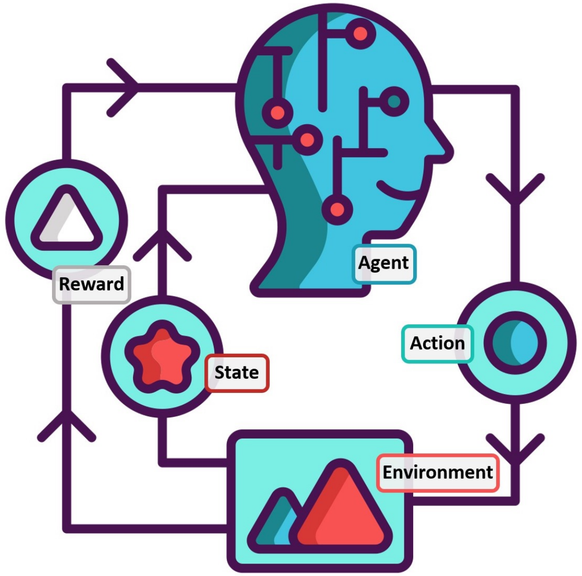
Reinforcement Learning 2025/2026

Lecture #09 Temporal Difference Learning

Gian Antonio Susto



Announcements before starting

- 1st partial exam list now open (Google form - no uniweb enrollment)

- Content for the 1st partial exam will end this week:

1. Lectures/slides: from lecture 1 to lecture 10
2. Book: from chapter 1 to chapter 6

Announcements before starting

- 1st partial exam list now open (Google form - no uniweb enrollment)

- Content for the 1st partial exam will end this week:
1. Lectures/slides: from lecture 1 to lecture 10
2. Book: from chapter 1 to chapter 6

- Next week:
1. Lecture on Wed. 5th of November: recap lecture! I will start preparing some materials based on your input! Send input, be prepared to ask questions!
2. Lecture on Thu. 6th of November: n-step bootstrapping + TD-lambda (content for 2nd partial)

Recap: TD learning vs MC (prediction)

The main difference between MC and TD is

- MC methods wait for the actual return G_t (that is available at the end of the episode) to update the estimation of $v_\pi(s)$

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$

- TD(0) uses the immediate reward (that is available after taking one action) to update the estimation of the so-called TD target

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

The TD target can be seen as the 'best' estimation of G_t at time t+1:

$$G_t = R_{t+1} + \gamma G(S_{t+1}) \sim R_{t+1} + \gamma V(S_{t+1})$$

Recap: TD learning vs MC (prediction)

The main difference between MC and TD is

- MC methods wait for the actual return G_t (that is available at the end of the episode) to update the estimation of $v_\pi(s)$

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

- This quantity is called the *TD error* available after taking one o-called TD target

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

The TD target can be seen as the 'best' estimation of G_t at time t+1:
$G_t = R_{t+1} + \gamma G(S_{t+1}) \sim R_{t+1} + \gamma V(S_{t+1})$

Recap: TD learning vs MC (prediction)

The main difference between MC

- MC methods wait for the actu
  end of the episode) to update

Today we'll also deal with the control problem: any ideas on how to approach it?

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

- available after taking one
  so-called TD target

This quantity is called the *TD error*

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

The TD target can be seen as the 'best' estimation of G_t at time t+1:
$G_t = R_{t+1} + \gamma G(S_{t+1}) \sim R_{t+1} + \gamma V(S_{t+1})$

Recap: TD learning vs MC (prediction)

The main difference between MC

- MC methods wait for the actu
end of the episode) to update

Today we'll also deal with the control problem: we'll consider Q instead of V

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$

- available after taking one
o-called TD target

This quantity is called the *TD error*

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

The TD target can be seen as the 'best' estimation of G_t at time t+1:

$G_t = R_{t+1} + \gamma G(S_{t+1}) \sim R_{t+1} + \gamma V(S_{t+1})$

Recap: TD learning vs MC (prediction)

What are the advantages/disadvantages of considering the TD error instead of the real return?

hat is available at the
on of $v_\pi(s)$

$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$

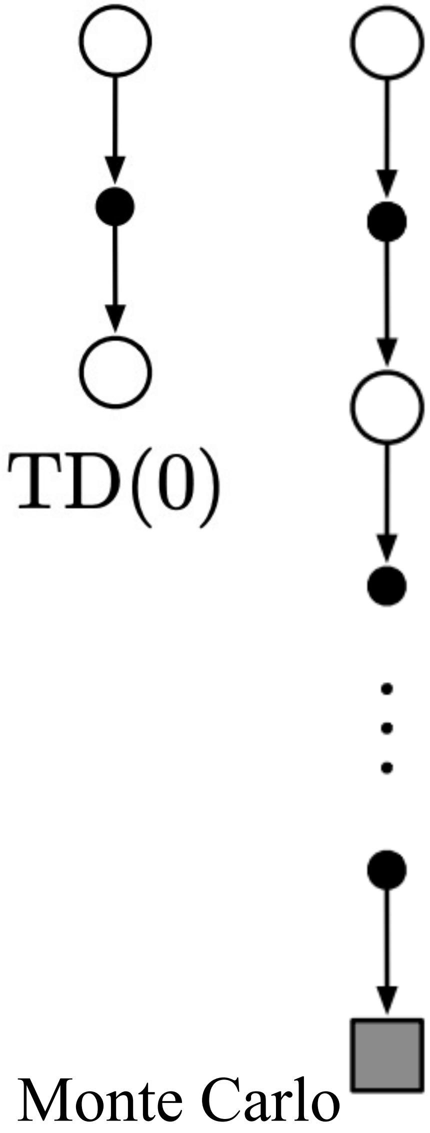
- This quantity is called the *TD error* available after taking one
action) to update the estimation of the so-called TD target

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

The TD target can be seen as the ‘best’ estimation of G_t at time t+1:
$G_t = R_{t+1} + \gamma G(S_{t+1}) \sim R_{t+1} + \gamma V(S_{t+1})$

Recap: TD learning vs MC (prediction)

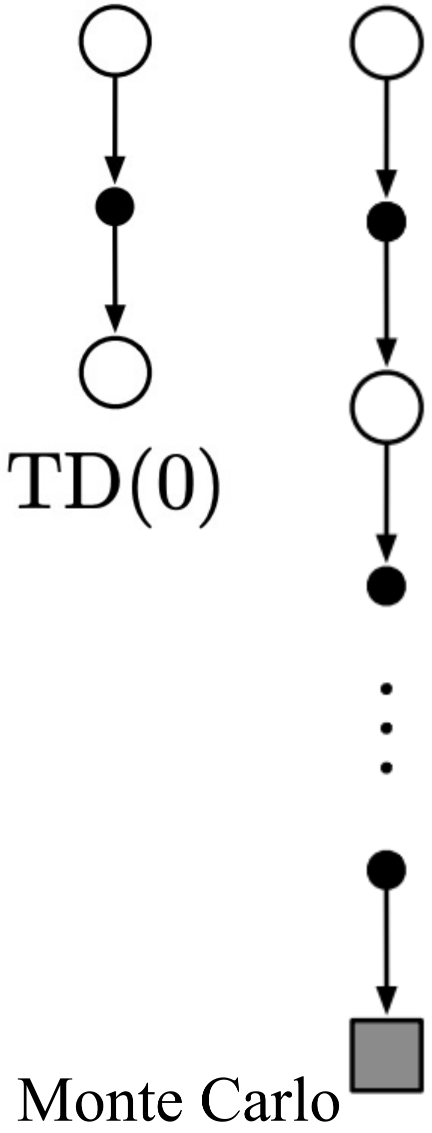
- TD methods update their estimates based on other estimates, ie. they bootstrap



Recap: TD learning vs MC (prediction)

- TD methods update their estimates based on other estimates, ie. they bootstrap
- TD can learn before knowing the final outcome:

TD can learn online after every step / MC must wait until end of episode before return is known

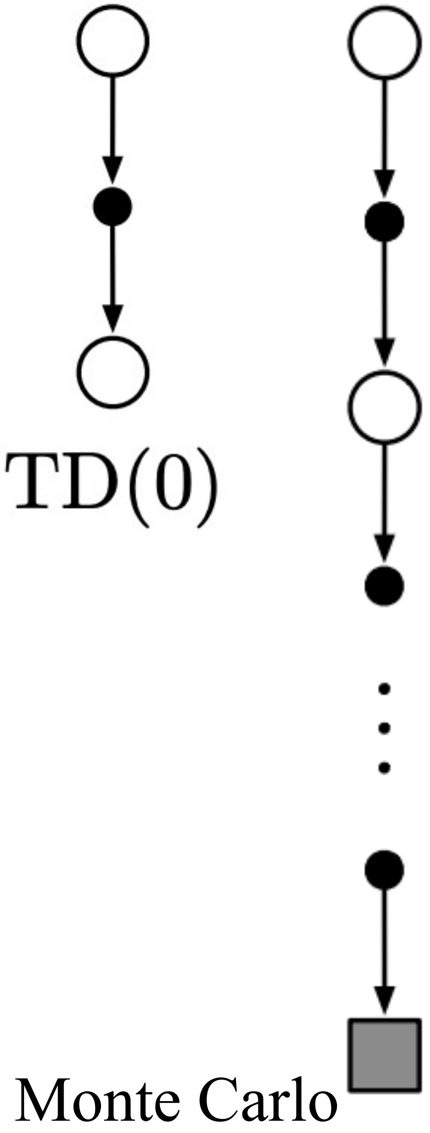


Recap: TD learning vs MC (prediction)

- TD methods update their estimates based on other estimates, ie. they bootstrap
- TD can learn before knowing the final outcome:

TD can learn online after every step / MC must wait until end of episode before return is known

- TD can learn without the final outcome
    i. TD can learn from incomplete sequences / MC can only learn from complete sequences
    ii. TD works in continuing (non-terminating) environments / MC only works for episodic (terminating) environments

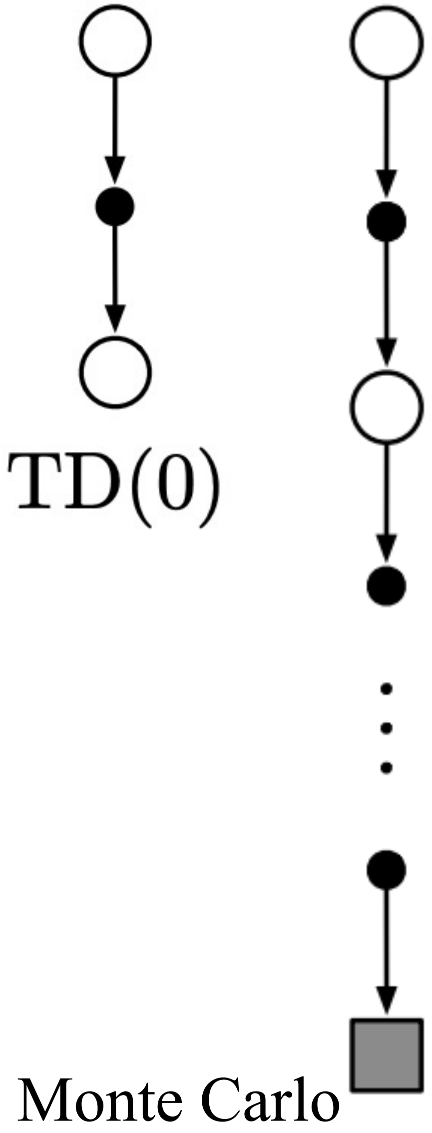


Recap: TD learning vs MC (prediction)

- TD methods update their estimates based on other estimates, ie. they bootstrap
- TD can learn before knowing the final outcome:

TD can learn online after every step / MC must wait until end of episode before return is known

- TD can learn without the final outcome
    i. TD can learn from incomplete sequences / MC can only learn from complete sequences
    ii. TD works in continuing (non-terminating) environments / MC only works for episodic (terminating) environments
- TD methods still converge to the right estimation of v_π



Recap: TD learning vs MC (prediction)

- TD methods update their estimates based on other estimates, ie. they bootstrap
- TD can learn before knowing the final outcome:

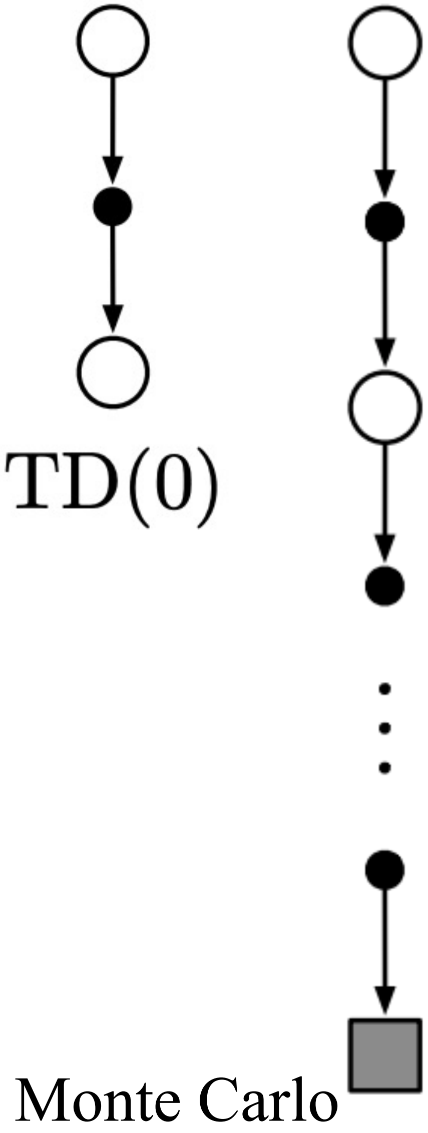
TD can learn online after every step / MC must wait until end of episode before return is known

- TD can learn without the final outcome

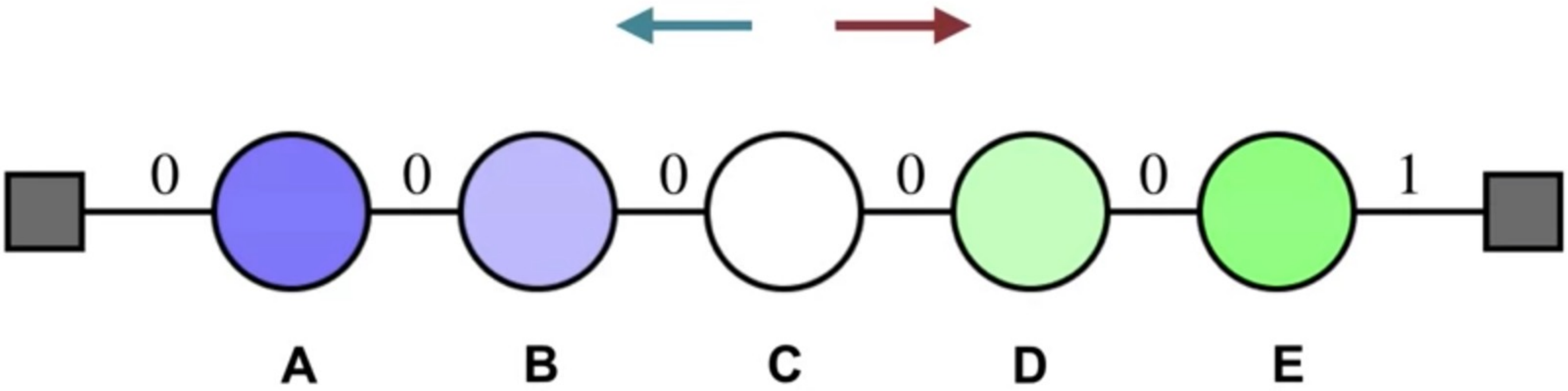
i. TD can learn from incomplete sequences / MC can only learn from complete sequences

ii. TD works in continuing (non-terminating) environments / MC only works for episodic (terminating) environments

- TD methods still converge to the right estimation of v_π
- Typically, TD approaches are faster to converge than MC!



