

# Lecture 5: Model-free RL (Part II)

Lei Zhang

# Outline

---

- TD Prediction
- TD Control
- Summary: DP vs. TD

\*Materials are modified from David Silver's RL lecture notes

# Outline

---

- TD Prediction
- TD Control
- Summary: DP vs. TD

# MC Learning (Refresher)

---

- Update  $V(s)$  incrementally after episode  $S_1, A_1, R_2, \dots, S_T$   
For each state  $S_t$  with return  $G_t$

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)} (G_t - V(S_t))$$

- In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes.

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

# Temporal Difference (TD) Learning

---

- TD methods learn directly from episodes of experience
- TD is *model-free*: no knowledge of MDP transitions / rewards
- TD learns from *incomplete* episodes, by *bootstrapping*
- TD updates a guess towards a guess

# TD Prediction

---

Goal: learn  $v_\pi$  online from experience under policy  $\pi$

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)

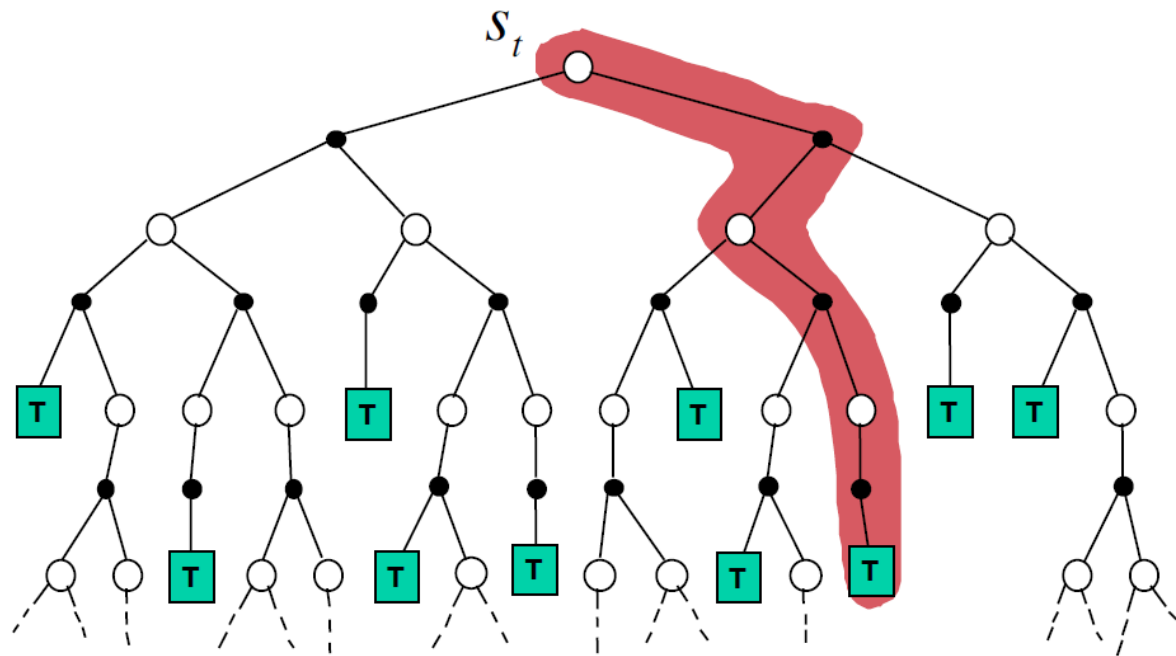
- Update value  $V(S_t)$  toward *estimated* return  $R_{t+1} + \gamma V(S_{t+1})$

