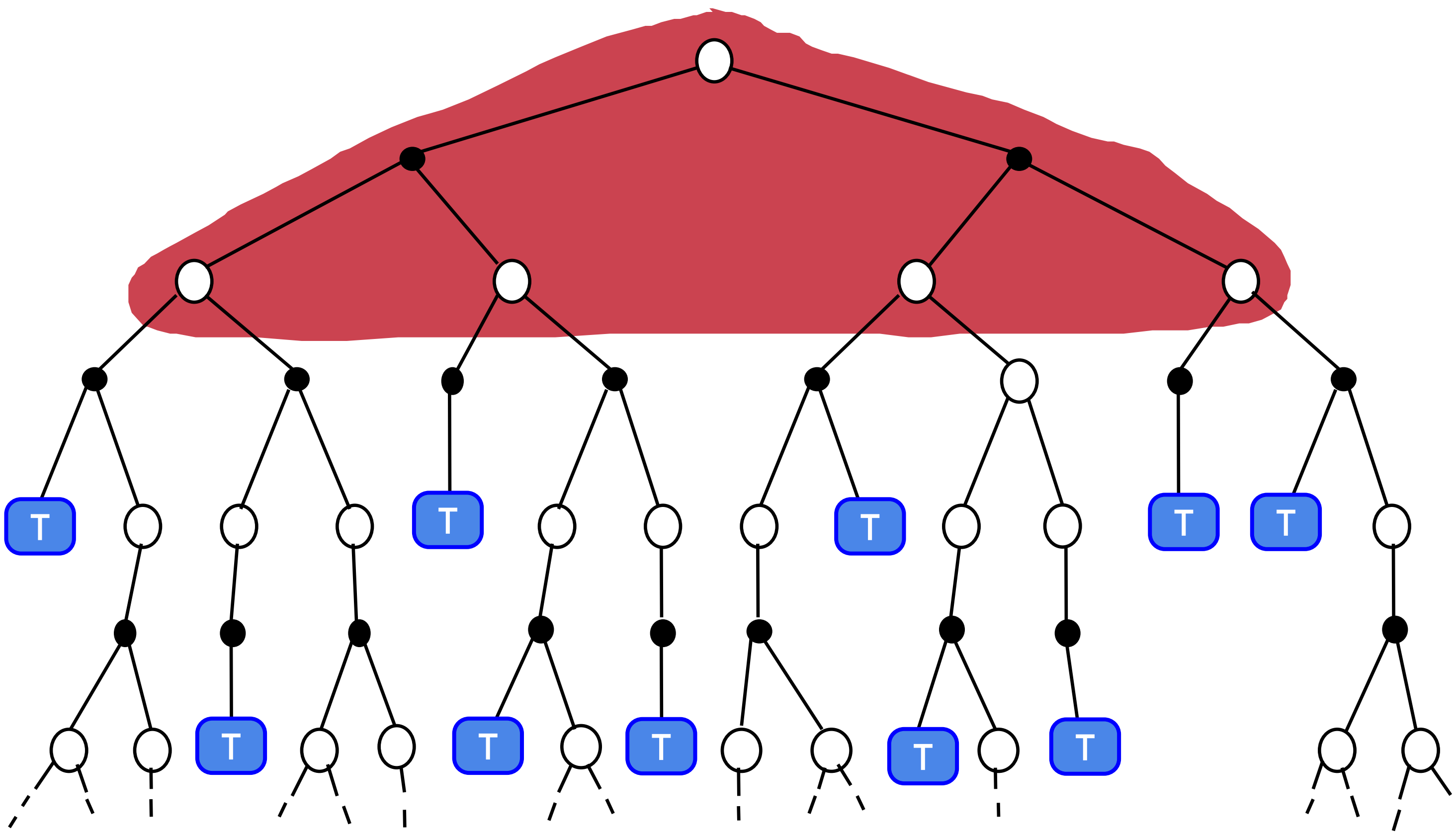
Lecture 5: Temporal Difference Learning and Monte-Carlo Methods

B. Ravindran

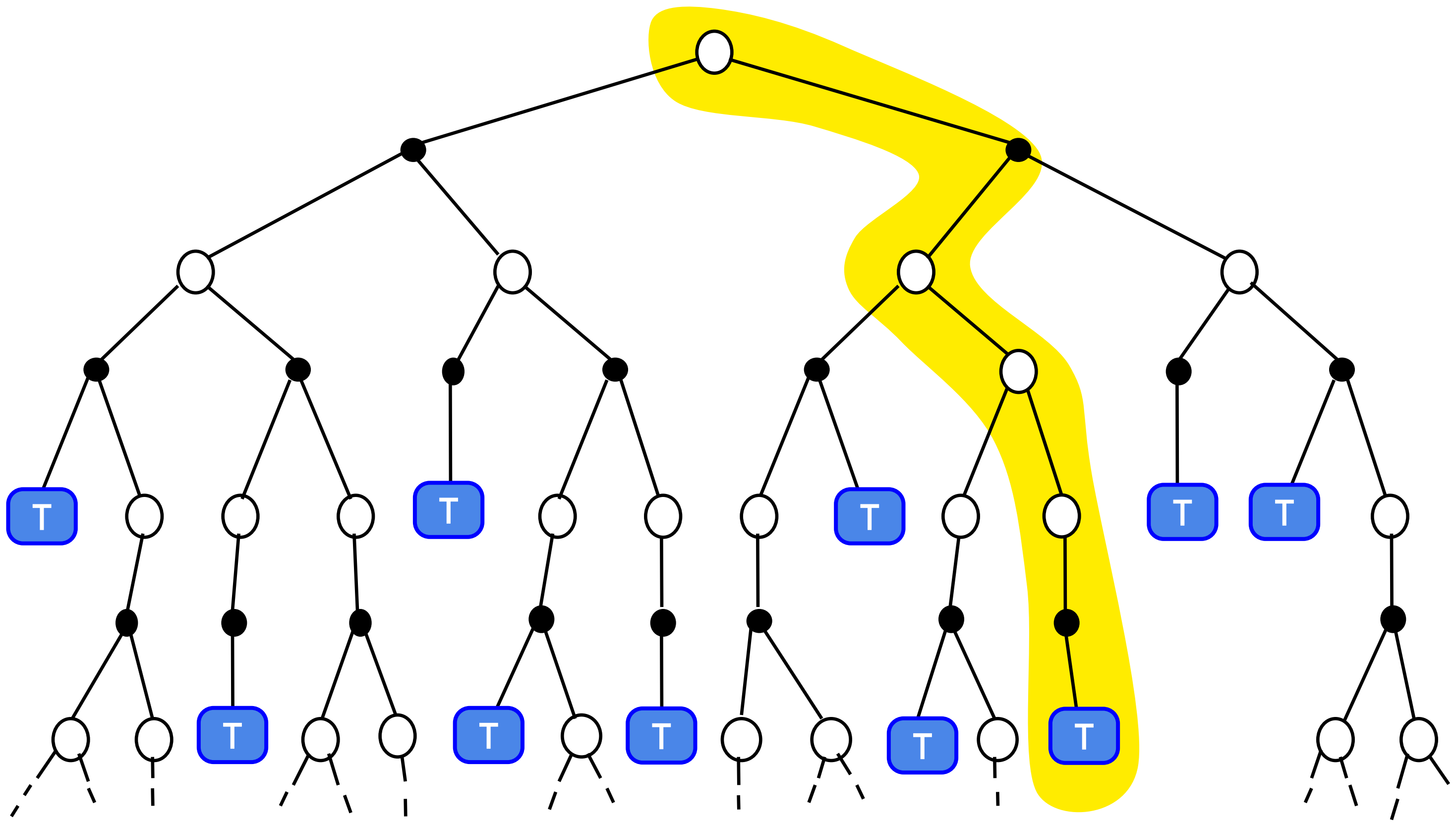
Dynamic Programming

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\Big[r + \gamma v_\pi(s')\Big]$$

