Lecture - 05

Temporal Difference Learning

→ Dynamic programming. (bootstrapping)

→ Monte Carlo. (does not requ. dynamics fn)

→ TD Learning → Comb. of MC & DP.

TD prediction:-

Constant - α MC

Simple every-visit Monte Carlo.

MC : $V(S_t) = V(S_t) + \alpha [G_t - V(S_t)]$

↑ sample return.

TD : $V(S_t) = V(S_t) + \alpha [R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$

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

$\gamma V(S_{t+1})$

Tabular TD for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha[R + \gamma V(S') - V(S)]$
    $S \leftarrow S'$
  until S is terminal

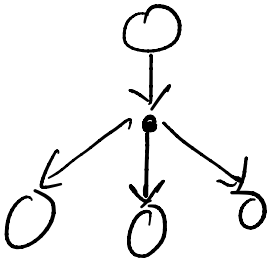
$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \quad \text{MC}$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma G_{t+1} \mid S_t = s]$$

$$= \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s] \quad \text{TD.}$$

Backup diagram of TD :-

DP



MC

TD

TDerror: $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

$$G_t - V(S_t) = R_{t+1} + \gamma G_{t+1} - V(S_t)$$

$$= R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1})$$

$$= \boxed{R_{t+1} + \gamma V(S_{t+1}) - V(S_t)} + \gamma G_{t+1} - \gamma V(S_{t+1})$$

$$= \delta_t + \gamma (G_{t+1} - V(S_{t+1}))$$

$$= \delta_t + \gamma \delta_{t+1} + \gamma^2 (G_{t+2} - V(S_{t+2}))$$

$$= \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \cdots + \gamma^{T-t-1} \delta_{T-1} + \gamma^{T-t} (G_T - V(S_T))$$

$$= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k .$$

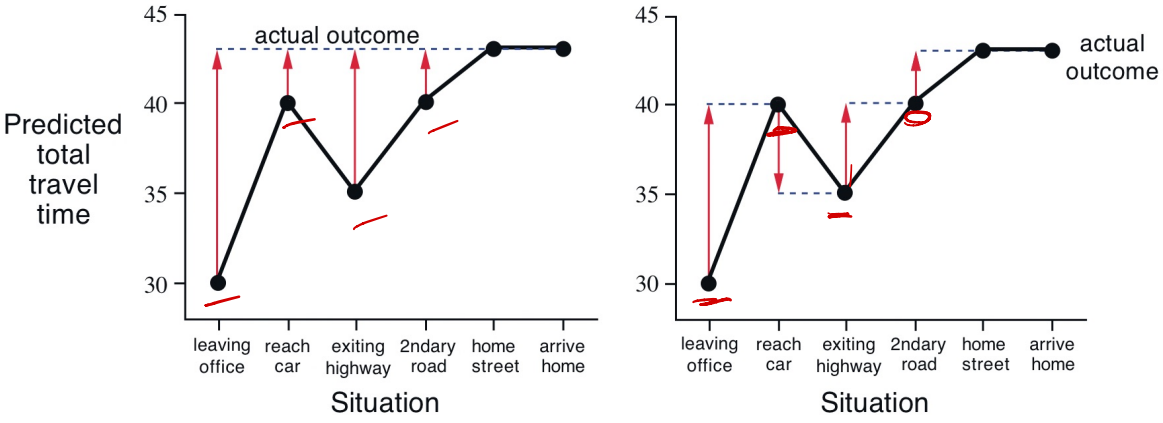
Driving home:

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Rewards — elapsed time on each leg of the journey.

$\gamma = 1$ No discount.

return → actual time to go from the state.



