

10703 Deep Reinforcement Learning and Control

Russ Salakhutdinov

Machine Learning Department
rsalakhu@cs.cmu.edu

Temporal Difference Learning

Used Materials

- **Disclaimer:** Much of the material and slides for this lecture were borrowed from Rich Sutton's class and David Silver's class on Reinforcement Learning.

MC and TD Learning

- ▶ Goal: learn $v_\pi(s)$ from episodes of experience under policy π

- ▶ Incremental every-visit Monte-Carlo:

- Update value $V(S_t)$ toward actual return G_t

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

- ▶ Simplest Temporal-Difference learning algorithm: TD(0)

- Update value $V(S_t)$ toward estimated returns $R_{t+1} + \gamma V(S_{t+1})$

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

- ▶ $R_{t+1} + \gamma V(S_{t+1})$ is called the TD target

- ▶ $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$ is called the TD error.

DP vs. MC vs. TD Learning

- ▶ Remember:

$$\begin{aligned} v_{\pi}(s) &\doteq \mathbb{E}_{\pi}[G_t \mid S_t = s] \\ &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right] \\ &= \mathbb{E}_{\pi}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \mid S_t = s\right] \\ &= \mathbb{E}_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) \mid S_t = s]. \end{aligned}$$

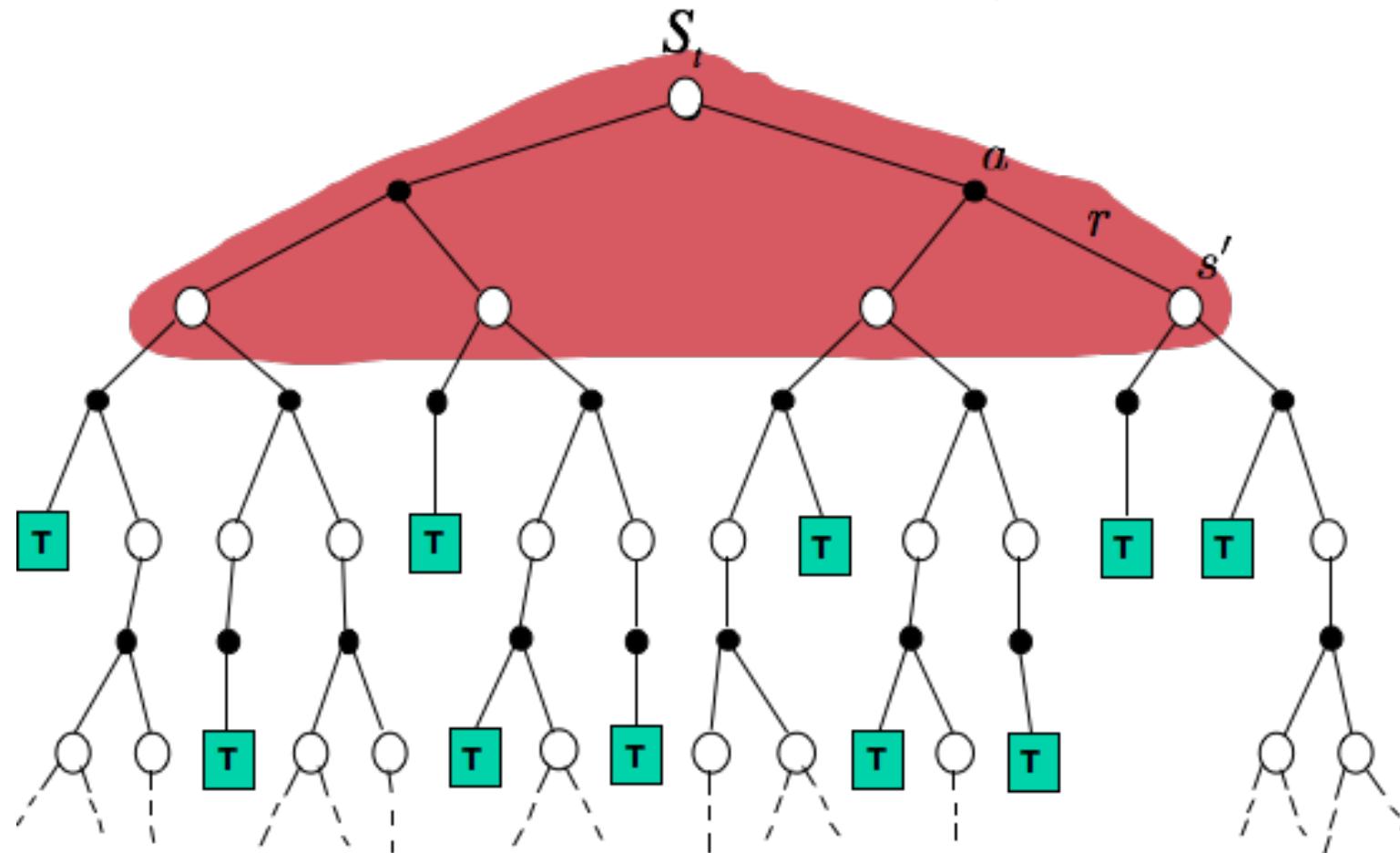
MC: sample average return approximates expectation

TD: combine both: Sample expected values and use a current estimate $V(S_{t+1})$ of the true $v_{\pi}(S_{t+1})$

DP: the expected values are provided by a model. But we use a current estimate $V(S_{t+1})$ of the true $v_{\pi}(S_{t+1})$

