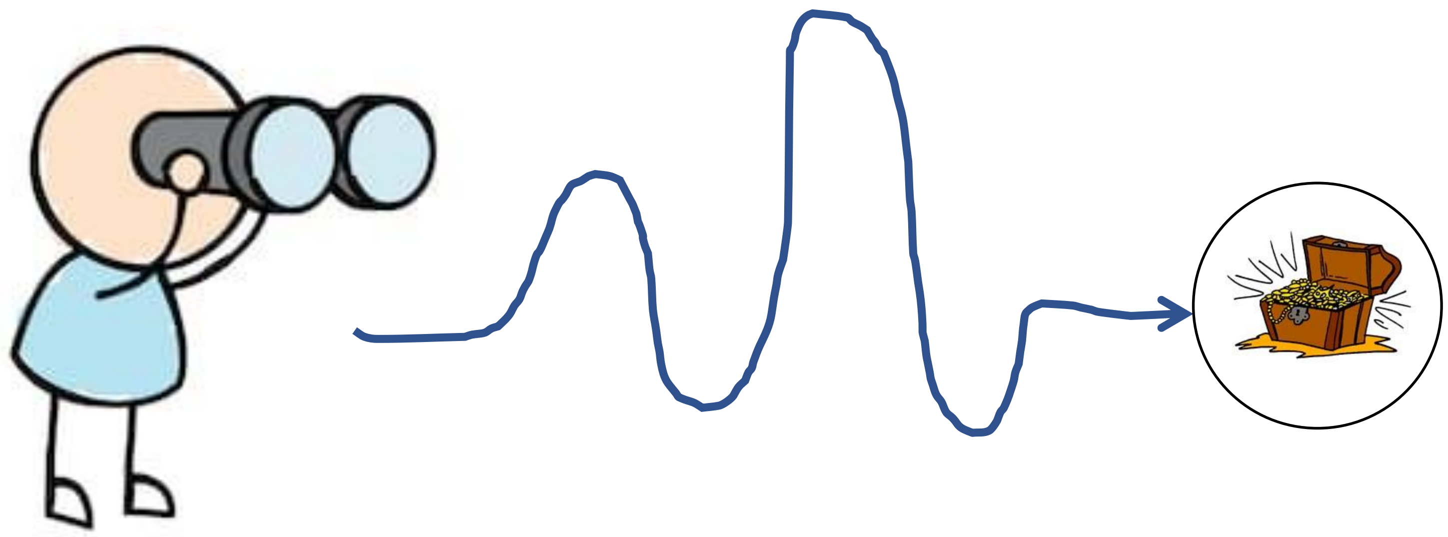
CSCE-642 Reinforcement Learning
Chapter 7: n-step Bootstrapping