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

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

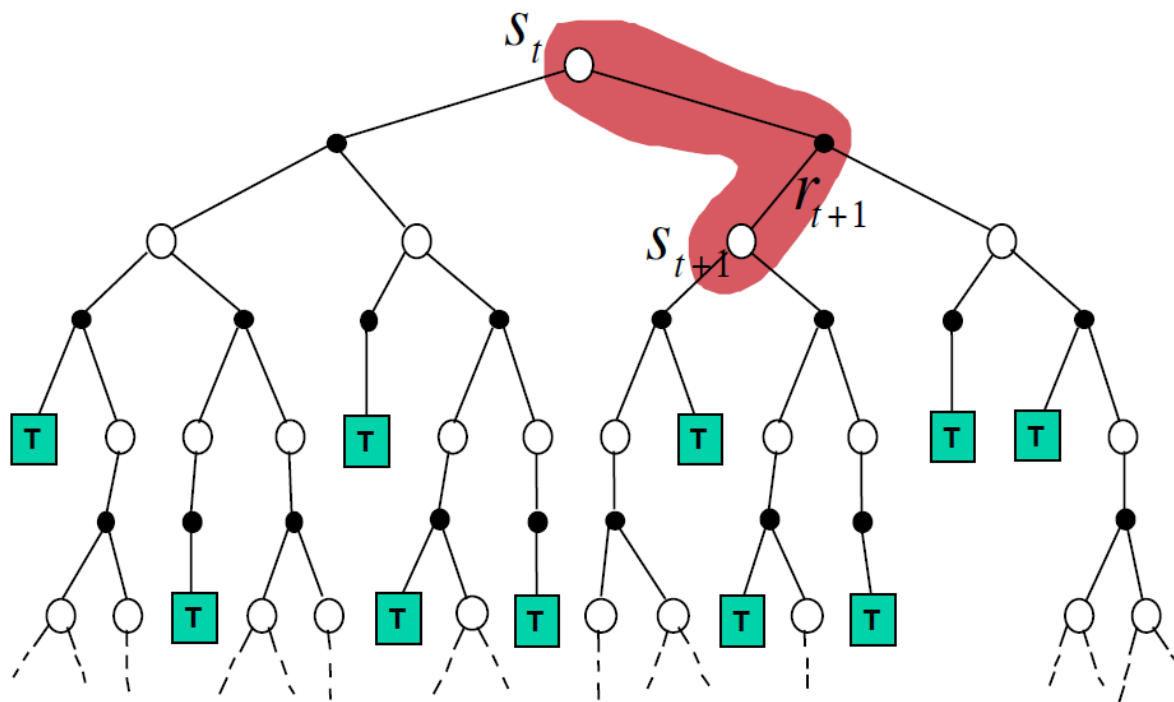
# MC backup

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



# TD Backup

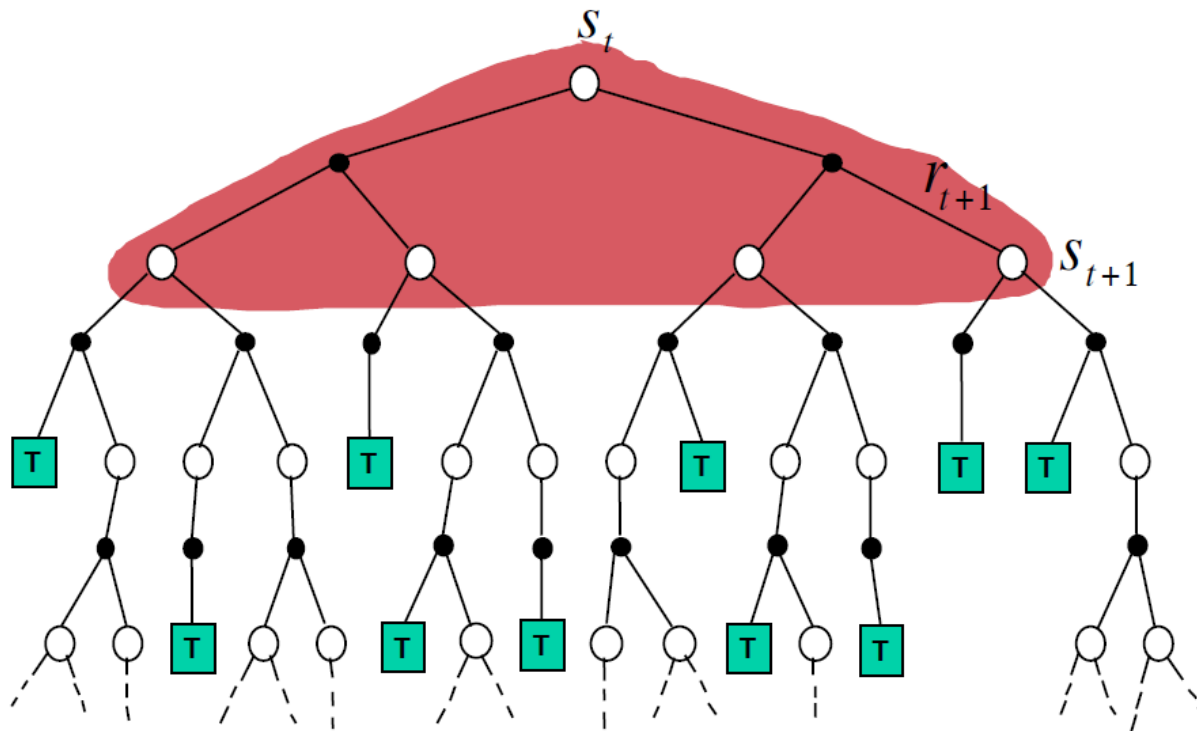
$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$