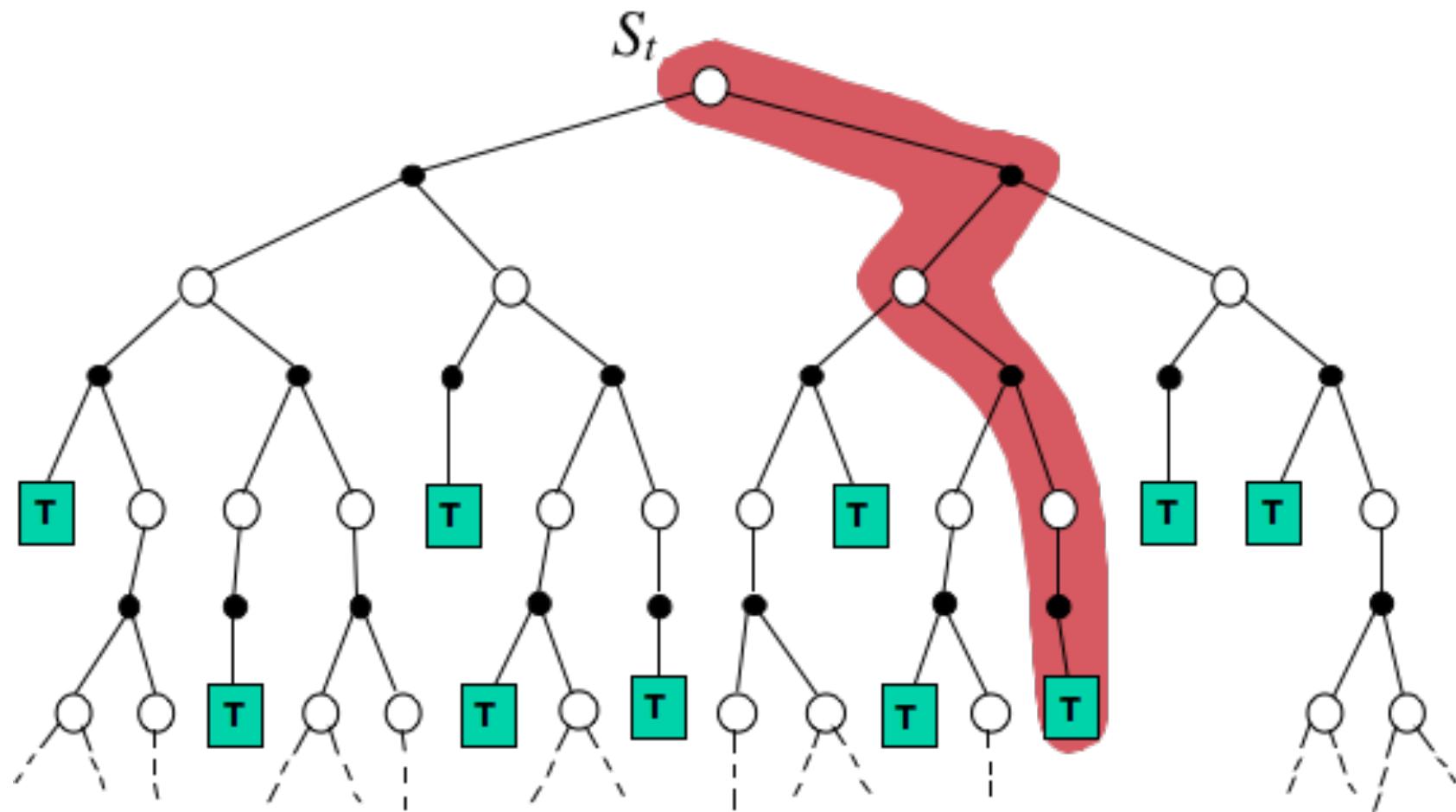
Dynamic Programming

$$V(S_t) \leftarrow E_{\pi} \left[R_{t+1} + \gamma V(S_{t+1}) \right] = \sum_a \pi(a|S_t) \sum_{s',r} p(s',r|S_t,a) [r + \gamma V(s')]$$



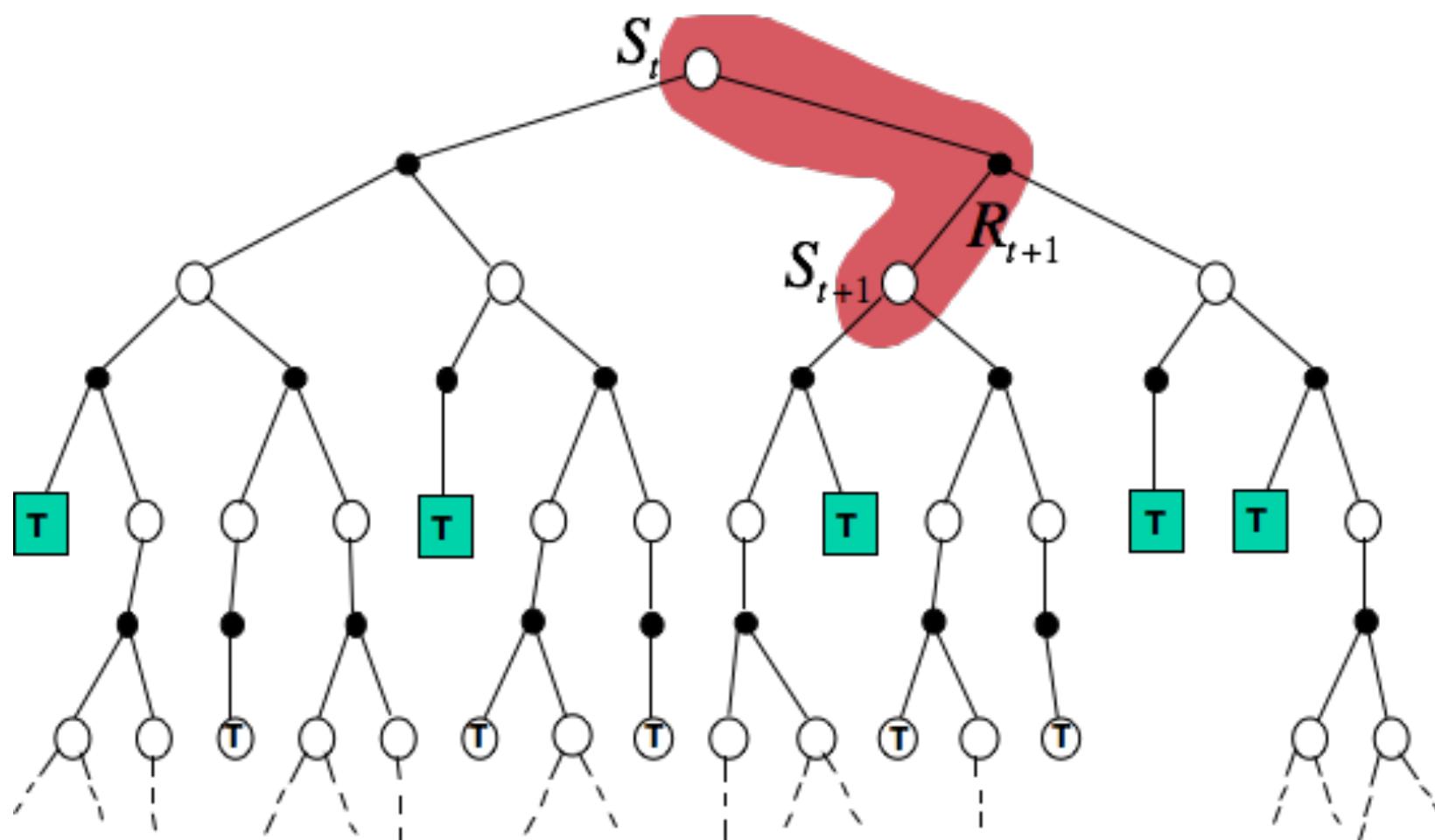
Monte Carlo

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



Simplest TD(0) Method

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



TD Methods Bootstrap and Sample

- ▶ **Bootstrapping:** update involves an estimate
 - MC does not bootstrap
 - DP bootstraps
 - TD bootstraps
- ▶ **Sampling:** update does not involve an expected value
 - MC samples
 - DP does not sample
 - TD samples