MC

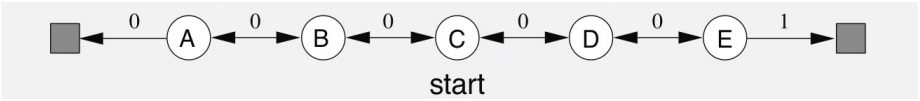
TD

Advantages of TD prediction methods:-

Random walk :-

Markov Reward Process (MRP).

undiscounted

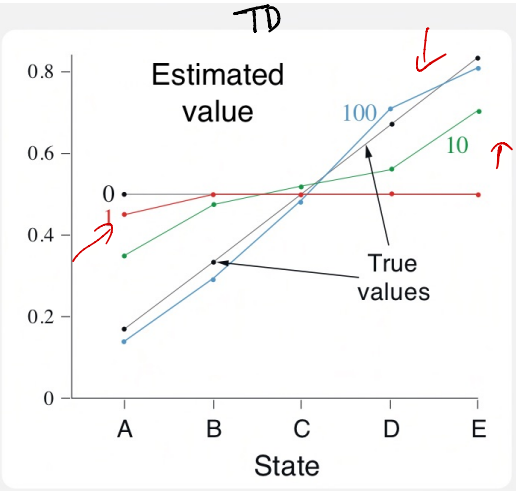


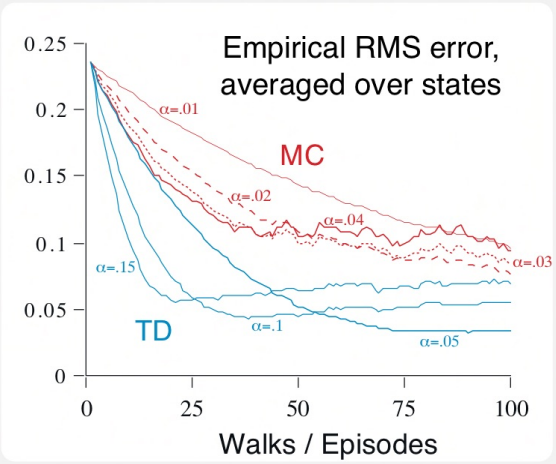
1/6 2/6 0.5 4/6 5/6

3/6

TD

initial state value of 0.5 for all s.



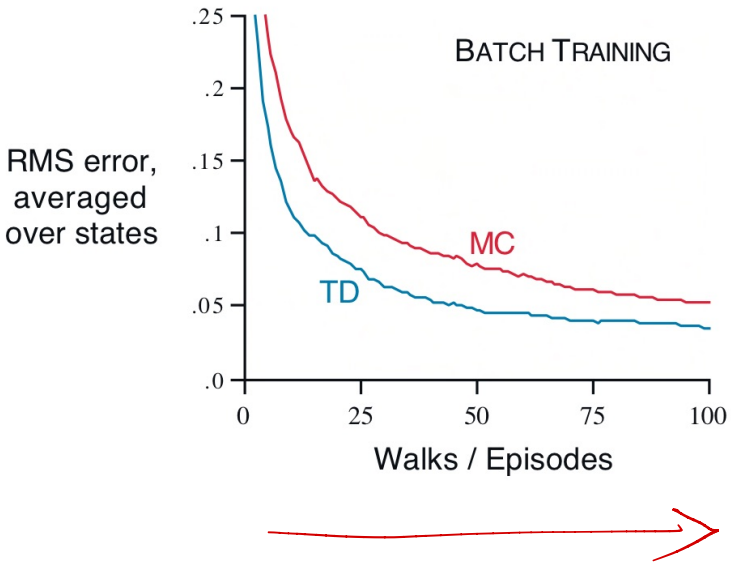


Batch-updating :-

1 ep.

2 eps

3 eps


BATCH TRAINING
RMS error, averaged over states
.25
.2
.15
.1
.05
.0
0
25
50
75
100
MC
TD
Walks / Episodes

A, 0, C, -5

1)	A, 0, B, 0	5) B, 1
2)	B, 1	6) B, 1
3)	B, 1	7) B, 1
4)	B, 1	8) B, 0

V(A), V(B)

$V(B) = 6/8 = 3/4$.