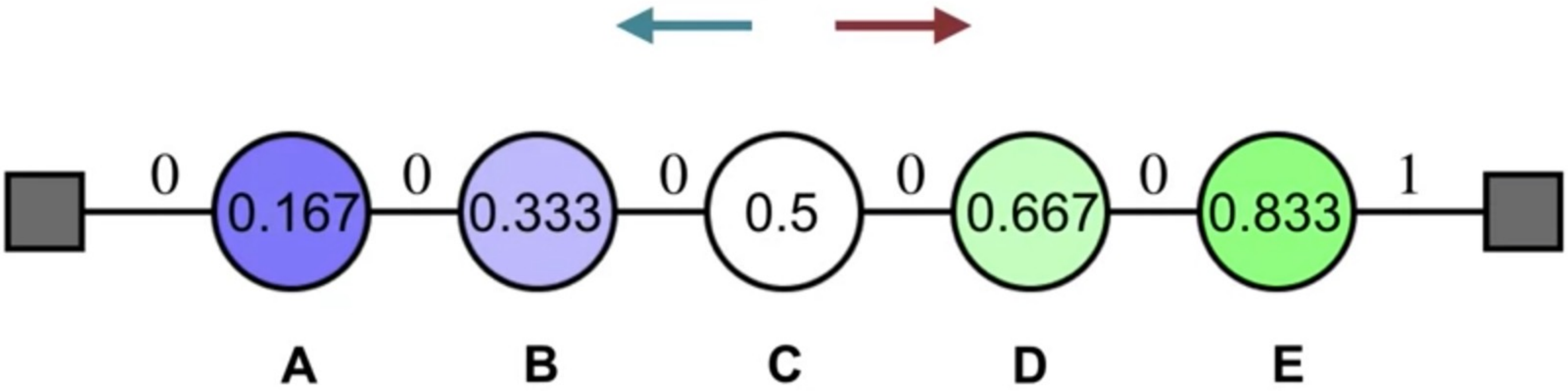
Prediction: TD(0) – Random Walk



$$\pi(\,.\,|\,s) = 1/2 \;\; \forall s \in \mathcal{S} \qquad\qquad \gamma = 1$$

A. White, M. White 'Sample-based Learning Methods'

Prediction: TD(0) – Random Walk

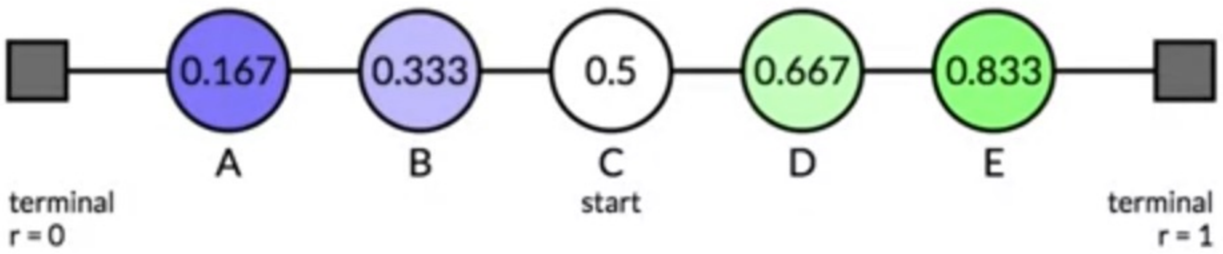


$$\pi(\,.\,|\,s) = 1/2 \;\; \forall s \in \mathcal{S} \qquad\qquad \gamma = 1$$

A. White, M. White 'Sample-based Learning Methods'

Episode #1

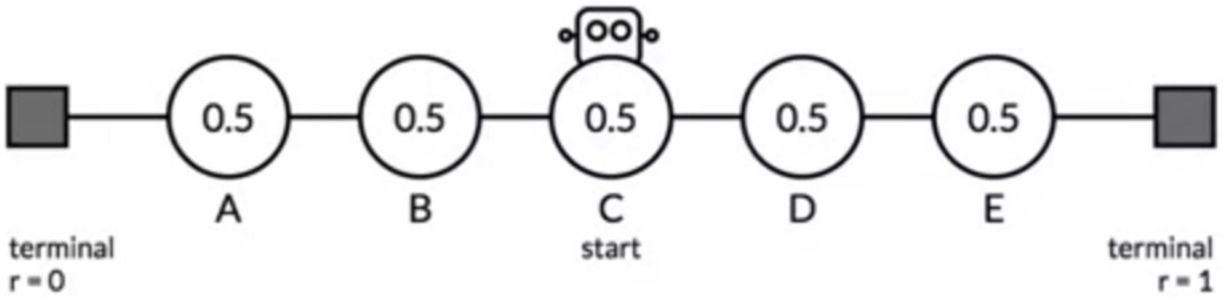
Target / Exact Values



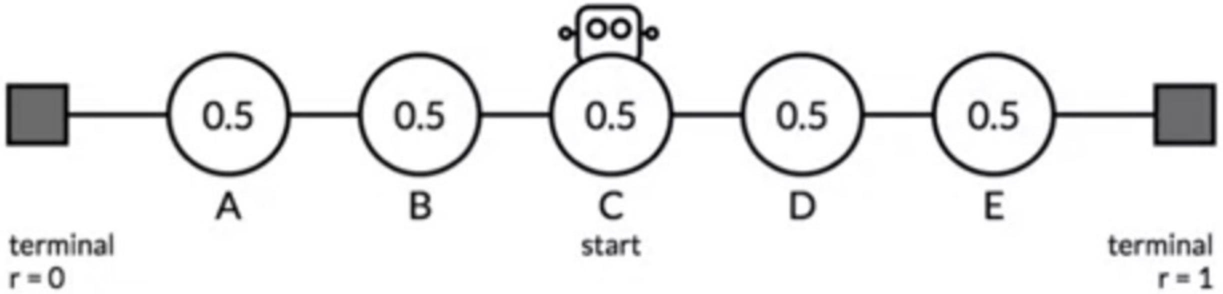
Updates using TD Learning

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

In the following:
$\alpha = 0.5$



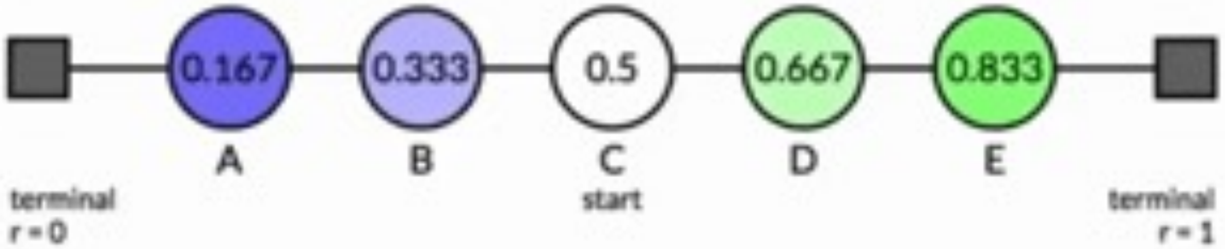
Updates using Monte Carlo



A. White, M. White 'Sample-based Learning Methods'

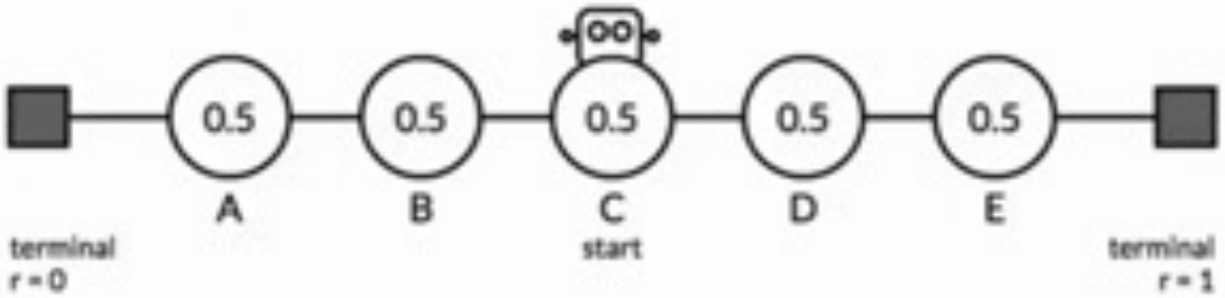
Episode #1

Target / Exact Values

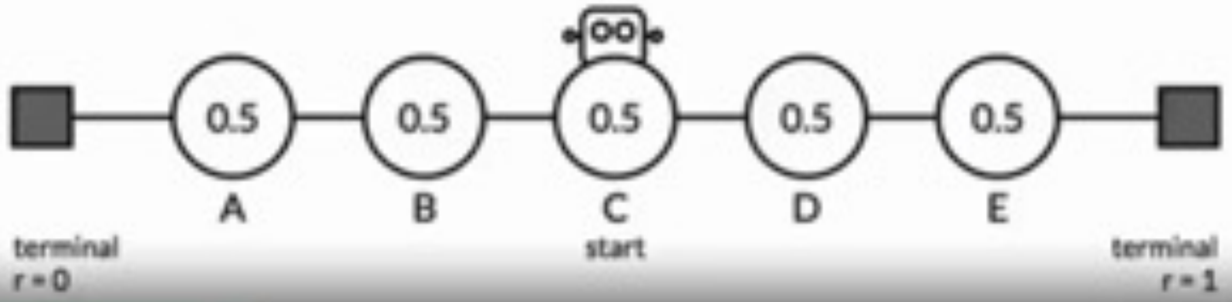


Updates using TD Learning

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



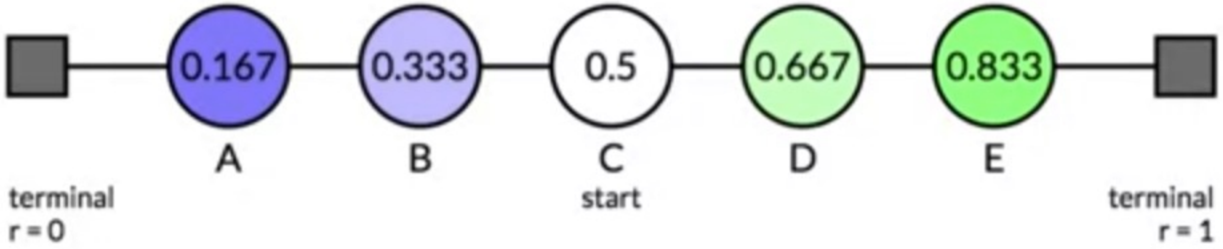
Updates using Monte Carlo



A. White, M. White 'Sample-based Learning Methods'

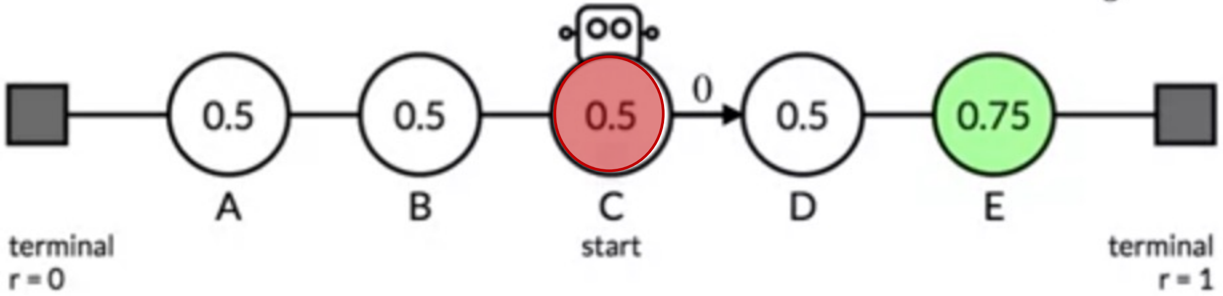
Episode #1

Target / Exact Values

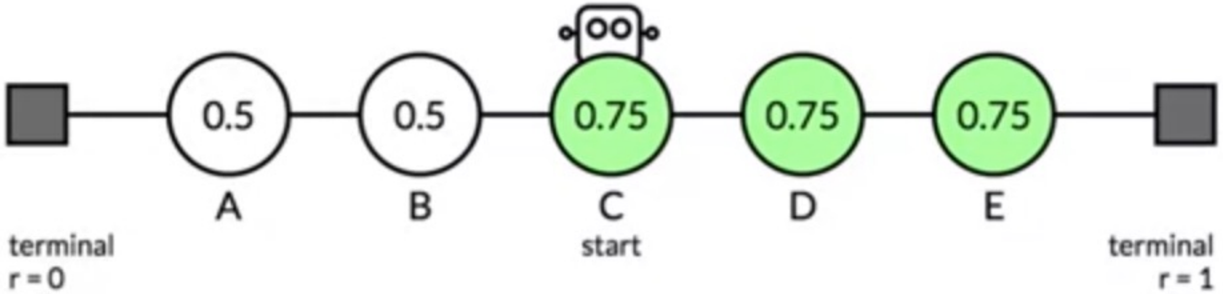


Updates using TD Learning

$$V(S_t) \leftarrow V(S_t) + \alpha[\underbrace{R_{t+1}}_{0} + \gamma \underbrace{V(S_{t+1})}_{0.5} - \underbrace{V(S_t)}_{0.5}]$$



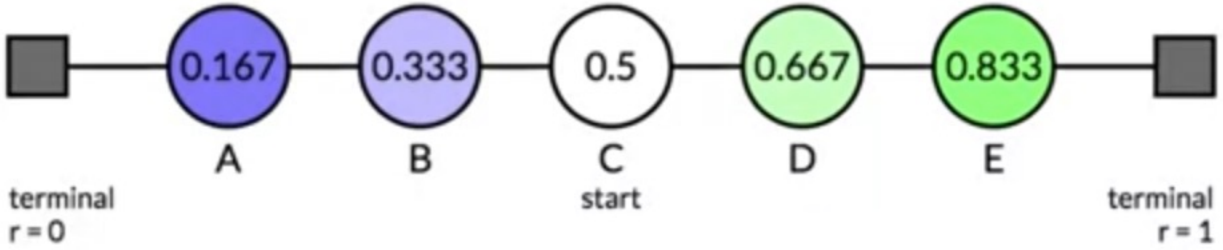
Updates using Monte Carlo



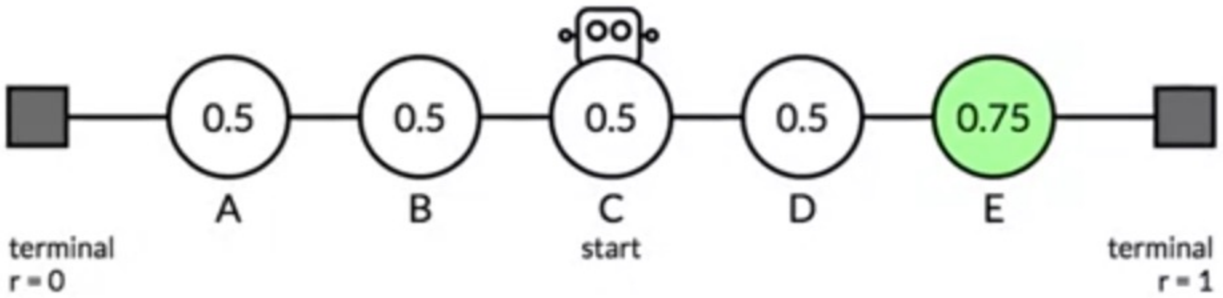
A. White, M. White 'Sample-based Learning Methods'

Episode #1

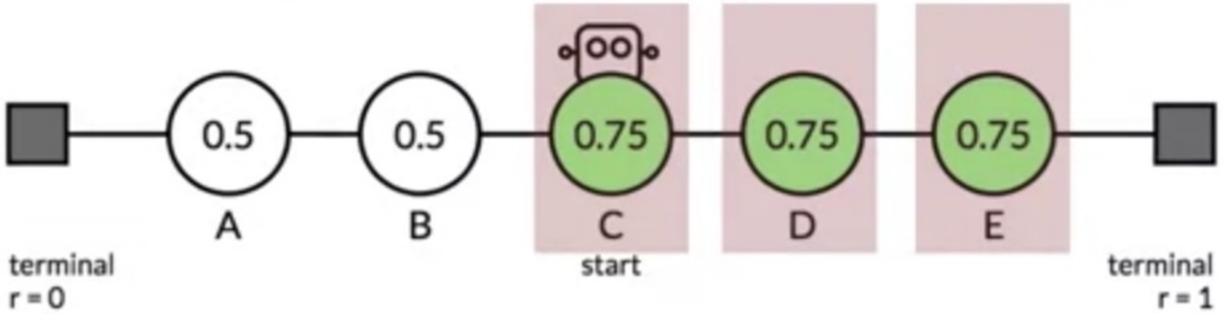
Target / Exact Values



Updates using TD Learning $V(S_t) \leftarrow V(S_t) + \alpha[\, R_{t+1} + \gamma V(S_{t+1}) \; - V(S_t)]$

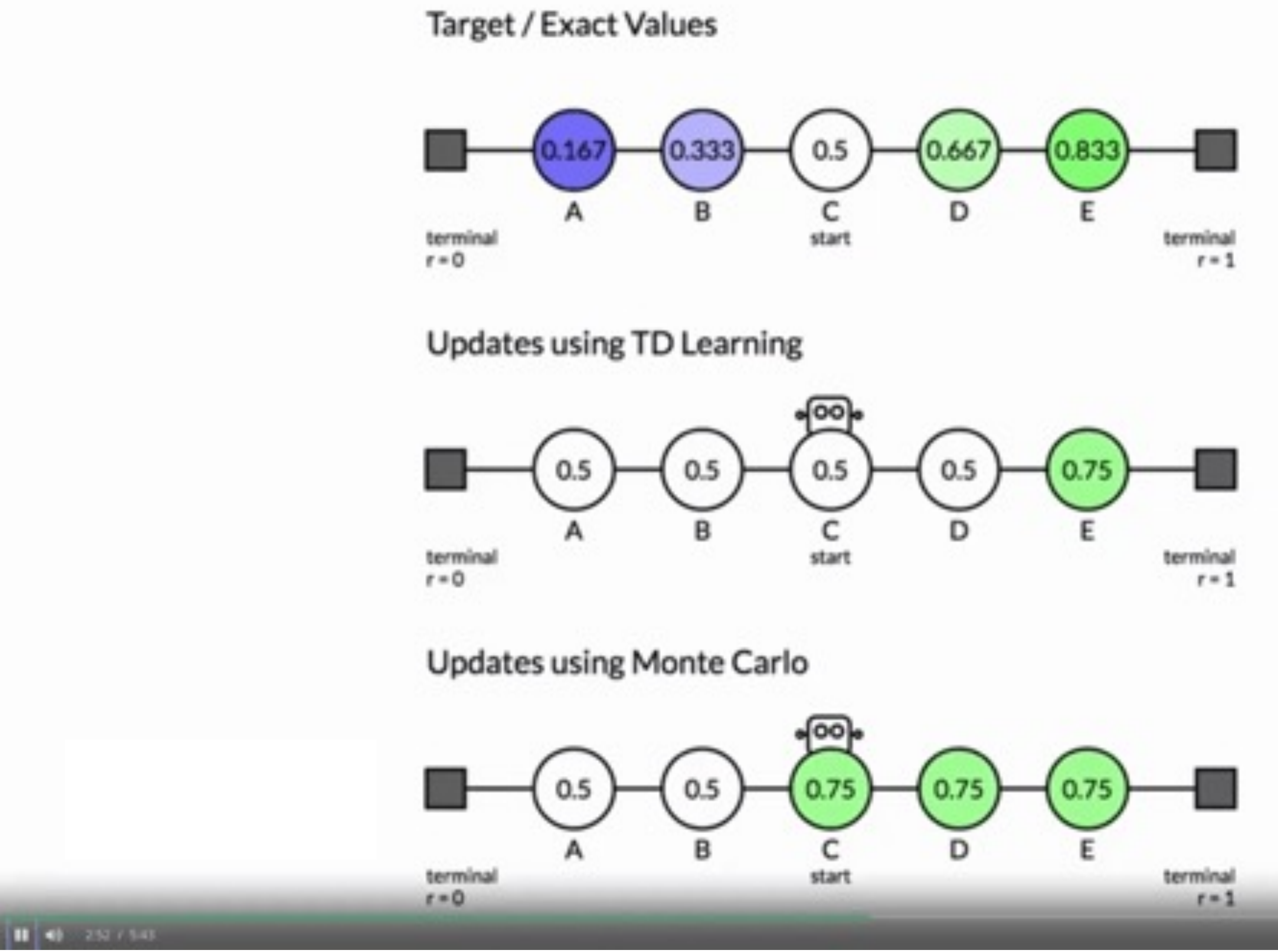


Updates using Monte Carlo $V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$



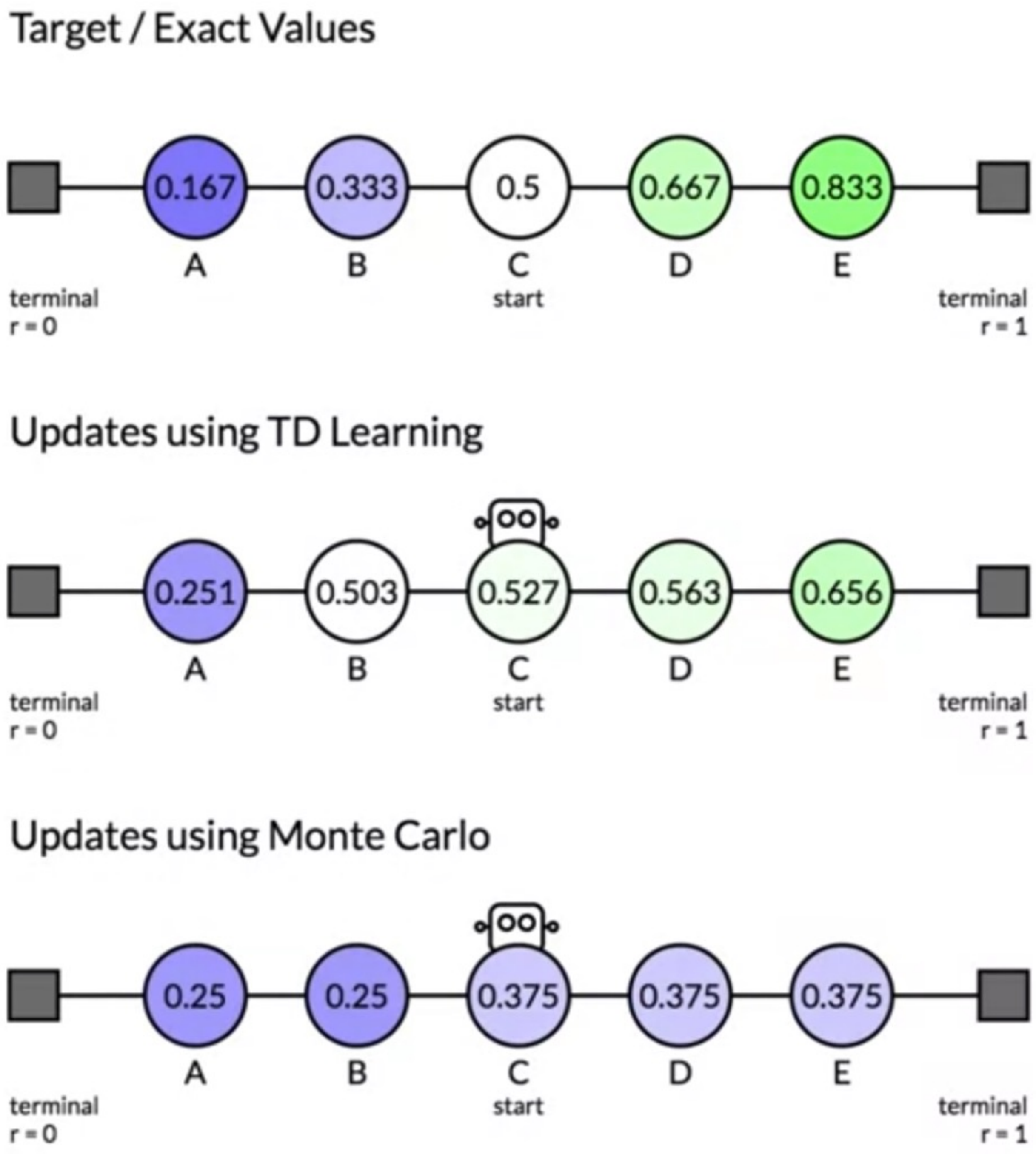
A. White, M. White 'Sample-based Learning Methods'