TD Prediction

- ▶ Policy Evaluation (the prediction problem):
 - for a given policy π , compute the state-value function v_π

- ▶ Remember: Simple every-visit Monte Carlo method:

$$V(S_t) \leftarrow V(S_t) + \alpha [G_t - V(S_t)]$$


target: the actual return after time t

- ▶ The simplest Temporal-Difference method TD(0):

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

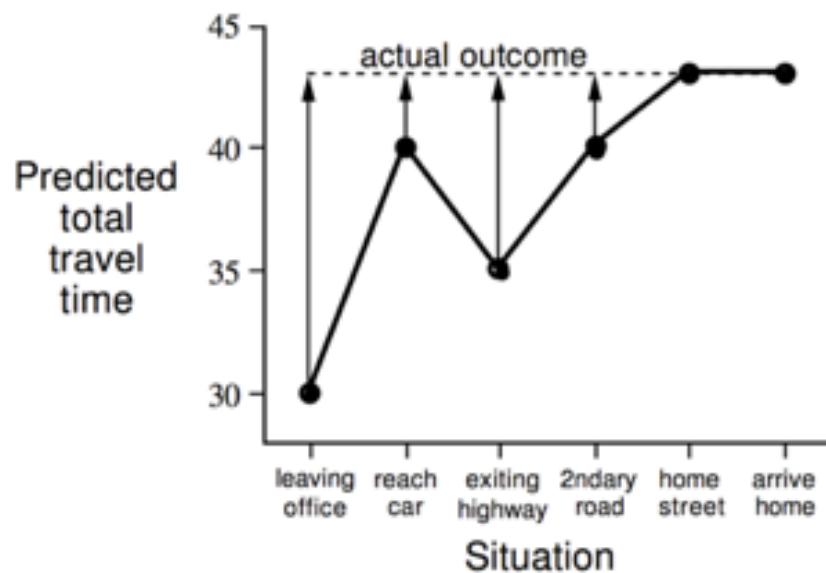

target: an estimate of the return

Example: Driving Home

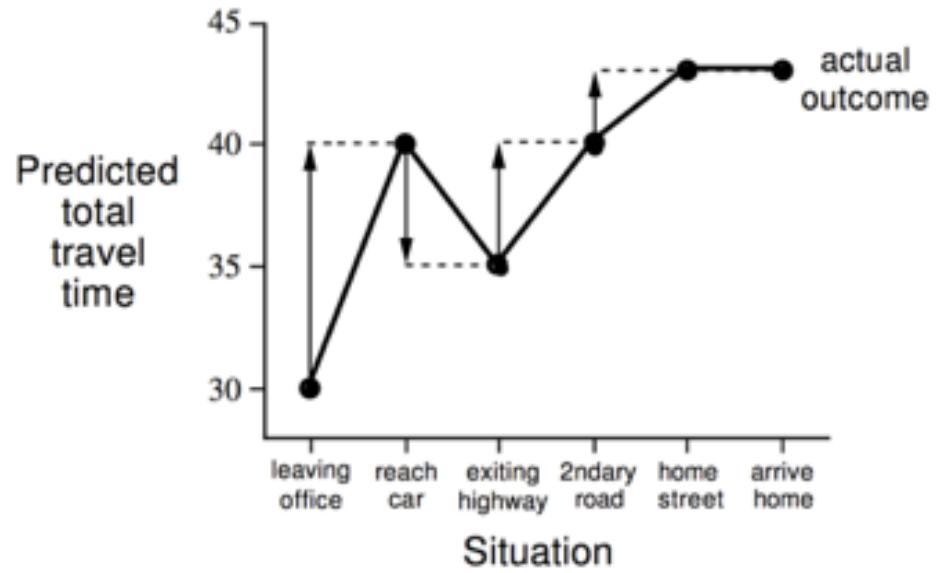
<i>State</i>	<i>Elapsed Time</i> (minutes)	<i>Predicted</i> <i>Time to Go</i>	<i>Predicted</i> <i>Total Time</i>
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Example: Driving Home

Changes recommended by
Monte Carlo methods ($\alpha=1$)



Changes recommended
by TD methods ($\alpha=1$)



Advantages of TD Learning

- ▶ TD methods **do not require a model of the environment**, only experience
- ▶ TD, but not MC, methods can be **fully incremental**
- ▶ You can learn **before** knowing the final outcome
 - Less memory
 - Less computation
- ▶ You can learn **without** the final outcome
 - From incomplete sequences
- ▶ Both MC and TD converge (under certain assumptions to be detailed later), but which is faster?

Bias-Variance Trade-Off

- ▶ Monte-Carlo: Update value $V(S_t)$ toward actual return G_t

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

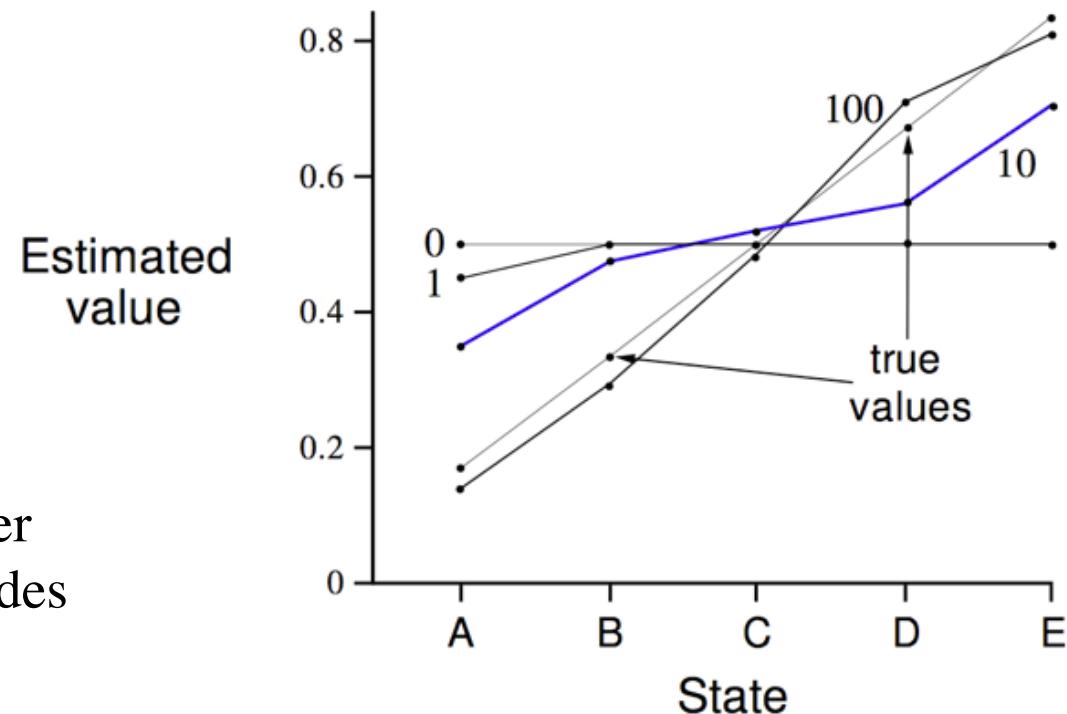
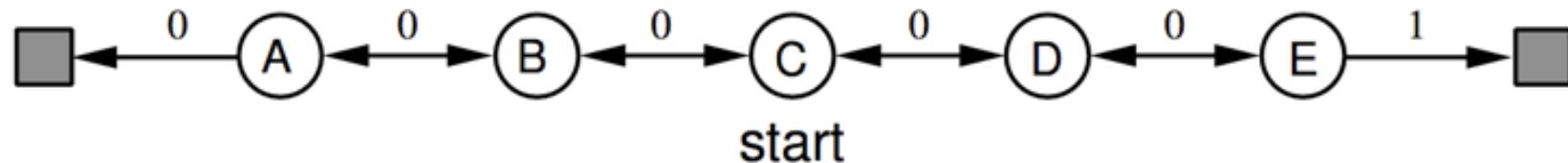
- ▶ Return $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$ is **unbiased estimate** of $v_\pi(S_t)$
- ▶ TD: Update value $V(S_t)$ toward estimated returns $R_{t+1} + \gamma V(S_{t+1})$
$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$
- ▶ True TD target: $R_{t+1} + \gamma v_\pi(S_{t+1})$ is **unbiased estimate** of $v_\pi(S_t)$
- ▶ TD target: $R_{t+1} + \gamma V(S_{t+1})$ is **biased estimate** of $v_\pi(S_t)$
- ▶ TD target is much lower variance than the return:
 - Return depends on many random actions, transitions, rewards
 - TD target depends on one random action, transition, reward