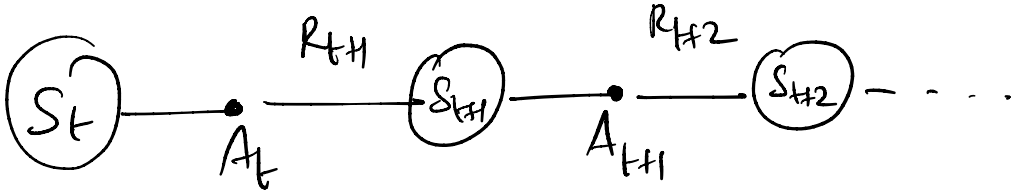
V(A) = 0 ? 3/4 ?

Batch MC → estimate that min. MSE on the train set.

Batch TD → Certainty-equivalence estimate

On-Policy TD Control: Sarsa.

for term. S_{t+1}, $Q(S_{t+1}, A_{t+1}) = 0$



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right]$$

$(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$

SARSA.

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

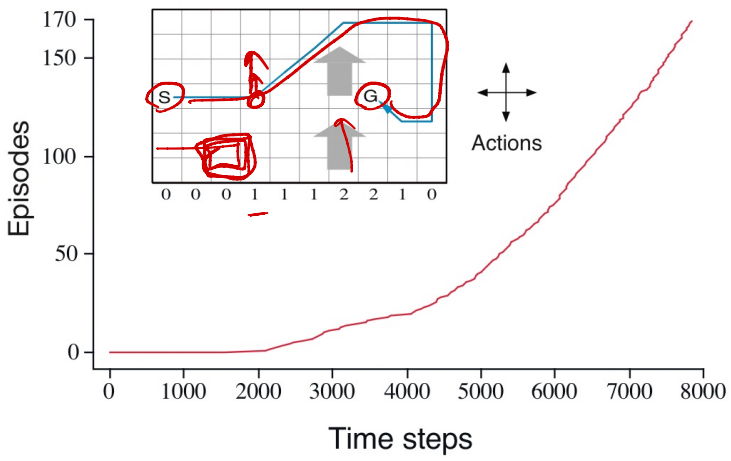
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop 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

Windy gridworld :

undiscounted.

episodic.



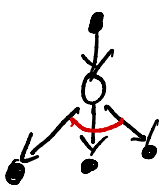
$\epsilon = 0.1$, $\alpha = 0.05$ initial $Q = 0$.

Off-Policy TD Control :

Q-learning [Watkins, 1989] :

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

Vs. Sarsa



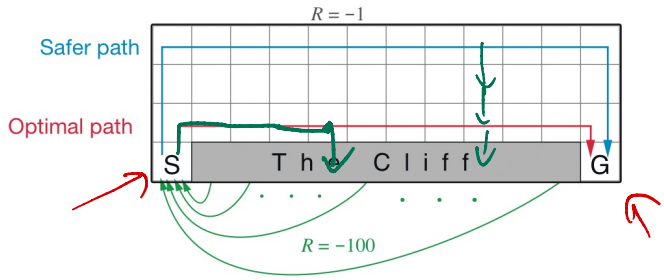
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

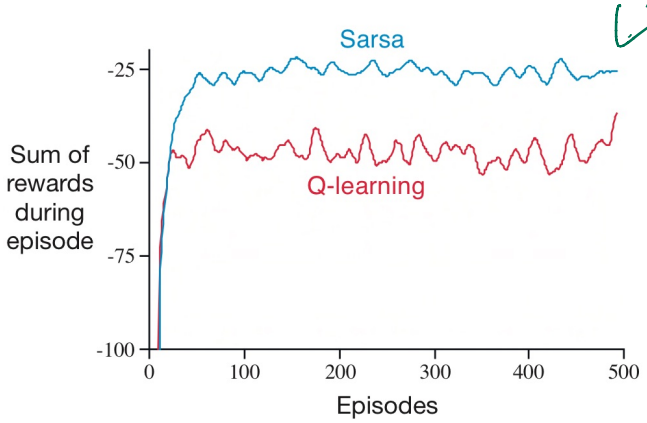
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