Episode #2



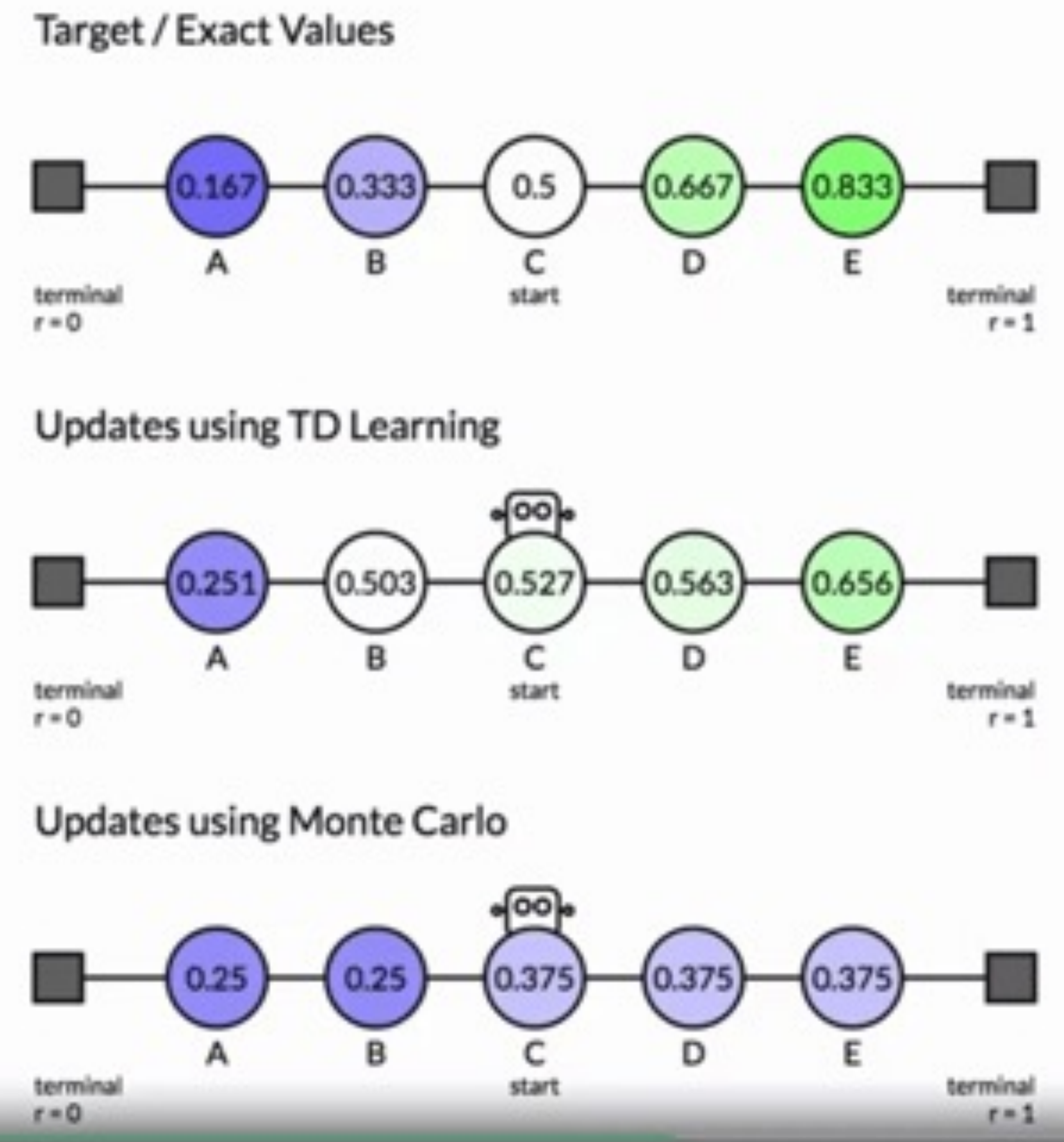
A. White, M. White 'Sample-based Learning Methods'

Episode #2



A. White, M. White 'Sample-based Learning Methods'

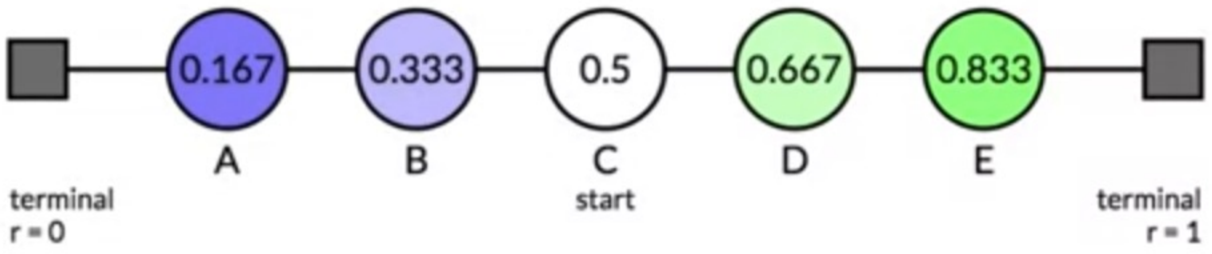
Additional Episodes



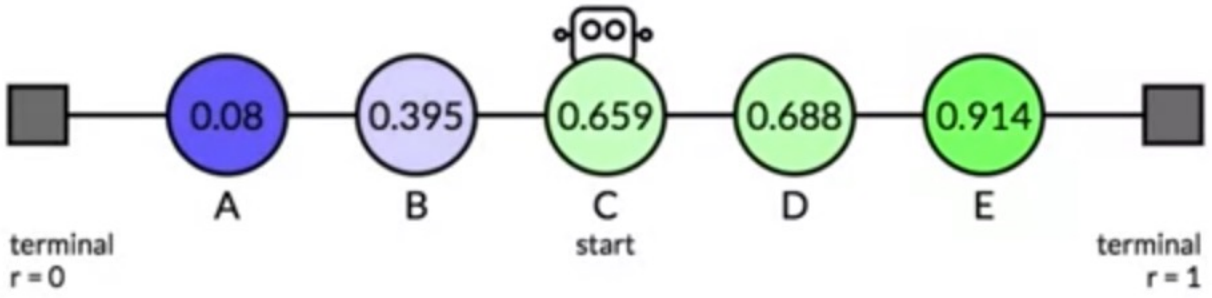
A. White, M. White 'Sample-based Learning Methods'

Additional Episodes

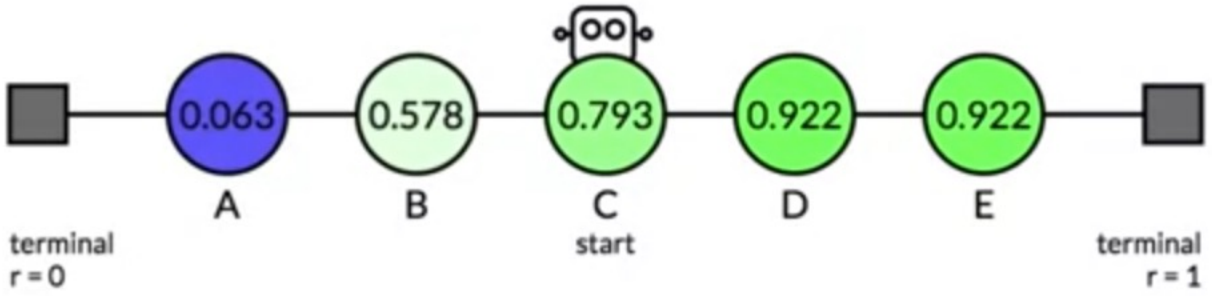
Target / Exact Values



Updates using TD Learning

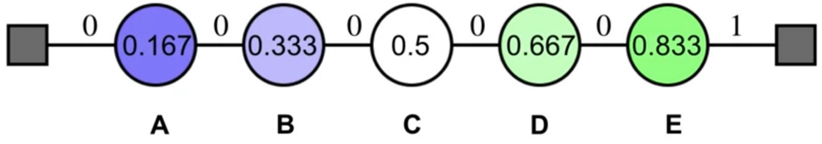


Updates using Monte Carlo

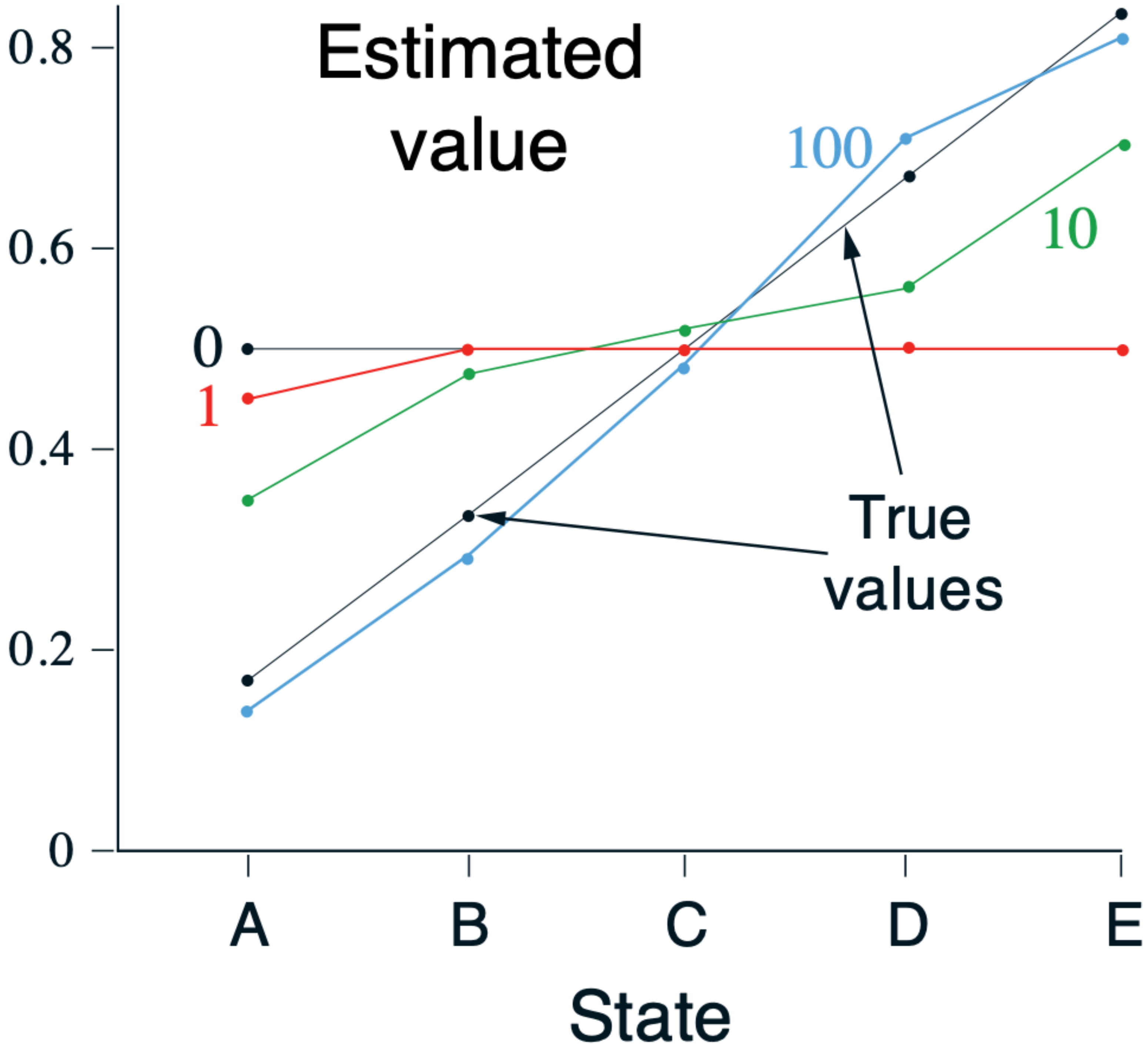


A. White, M. White 'Sample-based Learning Methods'

Recap Prediction: TD(0) – Random Walk

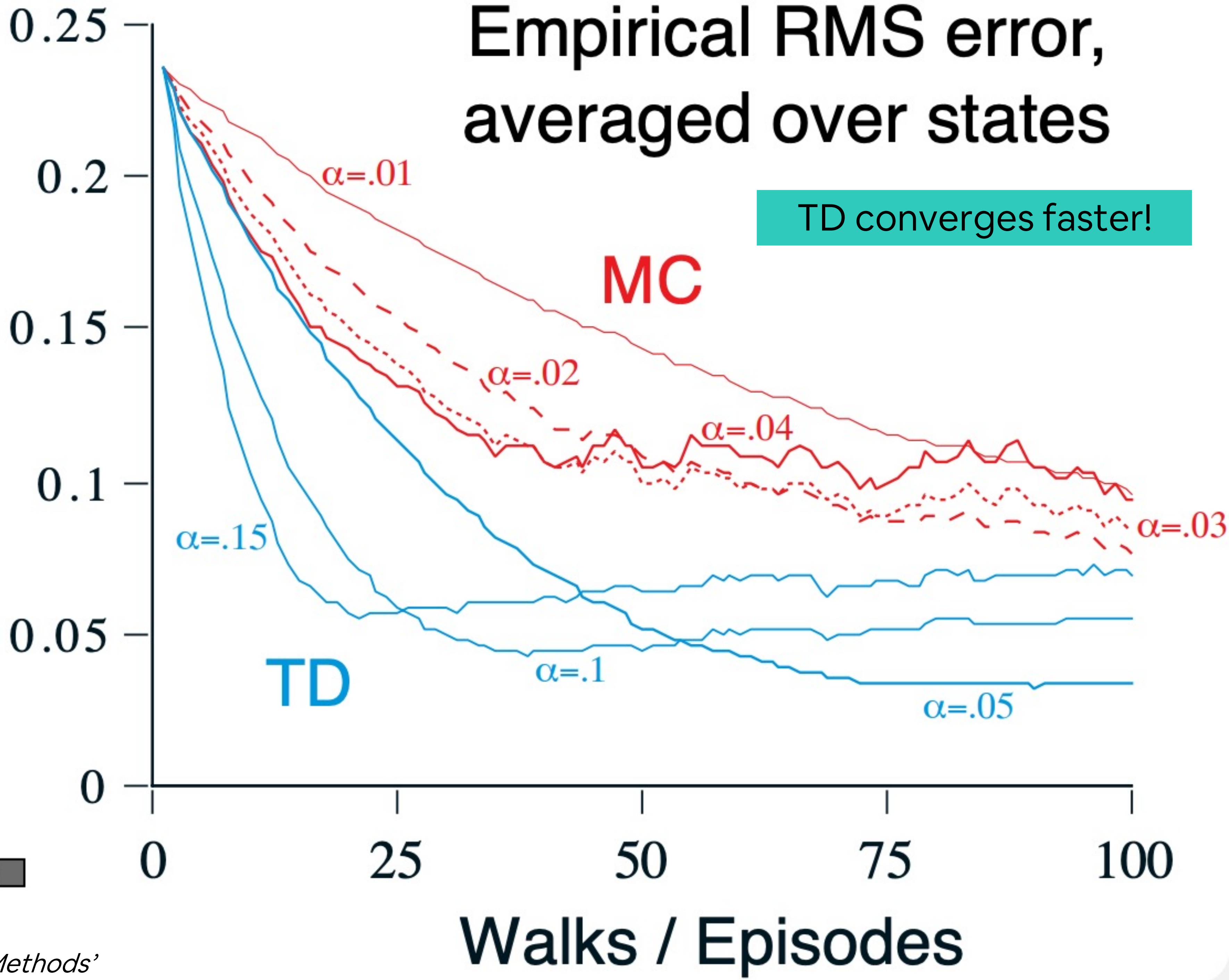


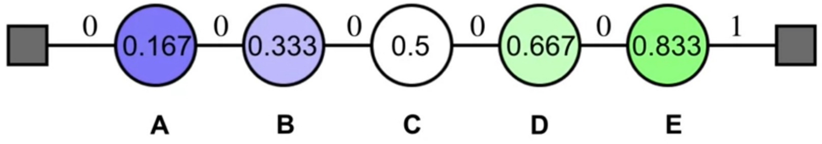
A. White, M. White 'Sample-based Learning Methods'



Recap Prediction: TD(0) – Random Walk

Estimation error - Root Mean Square (RMS) – averaged over the 5 states, the averaged over 100 runs

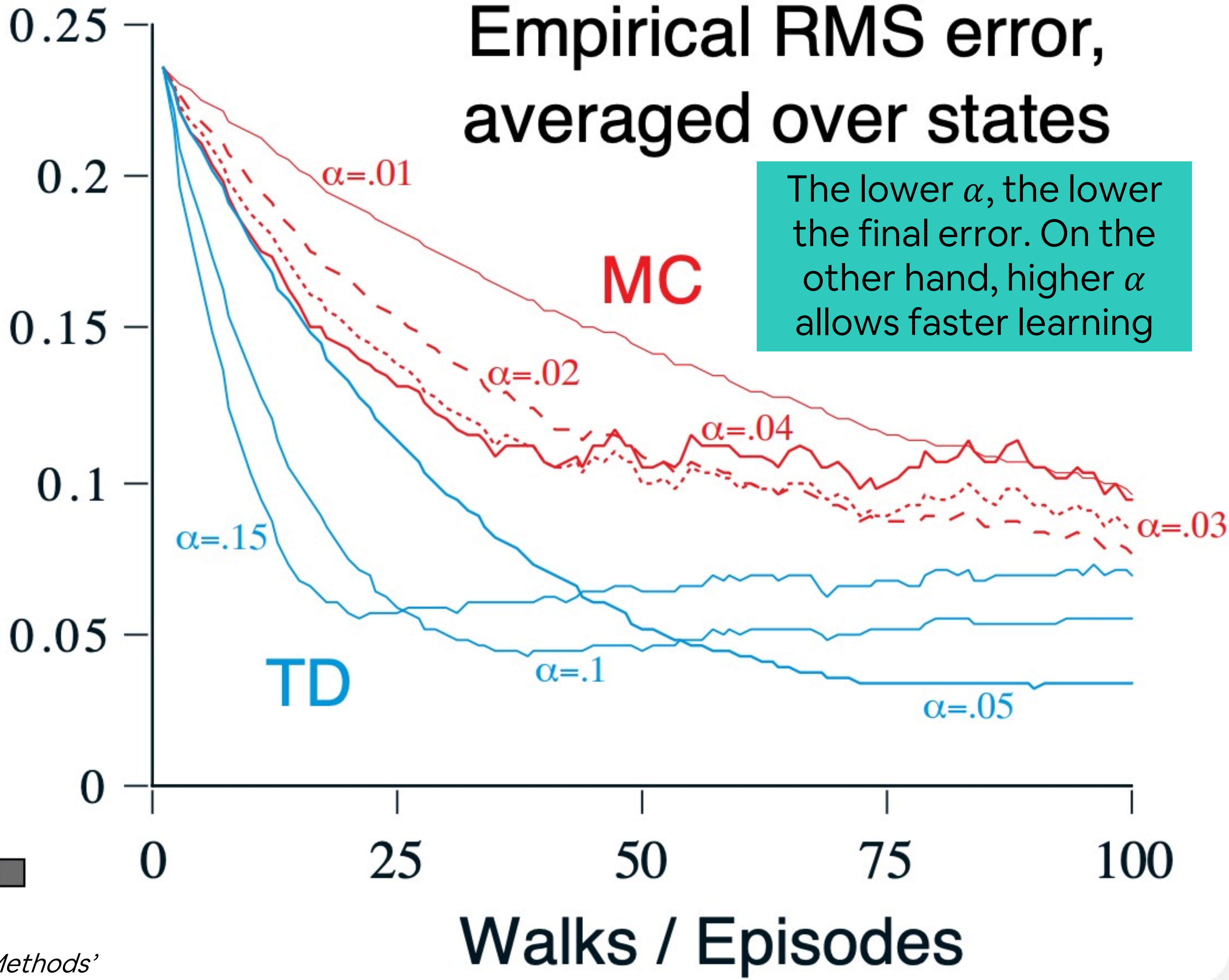




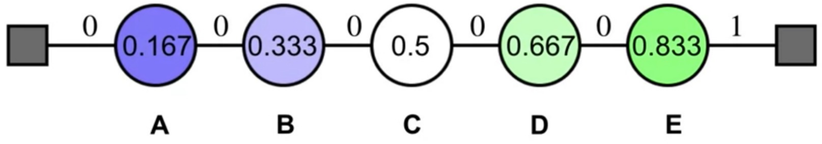
A. White, M. White 'Sample-based Learning Methods'

Recap Prediction: TD(0) – Random Walk

Estimation error - Root Mean Square (RMS) – averaged over the 5 states, the averaged over 100 runs