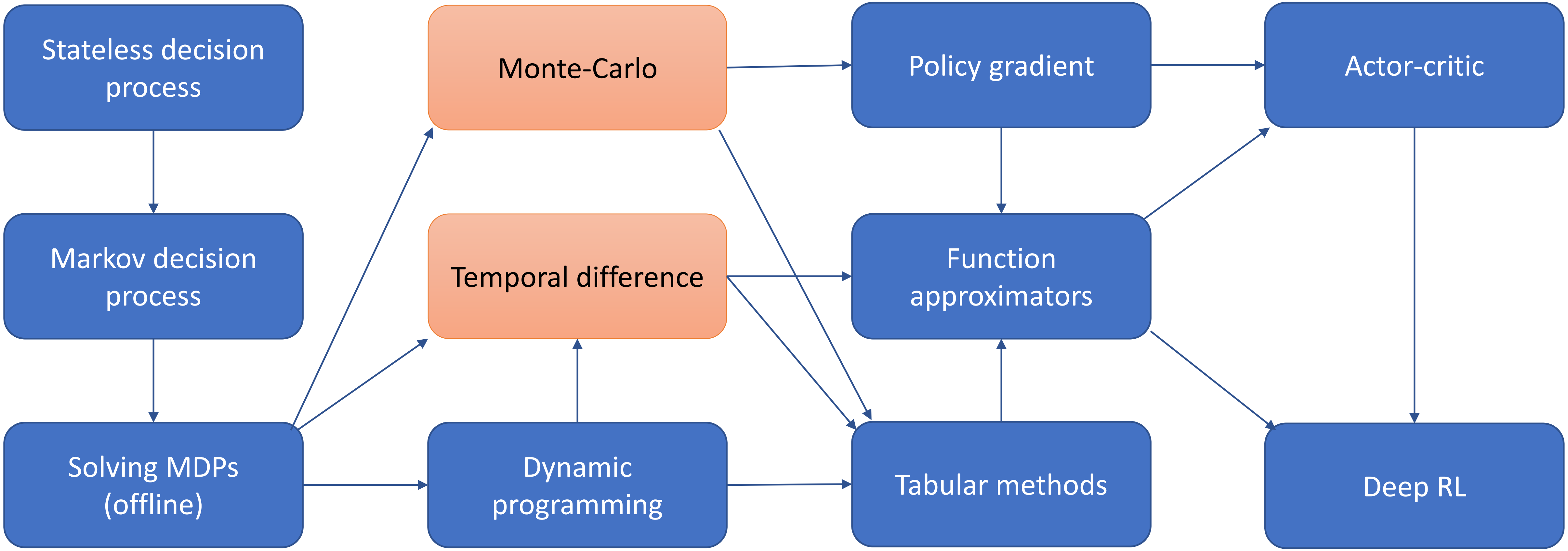


Instructor: Guni Sharon

TODOs:

- Quizzes:
  - Quiz3 Monte-Carlo Control Sep 16, EOD
  - Quiz4 TD-learning by Sep. 18, EOD
- Assignments:
  - Value Iteration, by September-23, EOD
  - Asynchronous Value Iteration, by September-23, EOD
  - Policy Iteration, by September-23, EOD
  - Monte-Carlo Control by September-30, EOD
  - Monte-Carlo Control with Importance Sampling by September-30, EOD
  - Tabular Q-Learning, by Oct. 7, EOD
  - SARSA, by Oct. 7, EOD
- Project:
  - Start writing, submit by Sep. 30

CSCE-689, Reinforcement Learning



Solving MDPs so far

Dynamic programming

- ☑ Off policy
- ☑ local learning, propagating values from neighbors (Bootstrapping)
- ☒ Model based

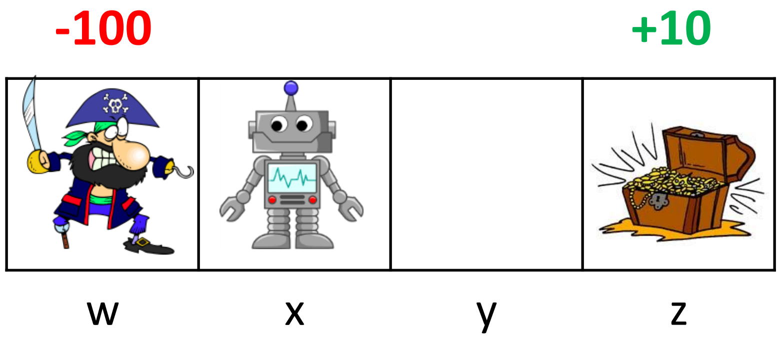
Monte-Carlo

- ☒ On-policy (though important sampling can be used)
- ☒ Requires a full episode to train on
- ☑ Model free, online learning

- $Q(z, exit) = 10$
- $Q(y, \rightarrow) = 0 + \gamma \max_a Q(z, a)$
- $Q(x, \rightarrow) = 0 + \gamma \max_a Q(y, a)$

$$q^*(s,a) = \sum_{s'} p(s'|s,a)\left(r(s,a,s') + \gamma \max_a [q^*(s',a)]\right)$$

$\gamma = 0.9$



- Episode = $\{x, y, z, exit\}$
- $Q(z, exit) = 10$
- $Q(y, \rightarrow) = 9$
- $Q(x, \rightarrow) = 8.1$

Fuse DP and MC

Dynamic programming

- ✓ Off policy
- ✓ local learning, propagating values from neighbors (Bootstrapping)
- ✗ Model based

Monte-Carlo

- ✗ On-policy (though important sampling can be used)
- ✗ Requires a full episode to train on
- ✓ Model free, online learning

TD Learning

- ✓ Off policy
- ✓ local learning, propagating values from neighbors (Bootsraping)
- ✓ Model free, online learning

Online Bellman update

- $q^*(s,a) = \sum_{s'} P(s'|s,a)\,(R(s,a,s') + \gamma \max_{a'}[q^*(s',a')])$
- Model and reward function are unknown
- Instead, we observe transitions: $< s_t, a_t, r_{t+1}, s_{t+1} >$
  - $\mathbb{E}[X] = \sum_{k=1}^{|X|} x_i \Pr\{x_i\} = \frac{\sum_{k=1}^{N} sample_k}{N}$
- $\mathbb{E}_{s'\sim P(s'|s,a)}\left[\mathbb{E}[R(s,a,s')] + \gamma \max_{a'}[q^*(s',a')]\right]$
- For a single sample $= r_{t+1} + \gamma \max_{a'}[q^*(s_{t+1},a')]$
- = unbiased estimation of the Bellman update