Bias-Variance Trade-Off

- ▶ MC has high variance, zero bias
 - Good convergence properties
 - Even with function approximation
 - Not very sensitive to initial value
 - Very simple to understand and use

- ▶ TD has low variance, some bias
 - Good Usually more efficient than MC
 - TD(0) converges to $v_{\pi}(s)$
 - More sensitive to initial value

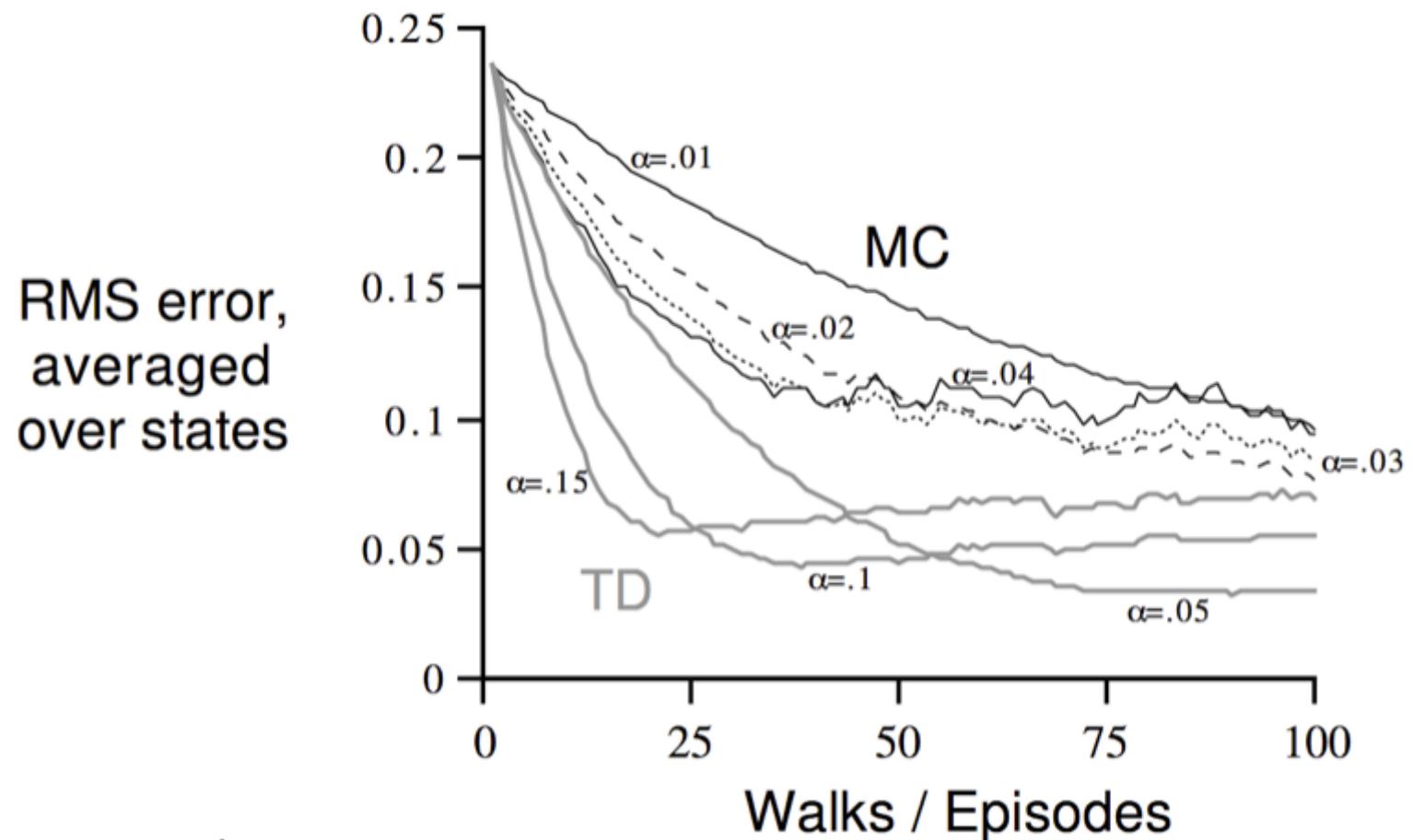
Random Walk Example



Values learned by TD after various numbers of episodes

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

TD and MC on the Random Walk

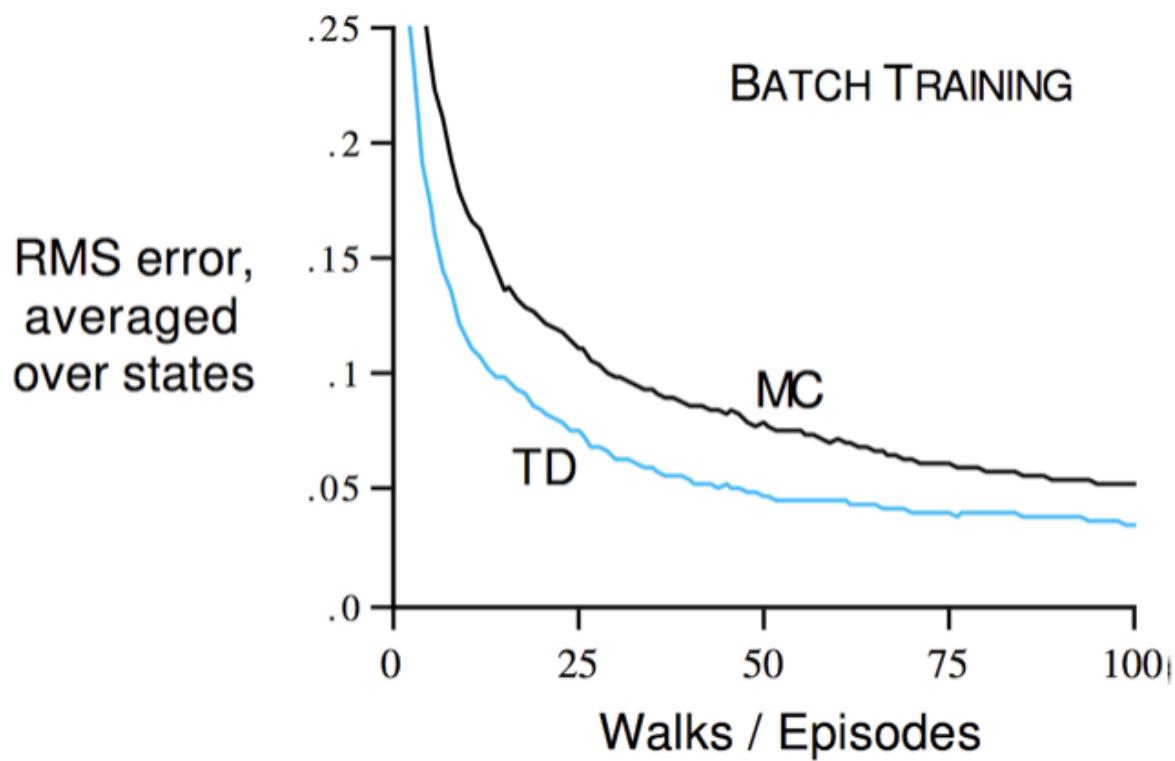


Data averaged over
100 sequences of episodes

Batch Updating in TD and MC methods

- ▶ Batch Updating: train completely on a finite amount of data,