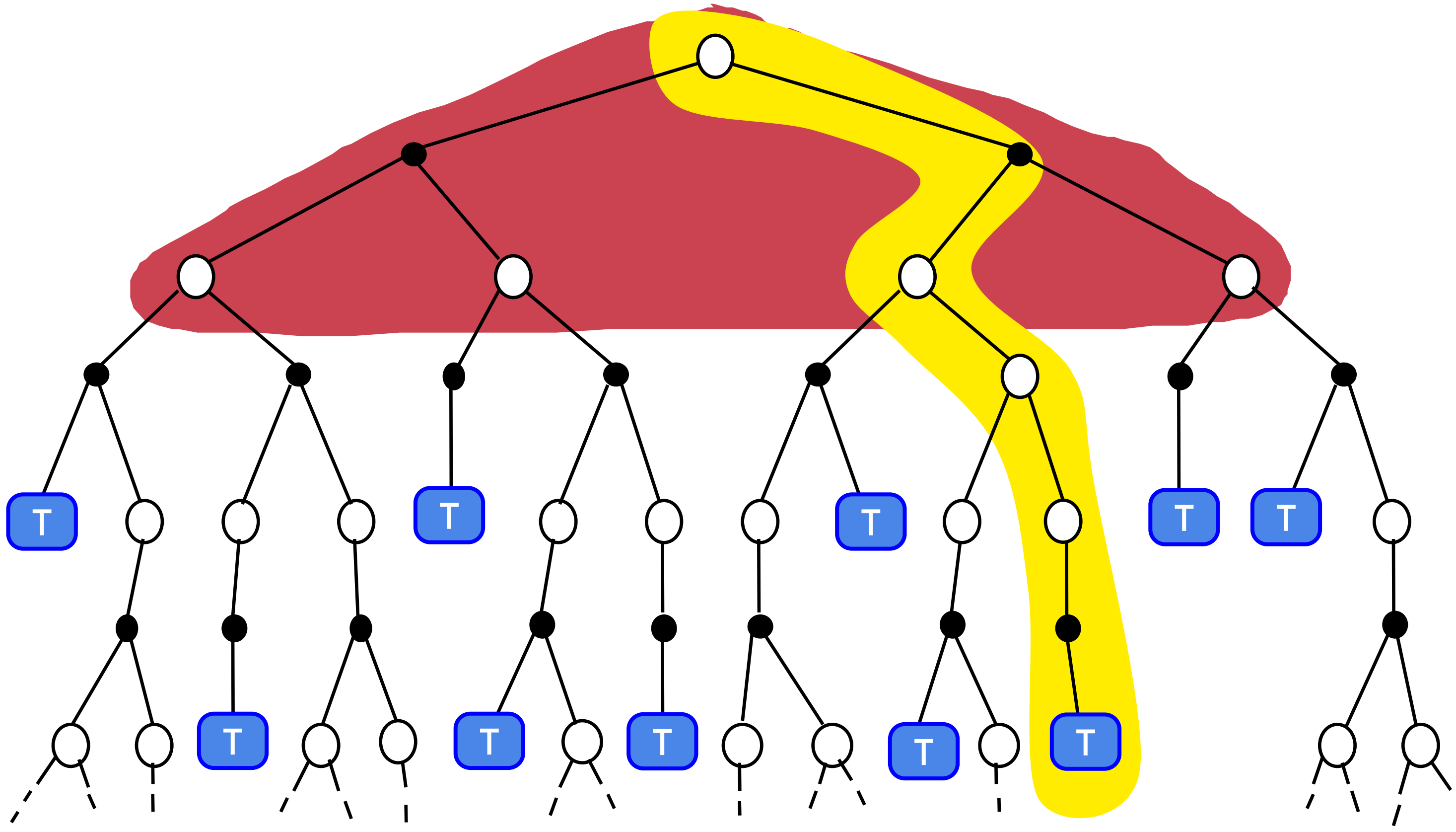


Monte-Carlo RL

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

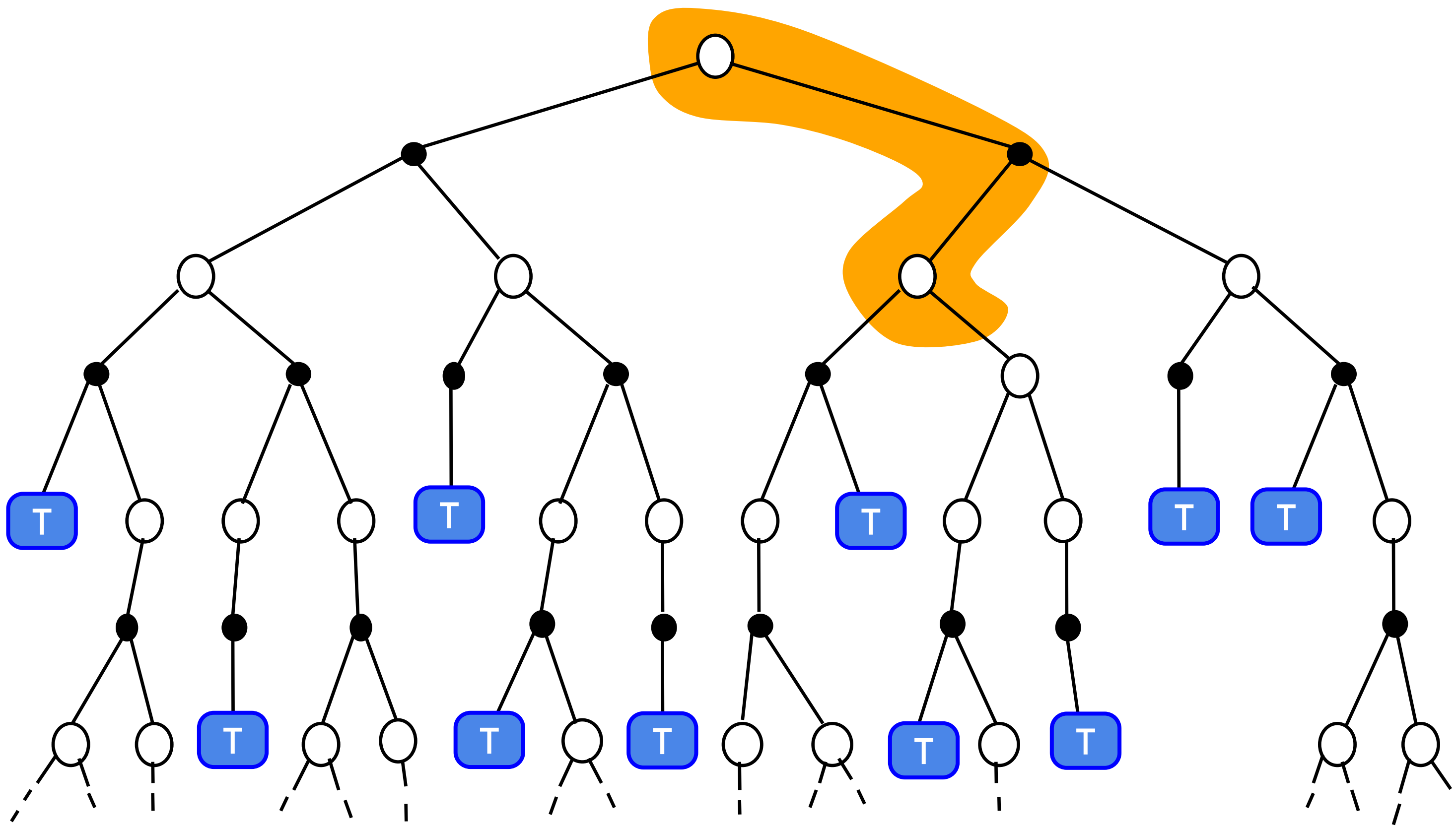


Best of Both Worlds !

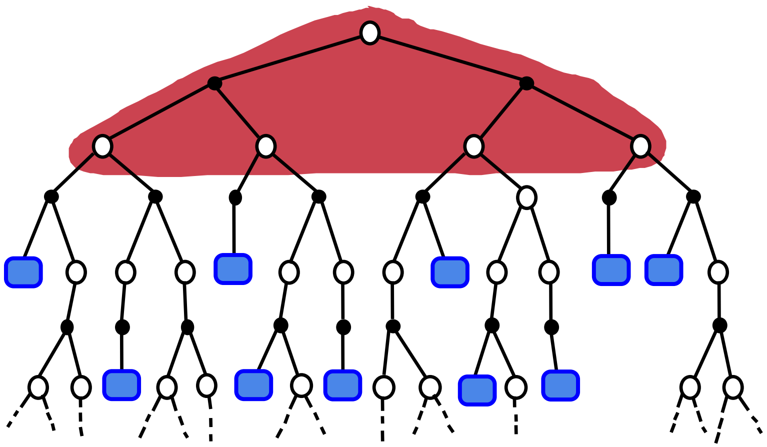


Simplest 'TD' Method

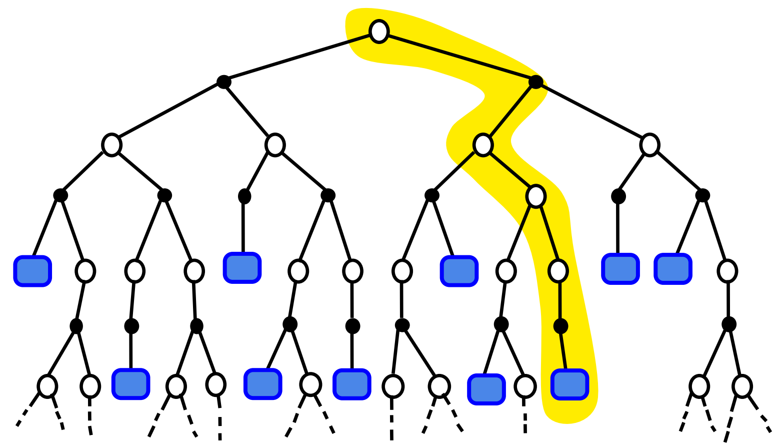
$$v_\pi(s) = R_{t+1} + \gamma v_\pi(s')$$



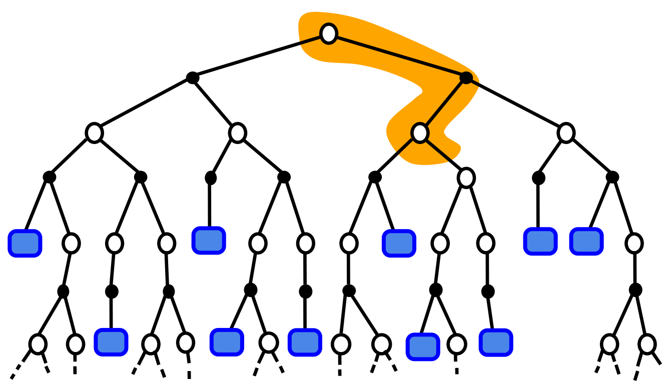
DP vs MC vs TD



DP



MC



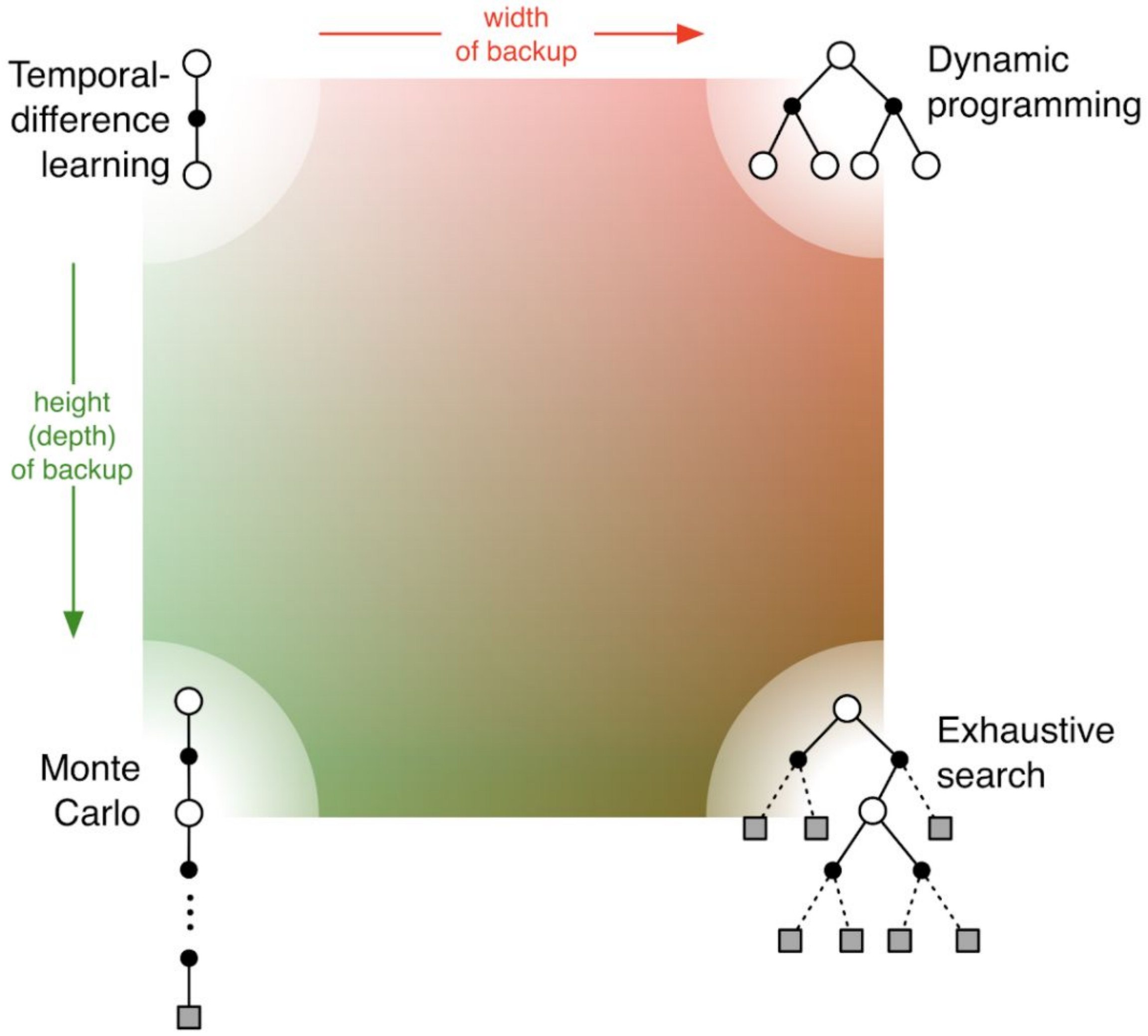
TD

Bootstrapping and Sampling

- ❑ Bootstrapping: Update using an estimate
  - ❑ DP and TD bootstrap ✅
  - ❑ Monte Carlo does not bootstrap ❌