# DP Backup

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



# Bootstrapping & Sampling

---

**Bootstrapping:** update involves an estimate

- MC does not bootstrap
- DP bootstraps
- TD bootstraps

**Sampling:** update samples an expectation

- MC samples
- DP does not sample
- TD samples

# MC vs. TD

---

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

- TD can learn from incomplete sequences
- MC can only learn from complete sequences
- TD works in continuing (non-terminating) environments
- MC only works for episodic (terminating) environments

# MC vs. TD

---

- Return  $G_t = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$  is *unbiased* estimate of  $v_\pi(S_t)$
- 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

# MC vs. TD

---

MC has high variance, zero bias

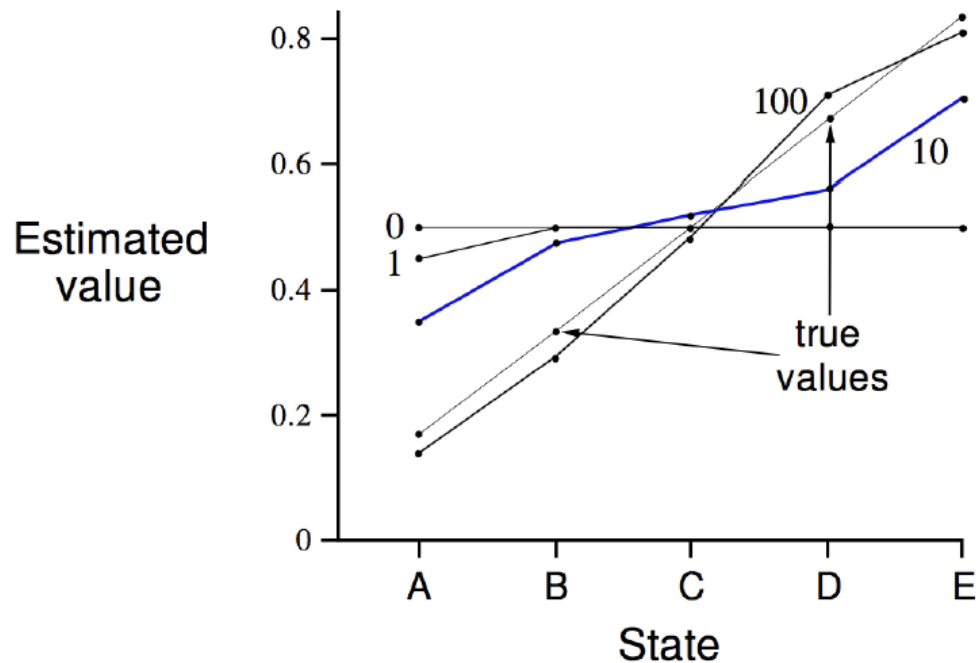
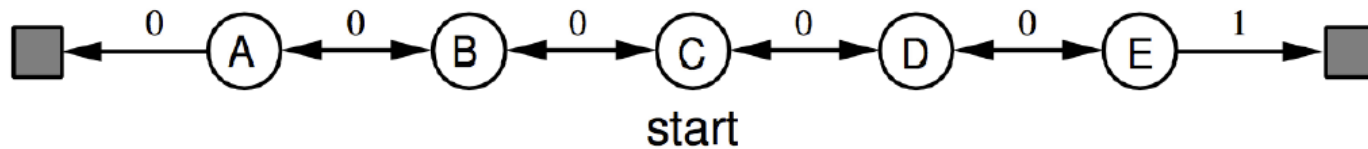
- Good convergence properties
- (even with function approximation)
- Not very sensitive to initial value
- Very simple to understand and use

TD has low variance, some bias

- Usually more efficient than MC
- TD(0) converges to  $v_{\pi}(s)$
- (but not always with function approximation)
- More sensitive to initial value