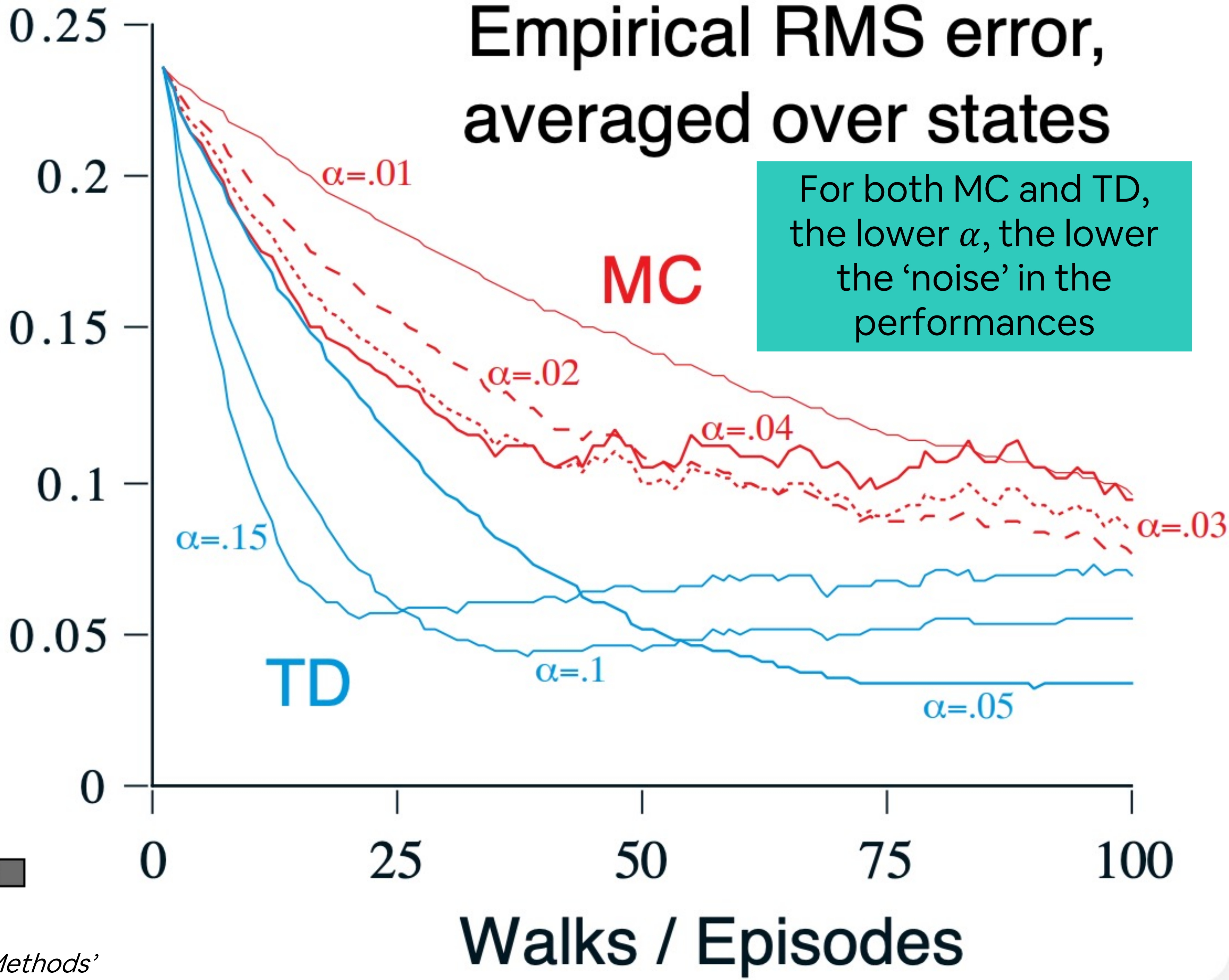
The lower α, the lower the final error. On the other hand, higher α allows faster learning



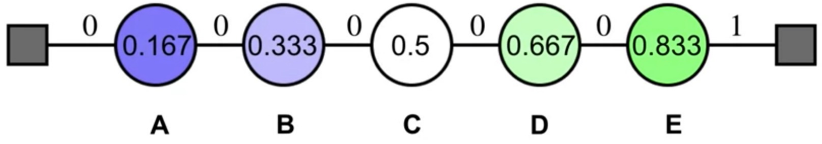
A. White, M. White 'Sample-based Learning Methods'

Recap
Prediction: TD(0) – Random Walk

Estimation error - Root Mean Square (RMS) – averaged over the 5 states, the averaged over 100 runs



For both MC and TD, the lower α, the lower the 'noise' in the performances



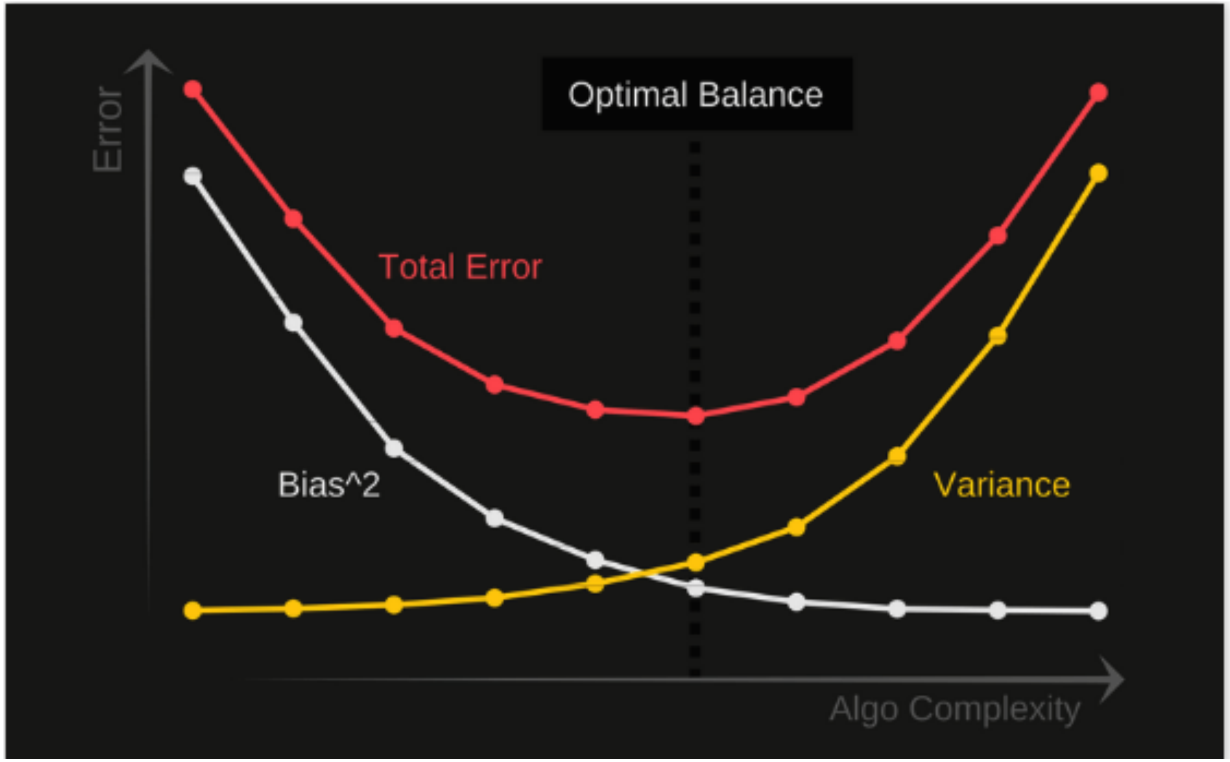
A. White, M. White 'Sample-based Learning Methods'

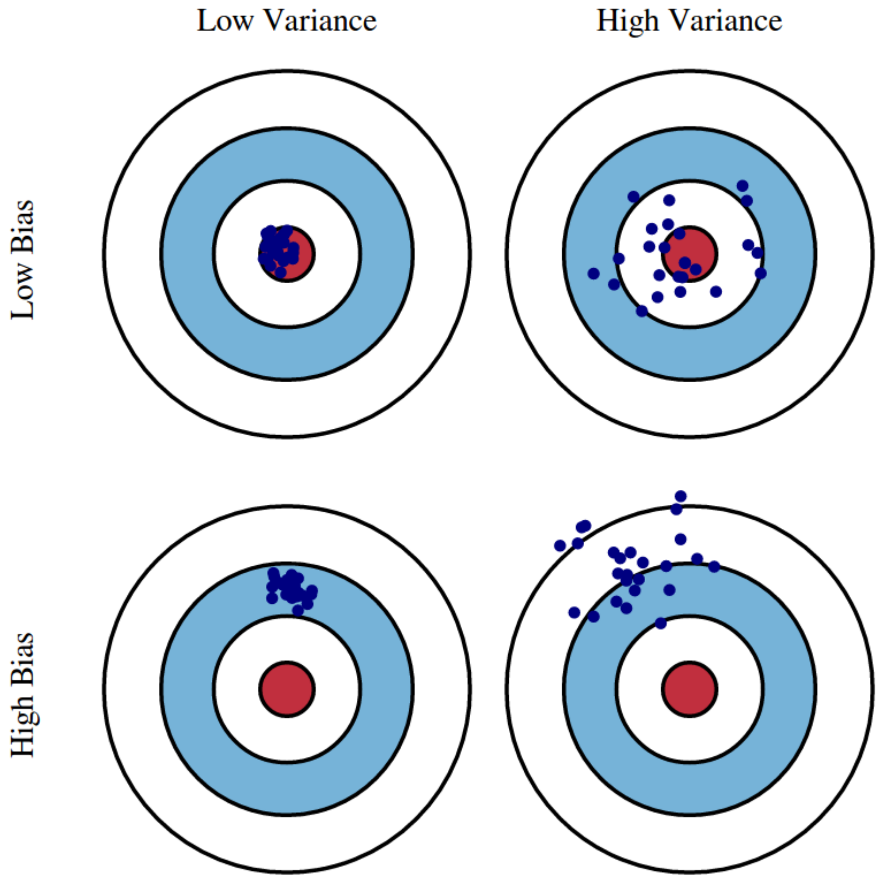
Bias/Variance Trade-off

- ?

Bias/Variance Trade-off

- A trade-off present in many Machine Learning settings





Bias/Variance Trade-off

- The return $G_t = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{T-1} R_T$ is an <u>unbiased</u> estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ (we don't know the true value function!) is an <u>unbiased</u> estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ (the target that we move to, it can be quite a wrong estimate!) is a <u>biased</u> estimate of $v_\pi(S_t)$

Bias/Variance Trade-off

- The return $G_t = R_{t+1} + \gamma R_{t+2} + \ \ldots + \gamma^{T-1} R_T$ is an unbiased estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ (we don't know the true value function!) is an unbiased estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ (the target that we move to, it can be quite a wrong estimate!) is a biased estimate of $v_\pi(S_t)$
- TD target is much lower variance than the return:

1. Return depends on many random actions, transitions, rewards
2. TD target depends on one random action, transition, reward

Bias/Variance Trade-off

- MC has high variance, zero bias

1. Good convergence properties (even with function approximation – Chapter 9, we will see this later in the course)
2. Not very sensitive to initial value
3. Very simple to understand and use

- TD has low variance, some bias

1. Usually more efficient than MC
2. TD(0) converges to $v_\pi(s)$ (but not always with function approximation -> we'll talk about this in few lectures!)
3. More sensitive to initial values

Prediction: (batch) MC and TD – AB Example

Two states, no discounting, 8 episodes of experience:

A, 0, B, 0

B, 1

B, 1

B, 1

B, 1

B, 1

B, 1

B, 0

What is $V(A)$ and what is $V(B)$?

Prediction: (batch) MC and TD – AB Example

Two states, no discounting, 8 episodes of experience:

A, 0, B, 0

B, 1

B, 1

B, 1

B, 1

B, 1

B, 1

B, 0

$V(B)$ = 3/4
$V(A)$ = ?

What is $V(A)$ and what is $V(B)$?

Prediction: (batch) MC and TD – AB Example

Two states, no discounting, 8 episodes of experience:

A, 0, B, 0

B, 1

B, 1

B, 1

B, 1

B, 1

B, 1

B, 0

What is $V(A)$ and what is $V(B)$?

$V(B)$ = 3/4
$V(A)$ = 0 is the Monte Carlo solution
-> MC converges to solution with minimun mean-squared error

Prediction: (batch) MC and TD – AB Example

Two states, no discounting, 8 episodes of experience:

A, 0, B, 0

B, 1

B, 1

B, 1

B, 1

B, 1

B, 1

B, 0

What is $V(A)$ and what is $V(B)$?

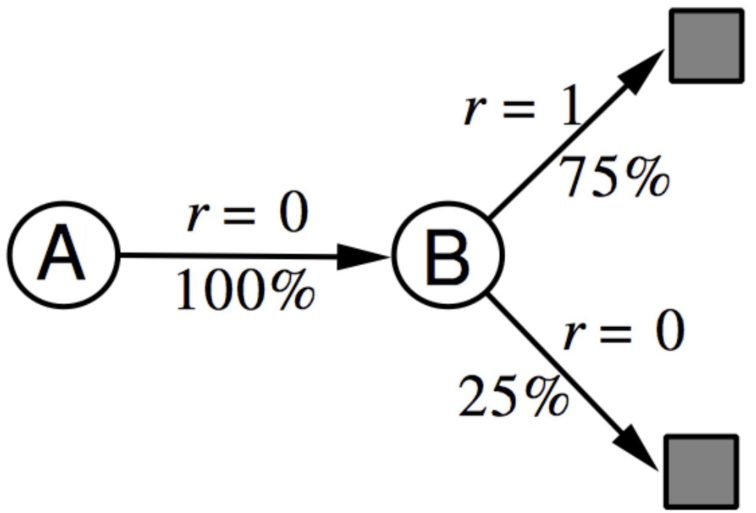
$V(B)$ = 3/4

$V(A)$ = 0 is the Monte Carlo solution

-> MC converges to solution with minimun mean-squared error

$V(A)$ = 3/4 is the TD(0) solution

-> TD(0) (implicitely) converges to solution of max likelihood Markov model



Prediction: MC and TD – AB Example

MC converges to solution with minimun mean-squared error: best fit to the observed returns

$$\sum_{k=1}^{K} \sum_{t=1}^{T_k} \left(G_t^k - V(s_t^k)\right)^2$$

TD(0) (implicitely) converges to solution of max likelihood Marxov model, ie the solution to the MDP $\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$ that best fits the data

$$\hat{\mathcal{P}}_{s,s'}^{a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k, s_{t+1}^k = s, a, s')$$

$$\hat{\mathcal{R}}_{s}^{a} = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{t=1}^{T_k} \mathbf{1}(s_t^k, a_t^k = s, a) r_t^k$$

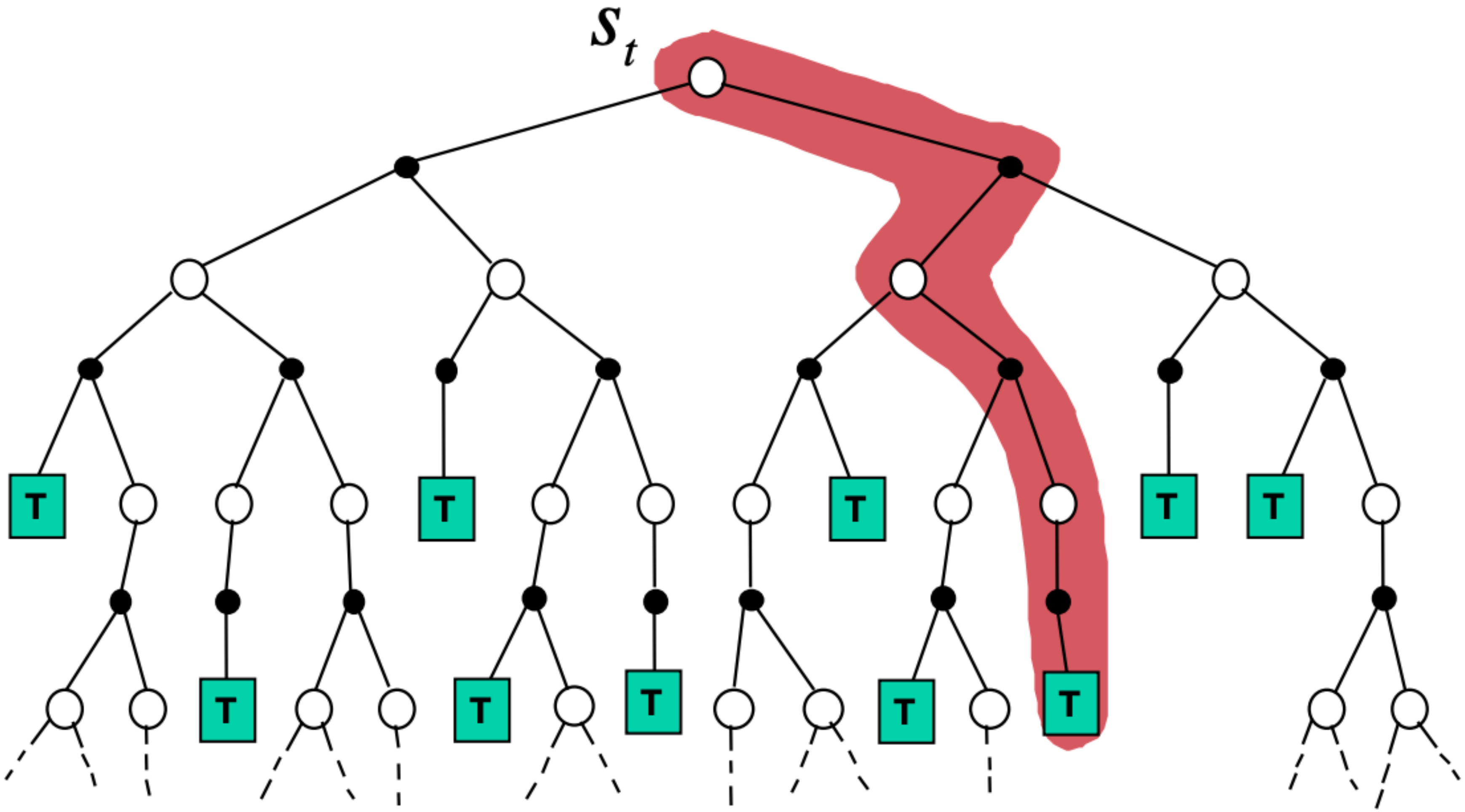
TD exploits Markov property and it is usually more efficient in Markov environment

(Prediction: batch MC and TD)

- If experience grows (amount of episodes increases), both MC and TD will converge to $V(s) \longrightarrow v_\pi(s)$
- In many cases we however have limited amount of experience: a common approach is to present the experience repeatedly until the method converges upon an answer
- We repeatedly sample episode $k \in [1, K]$ and we apply MC or TD(0) to that episode
- Under batch training, MC and TD(0) converges to an ‘optimal’ solution, but with different definition of optimality