Temporal difference learning

- $Q(s,a) = Q(s,a) + \alpha\left(R_{t+1} + \gamma \max_{a'}[q^*(s',a')] - Q(s,a)\right)$
- But we don't know $q^*(s',a)$
- Use the learned estimation $Q(s',a)$
- $Q(s,a) = Q(s,a) + \alpha\left(\underbrace{R_{t+1} + \gamma \max_{a'}[Q(s',a')] - Q(s,a)}_{\text{Temporal difference error: } \delta}\right)$

Introduces a maximization bias!

SARSA: On-policy TD Control

- $Q(s,a) = Q(s,a) + \alpha\left(R_{t+1} + \gamma \max_{a'}[Q(s',a')] - Q(s,a)\right)$
- Replace $\max_{a'}[Q(s',a')]$ with values from the observed transition
  - $< s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}, >$
- $Q(s_t,a_t) = Q(s_t,a_t) + \alpha\left(r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right)$
- SARSA converges with probability 1 to an optimal policy and action-values as long as all state–action pairs are visited infinitely often, and the policy converges in the limit to the greedy policy
  - Which can be arranged, for example, with ε-greedy policies by setting ε= 1/t

SARSA: On-policy TD Control

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Q-learning: Off-policy TD Control

- Use the original TD update rule
- $Q(s,a) = Q(s,a) + \alpha\left(R_{t+1} + \gamma \max_{a'}[Q(s',a')] - Q(s,a)\right)$
- Approximates the state-action value for the optimal policy, i.e., q^*
  - Assuming that every state-action pair is visited infinitely often
- Follows from the proof of convergence for the Bellman function
  - See slides #25,26 in “3MDPs+DP.pptx”

Q-learning: Off-policy TD Control

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q (e.g., ϵ-greedy)
    Take action A, observe R, S'
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
    $S \leftarrow S'$
  until S is terminal

Double Q-learning

Initialize $Q_1(s,a)$ and $Q_2(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily
Initialize $Q_1(\textit{terminal-state}, \cdot) = Q_2(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    Choose A from S using policy derived from Q_1 and Q_2 (e.g., ε-greedy in $Q_1 + Q_2$)
    Take action A, observe R, S'
    With 0.5 probabilility:
      $Q_1(S,A) \leftarrow Q_1(S,A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S',a)\big) - Q_1(S,A)\Big)$
    else:
      $Q_2(S,A) \leftarrow Q_2(S,A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S',a)\big) - Q_2(S,A)\Big)$
    $S \leftarrow S'$
  until S is terminal

TD learning

- Temporal difference = online computation of the TD error, δ
- Allows us to perform online Bellman updates without any knowledge of the model
- In many cases converges faster than MC. Performing online updates (no need to complete an episode)
- SARSA provides unbiased TD learning through on policy estimations
- Q learning might suffer from a maximization bias that can be addressed with double Q learning
- Both SARSA and Q learning are guaranteed to converge to the optimal state/action values in the tabular case + appropriate learning rate

Model free RL so far

Monte-Carlo

- ✗ On-policy (but important sampling can be used)
- ✗ Requires a full episode to train on
- ✗ Noisy learning (high variance)
- ✓ Efficient value propagation

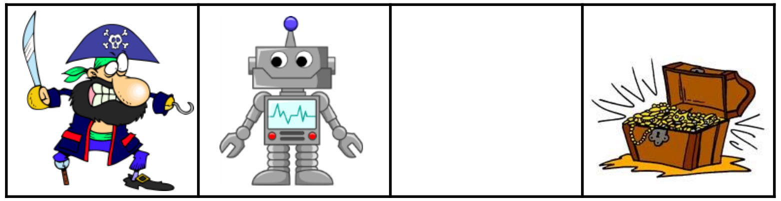
Temporal-Difference learning

- ✓ Off policy
- ✓ Local learning, propagating values from neighbors (Bootsraping)
- ✗ slow value propagation

- Episode = $\{x, y, z, exit\}$
- $Q(z, exit) = 10$
- $Q(y, \rightarrow) = 9$
- $Q(x, \rightarrow) = 8.1$