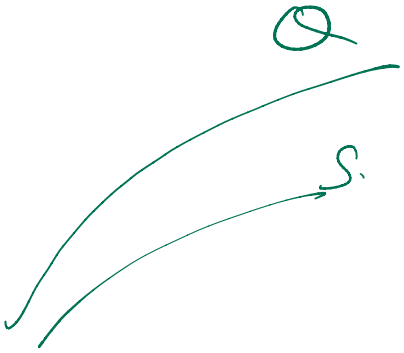
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

Cliff walking:

- undiscounted, episodic.





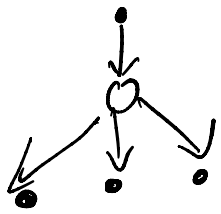


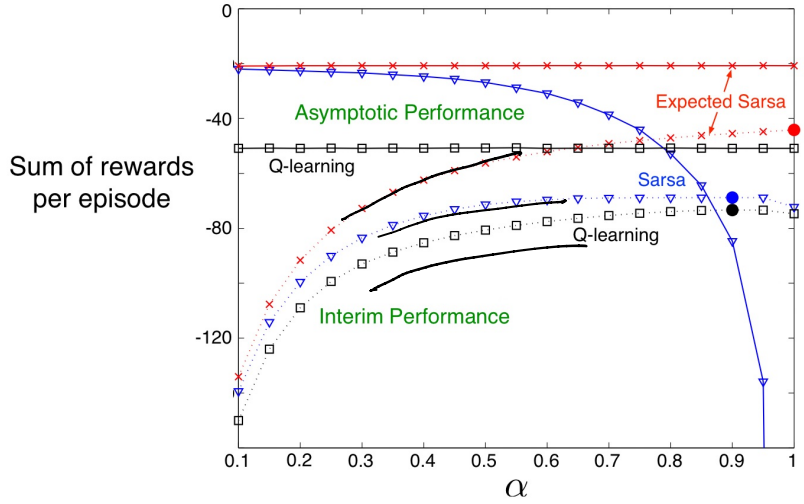
Expected Sarsa :-

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \, \mathbb{E}_\pi \big[Q(S_{t+1}, A_{t+1}) \mid S_{t+1} \big] - Q(S_t, A_t) \Big]$$

$$\leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \sum_a \pi(a \mid S_{t+1}) \, Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

Expected Sarsa.





π - greedy.

beh. π - ϵ-greedy

} Expected Sarsa.

Maximization bias :-

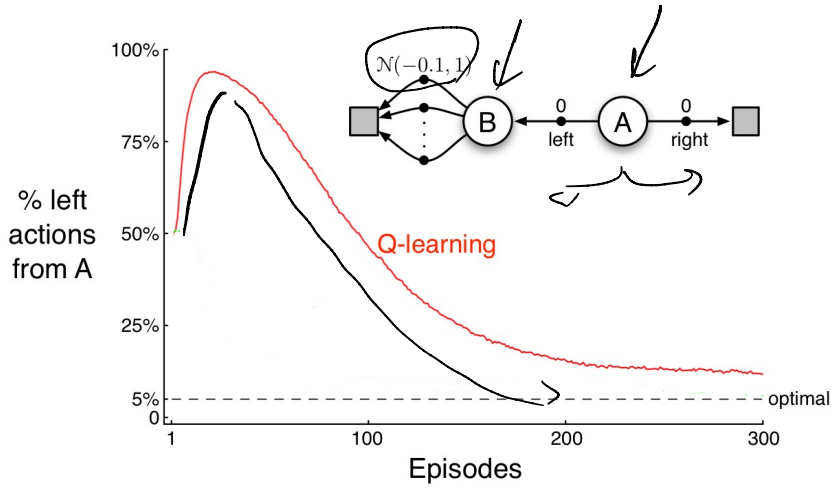
Q-learning : greedy :

Sarsa : ϵ greedy.