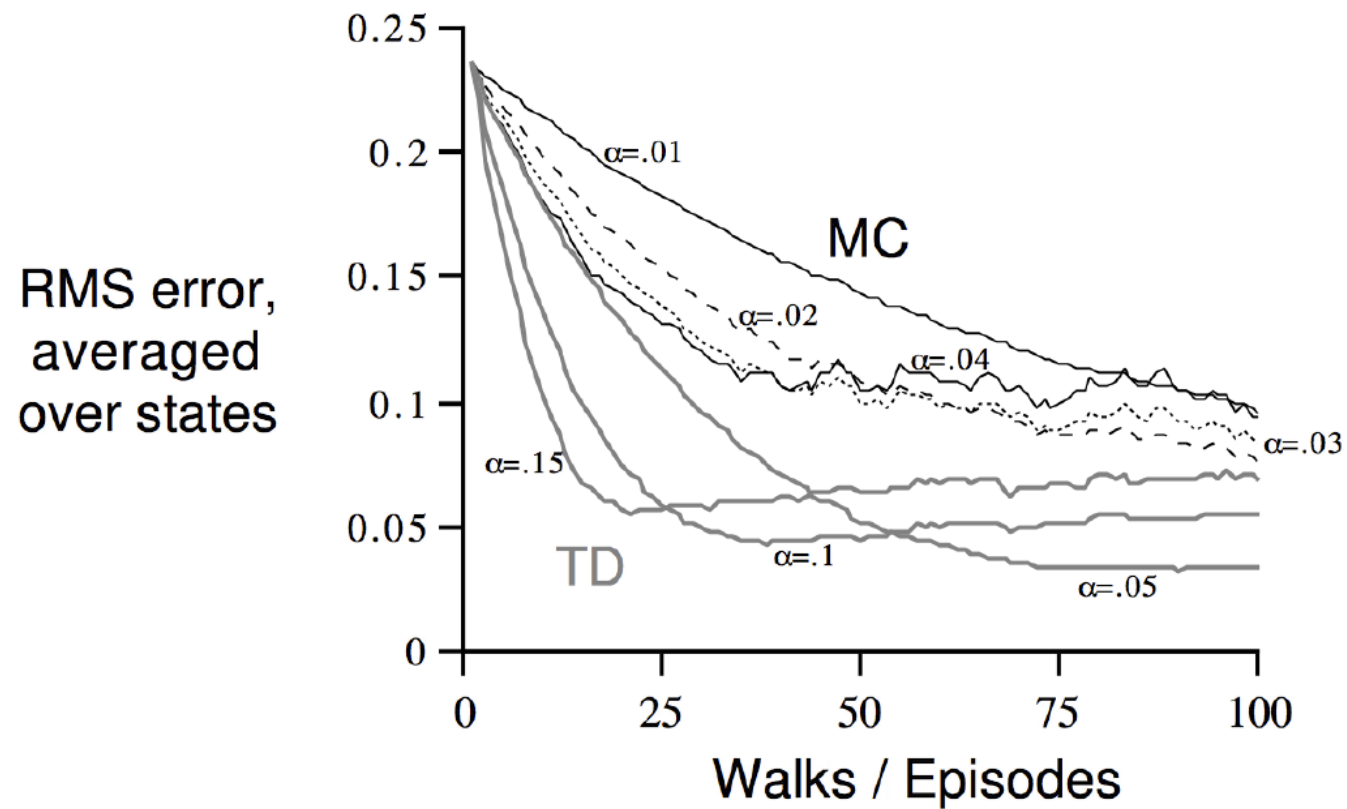
In practice, TD methods have usually been found to converge faster than constant step-size MC methods on stochastic tasks

# Example: Random Walk



- All episodes start in the center state C.
- Move either left or right by one state for each step, with equal probability.