- ❑ Sampling: Update using samples
  - ❑ TD and Monte Carlo sample ✅
  - ❑ DP *(typically)* does not sample ❌

Bootstrapping and Sampling



Monte-Carlo RL

- Learning directly from sample episodes of experience
- Does not use a known model and is model-free
- MC does not use bootstrapping
- Value functions are calculated as mean of discounted returns (G_t)

Monte-Carlo Prediction

- First-visit MC method estimates *V(s)* as the average of the returns following first visits to *s*

- Every-visit MC method estimates *V(s)* as the average of the returns following all visits to *s*

Monte-Carlo Prediction: First Visit

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, arbitrarily, for all $s \in \mathcal{S}$

$Returns(s) \leftarrow$ an empty list, for all $s \in \mathcal{S}$

Loop forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

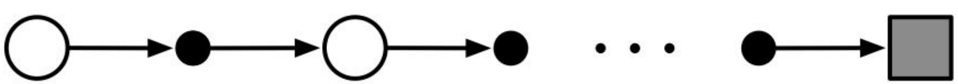
Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless S_t appears in $S_0, S_1, \ldots, S_{t-1}$:

Append G to $Returns(S_t)$

$V(S_t) \leftarrow \text{average}(Returns(S_t))$



Monte-Carlo Control: First Visit

Algorithm parameter: small $\varepsilon > 0$

Initialize:

$\pi \leftarrow$ an arbitrary ε-soft policy

$Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

$Returns(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following π: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair S_t, A_t appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:

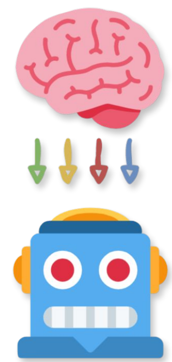
Append G to $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg\max_a Q(S_t, a)$ (with ties broken arbitrarily)

For all $a \in \mathcal{A}(S_t)$:

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$



Temporal Difference Learning

Temporal Difference

- *"If one had to identify one idea as central and novel to RL, it would undoubtedly be TD learning"*

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \Big]$$

- Simple rule to explain complex behaviors
- Has had profound impact in behavioral psychology and neuroscience!

Temporal Difference

- TD methods do not require a model of the environment, only experience

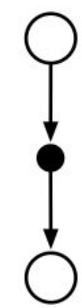
- TD methods can be fully incremental (bootstrapping)
  - You can learn before knowing the final outcome
    - Less memory & peak computation
  - You can learn without the final outcome
    - From incomplete sequences

- TD methods thus combine individual advantages of DP and MC

TD Prediction

- Policy Evaluation (the prediction problem): for a given policy, compute the state-value function
- No knowledge of *p* & *r*, but access to the real system, or a "sample" model assumed
- Uses bootstrapping and sampling
- The simplest TD method, TD(0):

$$V(S_t) \leftarrow V(S_t) + \alpha \Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \Big]$$



TD(0)

Stochastic Averaging Rule

$$E(x) \approx \frac{1}{n}\sum_{i=1}^{n} x_i$$

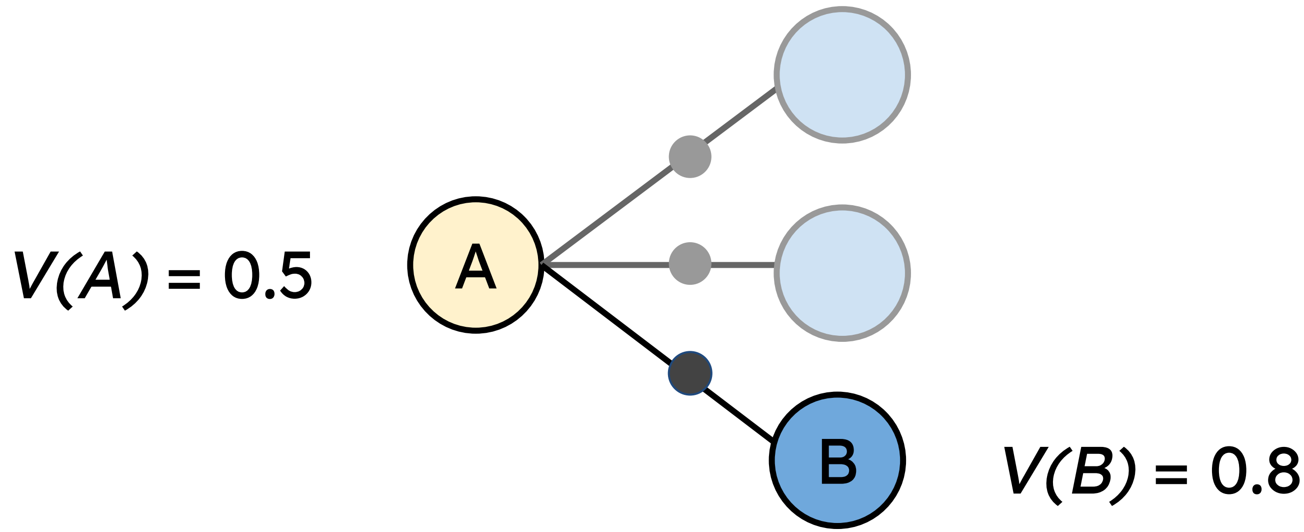
Let $\bar{x}_n$ be the average of the first n samples

$$\begin{aligned}
\bar{x}_{n+1} &= \frac{1}{n+1}\left(x_{n+1} + n\bar{x}_n\right) \\
&= \frac{1}{n+1}\left(x_{n+1} + n\bar{x}_n + \bar{x}_n - \bar{x}_n\right) \\
&= \frac{1}{n+1}\left(\left(n+1\right)\bar{x}_n + \left(x_{n+1} - \bar{x}_n\right)\right) \\
&= \bar{x}_n + \frac{1}{n+1}\left(x_{n+1} - \bar{x}_n\right) \\
&= \bar{x}_n + \alpha\left(x_{n+1} - \bar{x}_n\right)
\end{aligned}$$

new estimate = old estimate + α (new sample - old estimate)

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

TD Update Example



Assuming :

reward, r, A $\rightarrow$ B : 0

$\alpha : 0.2$

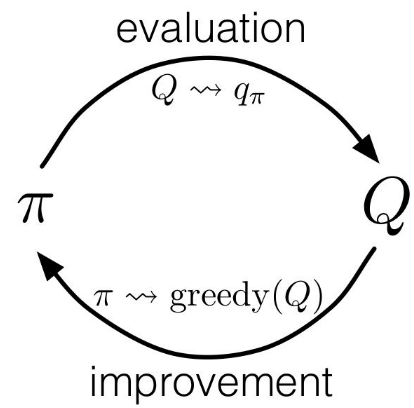
$\gamma : 0.9$

$$V(A) = V(A) + \alpha[r + \gamma V(B) - V(A)]$$

$$V(A) = 0.5 + 0.2[0 + 0.9 * 0.8 - 0.5] = 0.544$$

TD Control

- The control problem: approximate optimal policies
- Recall the idea of GPI:



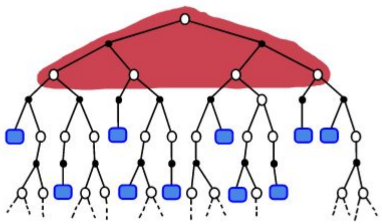
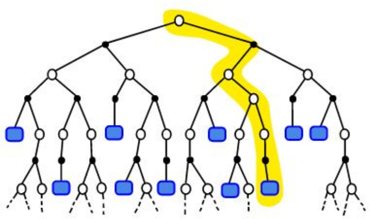
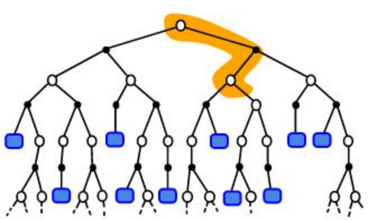
- Evaluation: use TD(0) to evaluate value function
- Improvement: make policy **greedy** wrt current value function
- Note that we estimate action values rather than state values in the absence of a model

ε-Greedy Policies

$$
\begin{aligned}
&A^* \leftarrow \arg\max_a Q(S_t, a)\\
&\text{For all } a \in \mathcal{A}(S_t)\text{:}\\
&\qquad \pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}
\end{aligned}
$$

- Any ε-greedy policy with respect to Q following π is an improvement over any ε-soft policy π is assured by the policy improvement theorem

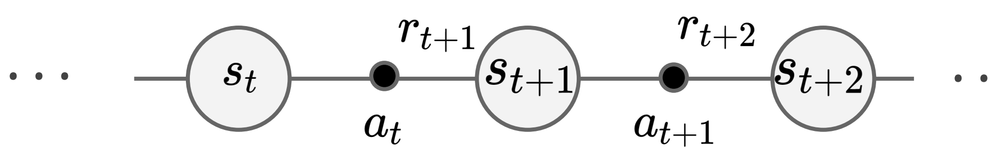
Summary: DP vs MC vs TD

		Bootstrapping?	Sampling?	Bias/Variance
DP		✅	❌	-
MC		❌	✅	Low Bias, High Variance
TD		✅	❌	High Bias, Low Variance