Positive bias / Maximization bias

S. $q(s,a) = 0$.

$Q(s,a)$



How to avoid max. bias?

Bandits $Q_1(a)$ $Q_2(a)$

Use Q_1 for max: $A^* = \underset{a}{\operatorname{argmax}}\, Q_1(a)$

Use Q_2 for estim: $Q_2(A^*) = Q_2\left(\underset{a}{\operatorname{argmax}}\, Q_1(a)\right)$

$\mathbb{E}[Q_2(A^*)] = q(A^*)$

$Q_1\left(\underset{a}{\operatorname{argmax}}\, Q_2(a)\right)$

Double learning:

Double Q-learning :-

$$Q_1(S_t, A_t) = Q_1(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q_2\big(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q_1(S_{t+1}, a)\big) - Q_1(S_t, A_t) \Big]$$

Double Q-learning, for estimating $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q_1(s, a)$ and $Q_2(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, such that $Q(\mathit{terminal}, \cdot) = 0$

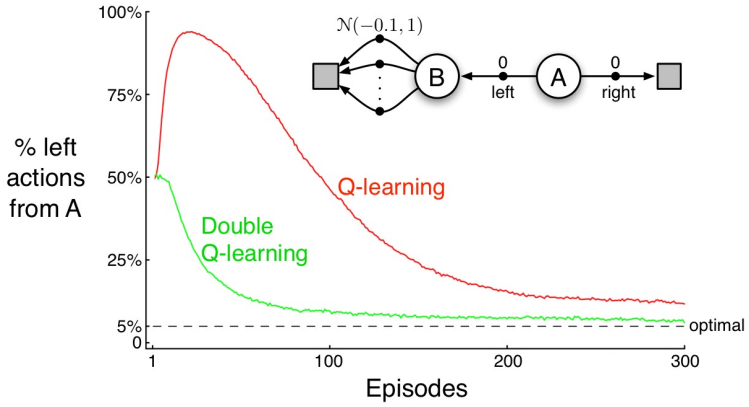
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using the policy ε-greedy in $Q_1 + Q_2$
    Take action A, observe R, S'
    With 0.5 probabilility:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha\Big(R + \gamma Q_2\big(S', \arg\max_a Q_1(S', a)\big) - Q_1(S, A)\Big)$$

    else:

$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha\Big(R + \gamma Q_1\big(S', \arg\max_a Q_2(S', a)\big) - Q_2(S, A)\Big)$$

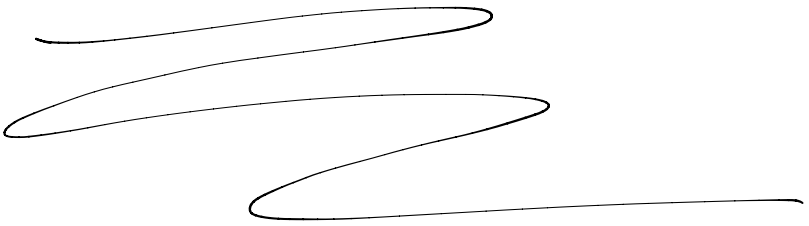
    $S \leftarrow S'$
  until S is terminal


100%
75%
50%
25%
5%
0
% left actions from A
$\mathcal{N}(-0.1, 1)$
B
0
left
A
0
right
Q-learning
Double Q-learning
optimal
1
100
200
300
Episodes