$\gamma = 0.9$

-100 +10



w x y z

- Episode = $\{x, y, z, exit\}$
- $Q(x, \rightarrow) = 0$
- $Q(y, \rightarrow) = 0$
- $Q(z, exit) = 10$

MC and TD(0) are two extremes of the same continuum

Monte-Carlo

- ✗ On-policy (but important sampling can be used)
- ✗ Requires a full episode to train on
- ✗ Noisy learning (high variance)
- ✓ Efficient value propagation

Temporal-Difference learning

- ✓ Off policy
- ✓ Local learning, propagating values from neighbors (Bootsraping)
- ✗ slow value propagation

n-step TD

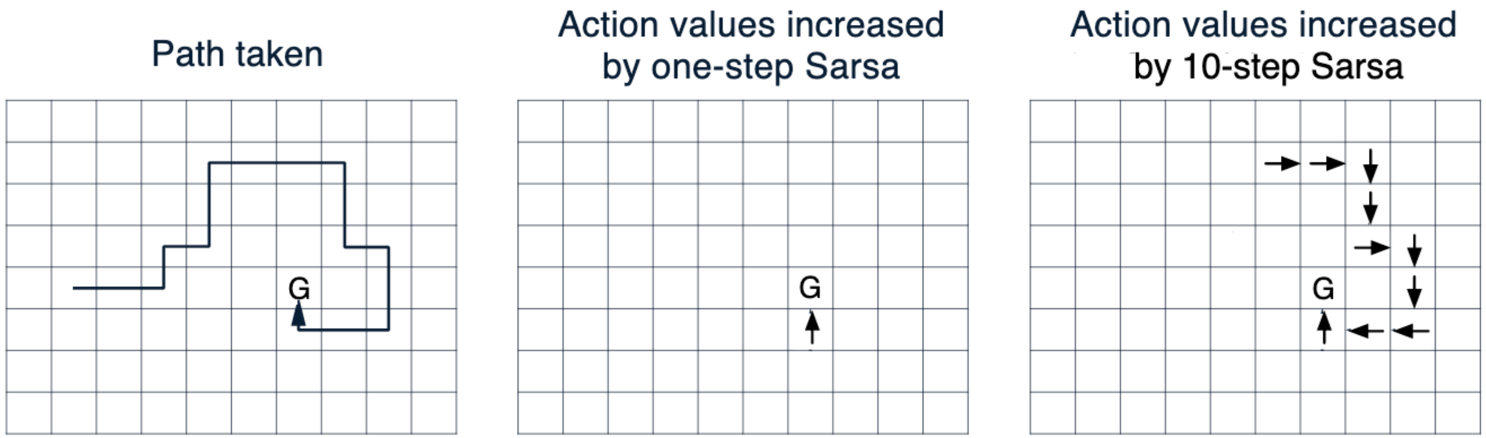
- ✗ On-policy (but important sampling can be used)
- ✓ Local learning, propagating values from neighbors (Bootsraping)
- ✓ Efficient value propagation

n-step TD

- Assume an episode $\{s_0, a_0, r_1, s_1, a_1, r_2 \ldots, s_{t-1}, a_{T-1}, r_T\}$
- How should $Q(s_0, a_0)$ be updated?
  - $Q(s_0, a_0) = Q(s_0, a_0) + \alpha\big(newval - Q(s_0, a_0)\big)$
- 1-tep: $newval = R_1 + \gamma Q(S_1, A_1)$
- MC: $newval = \sum_{k=1}^{T} \gamma^{k-1} R_k$
- n-step: $newval = [\sum_{k=1}^{n} \gamma^{k-1} R_k] + \gamma^n Q(S_n, A_n)$

n-step TD

- What is a reasonable n for the following scenario (assume $\gamma < 1$)?
- 1-step: low var returns but inefficient learning (slow value propagation)
- 10-steps? (assume 4-connected grid)
- 3-step?
- Tradeoff between learning speed and value variance



n-step SARSA

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

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 $\tau = T - 1$

n-step SARSA

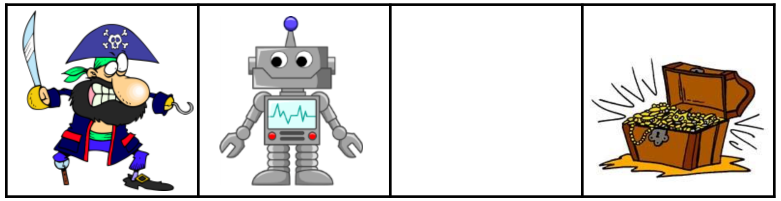
S_0	x
A_0	$\rightarrow$
R_1	–
S_1	–
A_1	–
R_2	–
S_2	–
A_2	–
R_3	–