Sarsa

Bellman Equation

$$q_\pi(s,a) \doteq \sum_{s',r} p(s',r|s,a)\left[r + \gamma \sum_{a'} \pi(a'|s')\, q(s',a')\right]$$



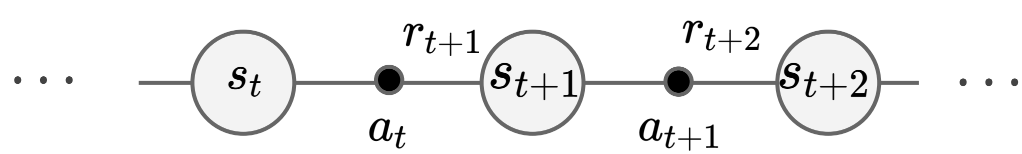
$$Q(S_t,A_t) \leftarrow Q(S_t,A_t) + \alpha\left[R_{t+1} + \gamma Q(S_{t+1},A_{t+1}) - Q(S_t,A_t)\right]$$

Temporal Difference Error

Sarsa: On-Policy TD Control

- On-policy control: Improve the behaviour policy

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

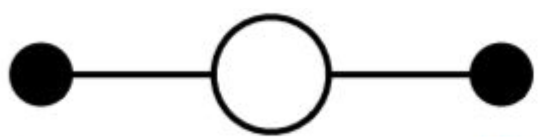


- Convergence is guaranteed as long as it is GLIE (Greedy in the Limit with Infinite Exploration)
  - all state-action pairs are visited an ∞ no. of times
  - the policy converges in the limit to the greedy policy

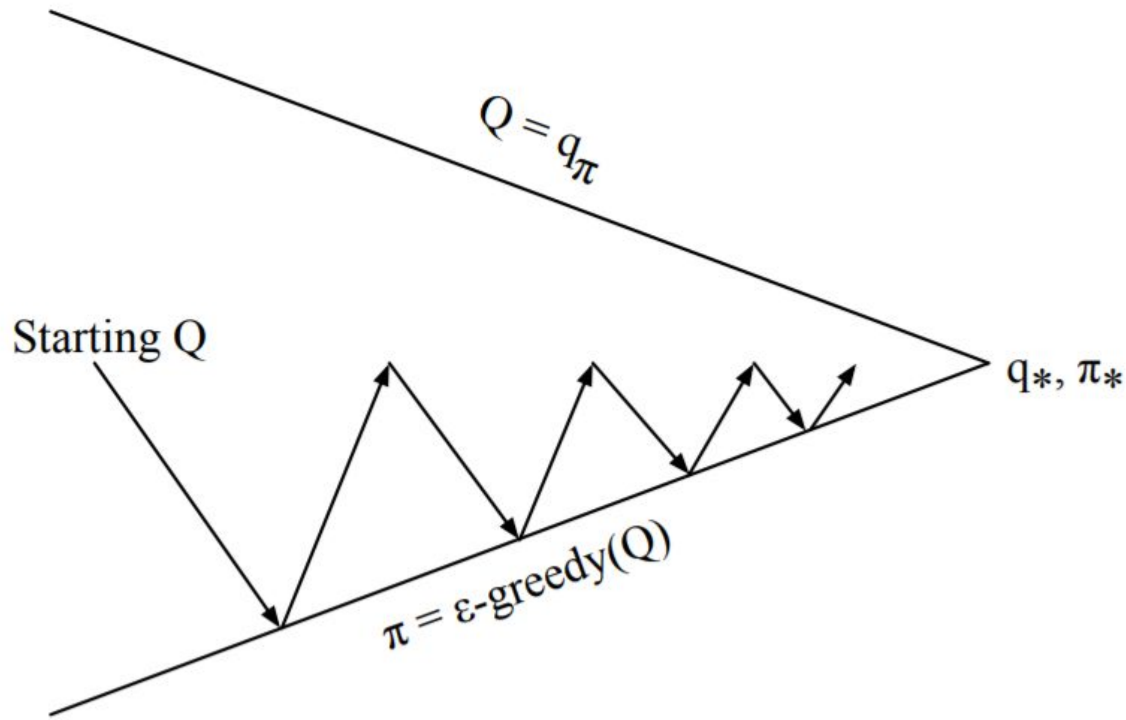
Sarsa: On-Policy TD Control

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
  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) IMPROVEMENT
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$ EVALUATION
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal



Sarsa: On-Policy TD Control



Every time-step:

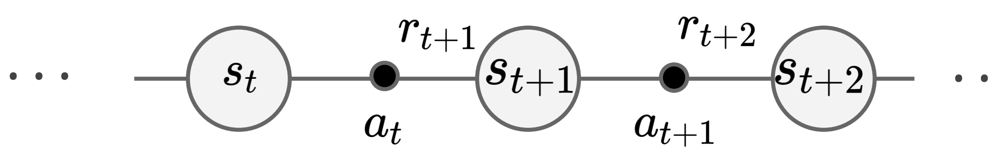
Policy evaluation Sarsa, $Q \approx q_\pi$

Policy improvement ϵ-greedy policy improvement

Q-Learning

Bellman Optimality Equation

$$q_*(s,a) \quad = \quad \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\Big|\; S_t = s, A_t = a\Big]$$



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

Temporal Difference
Error

Q-Learning: Off-Policy TD Control

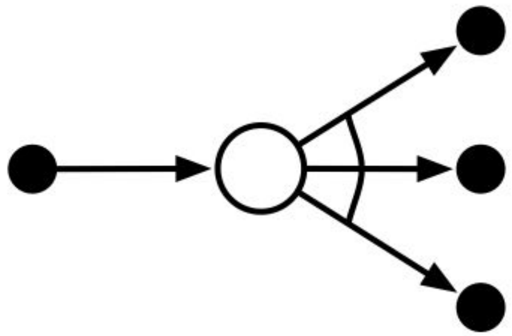
- In off-policy control, we have two policies:
  - behavior policy – used to generate behavior
  - estimation policy – the policy that is being evaluated and improved

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

Q-Learning: Off-Policy TD Control

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 $Q(terminal, \cdot) = 0$

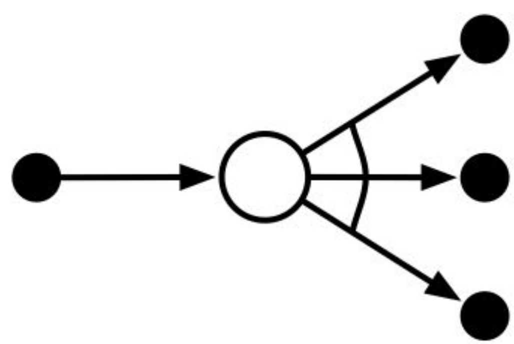
Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal



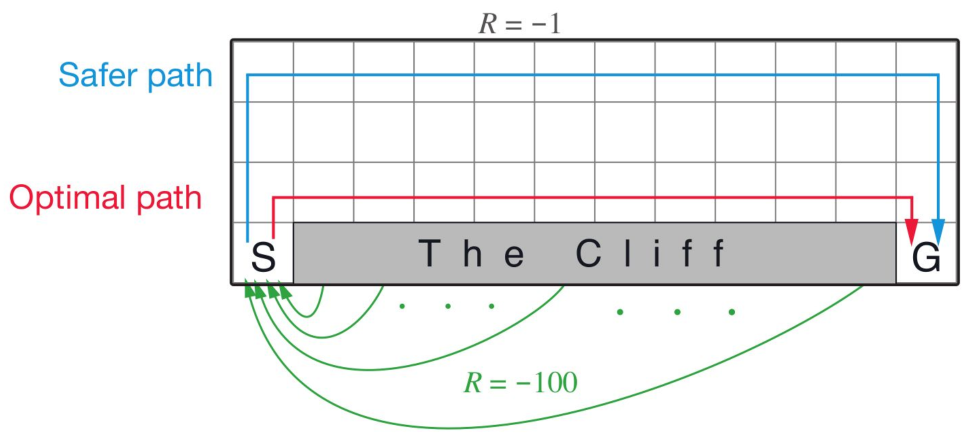
Q-Learning: Off-Policy TD Control

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 $Q(terminal, \cdot) = 0$

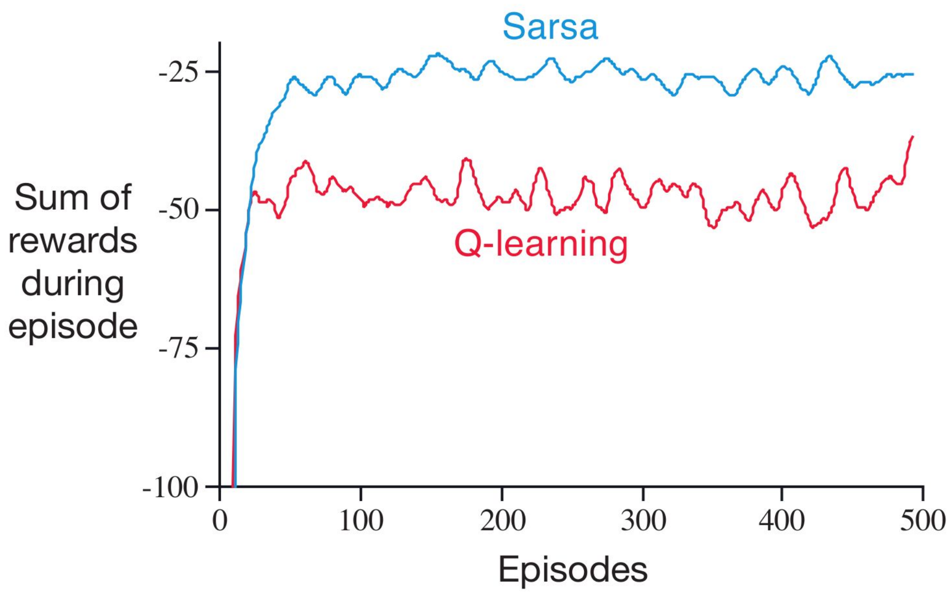
Loop for each episode:
    Initialize S
    Loop for each step of episode:
        Choose A from S ~~using policy derived from Q (e.g., ε-greedy)~~
        Take action A, observe R, S'
        $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \boxed{\gamma \max_a Q(S', a)} - Q(S, A)\big]$
        $S \leftarrow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



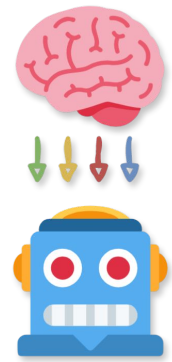
Cliff Walking: SARSA vs Q-learning



- R = -1 on all transitions except those into the region marked "The Cliff"
- Stepping into this region incurs a reward of -100 and the agent restarts