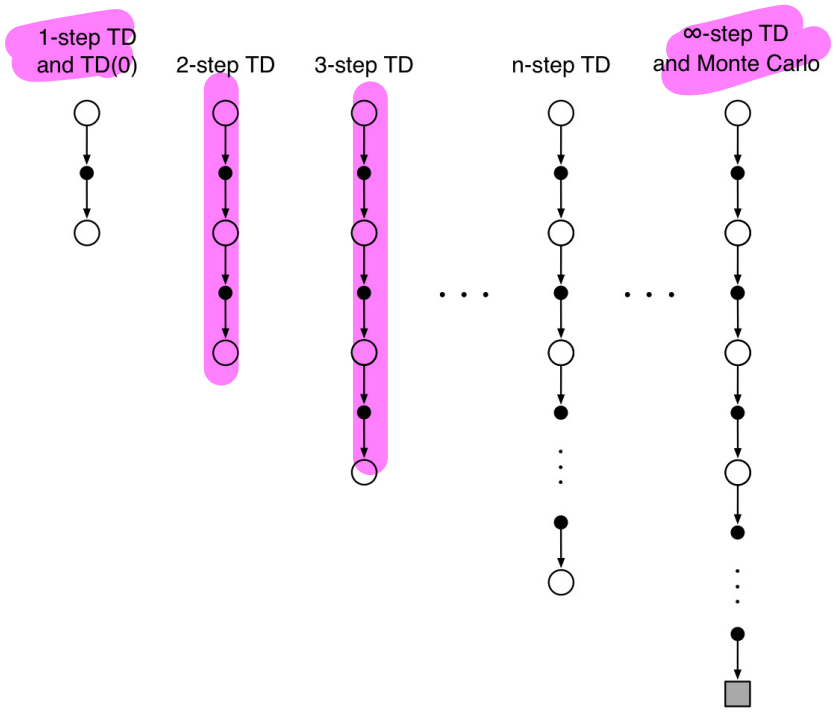
1-step TD

MC

n-step bootstrapping



n-step TD prediction:



$S_t, R_{t+1}, S_{t+1}, R_{t+2}, \ldots R_T, S_T$

MC: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T$

1-step TD: $G_{t:t+1} = R_{t+1} + \gamma V_t(S_{t+1})$

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

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

$t+n \geq T$, missing — zero.

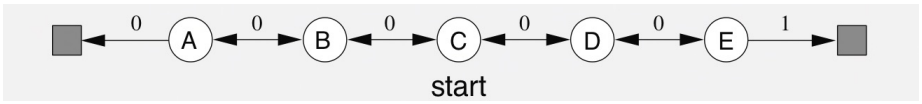
$$V_{t+n}(S_t) = V_{t+n-1}(S_t) +$$

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

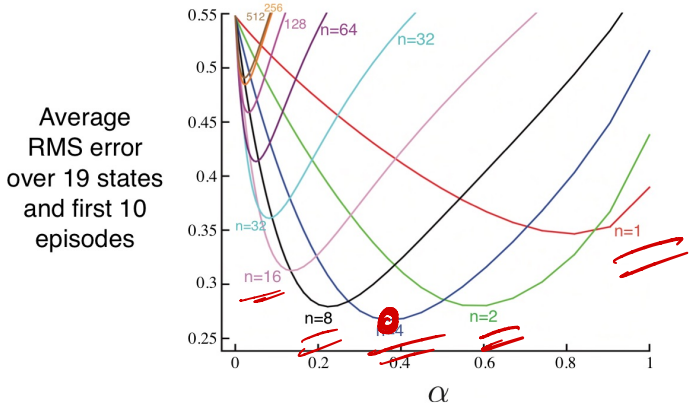
n-step TD for estimating $V \approx v_\pi$

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 $\tau = T - 1$



19 states instead of 5, with -1 outcome on left, all values initialized to 0.



n-step hyperparameter.