Q

w	x	y	z
0	**0,0**	**0,0**	**0**

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 $\tau = T - 1$

n-step SARSA

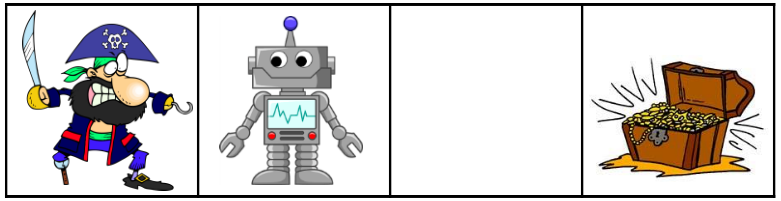
S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	–
R_2	–
S_2	–
A_2	–
R_3	–

Q

w	x	y	z
0	**0,0**	**0,0**	**0**

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 $\tau = T - 1$

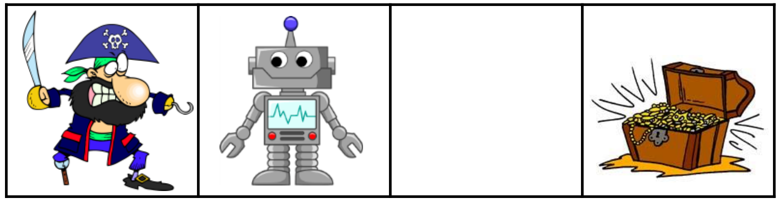
n-step SARSA

Q	0	0,0	0,0	0
	w	x	y	z

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	–
S_2	–
A_2	–
R_3	–

$\gamma = 0.9$

-100 +10



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t + 1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA

$\tau = -1$

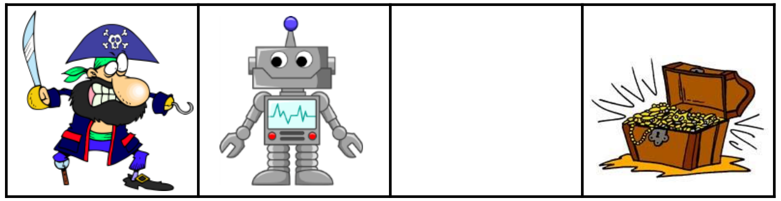
S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	$-$
S_2	$-$
A_2	$-$
R_3	$-$

Q

w	x	y	z
0	**0,0**	**0,0**	**0**

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t + 1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA

$\tau = -1$

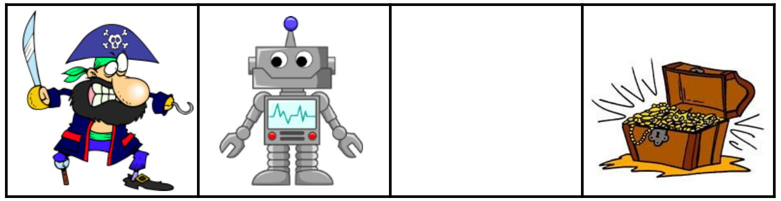
S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$-$
R_3	$-$

Q

w	x	y	z
0	**0,0**	**0,0**	**0**

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t+1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ ($G_{\tau:\tau+n}$)
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA

$\tau = -1$

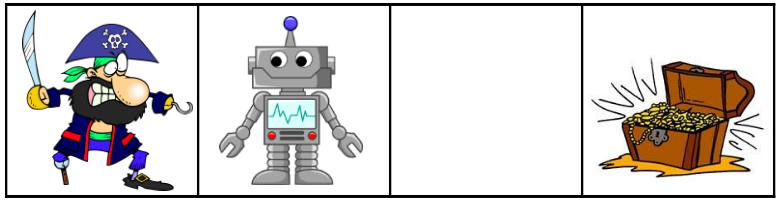
S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	–

Q

w	x	y	z
0	**0,0**	**0,0**	**0**

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 $\tau = T - 1$

n-step SARSA

$\tau = 0$

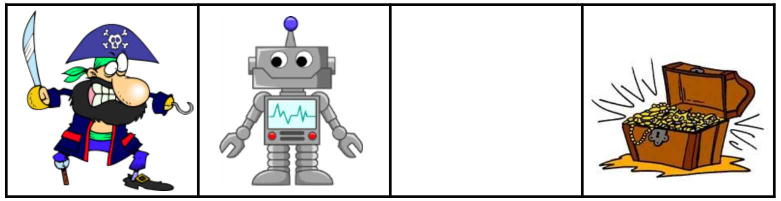
S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	$-$