$\varepsilon = 0.1$



n - step Bootstrapping

n-Step TD Prediction

- Consider TD(0): $V(S_t) \leftarrow V(S_t) + \alpha\Big[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\Big]$

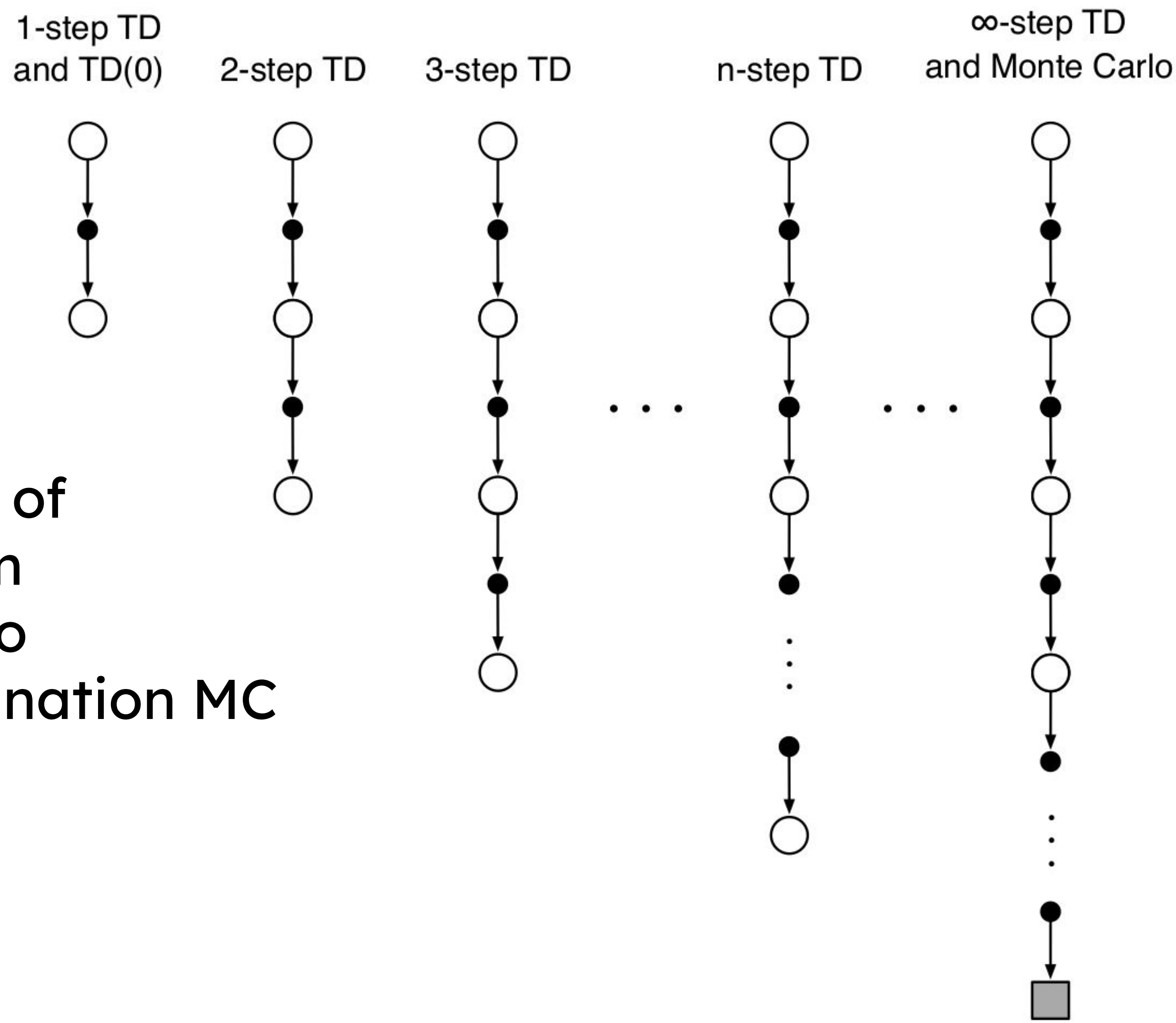
- Here, the target (for estimating TD error) contains only the next step reward: $G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$

- Alternatively, we can consider the rewards received in the next *n* steps:

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

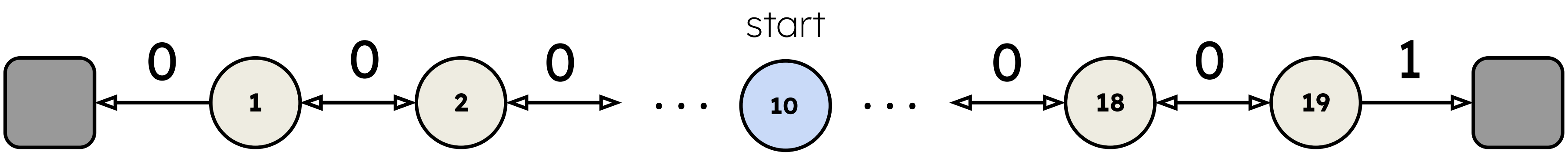
- The extreme would be to consider rewards till the end of the episode (Monte Carlo)