 - e.g., train repeatedly on 10 episodes until convergence.
- ▶ Compute updates according to TD or MC, but only update estimates after each complete pass through the data.
- ▶ For any finite Markov prediction task, under batch updating, TD converges for sufficiently small α .
- ▶ Constant- α MC also converges under these conditions, but may converge to a different answer.

Random Walk under Batch Updating



- After each new episode, all previous episodes were treated as a batch, and algorithm was trained until convergence. All repeated 100 times.

AB Example

- ▶ Suppose you observe the following 8 episodes:

A, 0, B, 0

B, 1

B, 1

$V(B)?$ 0.75

B, 1

$V(A)?$ 0?

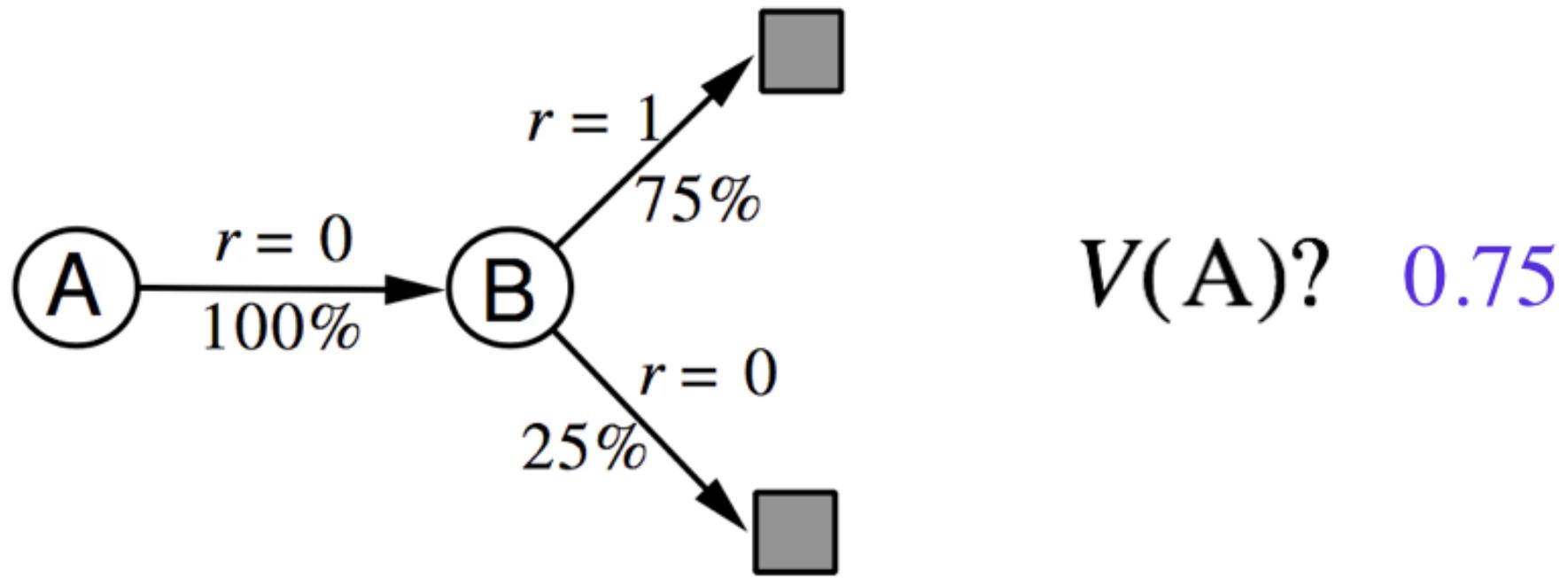
B, 1

B, 1

B, 0

- ▶ Assume Markov states, no discounting ($\gamma = 1$)