Monte-Carlo Backup

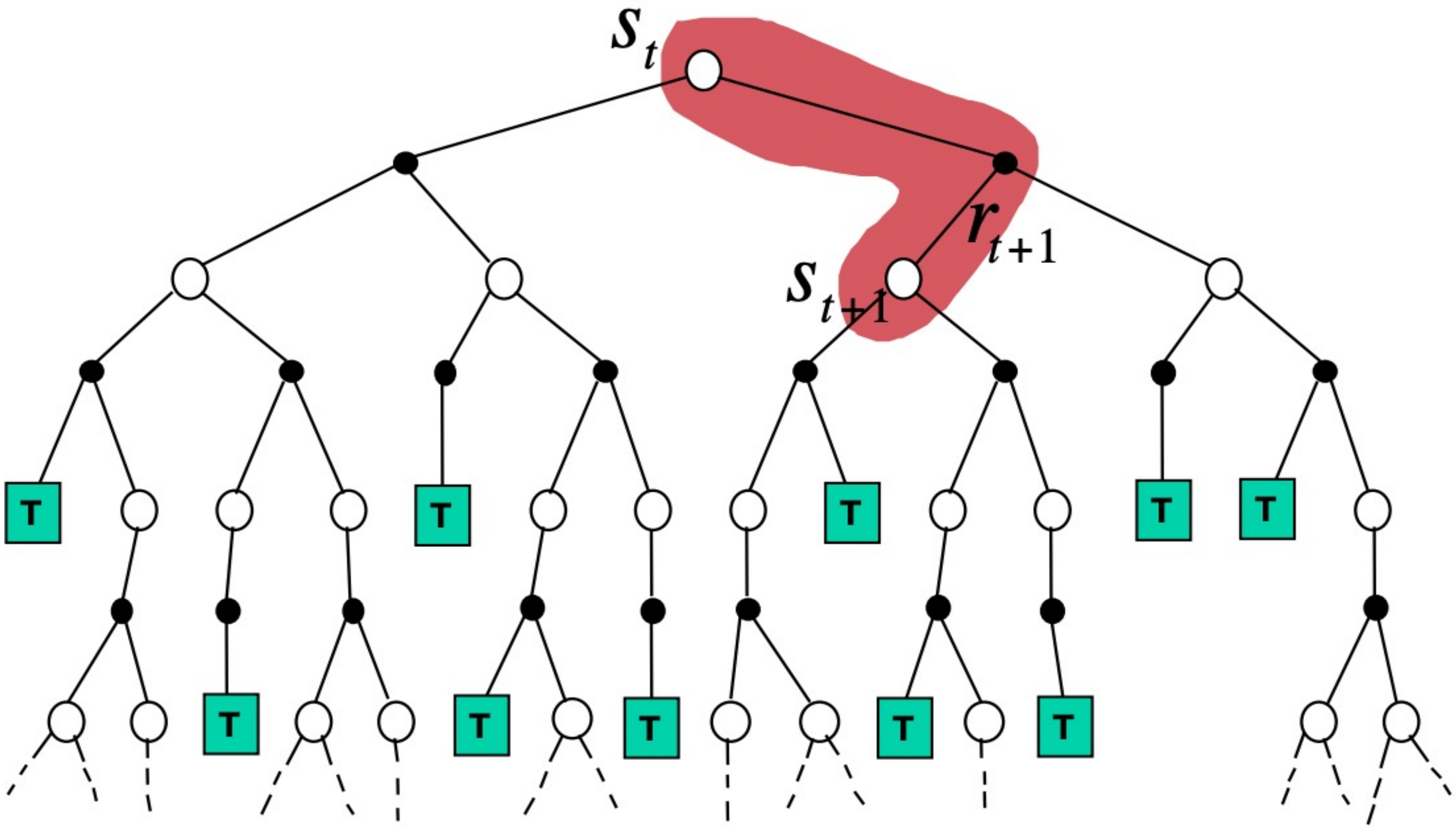
$$V(S_t) \leftarrow V(S_t) + \alpha\left(G_t - V(S_t)\right)$$



D. Silver
'Reinforcement
Learning' @ UCL

Temporal Difference Backup

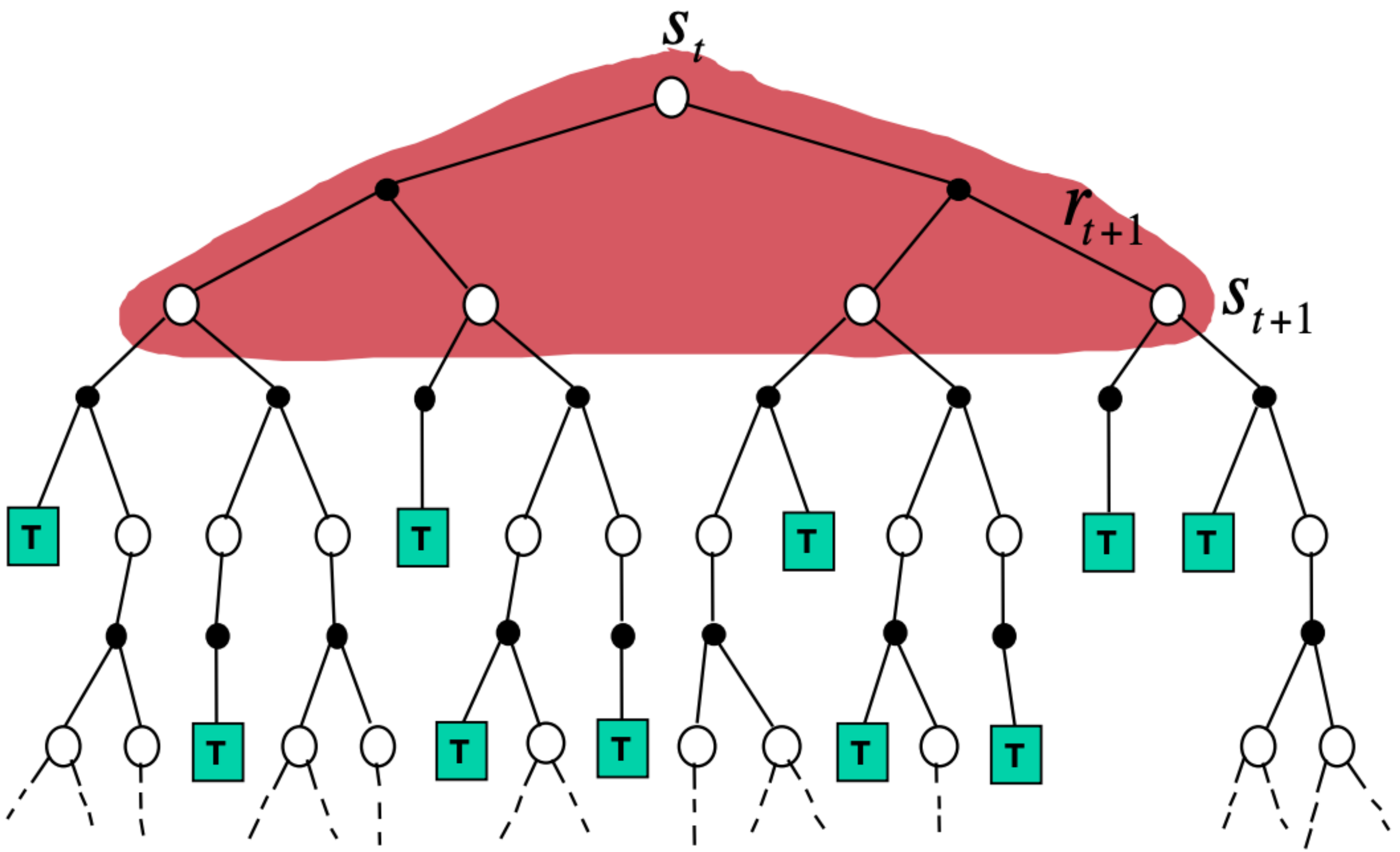
$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right)$$



D. Silver
'Reinforcement
Learning' @ UCL

Dynamic Programming Backup

$$V(S_t) \leftarrow \mathbb{E}_\pi \left[R_{t+1} + \gamma V(S_{t+1})\right]$$

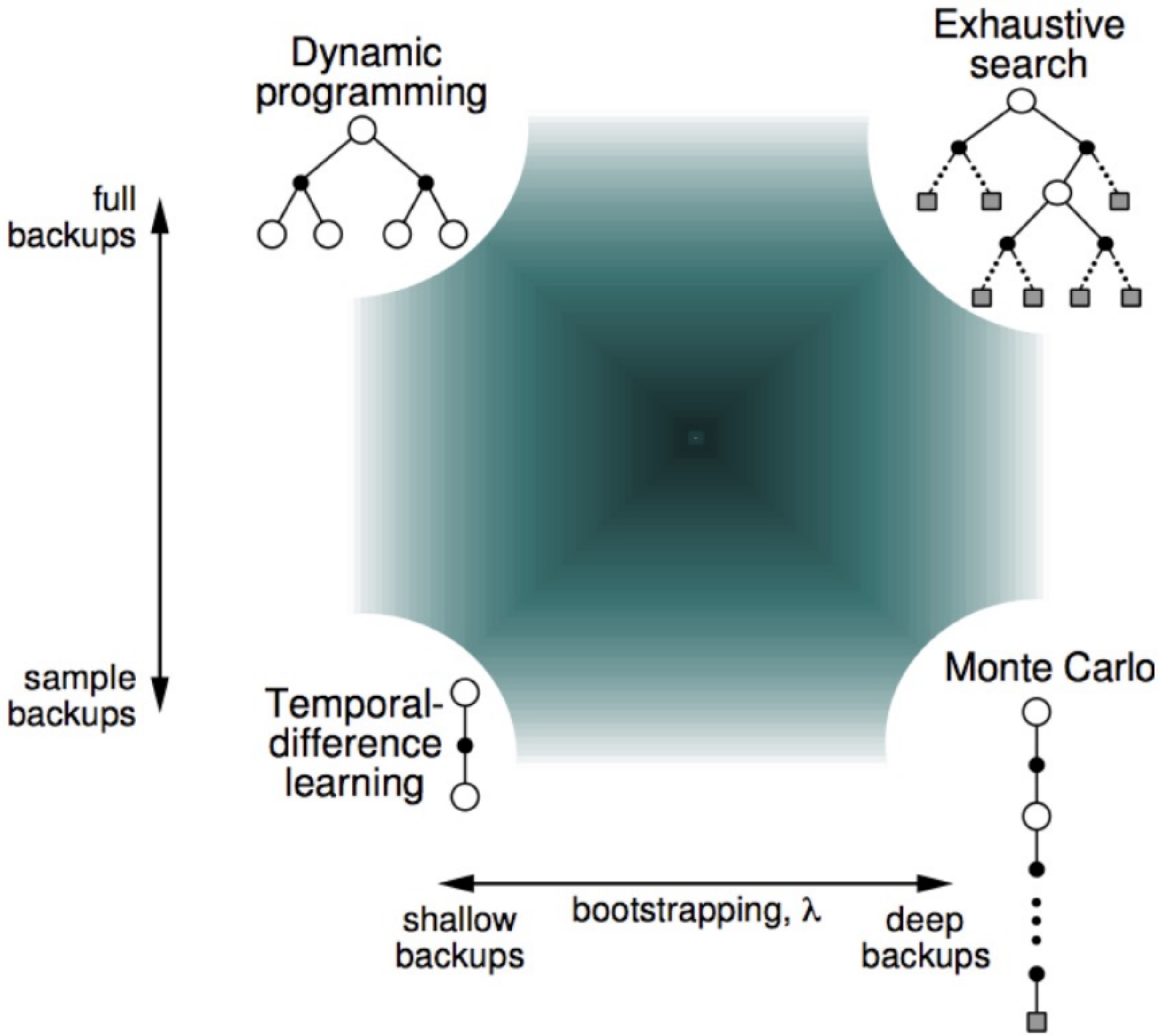


D. Silver 'Reinforcement Learning' @ UCL

Unified View of Reinforcement Learning

Bootstrap (update involves an estimate)
- MC does not bootstrap
- DP bootstraps
- TD bootstraps

Sampling (use samples to estimate expectation)
- MC samples
- DP does not sample
- TD samples



D. Silver 'Reinforcement Learning' @ UCL

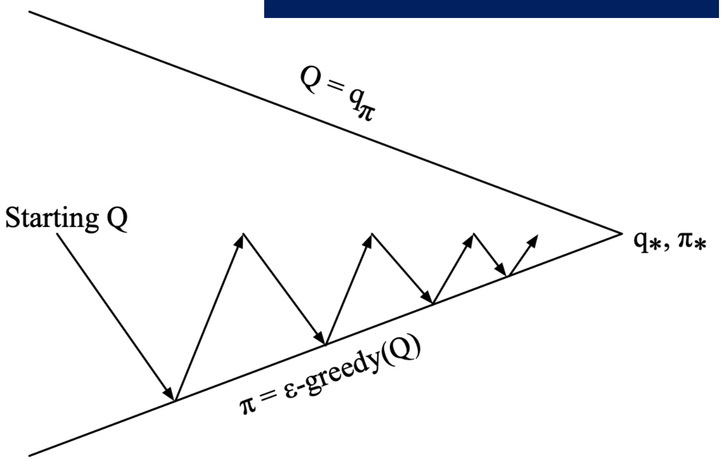
TD-Learning Control

- (On Policy) SARSA
- (Off Policy) Q-Learning

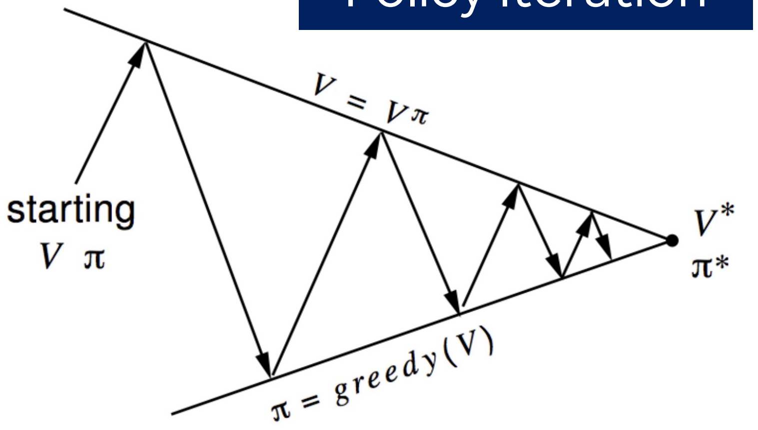
Control: Model-free Generalized Policy Iteration (GPI) with TD learning

- We apply again the GPI approach (iterations between prediction and improvement) for solving control
- Since we are model-free, we use interactions over q_π
- With TD(0) we need a way to handle incomplete sequences and we will consider improvements over the TD target (updates at every step of the episode)
- We start by considering the on-policy case

Value Iteration



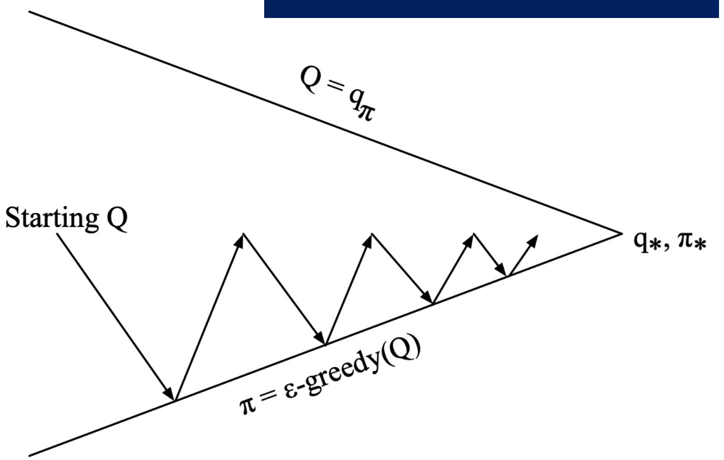
Policy Iteration



Control: Model-free Generalized Policy Iteration (GPI) with TD learning

- We apply again the GPI approach (iterations between prediction and improvement) for solving control
- Since we are model-free, we use interactions over q_π
- With TD(0) we need a way to handle incomplete sequences and we will consider improvements over the TD target (<u>updates at every step of the episode</u>)
- We start by considering the on-policy case

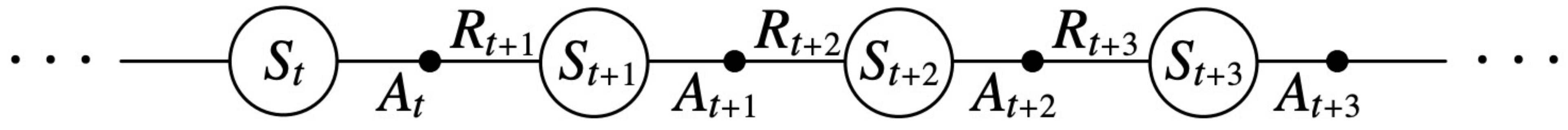
Value Iteration

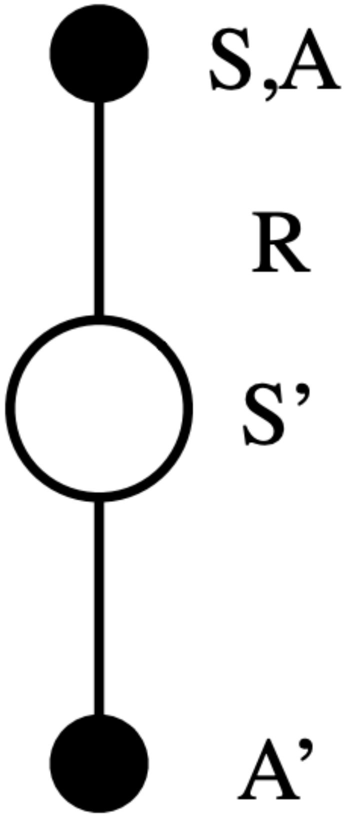


We'll consider:

- Similarly to value iterations, only partial evaluation of q_π (in line with the principle of TD(0) of using the 'newest' estimation)
- As in MC, we can consider approaches for dealing with exploration, like ε-greedy

Control: SARSA - On-policy TD learning for Control

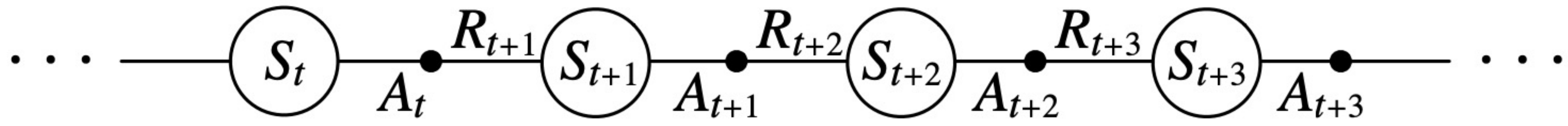


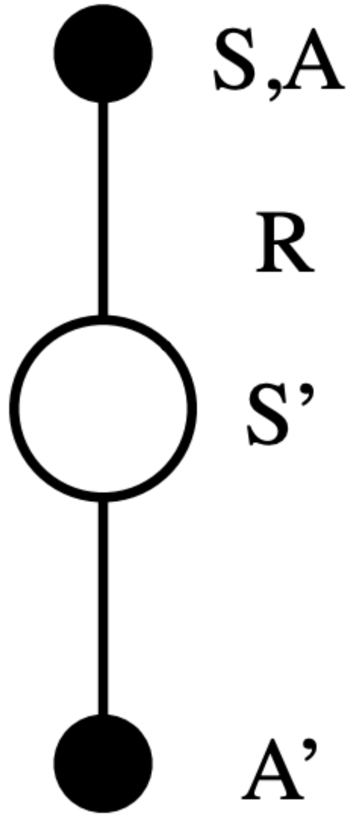


- We need to consider transactions from (state, action) to (state, action)
- TD target
- TD error

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

Control: SARSA - On-policy TD learning for Control

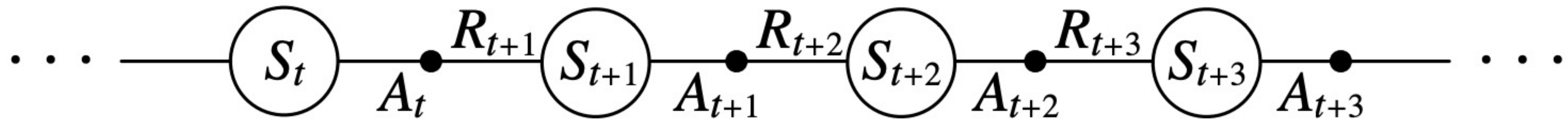


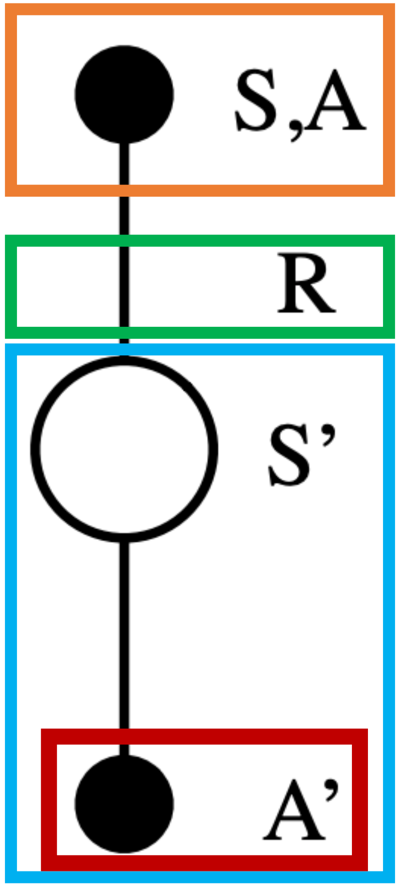


- We need to consider transactions from (state, action) to (state, action)
- TD target
- TD error

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

Control: SARSA - On-policy TD learning for Control





- We need to consider transactions from (state, action) to (state, action)
- TD target
- TD error

Pay attention: A' (A_{t+1}) is taken accordingly to your policy π

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

Control: SARSA - On-policy TD learning for Control

Which elements will be on the algorithm?