n-Step TD Prediction

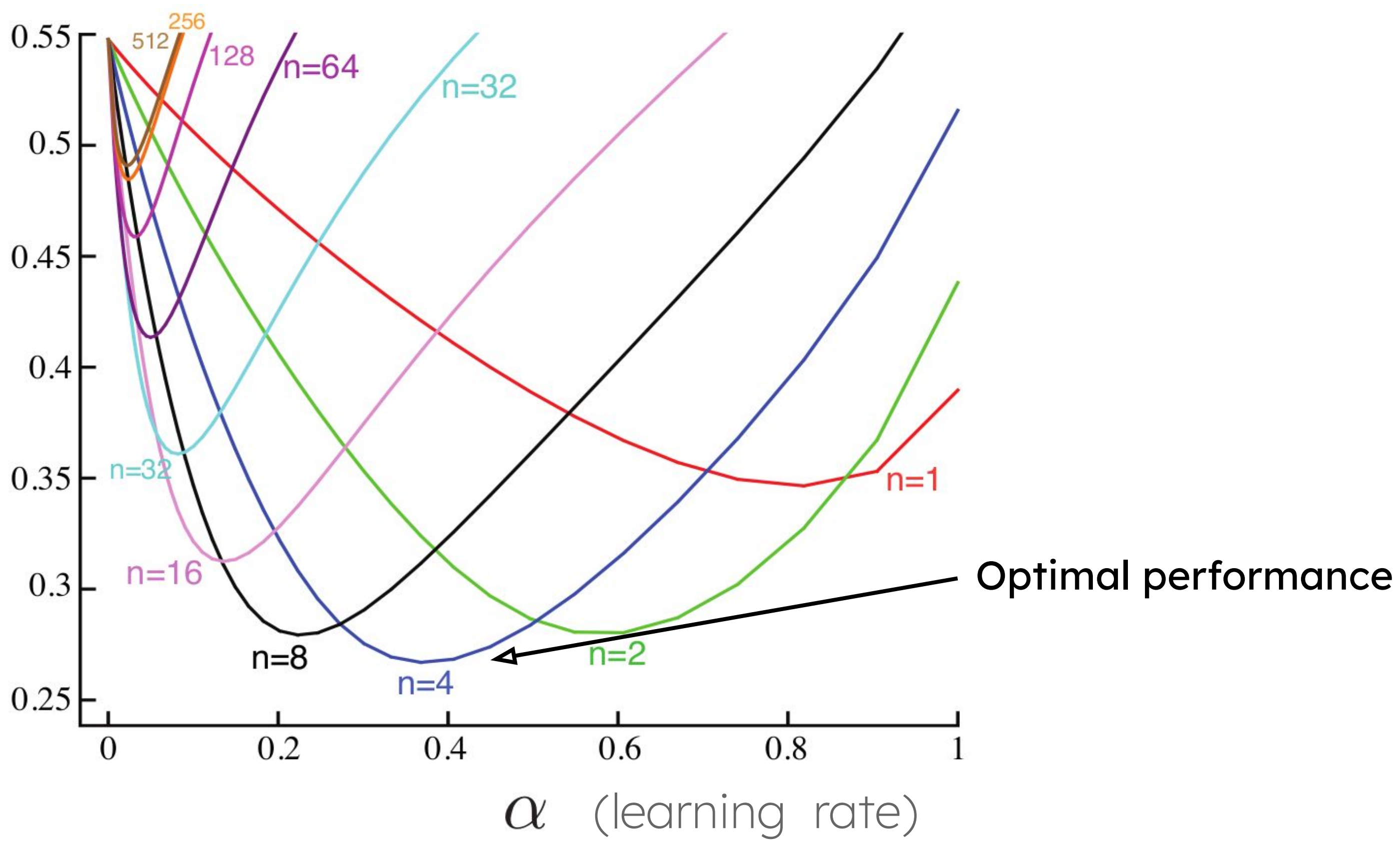


The spectrum of back-ups from one-step TD to up-until-termination MC

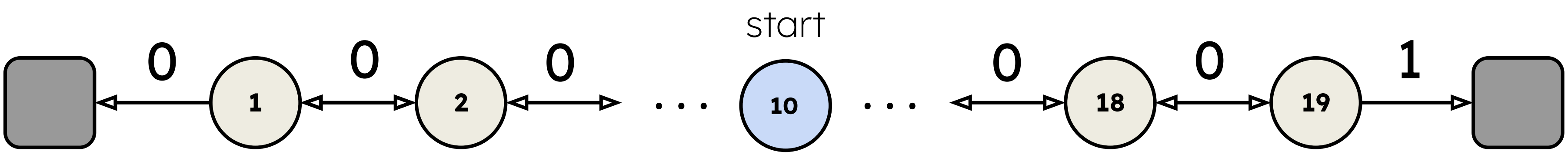
n-Step TD Prediction - Example

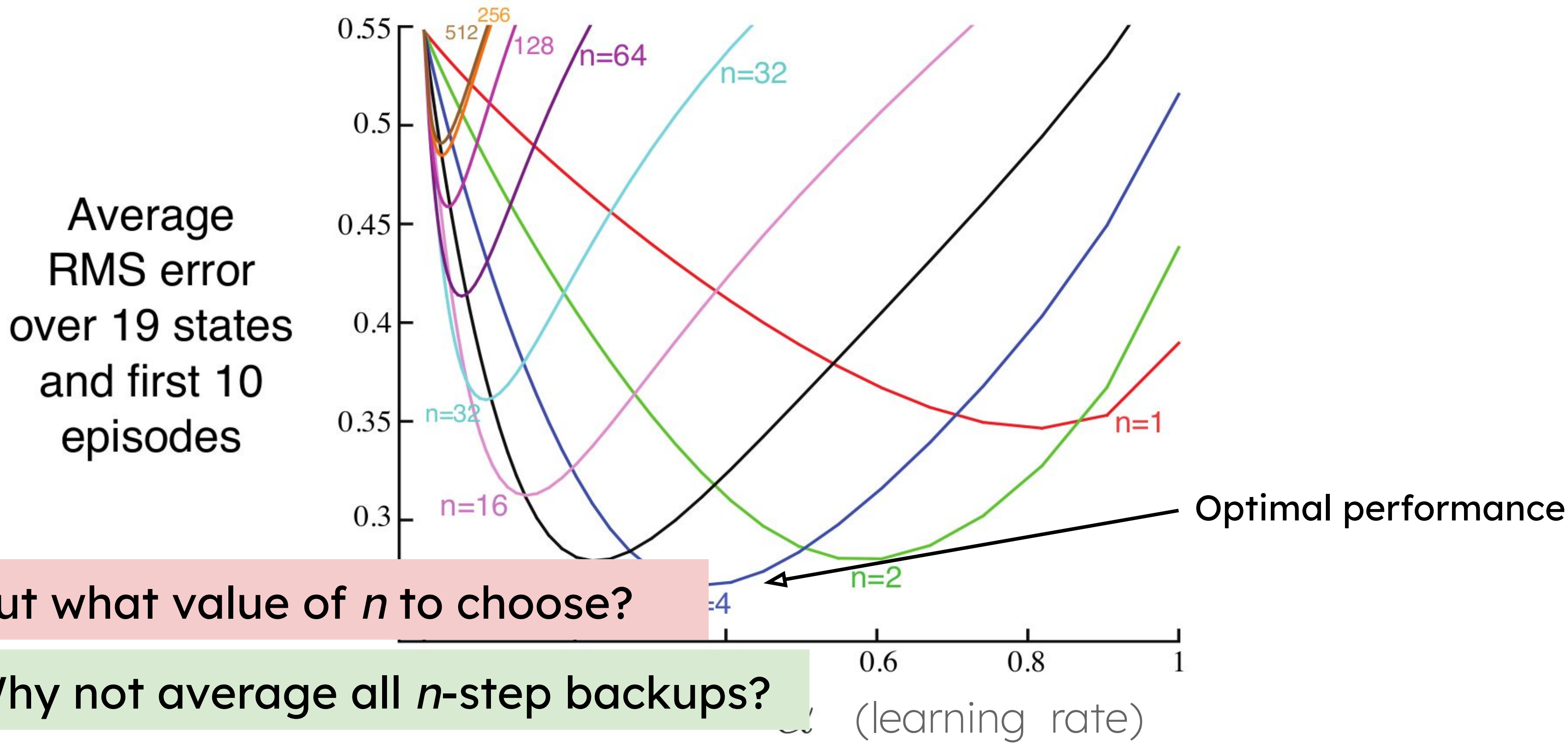


Average RMS error over 19 states and first 10 episodes



n-Step TD Prediction - Example





But what value of n to choose?

Why not average all n-step backups?

From *n*-Step TD to TD(λ)

- Instead of using *1 n-step backup*, we can consider an average of *multiple n-step backups*

- Example: $G_t^{avg} = \frac{1}{2} G_{t:t+10} + \frac{1}{2} G_{t:t+20}$

estimates of the same value - $V(s_t)$

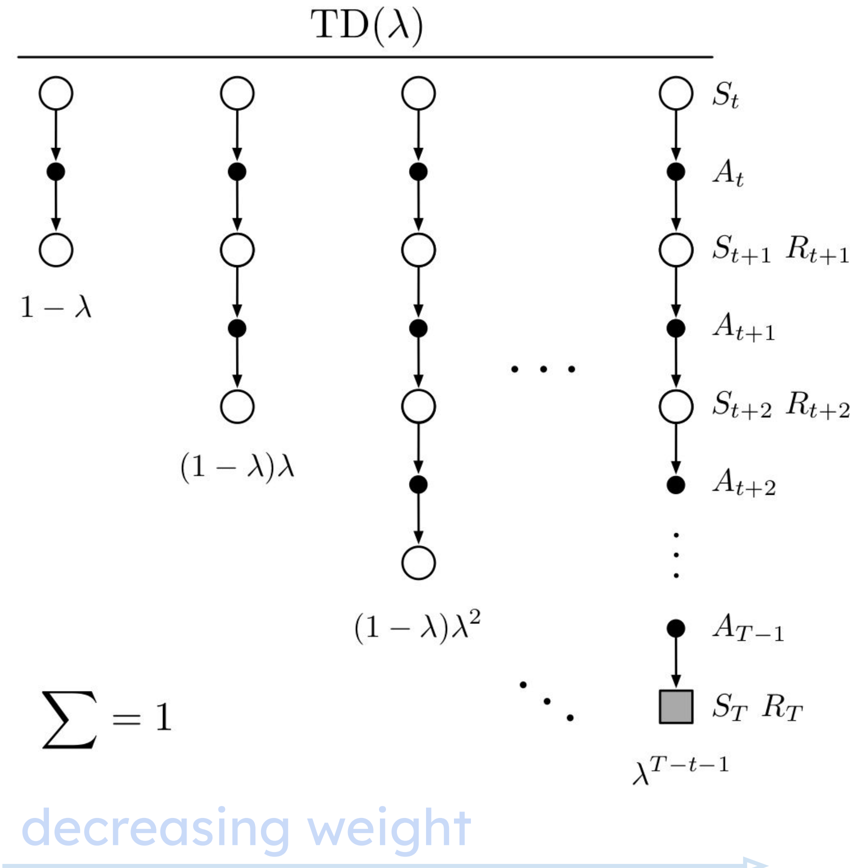
- In TD(λ), the average contains all the n-step backups each weighted proportional to λ^{n-1}, where $0 \leq \lambda \leq 1$.

TD(λ)

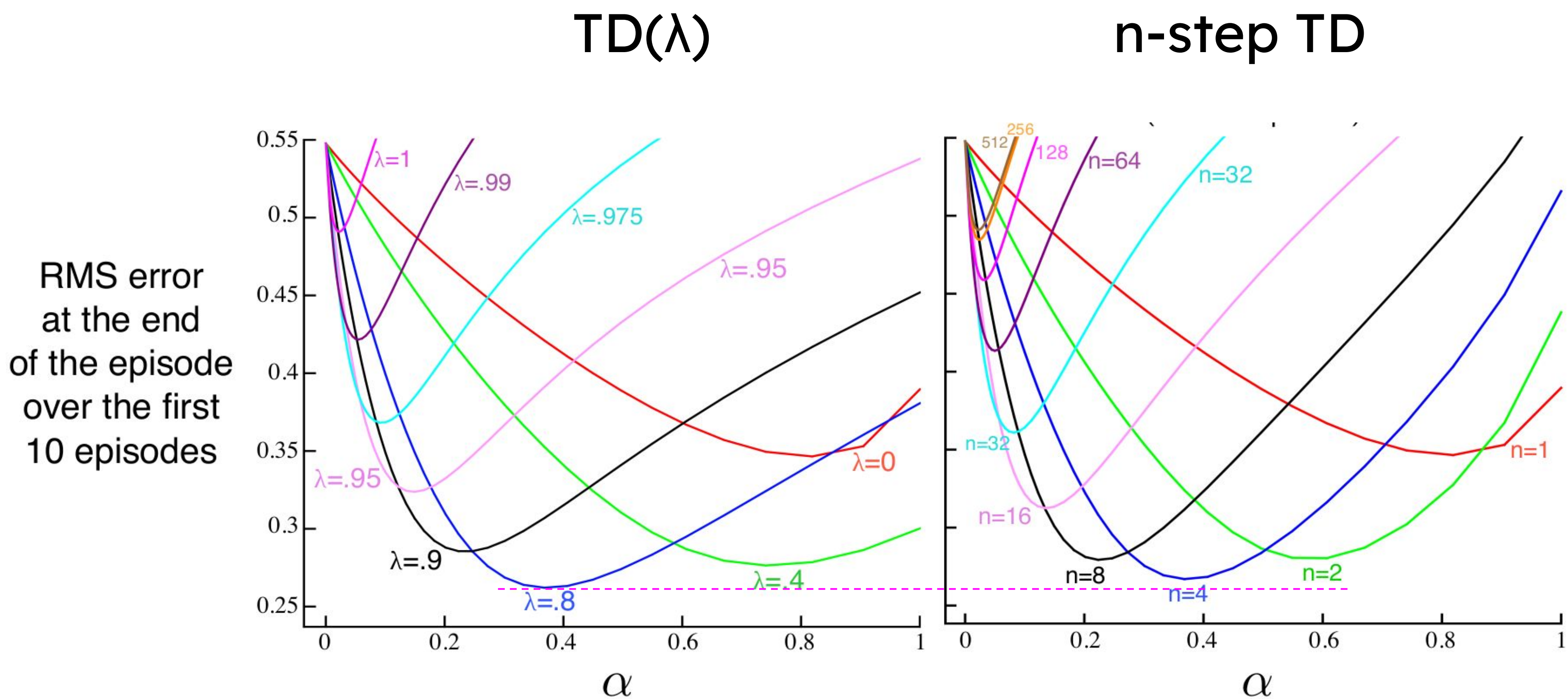
- In TD(λ), the average contains all the n-step backups each weighted proportional to λ^{n-1}, where $0 \le \lambda \le 1$.