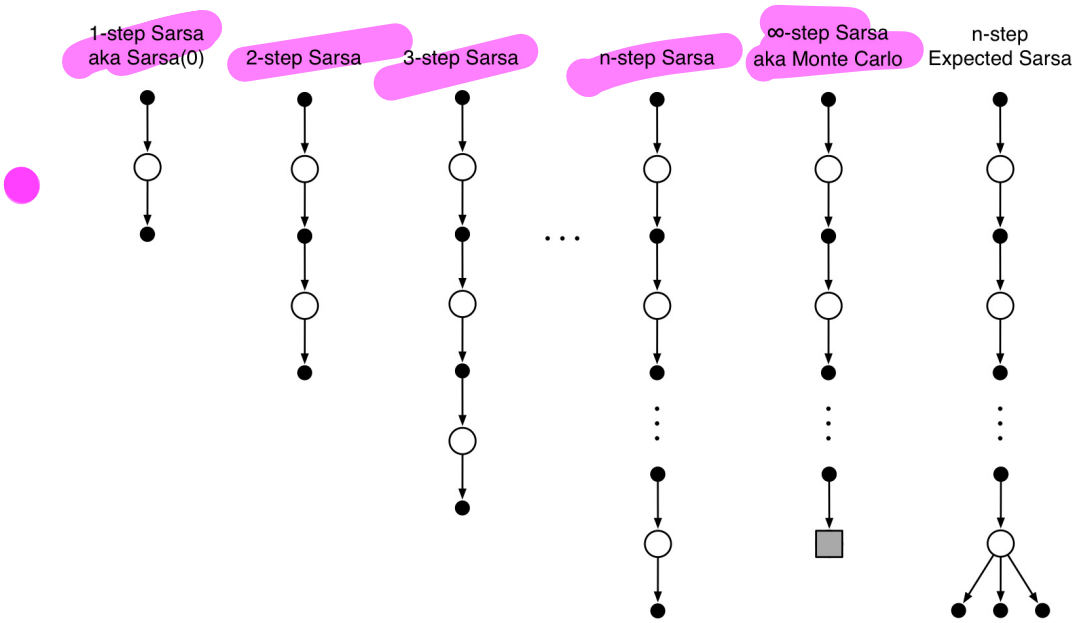
---

n-step Sarsa:-

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

with $G_{t:t+n} = G_t$ if $t+n \geq T$.

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

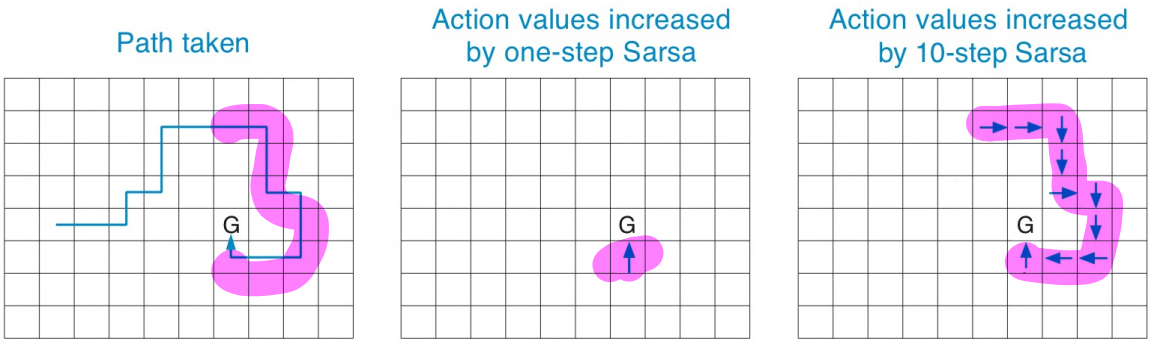

1-step Sarsa
aka Sarsa(0)
2-step Sarsa
3-step Sarsa
n-step Sarsa
∞-step Sarsa
aka Monte Carlo
n-step
Expected Sarsa

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

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

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

Illustration: 1-step Sarsa vs. 10-step Sarsa.



n-step expected Sarsa:

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

$$\bar{V}_t(s) = \sum_a \pi(a|s)\, Q_t(s,a).$$

n-step off-Policy learning:

Importance sampling:

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

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

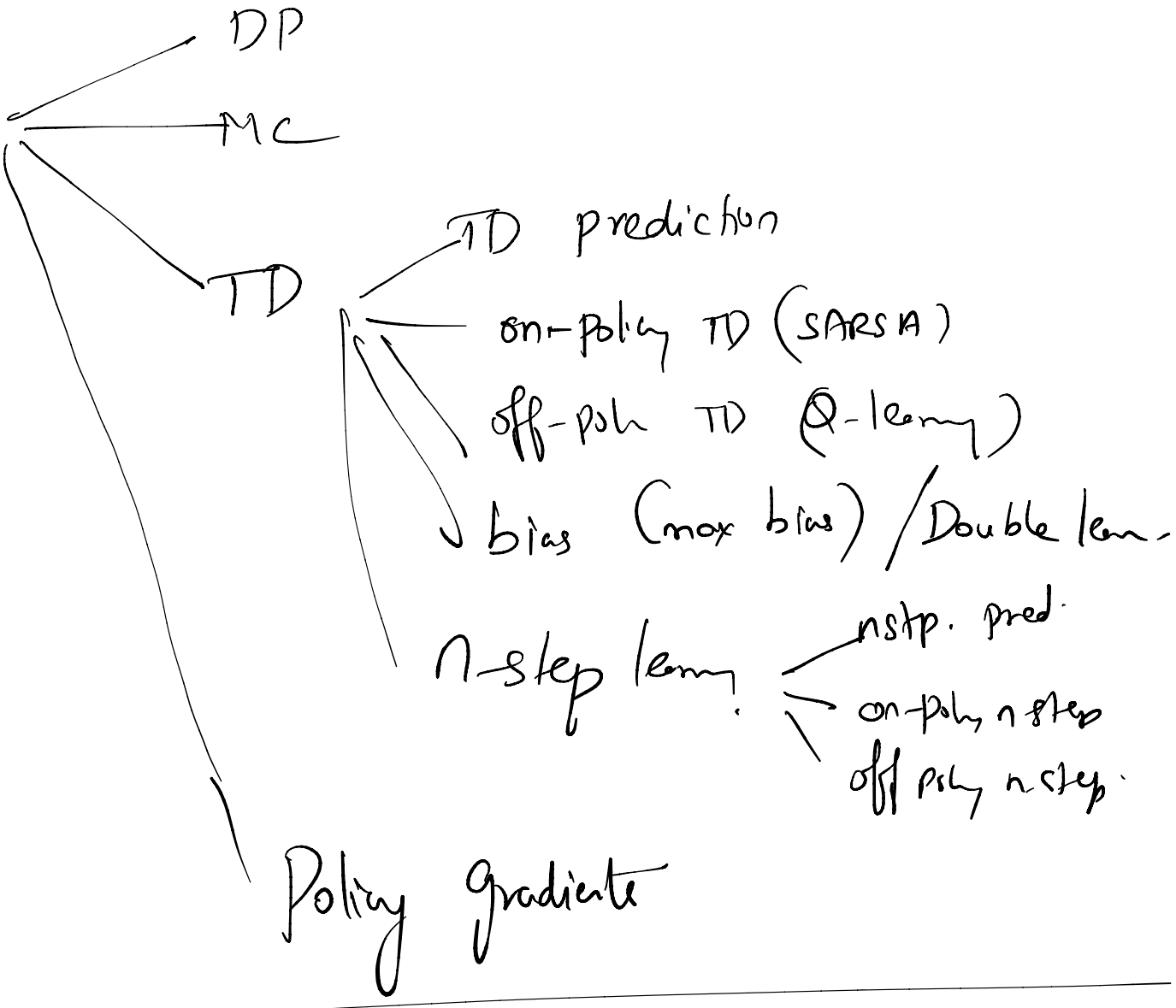
off-Policy n-step Sarsa:

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

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

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

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



— Function approximation