Control: SARSA - On-policy TD learning for Control

Which elements will be on the algorithm?

- GPI: policy evaluation + policy improvement
- Since we are doing TD learning, for loops both over the various episodes (we are model-free, we need data) and over the various steps in an episode
- In TD learning we will also need to consider incremental updates (so we need to set up an α parameter)
- For exploration we may consider epsilon greedy approach

Control: SARSA - On-policy TD learning for Control

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 $Q(\mathit{terminal}, \cdot) = 0$

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

Control: SARSA - On-policy TD learning for Control

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 $Q(terminal, \cdot) = 0$

Initialization: α is an hyperparameter

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

Control: SARSA - On-policy TD learning for Control

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

Algorithm parameters: step size $\alpha \in (0, 1]$, $\boxed{\text{small } \varepsilon > 0}$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

We are considering ε-greedy to ensure exploration

Loop for each episode:

  Initialize S

  Choose A from S using policy derived from Q $\boxed{\text{(e.g., } \varepsilon\text{-greedy)}}$

  Loop for each step of episode:

    Take action A, observe R, S'

    Choose A' from S' using policy derived from Q $\boxed{\text{(e.g., } \varepsilon\text{-greedy)}}$

    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') - Q(S, A)\big]$

    $S \leftarrow S'$; $A \leftarrow A'$;

  until S is terminal

Control: SARSA - On-policy TD learning for Control

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

In TD we always consider a double loop where we make updates for each step in each episode!

Control: SARSA - On-policy TD learning for Control

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

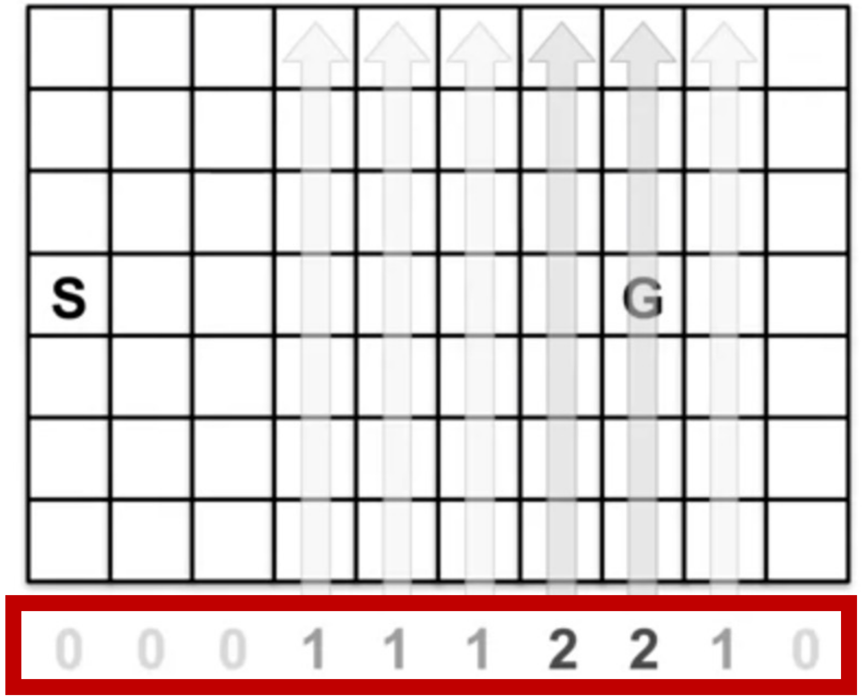
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except

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

We actual perform the next action, according to the policy, and the update Q. We will act epsilon greedily on Q at next step!

These are just to ‘move’ for next steps (S’, A’) -> (S, A)

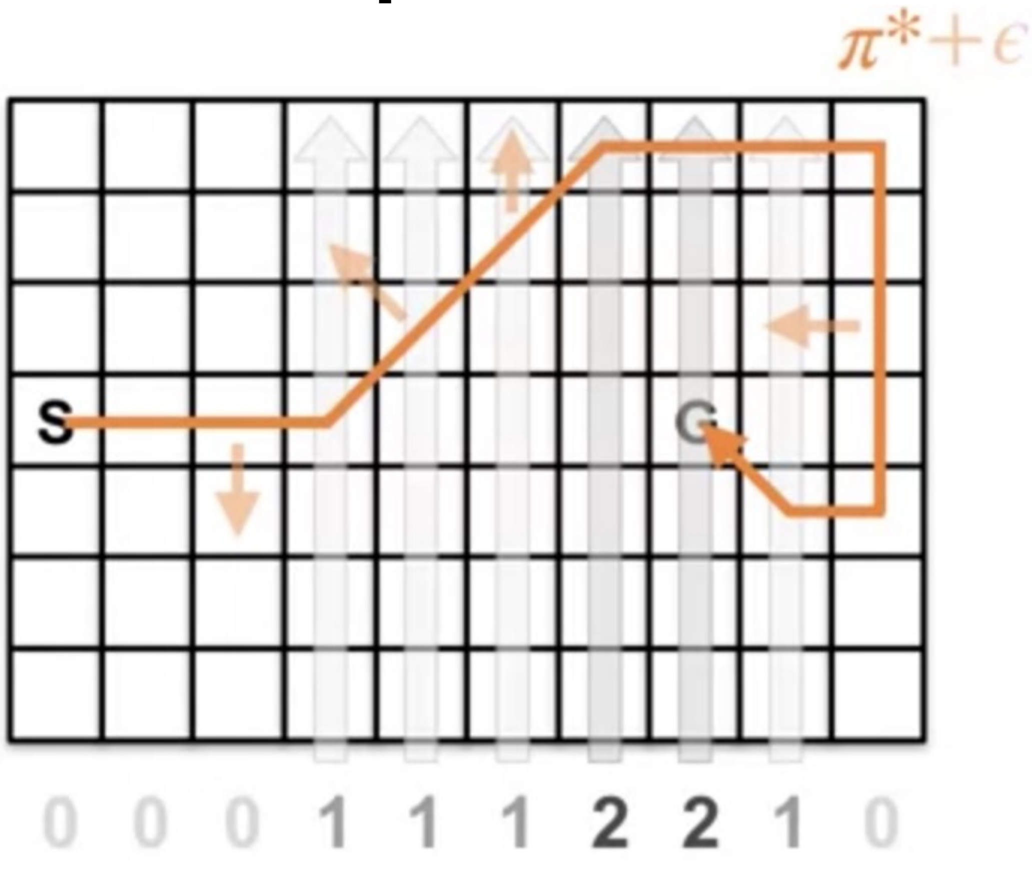
Control: SARSA – Windy Grid World Example



$$R_{step} = -1$$
$$\gamma = 1$$

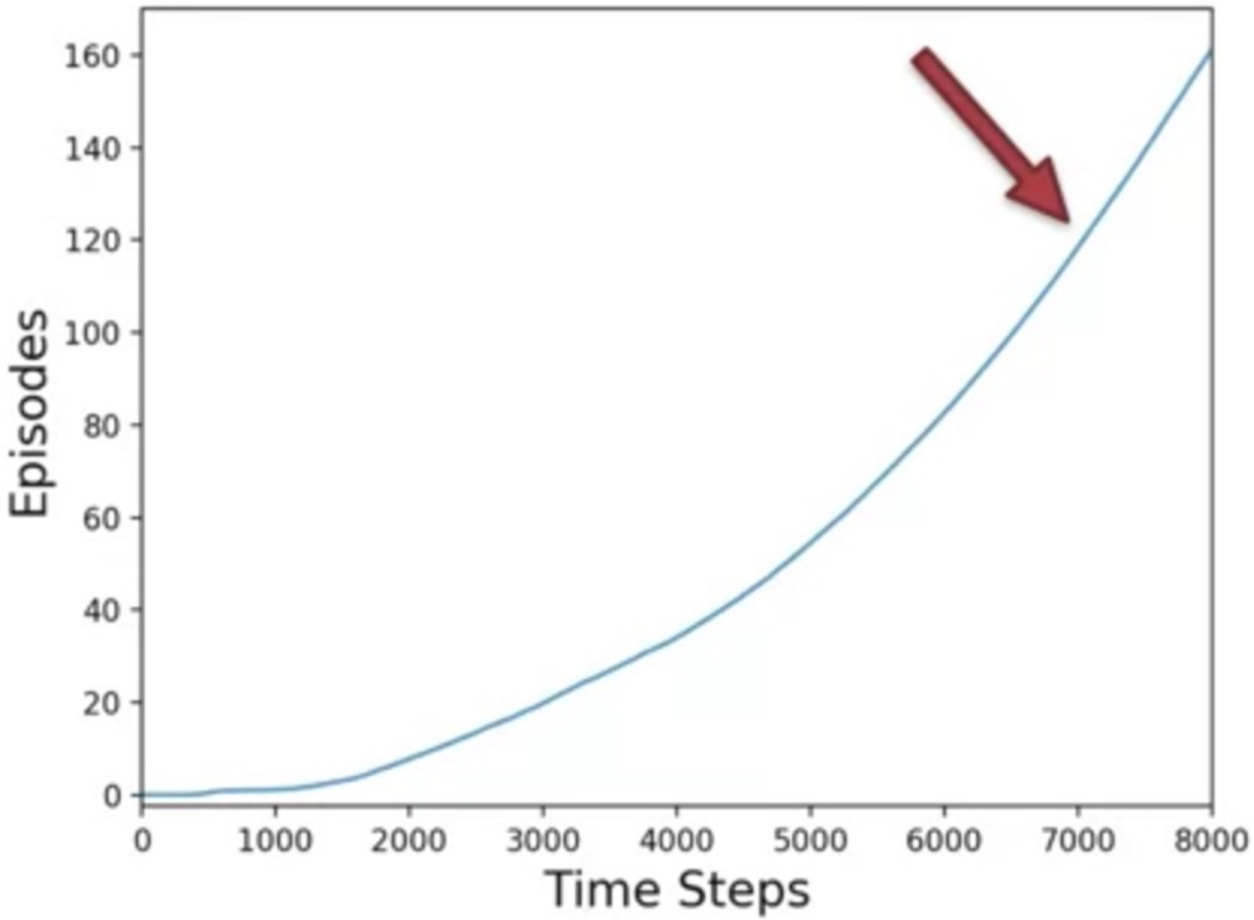
- The wind brings the agent up in the column of a number of cells equal to the number reported in the bottom
- Going outside the grid world does nothing
- This is a case where MC doesn't work well since many episodes may not end

Control: SARSA – Windy Grid World Example



Sarsa: $\epsilon = 0.1$

$\alpha = 0.5$



A. White, M. White 'Sample-based Learning Methods'

Control: Q-learning – (Off-policy) TD learning for Control

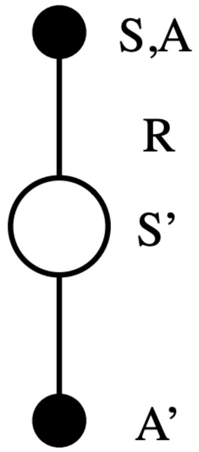
- Q-learning is the most popular approach to RL control
- We'll see how Q-learning (for TD(0)):

1. Can be derived as a slight modification from SARSA
2. Is associated with the Bellman Optimality Equation
3. (Can be considered off-policy)

Control: Q-learning vs. SARSA

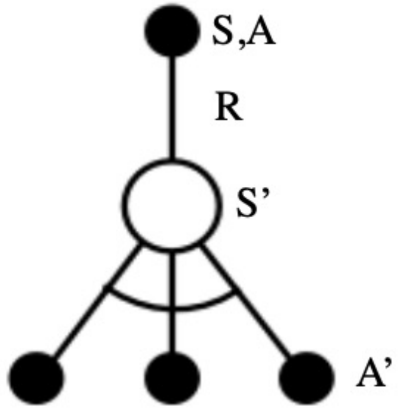
SARSA

Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;



Q-learning

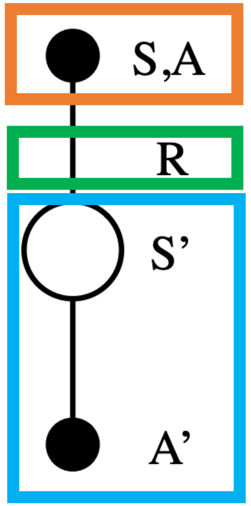
Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
$S \leftarrow S'$



Control: Q-learning vs. SARSA

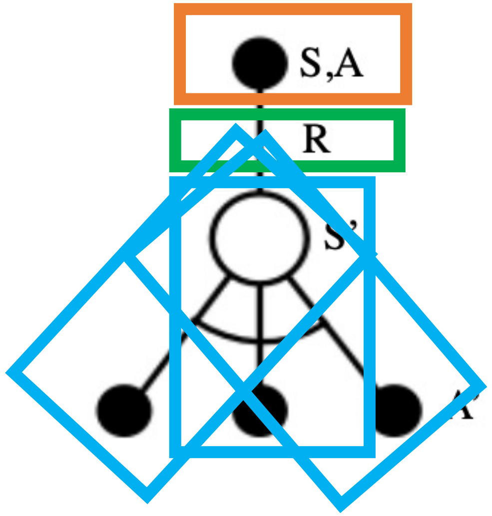
SARSA

Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma Q(S', A') - Q(S, A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;



Q-learning

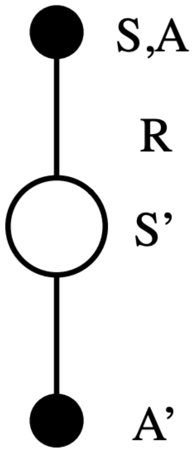
Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
$S \leftarrow S'$



Control: Q-learning vs. SARSA

SARSA

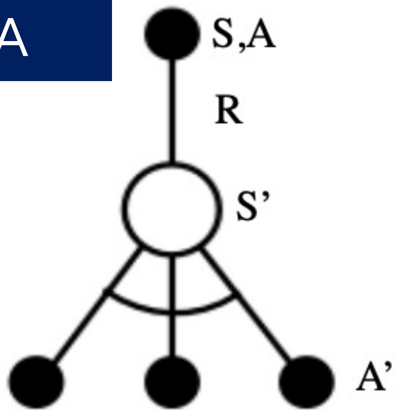
Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;



Q-learning is usually faster than SARSA

Q-learning

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
$S \leftarrow S'$

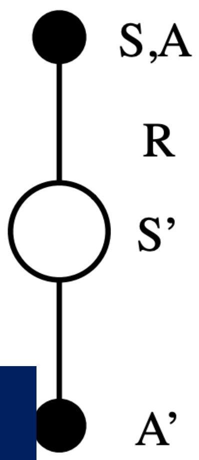


Control: Q-learning vs. SARSA

SARSA

Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;

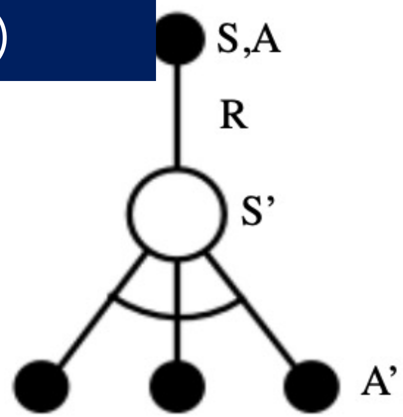
You actual perform next action, according to the policy and then update Q(s,a)



Q-learning

You look ahead and imagine greedy next action to update Q(s,a) (but you then perform the actual next action based on your current policy)

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
$S \leftarrow S'$



Control: Q-learning – (Off-policy) TD learning for Control

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 $Q(\textit{terminal}, \cdot) = 0$

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

Control: Q-learning – (Off-policy) TD learning for Control

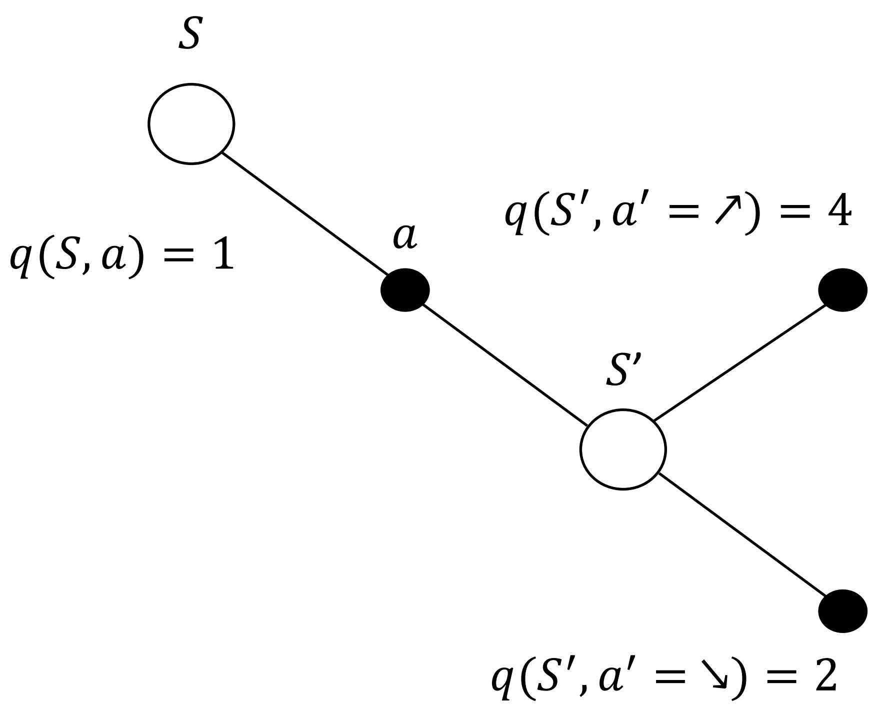
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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

Control: Q-learning vs. SARSA – example

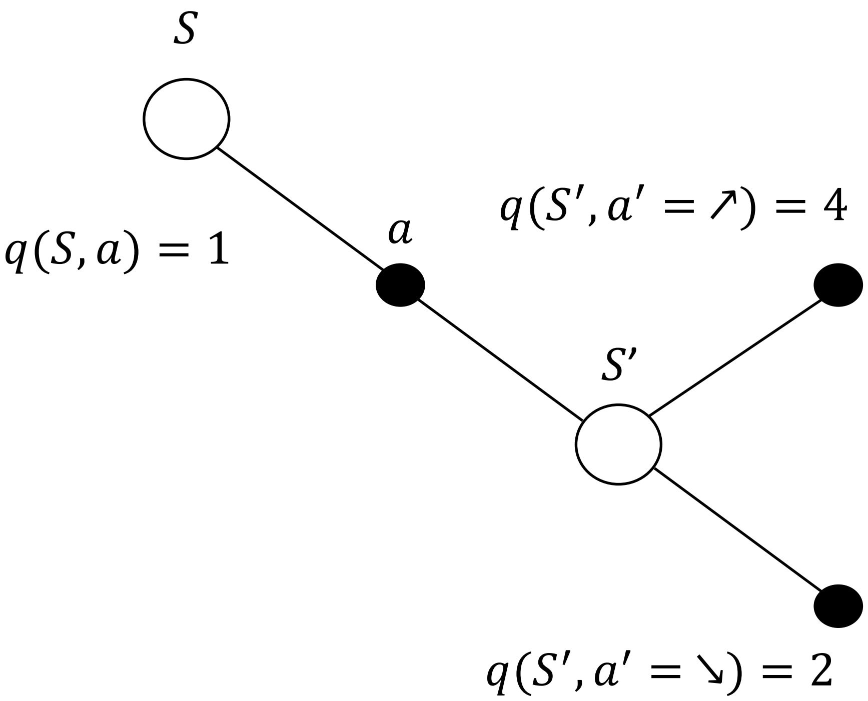
$\gamma = 0.5$
$\alpha = 0.1$



First episode we transition from S to S' by taking action a and we get a reward of +1

Control: Q-learning vs. SARSA – example

$\gamma = 0.5$
$\alpha = 0.1$



First episode we transition from S to S' by taking action a and we get a reward of +1