- λ-return:

$$G_t^\lambda \doteq (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}$$



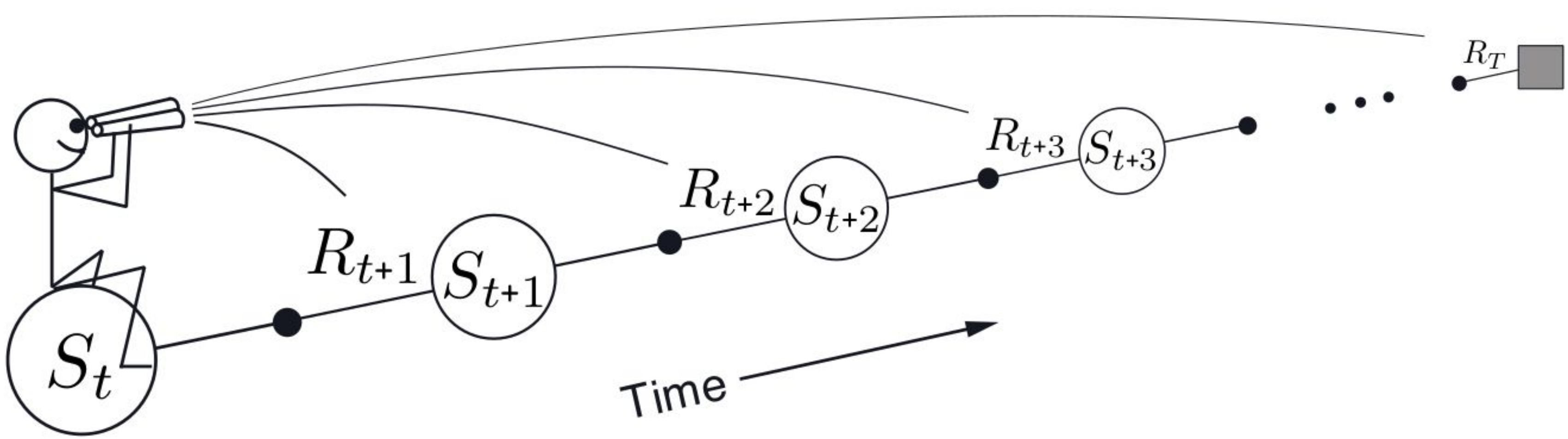
TD(λ) vs n-step TD



- The results with the TD(λ) are slightly better at the best values of α and λ, and at high α

TD(λ) - Forward View

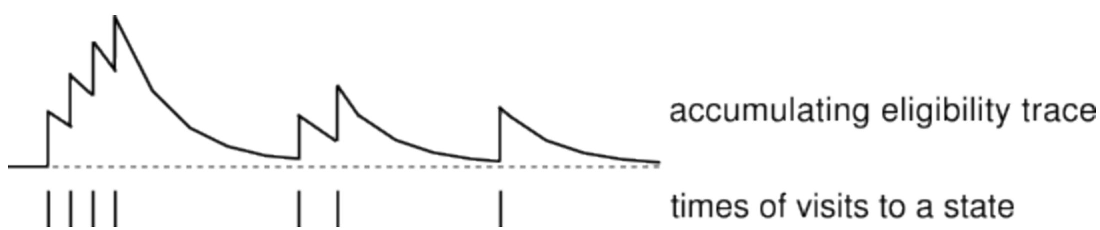
- TD(λ) as a *forward* - view algorithm
- Wait till the episode terminates
- Then for each state visited, we look *forward* in time to all the future rewards and combine them



Eligibility Traces

- To implement TD(λ) on-the-fly we use the concept of eligibility traces
- These are variables associated with each state denoted by $E_t(s)$
- They indicate the degree to which each state is eligible for undergoing learning changes

- On each step: $$E_t(s) = \begin{cases} \gamma\lambda E_{t-1}(s) & \text{if } s \neq S_t; \\ \gamma\lambda E_{t-1}(s) + 1 & \text{if } s = S_t, \end{cases}$$

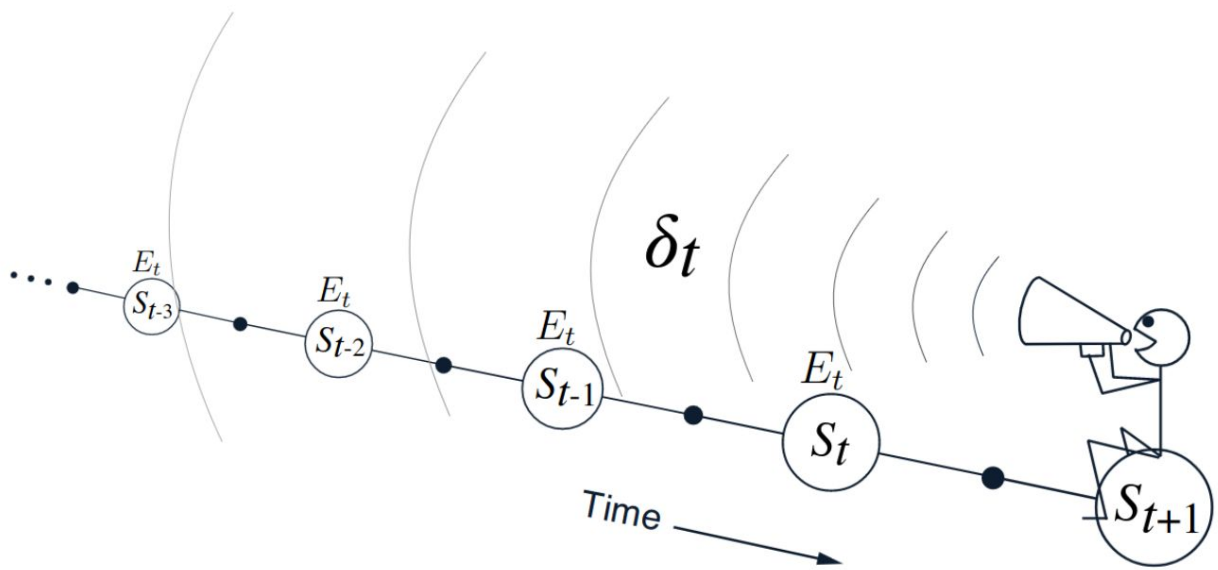


TD(λ) using Eligibility Traces

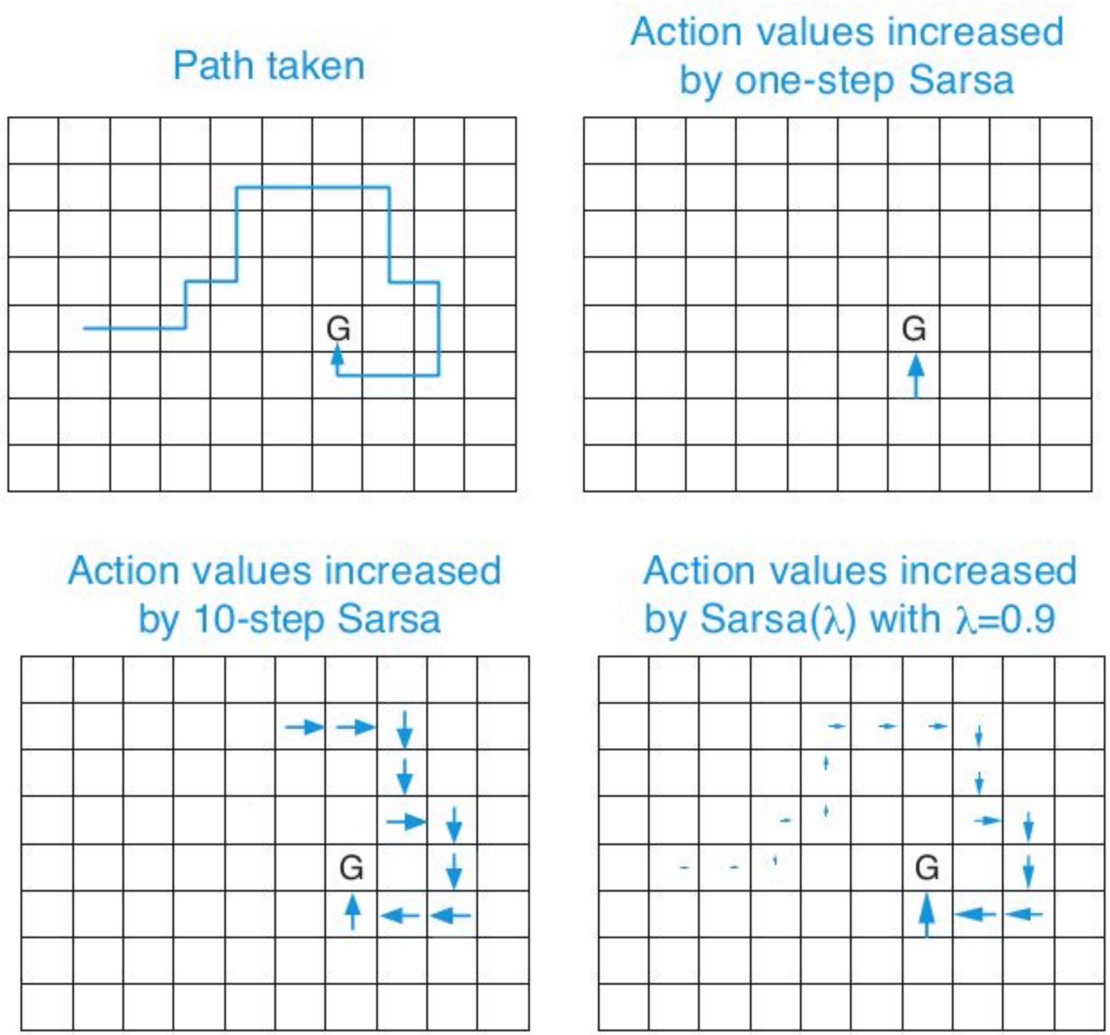
Initialize $V(s)$ arbitrarily (but set to 0 if s is terminal)
Repeat (for each episode):
  Initialize $E(s) = 0$, for all $s \in \mathcal{S}$
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe reward, R, and next state, S'
    $\delta \leftarrow R + \gamma V(S') - V(S)$
    $E(S) \leftarrow E(S) + 1$ (accumulating traces)
    For all $s \in \mathcal{S}$:
      $V(s) \leftarrow V(s) + \alpha \delta E(s)$
      $E(s) \leftarrow \gamma \lambda E(s)$
    $S \leftarrow S'$
  until S is terminal

TD(λ) - Backward View

- TD(λ) as a *backward* - view algorithm
- At each moment the current TD error is assigned it *backwards* to each prior state according to how eligible it is
- Need not wait for the episode to terminate!