# Example: Random Walk



# Batch Update

---

- MC and TD converge:  $V(s) \rightarrow v_{\pi}(s)$  as experience  $\rightarrow \infty$
- But what about batch solution for finite experience?

$$s_1^1, a_1^1, r_2^1, \dots, s_{T_1}^1$$

$$\vdots$$

$$s_1^K, a_1^K, r_2^K, \dots, s_{T_K}^K$$



# A-B Example

---

Two states  $A, B$ ; no discounting; 8 episodes of experience

$A, 0, B, 0$

$B, 1$

$B, 1$

$B, 1$

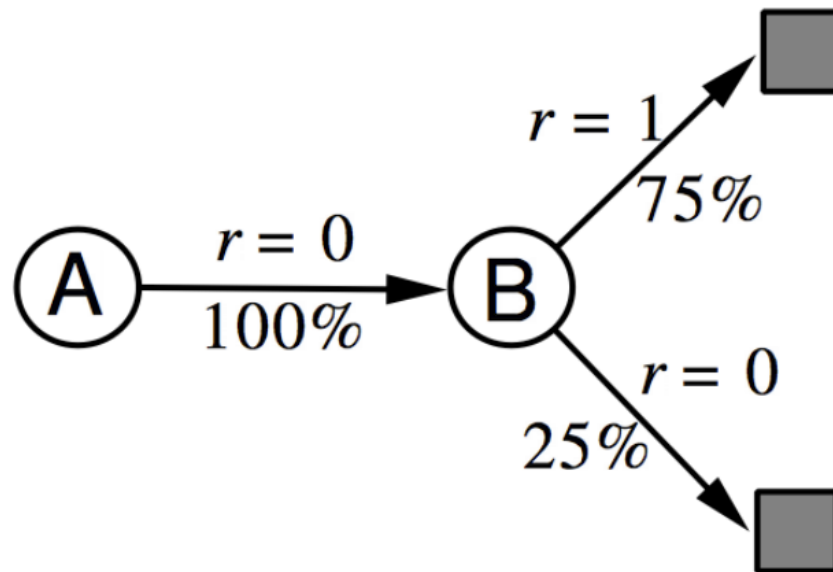
$B, 1$

$B, 1$

$B, 1$

$B, 0$

What is  $V(A), V(B)$ ?



# MC vs. TD

---

MC converges to solution with minimum mean-squared error

- Best fit to the observed returns

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

- In the AB example,  $V(A) = 0$

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

- Solution to the MDP  $\langle \mathcal{S}, \mathcal{A}, \hat{\mathcal{P}}, \hat{\mathcal{R}}, \gamma \rangle$  that best fits the data

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

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

- In the AB example,  $V(A) = 0.75$

# Outline

---

- TD Prediction
- TD Control
- Summary: DP vs. TD

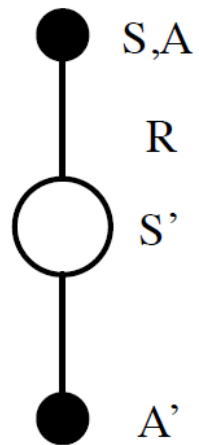
# TD Control

---

- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to  $Q(S, A)$
  - Use  $\epsilon$ -greedy policy improvement
  - Update every time-step

# Sarsa

---



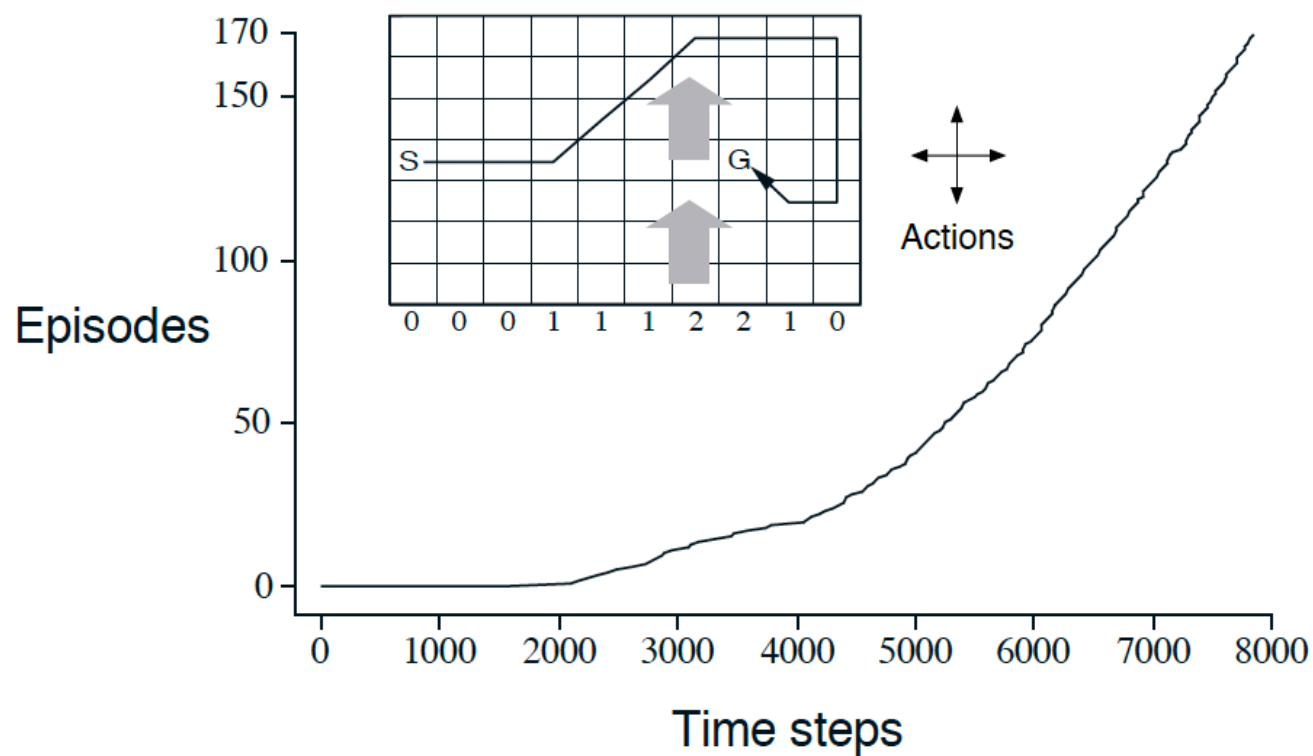
$$Q(S, A) \leftarrow Q(S, A) + \alpha (R + \gamma Q(S', A') - Q(S, A))$$