AB Example



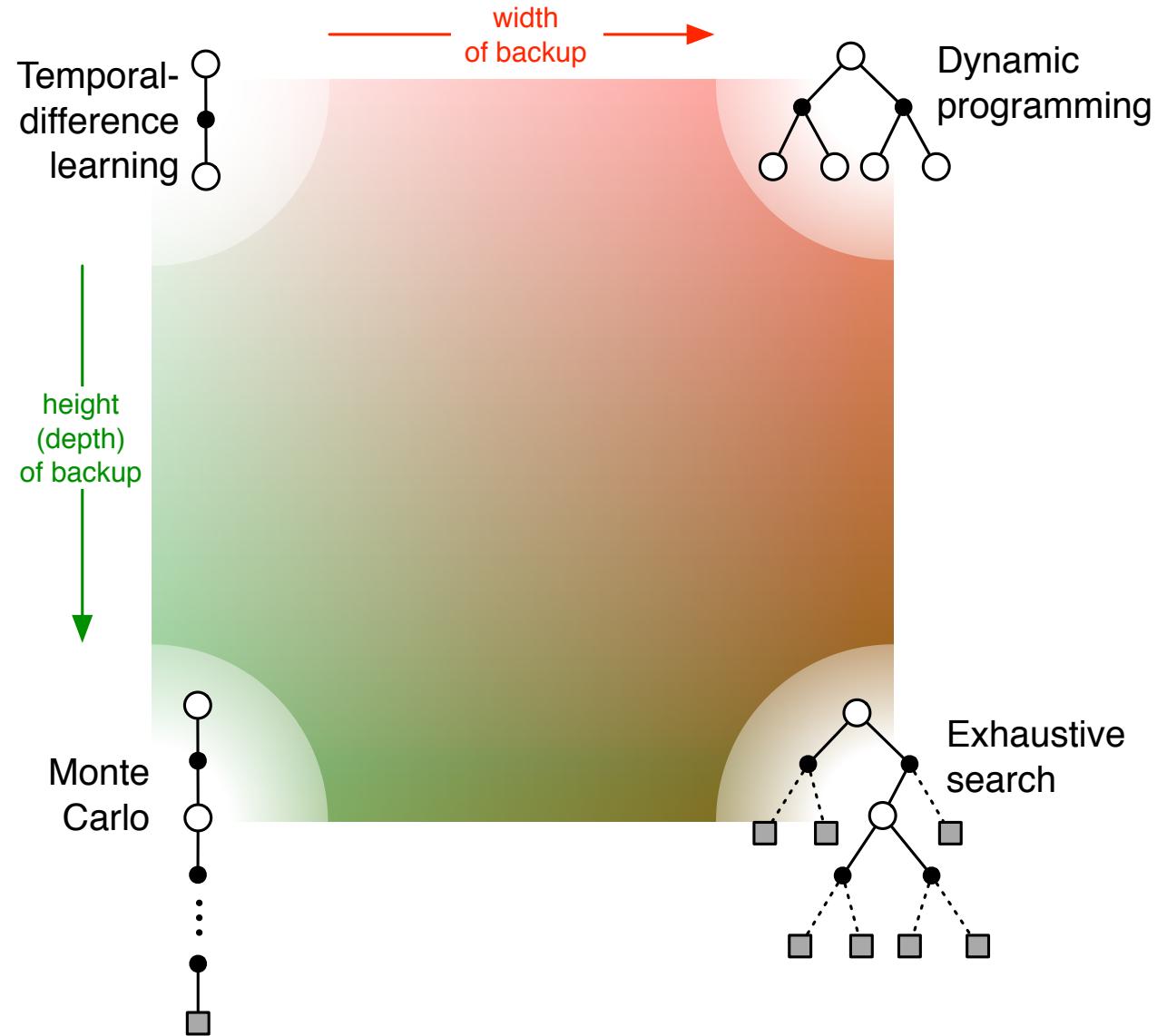
AB Example

- ▶ The prediction that best matches the training data is $V(A)=0$
 - This minimizes the mean-square-error on the training set
 - This is what a batch Monte Carlo method gets
- ▶ If we consider the sequentiality of the problem, then we would set $V(A)=.75$
 - This is correct for the maximum likelihood estimate of a Markov model generating the data
 - i.e, if we do a best fit Markov model, and assume it is exactly correct, and then compute what it predicts.
 - This is called the certainty-equivalence estimate
 - This is what TD gets

Summary so far

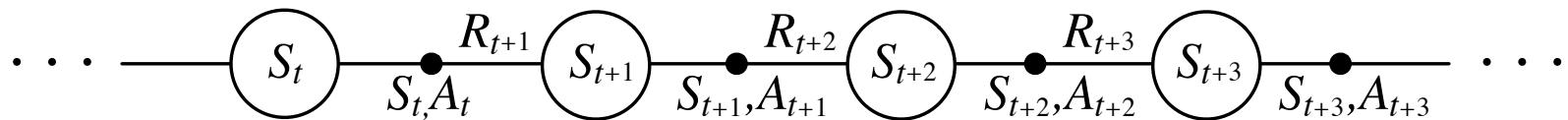
- ▶ Introduced one-step tabular model-free TD methods
- ▶ These methods bootstrap and sample, combining aspects of DP and MC methods
- ▶ TD methods are computationally congenial
- ▶ If the world is truly Markov, then TD methods will learn faster than MC methods

Unified View



Learning An Action-Value Function

- ▶ Estimate q_{π} for the **current policy** π



After every transition from a nonterminal state, S_t , do this:

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

If S_{t+1} is terminal, then define $Q(S_{t+1}, A_{t+1}) = 0$

Sarsa: On-Policy TD Control

- Turn this into a control method by always updating the policy to be **greedy** with respect to the current estimate:

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

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'

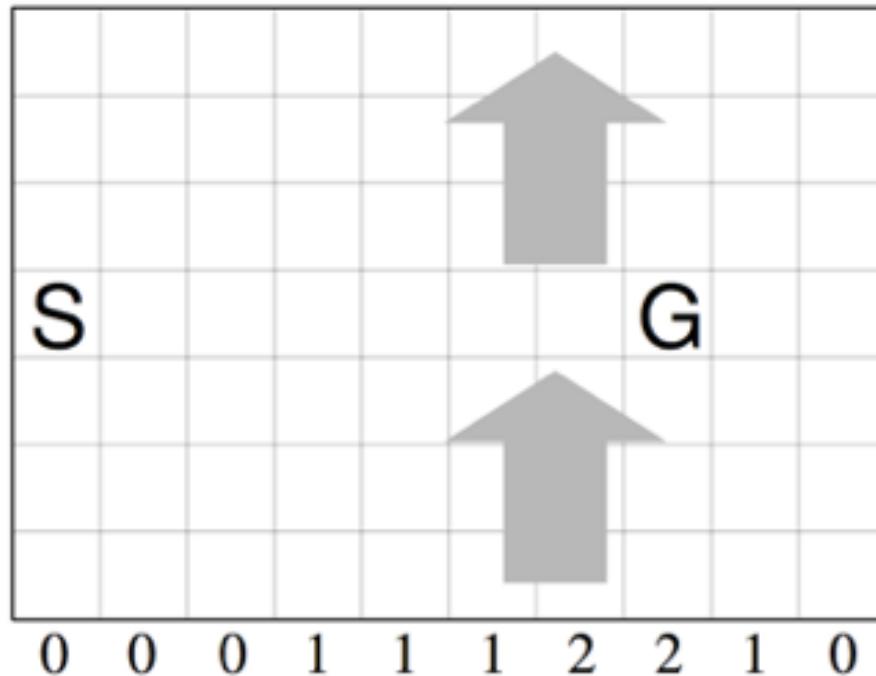
 Choose A' from S' using policy derived from Q (e.g., ε -greedy)

