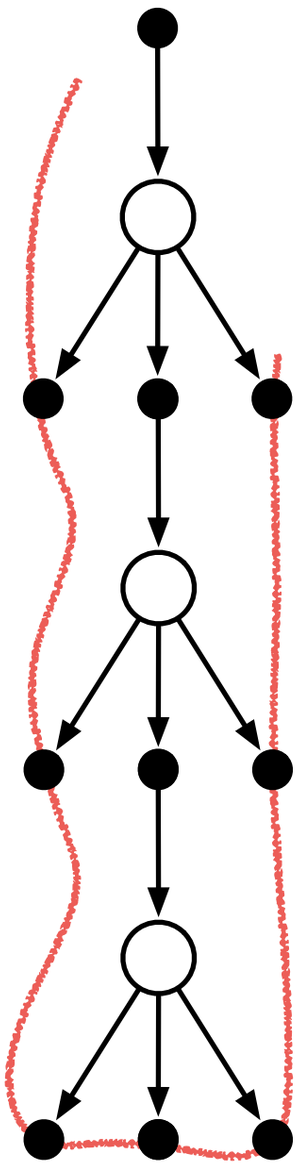


$$R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a)$$

2-step Tree Backup



3-step TB

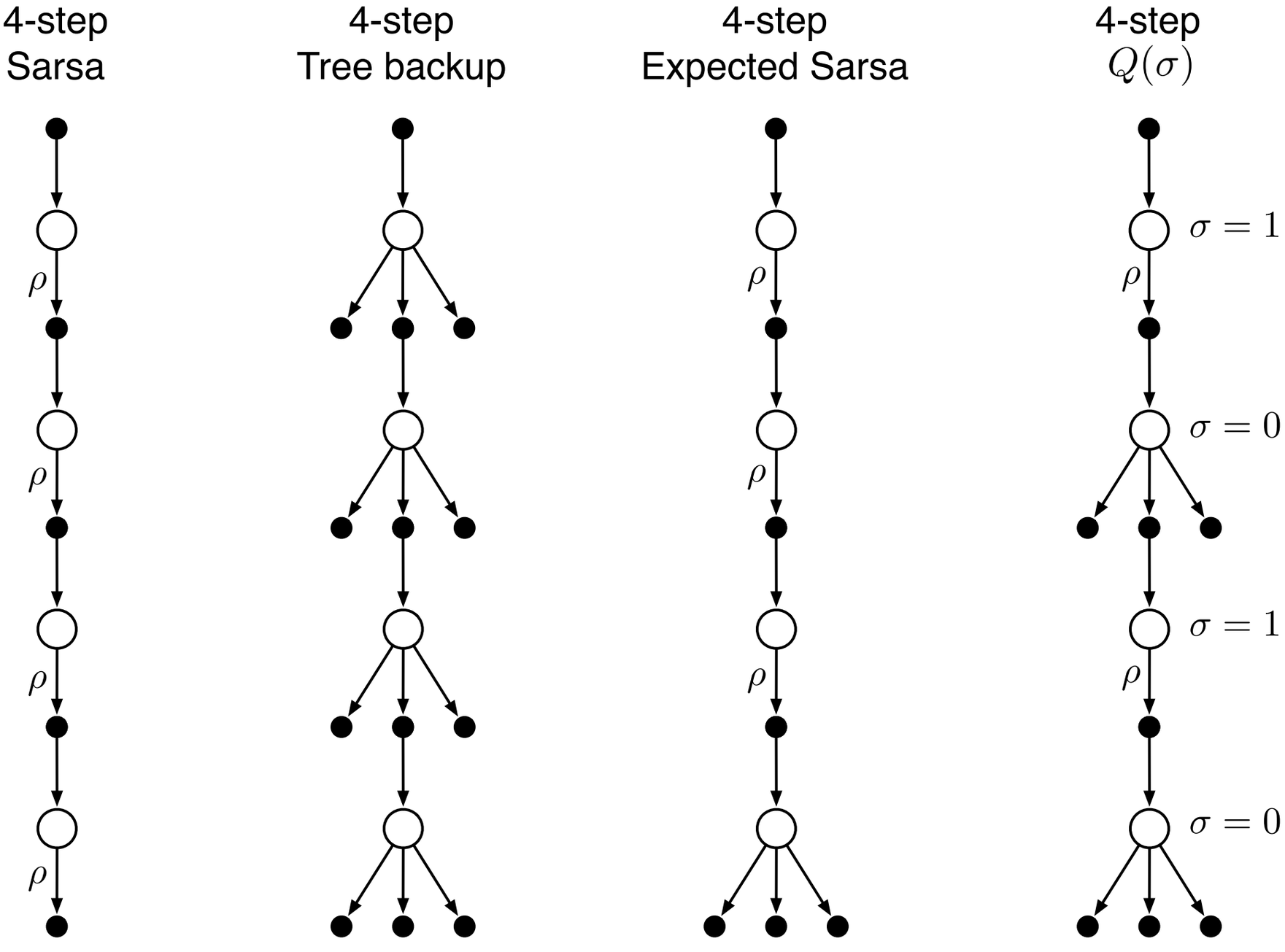


Target

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

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

A Unifying Algorithm: *n*-step Q(σ)



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

Computational costs

- must to remember the last **n** states
- learning is delayed by **n** steps
- **but,** per-step computation is small and uniform, like TD

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!

Unified View