SARSA:

- Target:

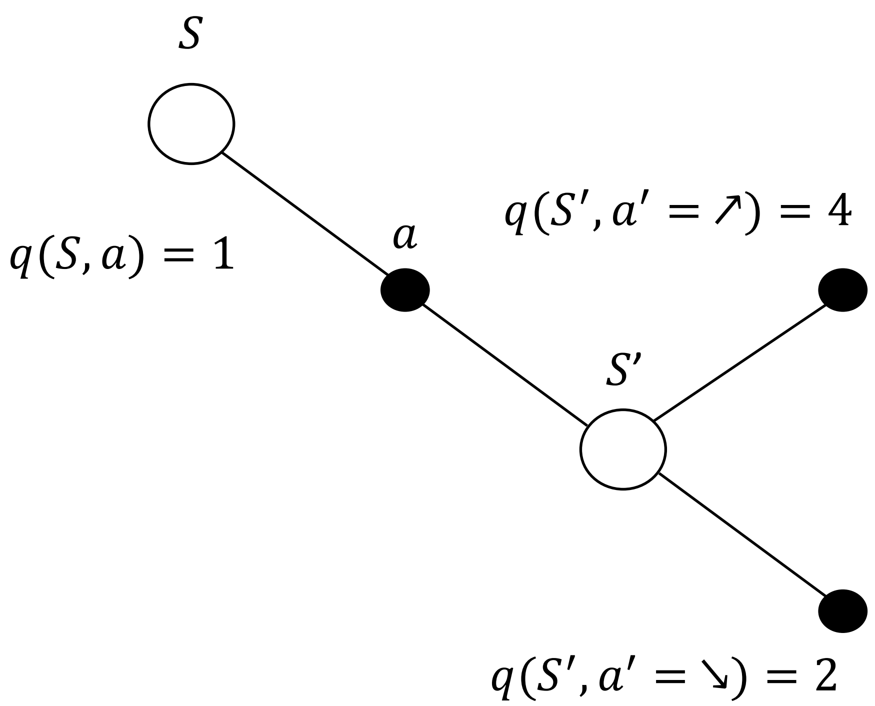
$r + \gamma q(s', \nearrow)$ = +1+0.5(+4) = +3 if by policy π we have $a' = \nearrow$ in s'
$r + \gamma q(s', \searrow)$ = +1+0.5(+2) = +2 if by policy π we have $a' = \searrow$ in s'

- Update

$q(S, a) = 1 + 0.1 * (3 - 1) = 1.2$ if by policy π we have $a' = \nearrow$ in s'
$q(S, a) = 1 + 0.1 * (2 - 1) = 1.1$ if by policy π we have $a' = \searrow$ in s'

Control: Q-learning vs. SARSA – example

$\gamma = 0.5$
$\alpha = 0.1$



First episode we transition from S to S' by taking action a and we get a reward of +1

SARSA:

- Target:

$r + \gamma q(s', \nearrow)$ = +1+0.5(+4) = +3 if by policy π we have $a' = \nearrow$ in s'
$r + \gamma q(s', \searrow)$ = +1+0.5(+2) = +2 if by policy π we have $a' = \searrow$ in s'

- Update

$q(S, a) = 1 + 0.1 * (3 - 1) = 1.2$ if by policy π we have $a' = \nearrow$ in s'
$q(S, a) = 1 + 0.1 * (2 - 1) = 1.1$ if by policy π we have $a' = \searrow$ in s'

Q-learning

- Target:

$r + \gamma \max_{a'} q(s', a')$ = +1+0.5(+4) = +3 indipendently from the current policy π (for this reason it is off-policy!)

- Update:

$q(S, a) = 1 + 0.1 * (3 - 1) = 1.2$

Control: Q-learning vs. SARSA and connection with Dynamic Programming

Sarsa: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\big)$

Bellman Expectation Equation

$$q_\pi(s,a) = \sum_{s',r} p(s', r \mid s, a)\Big(r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s', a')\Big)$$

Q-learning: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\big(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\big)$

Bellman Optimality Equation

$$q_*(s,a) = \sum_{s',r} p(s', r \mid s, a)\big(r + \gamma \max_{a'} q_\pi(s', a')\big)$$

A. White, M. White 'Sample-based Learning Methods'

Control: Q-learning vs. SARSA and connection with Dynamic Programming

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

Bellman Expectation Equation

$$q_\pi(s,a) = \sum_{s',r} p(s',r \mid s,a)\left(r + \gamma \sum_{a'} \pi(a' \mid s') q_\pi(s',a')\right)$$

SARSA is a sample-based version of Policy Iteration

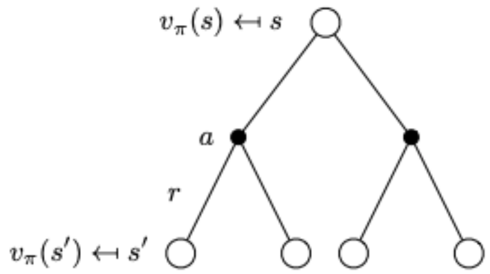
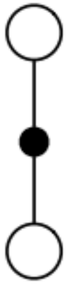
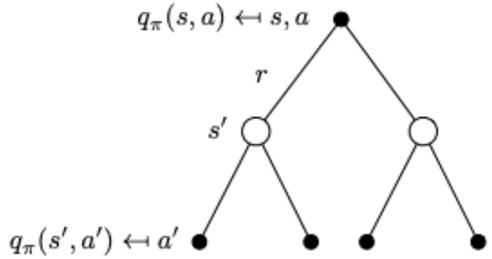
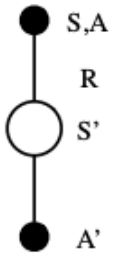
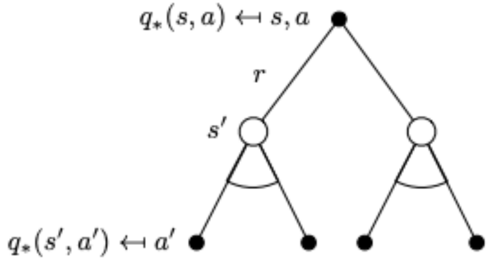
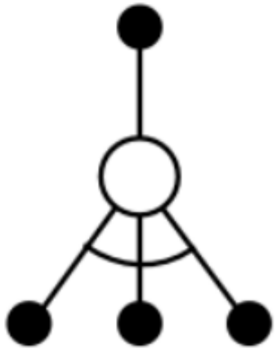
Q-learning: $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a') - Q(S_t, A_t)\right)$$

Bellman Optimality Equation

$$q_*(s,a) = \sum_{s',r} p(s',r \mid s,a)\left(r + \gamma \max_{a'} q_\pi(s',a')\right)$$

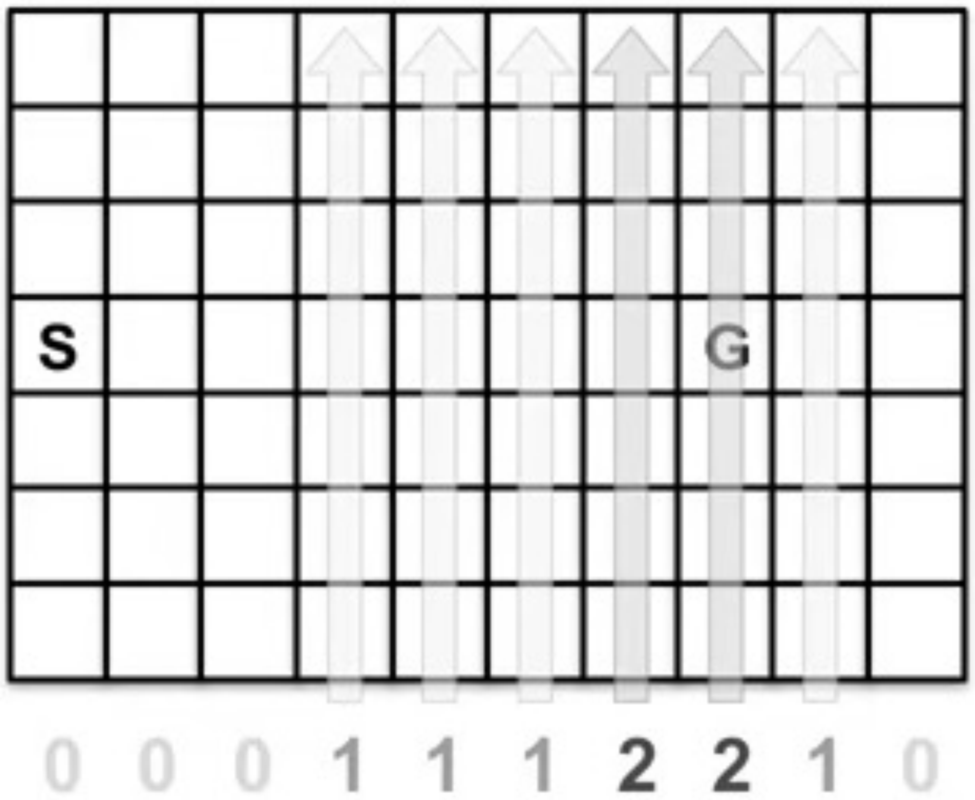
Q-learning is a sample-based version of Value Iteration

A. White, M. White 'Sample-based Learning Methods'

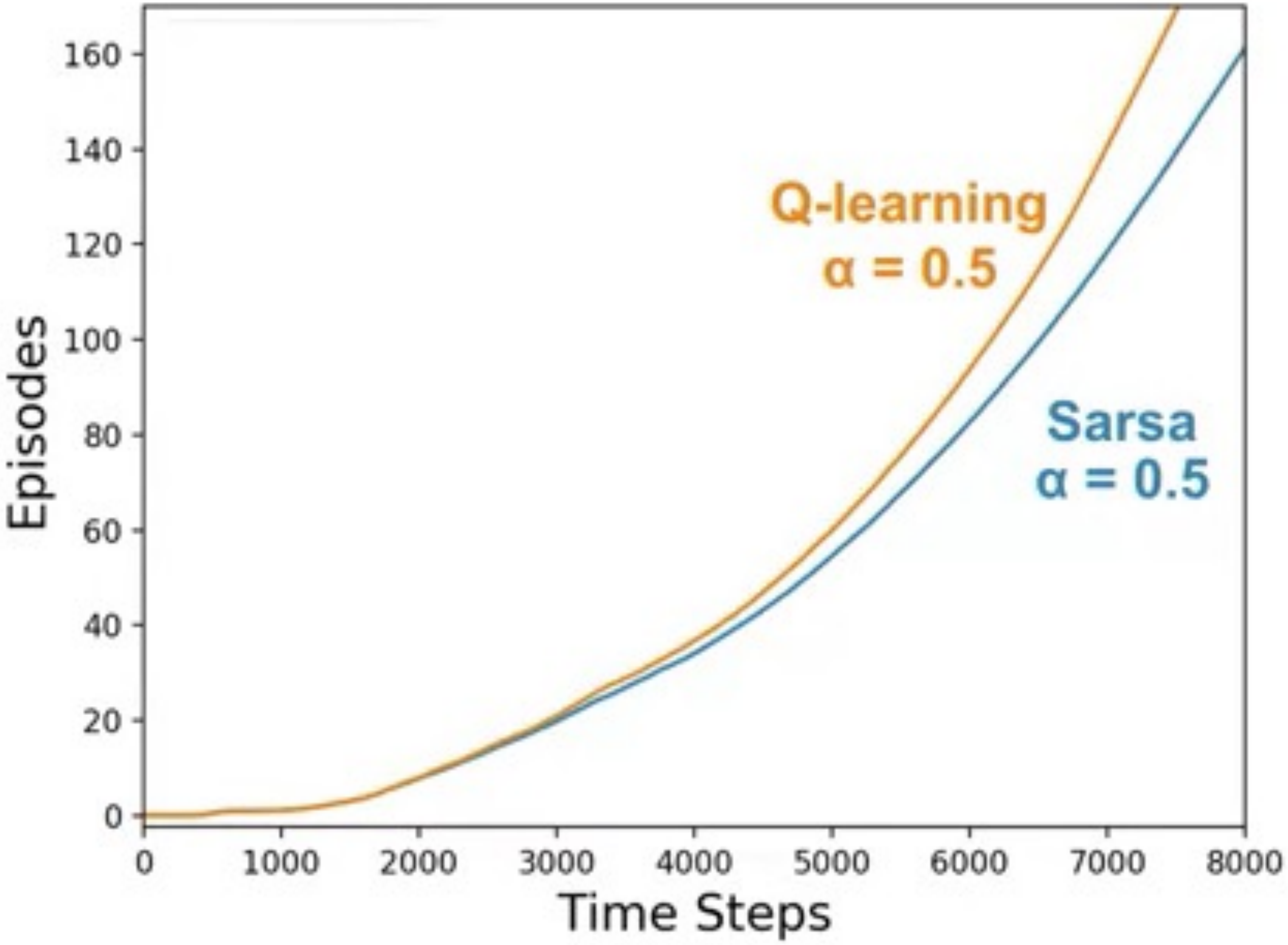
	Full Backup (DP)	*Sample Backup (TD)*
Bellman Expectation Equation for $v_\pi(s)$	 Iterative Policy Evaluation	 TD Learning
Bellman Expectation Equation for $q_\pi(s, a)$	 Q-Policy Iteration	 Sarsa
Bellman Optimality Equation for $q_*(s, a)$	 Q-Value Iteration	 Q-Learning

D. Silver 'Reinforcement Learning' @ UCL

Control: SARSA vs Q-learning – Windy Grid World Example

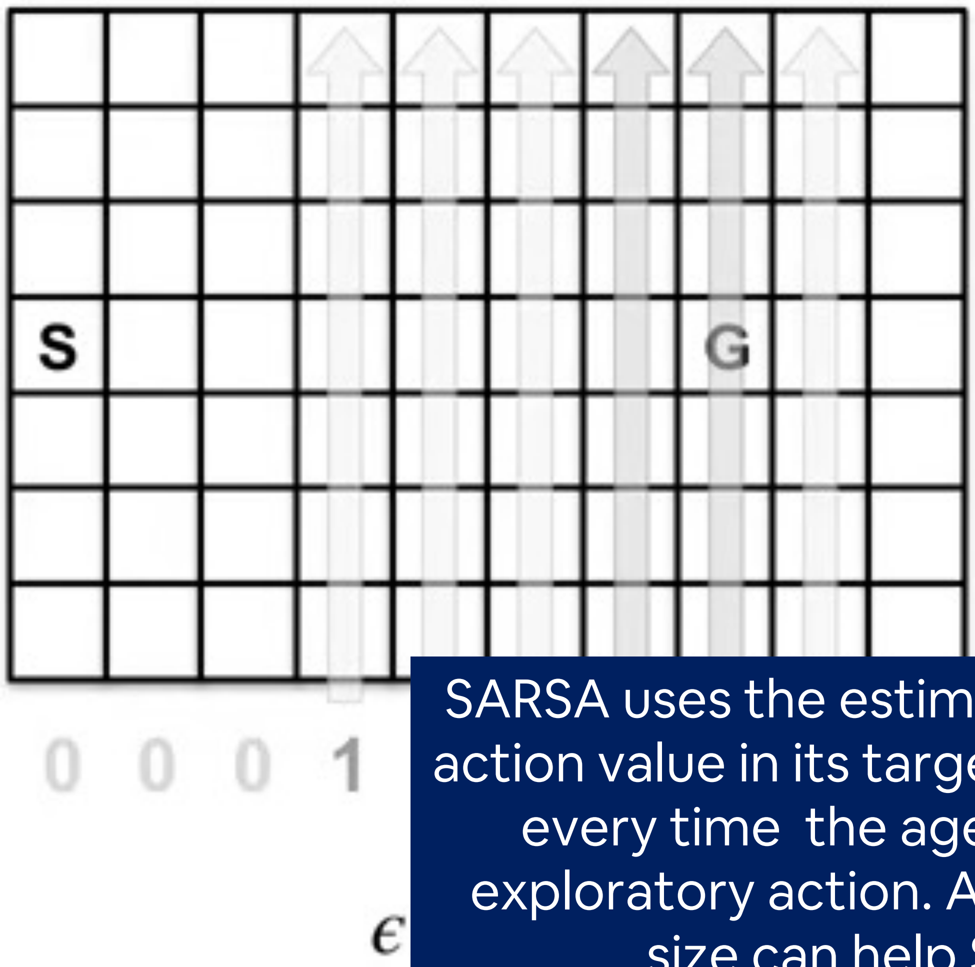


$\epsilon = 0.1$

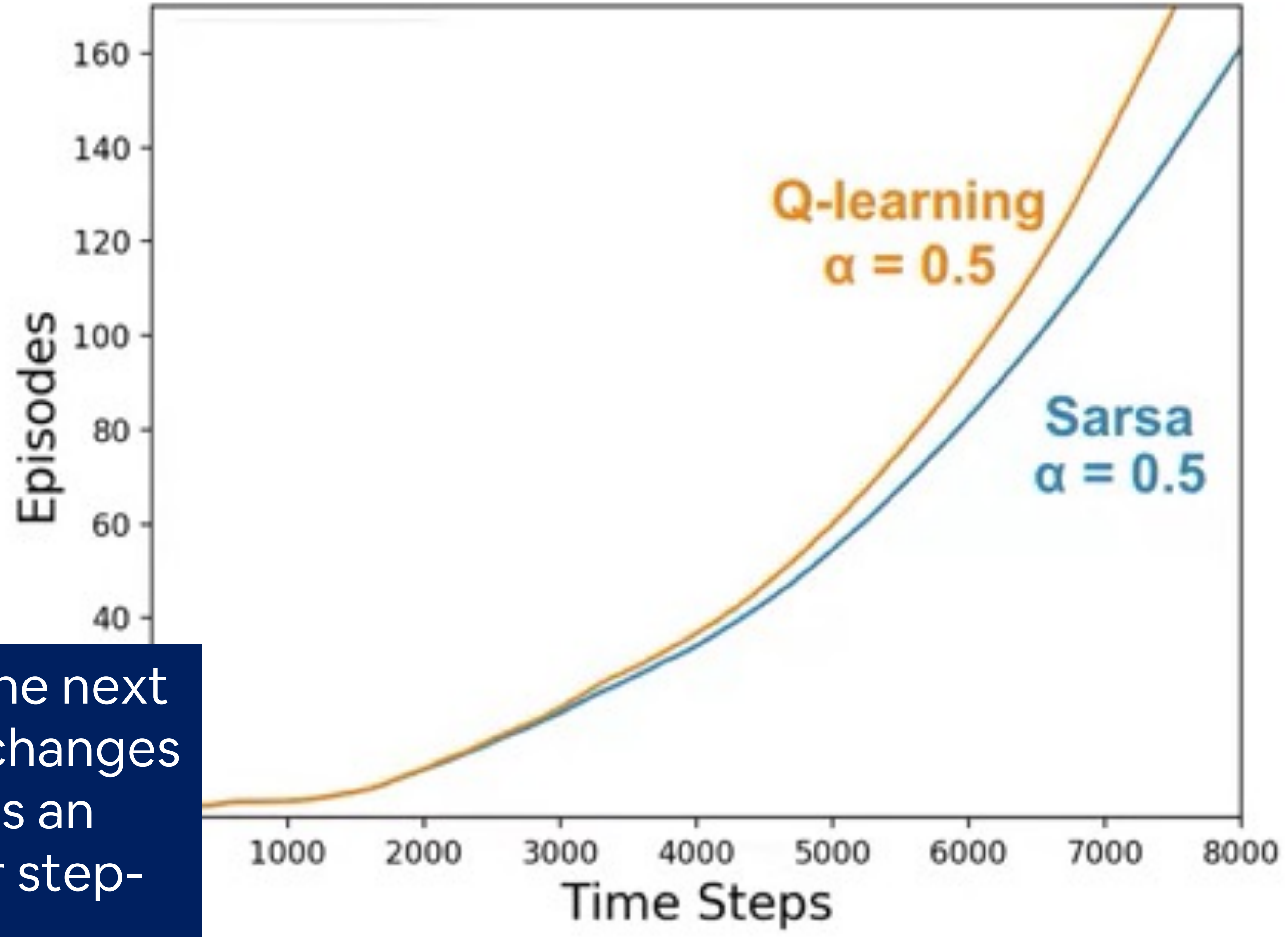


A. White, M. White 'Sample-based Learning Methods'

Control: SARSA vs Q-learning – Windy Grid World Example



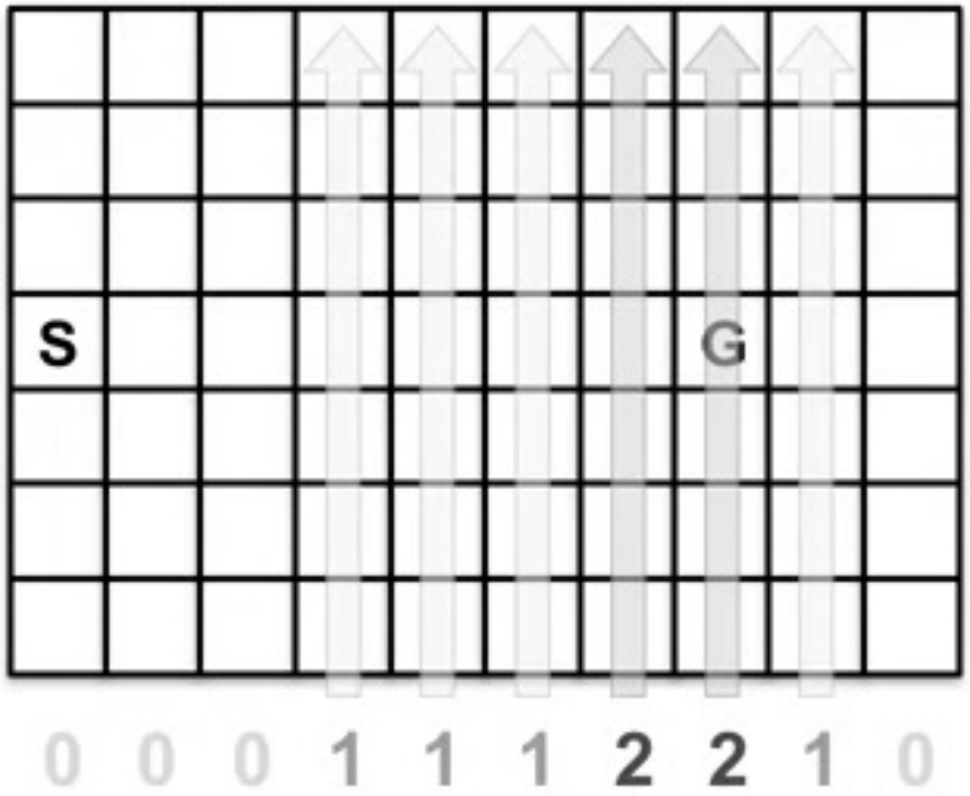
SARSA uses the estimate of the next action value in its target: this changes every time the agent takes an exploratory action. A smaller step-size can help SARSA