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

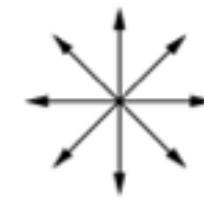
$S \leftarrow S'; A \leftarrow A'$;

 until S is terminal

Windy Gridworld



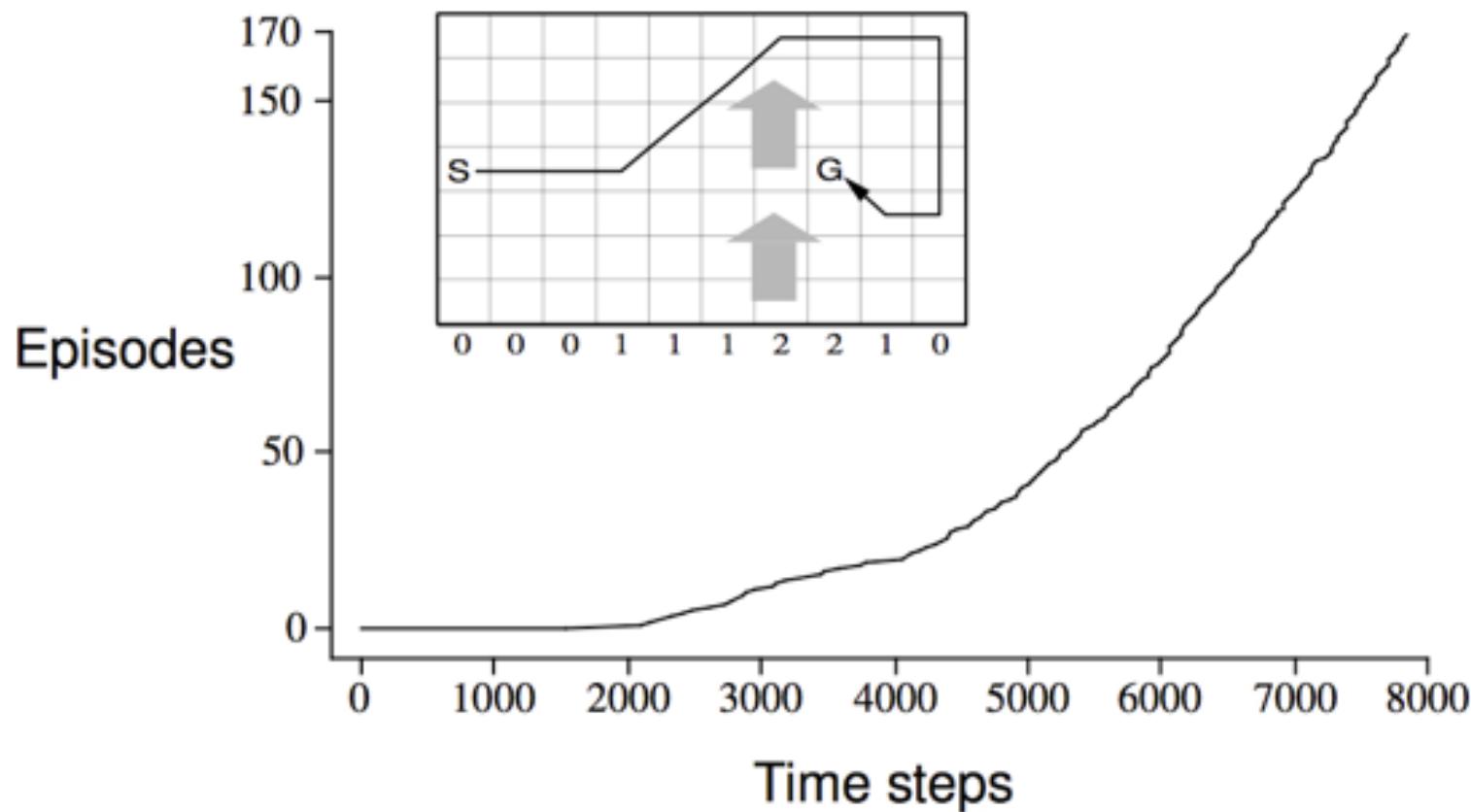
standard
moves



king's
moves

- undiscounted, episodic, reward = -1 until goal

Results of Sarsa on the Windy Gridworld



Q-Learning: Off-Policy TD Control

- ▶ One-step Q-learning:

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

Initialize $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\text{terminal-state}, \cdot) = 0$

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)

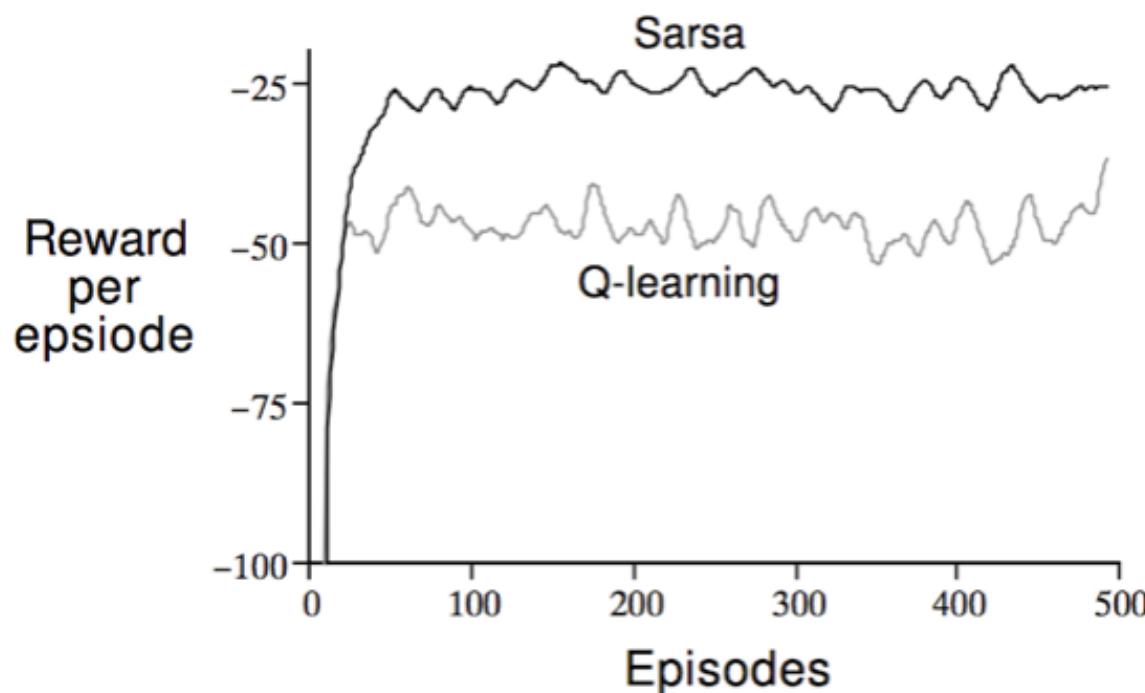
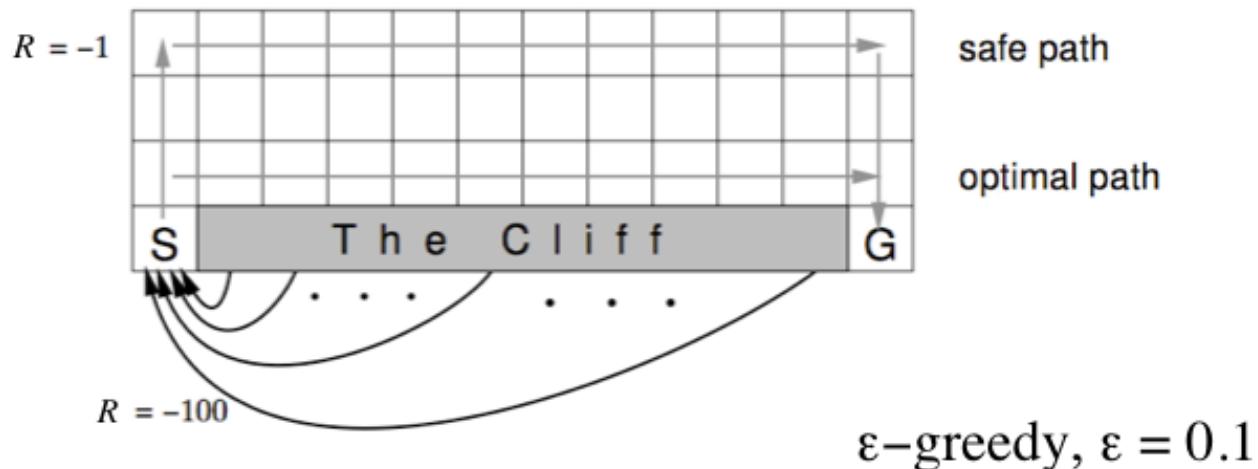
 Take action A , observe R, S'