Q

w	x	y	z
0	**0,0**	**0,0**	**0**

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t+1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA

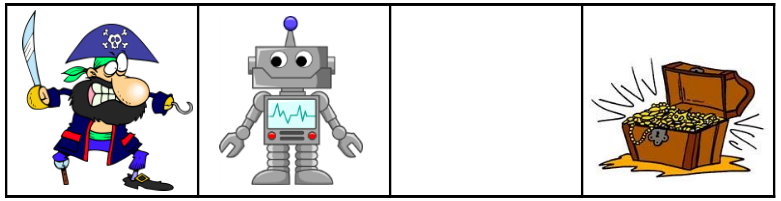
$\tau = 0$

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	$-$

Q	**0**	**0,0**	**0,0**	**0**
	w	x	y	z

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 $\tau = T - 1$

n-step SARSA

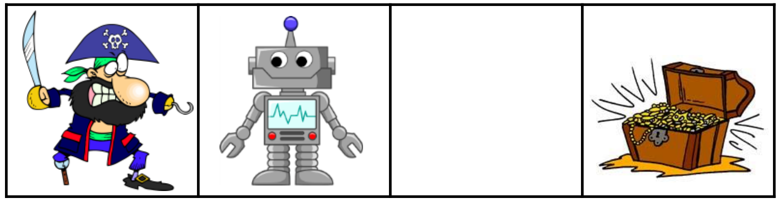
$\tau = 0$

Q	**0**	**0,0**	**0,0**	**0**
	w	x	y	z

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \dots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t+1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA

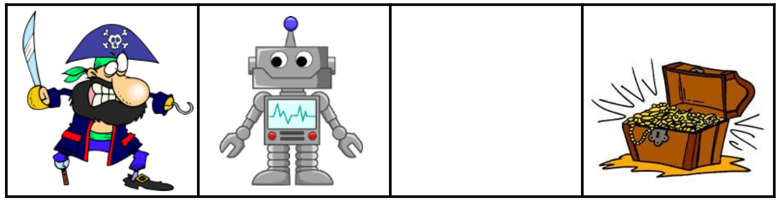
$\tau = 0$

Q	w	x	y	z
	0	0,0	0,0	0

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

$\gamma = 0.9$

-100 +10



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
- Initialize and store $S_0 \neq$ terminal
- Select and store an action $A_0 \sim \pi(\cdot|S_0)$
- $T \leftarrow \infty$
- 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:
      - $\boxed{T \leftarrow t+1}$
    - else:
      - Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  - $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  - If $\tau \geq 0$:
    - $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
    - If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
    - $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[G - Q(S_\tau, A_\tau)]$
    - If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
- Until $\tau = T - 1$

n-step SARSA

$\tau = 1$

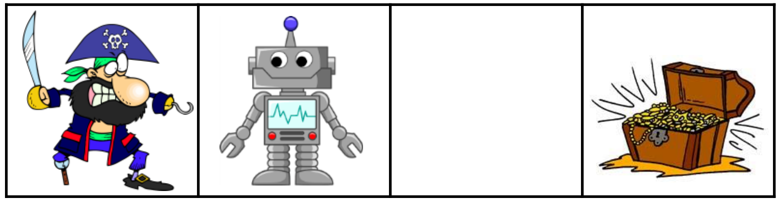
Q

0	**0,0**	**0,0**	**0**
w	x	y	z

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
- Initialize and store $S_0 \neq$ terminal
- Select and store an action $A_0 \sim \pi(\cdot|S_0)$
- $T \leftarrow \infty$
- For $t = 0, 1, 2, \dots$:
  - If $t < T$, then:
    - Take action A_t
    - Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
    - If S_{t+1} is terminal, then:
      - $T \leftarrow t+1$
    - else:
      - Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  - $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  - If $\tau \geq 0$:
    - $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
    - If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
    - $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
    - If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
- Until $\tau = T - 1$

n-step SARSA

$\tau = 1$

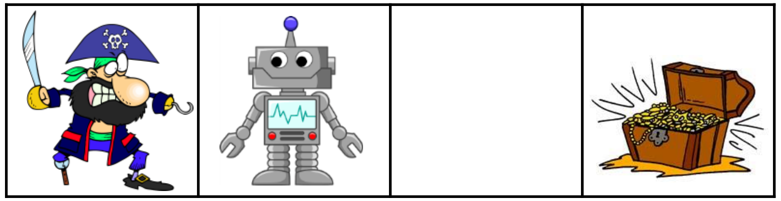
S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