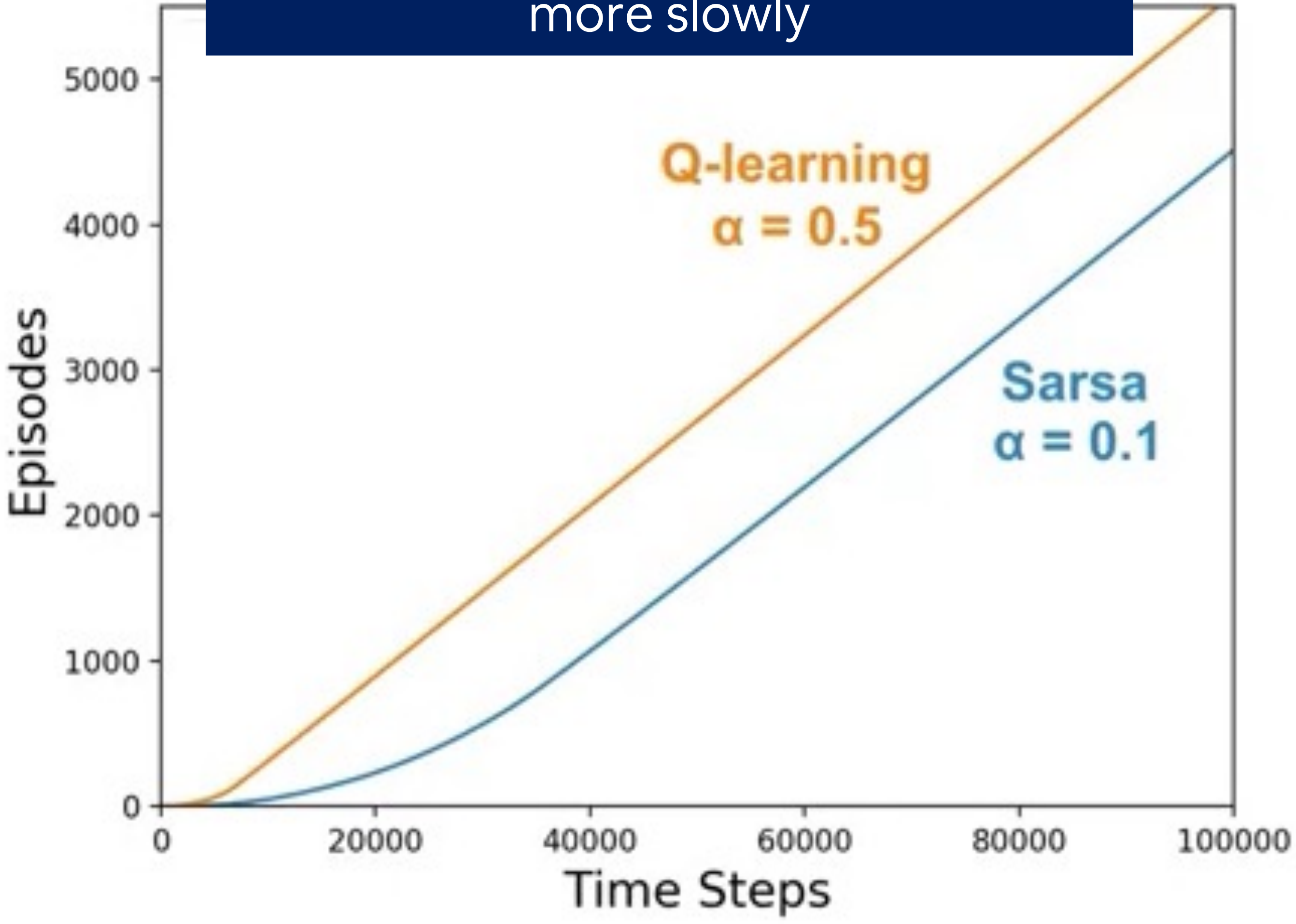
A. White, M. White 'Sample-based Learning Methods'

Control: SARSA vs Q-learning – Windy Grid World Example

SARSA finds the same solution of Q-learning (see the final slope), but more slowly



$\epsilon = 0.1$



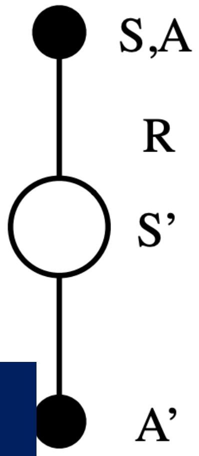
A. White, M. White 'Sample-based Learning Methods'

Control: why is Q-learning off-policy?

SARSA

Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
$S \leftarrow S'$; $A \leftarrow A'$;

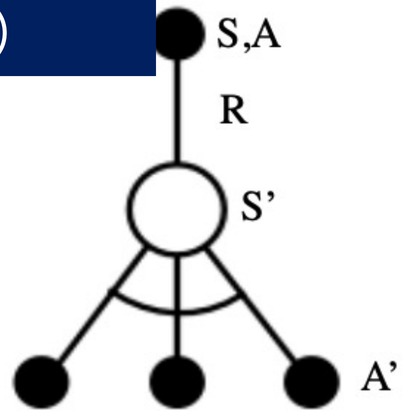
You actual perform next action, according to the policy and then update Q(s,a)



Q-learning

You look ahead and imagine greedy next action to update Q(s,a) (but you then perform the actual next action based on your current policy)

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q(S',a) - Q(S,A)\right]$
$S \leftarrow S'$

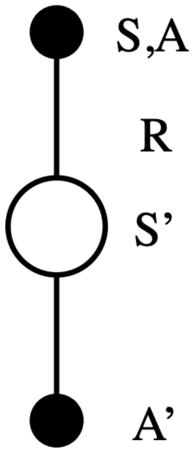


Control: why is Q-learning off-policy?

SARSA

Take action A, observe R, S'
Choose A' from S' using policy derived from Q (e.g., ε-greedy)
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
$S \leftarrow S'$; $A \leftarrow A'$;

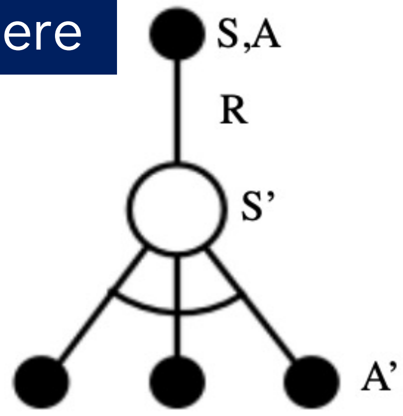
We only have one (target) policy here



Q-learning

Choose A from S using policy derived from Q (e.g., ε-greedy)
Take action A, observe R, S'
$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
$S \leftarrow S'$

We have a behaviour (epsilon-greedy) and a target policy (greedy!) here



Control: why is Q-learning off-policy?

Sarsa:

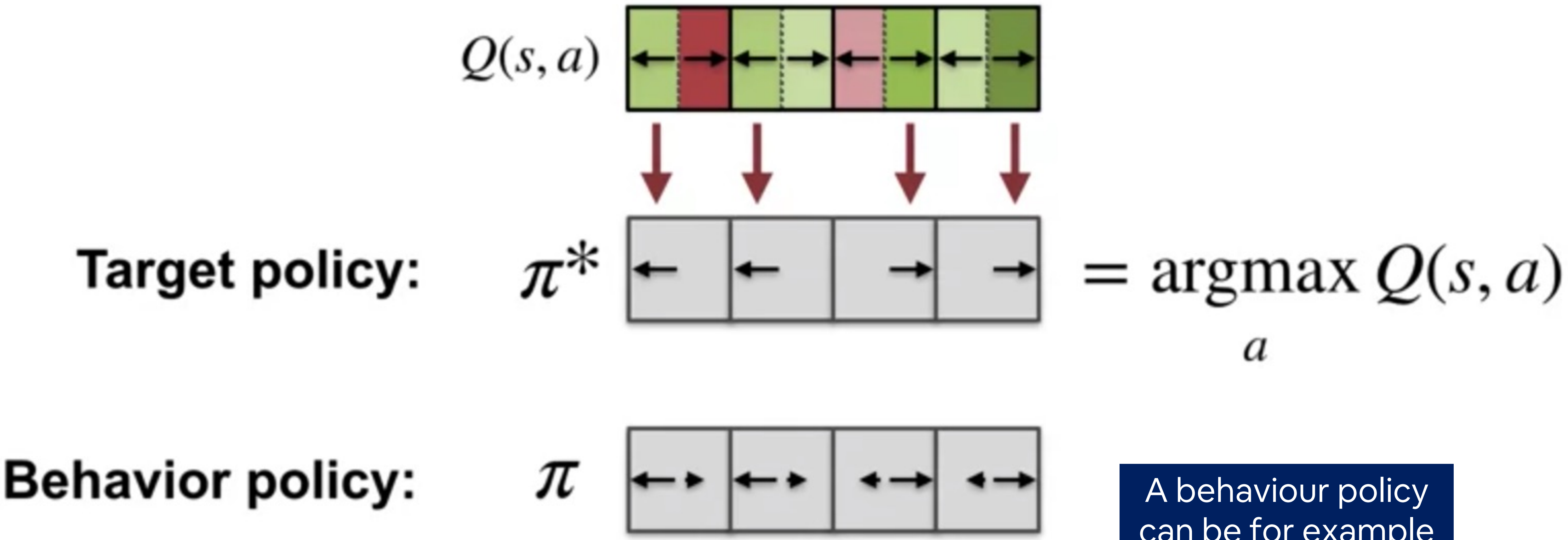
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma Q(S_{t+1}, \underset{\sim \pi}{A_{t+1}}) - Q(S_t, A_t)\right)$$

Q-learning:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\left(R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, \underset{\sim \pi_* \neq \pi}{a'}) - Q(S_t, A_t)\right)$$

A. White, M. White 'Sample-based Learning Methods'

Control: why is Q-learning off-policy?

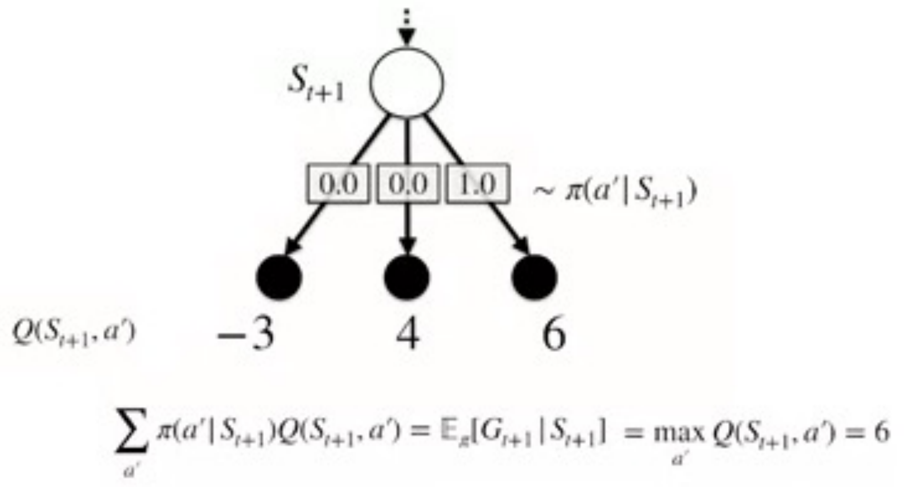


A behaviour policy can be for example ε-greedy

A. White, M. White 'Sample-based Learning Methods'

Control: why is Q-learning off-policy?

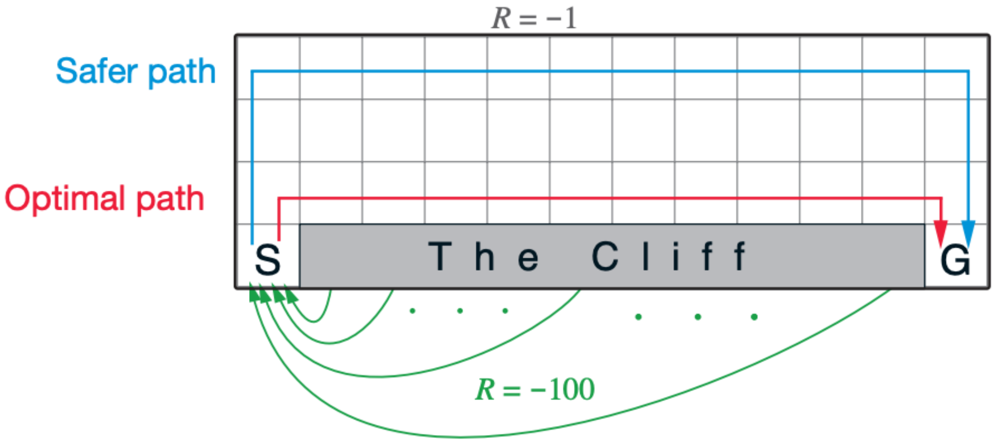
- No importance sampling is required: it is because the agent is estimating action values with unknown policy and it does not need important sampling ratios to correct for the difference in action selection.
- The action value function represents the returns following each action in a given state: the agents target policy represents the probability of taking each action in a given state.
- Putting these two elements together, the agent can calculate the expected return under its target policy from any given state,
- Q-learning uses exactly this technique to learn off-policy.
- Since the agents target policies greedy, with respect to its action values, all non-maximum actions have probability 0.
- As a result, the expected return from that state is equal to a maximal action value from that state.



A. White, M. White 'Sample-based Learning Methods'

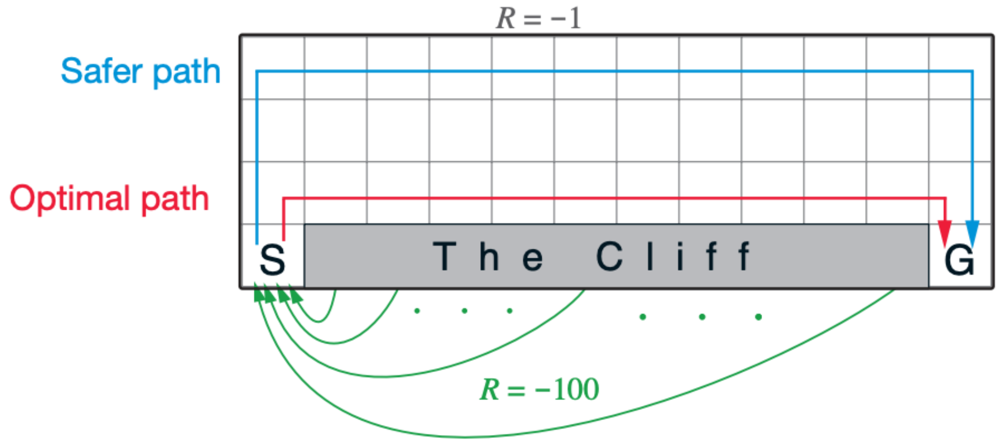
Control: why is Q-learning off-policy?

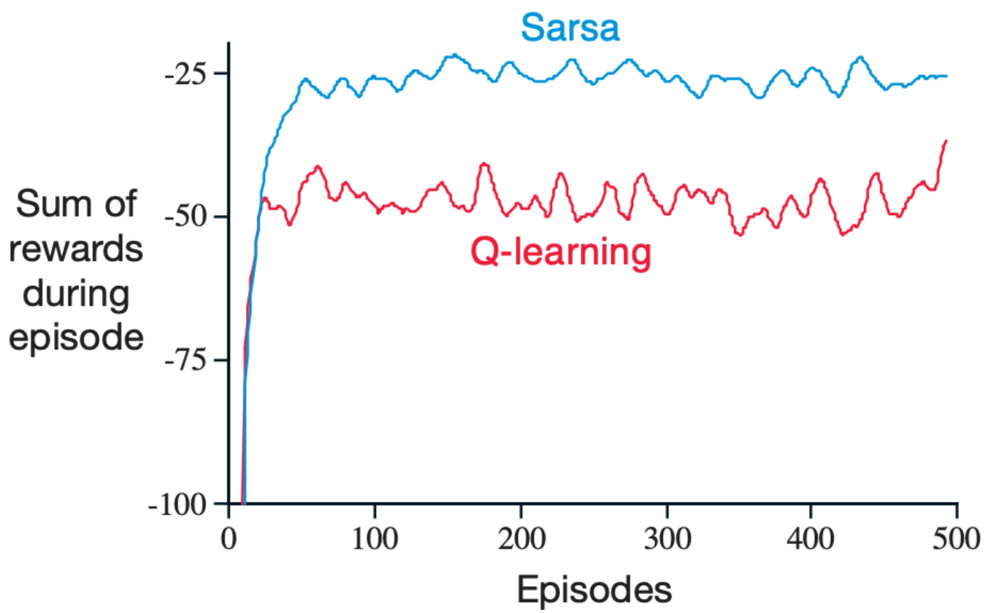
- Q-learning doesn't iterate between policy evaluation and policy improvement, but rather learns the optimal values directly. Not always ideal!



Control: why is Q-learning off-policy?

- Q-learning doesn't iterate between policy evaluation and policy improvement, but rather learns the optimal values directly. Not always ideal!
- Since Q-learning learns the optimal value function, it quickly learns that an optimal policy travels right alongside the cliff.
- However, since his actions are epsilon greedy, traveling alongside the cliff occasionally results and falling off of the cliff.
- Sarsa learns about his current policy, taking into account the effect of epsilon greedy action selection.





Credits

- Image of the course is taken from C. Mahoney ‘Reinforcement Learning’ https://towardsdatascience.com/reinforcement-learning-fda8ff535bb6
- Unified view of RL was taken from D. Silver ‘Reinforcement Learning’ course @ UCL

Thank you!
Questions?

Lecture #09
Temporal Difference Learning

Gian Antonio Susto