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

$S \leftarrow S'$;

 until S is terminal

Cliffwalking



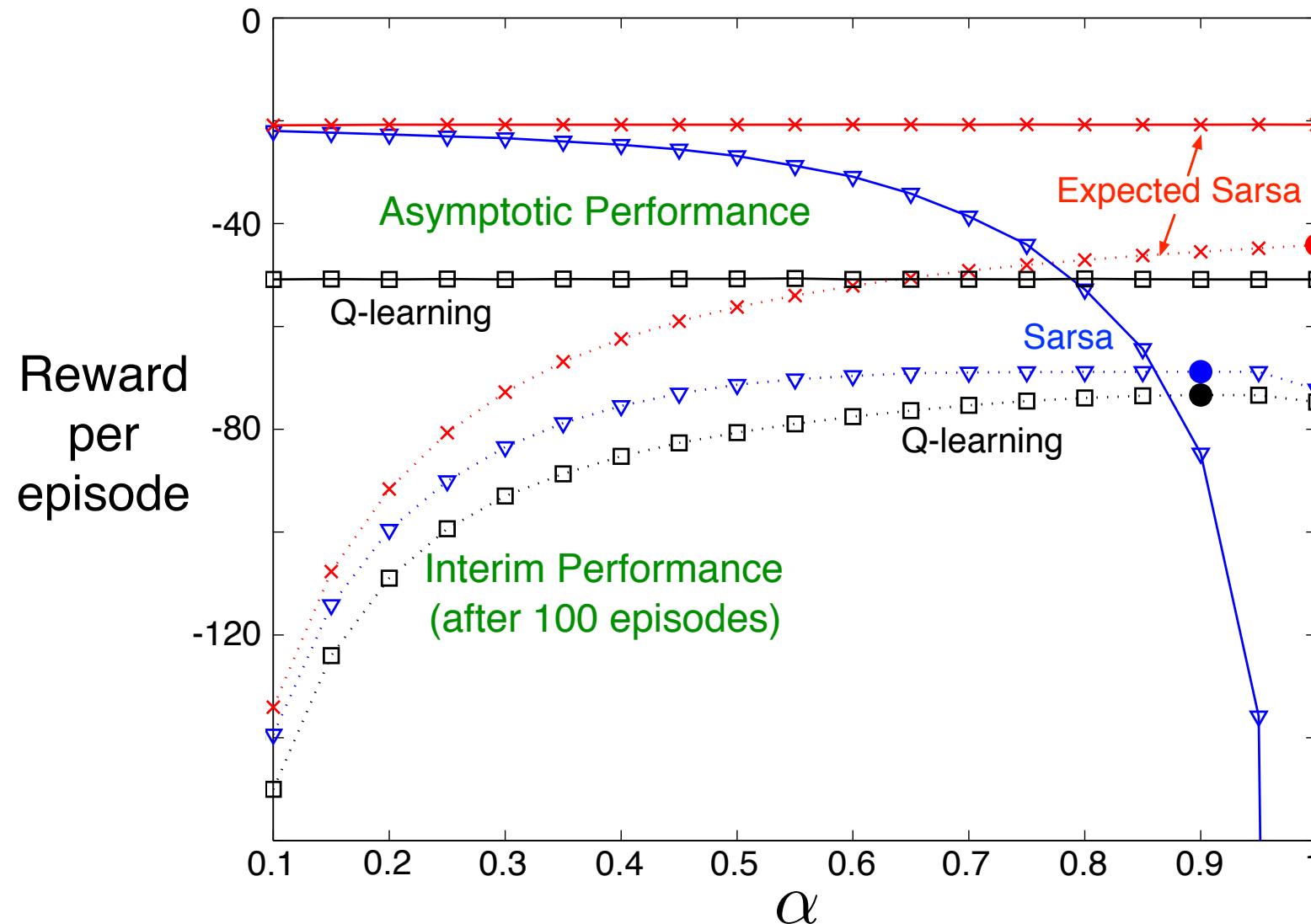
Expected Sarsa

- ▶ Instead of the sample value-of-next-state, use the expectation!

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t) \right] \\ &\leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \sum \pi(a|S_{t+1})Q(S_{t+1}, a) - Q(S_t, A_t) \right] \end{aligned}$$

- ▶ Expected Sarsa performs better than Sarsa (but costs more)

Performance on the Cliff-walking Task



Summary

- ▶ Introduced one-step tabular model-free TD methods
- ▶ These methods bootstrap and sample, combining aspects of DP and MC methods
- ▶ TD methods are computationally congenial
- ▶ If the world is truly Markov, then TD methods will learn faster than MC methods
- ▶ Extend prediction to control by employing some form of GPI
 - On-policy control: Sarsa, Expected Sarsa
 - Off-policy control: Q-learning