Q

0	**0,0**	**0,0**	**0**
w	x	y	z

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$ = 9
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA

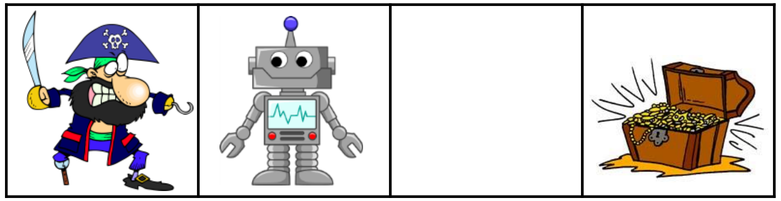
$\tau = 1$

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

Q	**0**	**0,0**	**0,9**	**0**
	w	x	y	z

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
- Initialize and store $S_0 \neq$ terminal
- Select and store an action $A_0 \sim \pi(\cdot|S_0)$
- $T \leftarrow \infty$
- 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 $\tau \geq 0$:
    - $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$ = 9
    - If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
    - $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
    - If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
- Until $\tau = T - 1$

n-step SARSA

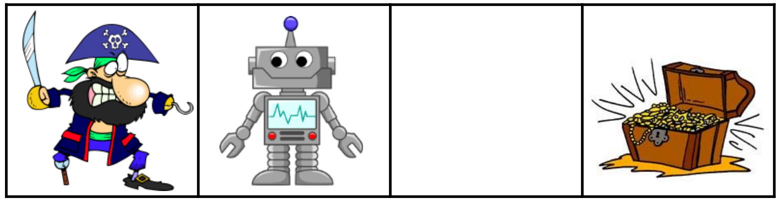
$\tau = 1$

Q	0	0,0	0,9	0
	w	x	y	z

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

$\gamma = 0.9$

-100 +10



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \dots$:
  | If $t < T$, then: False
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t + 1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA

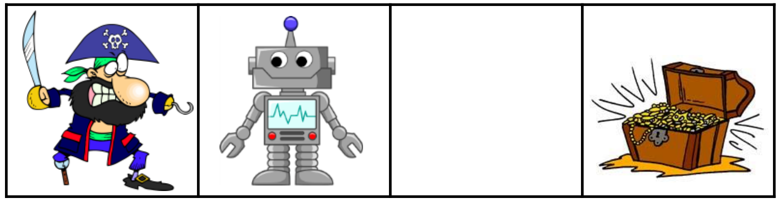
$\tau = 1$

Q	w	x	y	z
	0	**0,0**	**0,9**	**0**

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$ = 10
  | 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 $\tau = T - 1$

n-step SARSA

$\tau = 1$

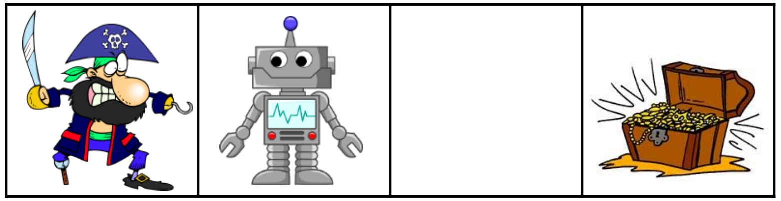
S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

Q

w	x	y	z
0	**0,0**	**0,9**	**10**

$\gamma = 0.9$

-100 **+10**



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  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 $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$ = 10
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA

$\tau = 1$

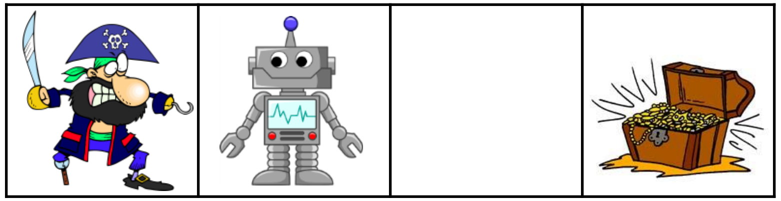
Q

0	0,0	0,9	10
w	x	y	z

S_0	x
A_0	$\rightarrow$
R_1	0
S_1	y
A_1	$\rightarrow$
R_2	0
S_2	z
A_2	$exit$
R_3	10