Speedup in Policy Learning



TD(0) vs TD(λ)

Example grid world domain

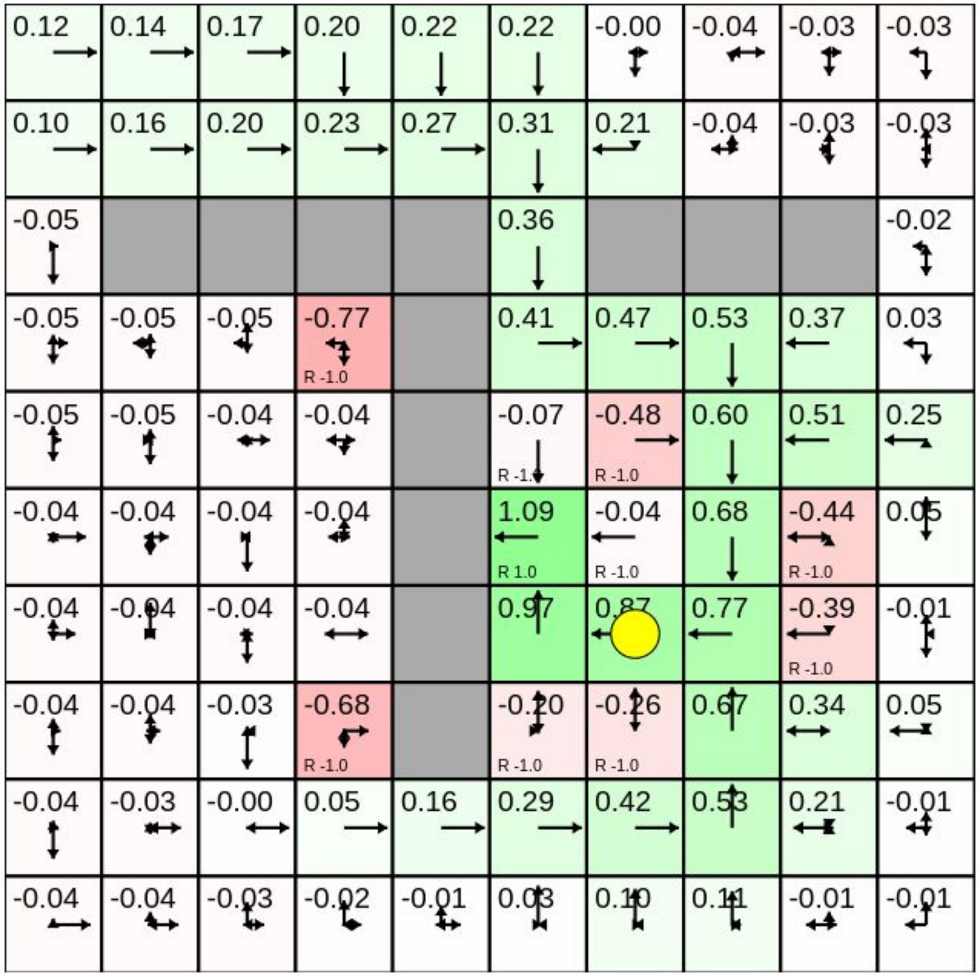
Grey cells = obstacles

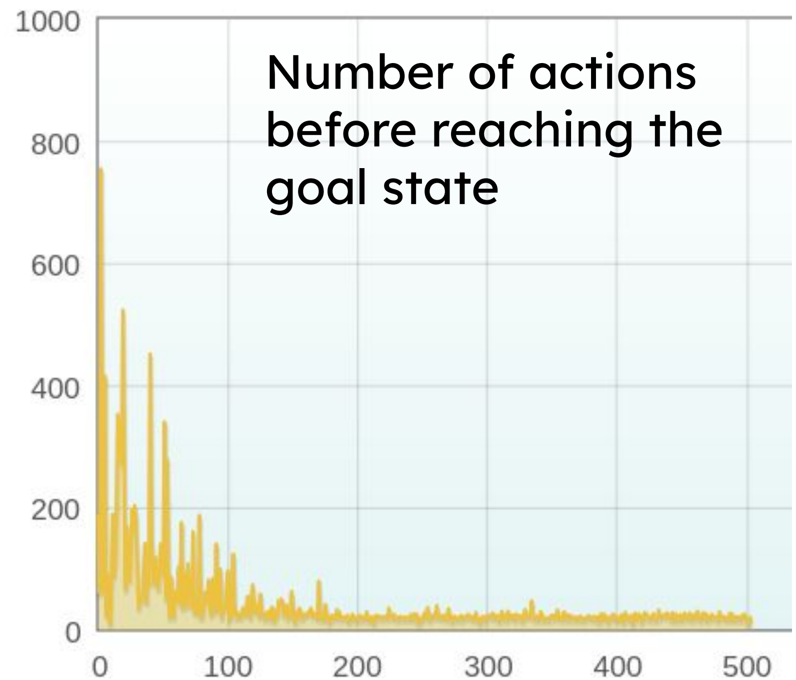
Notice R = 1.0 for centre cell with most surrounding cells having R = -1.0

Cell values = value function estimates

https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

TD(0) vs TD(λ)

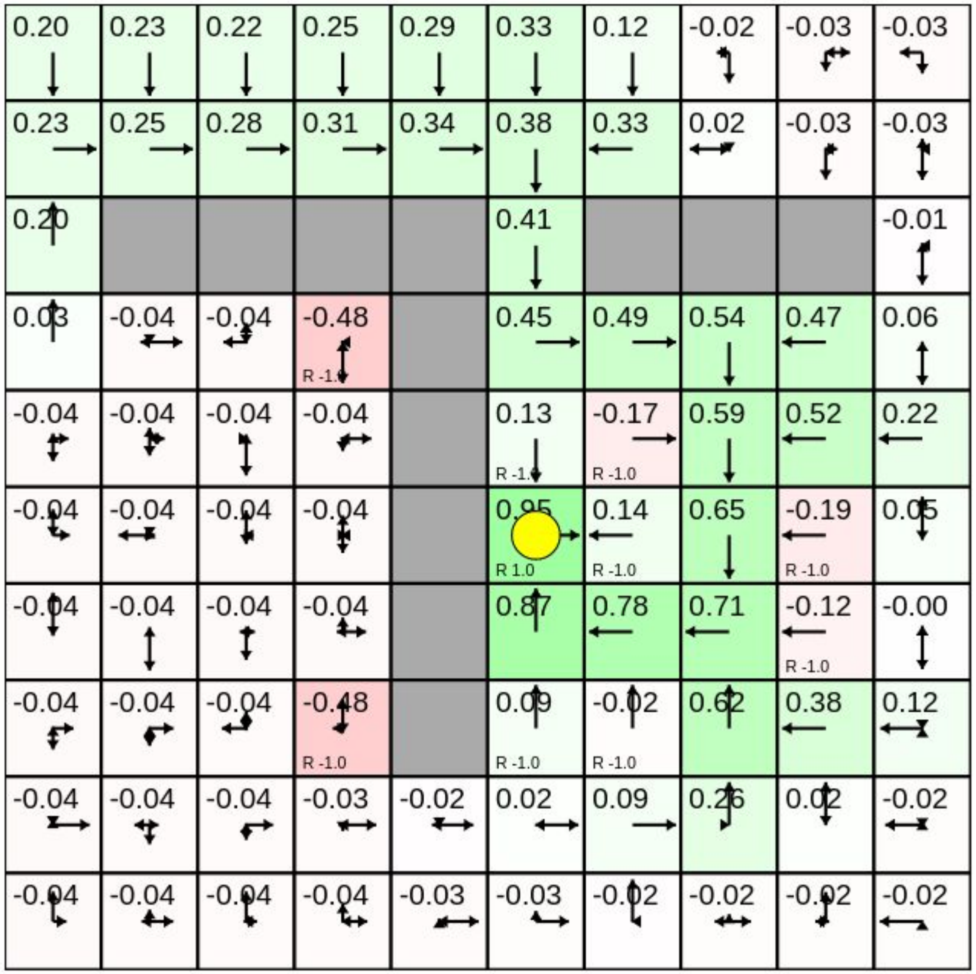


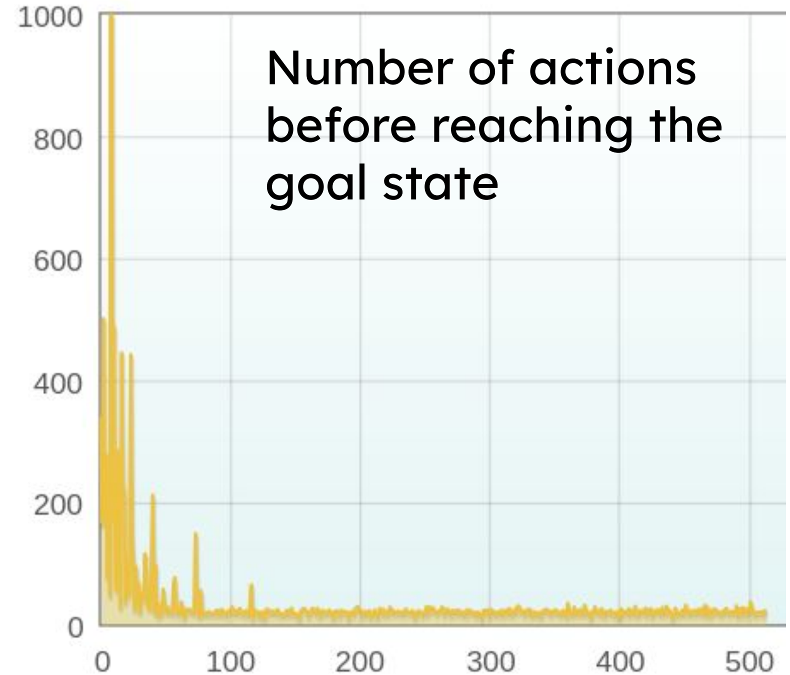


$\text{Algo}: \text{Q-learning}$

$\gamma = 0.9, \varepsilon = 0.2, \alpha = 0.1, \lambda = 0$

TD(0) vs TD(λ)

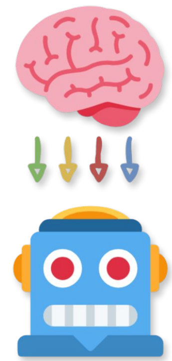




Number of actions before reaching the goal state

Algo : Q - learning

$\gamma = 0.9, \varepsilon = 0.2, \alpha = 0.1, \lambda = 0.2$



Importance Sampling

More on Off-Policy Learning

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s, a)$
- Follow behaviour policy $\mu(a|s)$
- Assumption of coverage:
  - Every action taken under π is also taken, at least occasionally, under μ
  - $\pi(a|s) > 0$ implies $\mu(a|s) > 0$
- e.g., π can be greedy while μ can be ε-greedy

Importance Sampling

- A general technique for estimating expected values under one distribution given samples from another.

$$
\begin{aligned}
\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X) f(X) \\
&= \sum Q(X) \frac{P(X)}{Q(X)} f(X) \\
&= \mathbb{E}_{X\sim Q}\left[\frac{P(X)}{Q(X)} f(X)\right]
\end{aligned}
$$

Off-Policy MC with Weighted Importance Sampling

Importance-sampling ratio:

$$\rho_t^T = \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} \mu(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}.$$

Weighted average return for *V* :

$$V(s) \doteq \frac{\sum_{t \in \mathcal{T}(s)} \rho_{t:T(t)-1} G_t}{|\mathcal{T}(s)|}$$