$\gamma = 0.9$

-100 +10



w x y z

n-step Sarsa for estimating $Q \approx q_*$, or $Q \approx q_\pi$ for a given 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
Parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n n=2
All store and access operations (for S_t, A_t, and R_t) can take their index mod n

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t + 1$
  | else:
  | Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
Until $\tau = T - 1$

n-step TD is on-policy

- 1-tep: $newval = R_1 + \gamma Q(S_1, A_1)$
- MC: $newval = \sum_{k=1}^{T} \gamma^k R_k$
- n-step: $newval = \sum_{k=1}^{n} \gamma^n R_n + \gamma^{n+1} Q(S_n, A_n)$

- $Q(S_n, A_n)$ assumes a policy $\pi(S_n) = A_n$
- Off-policy TD assumes π^* when computing $\max_{a\prime} Q(S_n, a')$
- Can we perform off-policy, n-step TD?
  - Yes! with important sampling

Importance sampling reminder

- Given a trajectory τ drawn by running b
- We can define the probability $\Pr\{\tau|b\}$
- We can also define $\Pr\{\tau|\pi\}$
- Define the ***importance sampling ratio*** as: $\rho_t = \frac{\Pr\{\tau_t|\pi\}}{\Pr\{\tau_t|b\}}$
- Can we compute ρ without a model, $p(S_{K+1}|S_K, A_K)$?
- $\rho_t = \frac{\prod_{k=t}^{T-1} \pi(A_K|S_K)\cancel{p(S_{K+1}|S_K,A_K)}}{\prod_{k=t}^{T-1} b(A_K|S_K)\cancel{p(S_{K+1}|S_K,A_K)}} = \prod_{k=t}^{T-1} \frac{\pi(A_K|S_K)}{b(A_K|S_K)}$ **YES!**

n-step SARSA + IS

- Similar to on-policy, n-step SARSA
- The observed n-step return is now multiplied by the IS ratio
- This can be used for learning q^* while following soft ε-greedy exploration

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

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

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim b(\cdot|S_0)$
  $T \leftarrow \infty$
  For $t = 0, 1, 2, \ldots$:
  | If $t < T$, then:
  | Take action A_t
  | Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  | If S_{t+1} is terminal, then:
  | $T \leftarrow t + 1$
  | else:
  | Select and store an action $A_{t+1} \sim b(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  | $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1, T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$ $(\rho_{\tau+1:t+n-1})$
  | $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
  | If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ $(G_{\tau:\tau+n})$
  | $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\left[G - Q(S_\tau, A_\tau)\right]$
  | If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε-greedy wrt Q
  Until $\tau = T - 1$

n-step SARSA + IS

- When attempting to learn q^*:
- How should we define the target policy?
  - $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$
- How should we define the behavior policy?
  - Any policy that provides coverage

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

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

```
Repeat (for each episode):
    Initialize and store S_0 ≠ terminal
    Select and store an action A_0 ~ b(·|S_0)
    T ← ∞
    For t = 0, 1, 2, ... :
    |   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 ← t + 1
    |       else:
    |           Select and store an action A_{t+1} ~ b(·|S_{t+1})
    |   τ ← t − n + 1    (τ is the time whose estimate is being updated)
    |   If τ ≥ 0:
    |       ρ ← ∏_{i=τ+1}^{min(τ+n−1, T−1)} π(A_i|S_i) / b(A_i|S_i)          (ρ_{τ+1:t+n−1})
    |       G ← Σ_{i=τ+1}^{min(τ+n, T)} γ^{i−τ−1} R_i
    |       If τ + n < T, then: G ← G + γ^n Q(S_{τ+n}, A_{τ+n})          (G_{τ:τ+n})
    |       Q(S_τ, A_τ) ← Q(S_τ, A_τ) + αρ [G − Q(S_τ, A_τ)]
    |       If π is being learned, then ensure that π(·|S_τ) is ε-greedy wrt Q
    Until τ = T − 1
```

What did we learn?

- TD learning results in slow propagation of future rewards across the state space
  - Especially problematic in sparse reward settings
- MC, on the other hand, suffers from high variance in observed returns
  - Especially problematic in stochastic environments and long episodes
- TD learning with n-step return gaps these two extremes
  - The best *n* is domain specific and is usually chosen empirically
  - Careful! Like MC, it is off-policy (use IS when needed)

What next?

- **Lecture**: Model-based RL (as opposed to model-free RL)
- **Quiz (on Canvas)**:
  - n-step Bootstrapping
  - By Sep. 23, EoD
- **Project**:
  - Start writing the project proposal document (due Sep. 30)