# Sarsa for On-Policy Control

---

```
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.,  $\epsilon$ -greedy)  
  Repeat (for each step of episode):  
    Take action  $A$ , observe  $R, S'$   
    Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -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
```

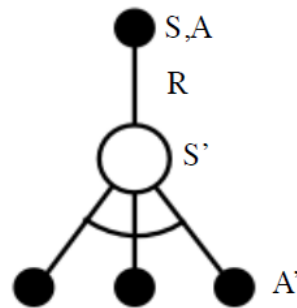
# Example: Windy Gridworld



# Q-Learning

---

- One of the most important breakthroughs in RL
- An Off-policy TD control algorithm (Watkins, 1989), why off-policy?



$$Q(S, A) \leftarrow Q(S, A) + \alpha \left( R + \gamma \max_{a'} Q(S', a') - Q(S, A) \right)$$

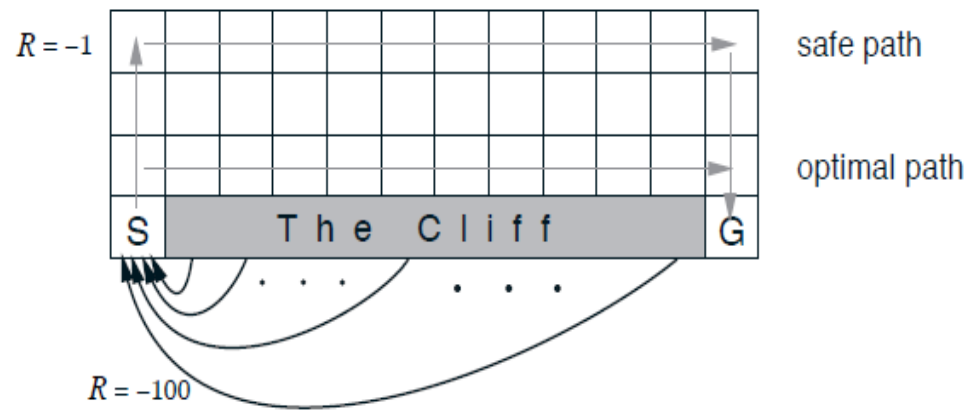


# Q-Learning for Off-Policy Control

---

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.,  $\epsilon$ -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

# Example: Cliff Walking



- $\epsilon = 0.1$ , Q-learning is worse than Sarsa, why?
- If  $\epsilon \rightarrow 0$ , Q-learning vs. Sarsa ?

# Extensions

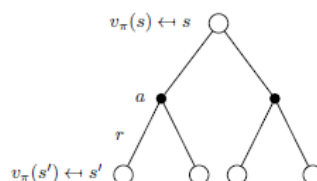
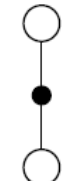
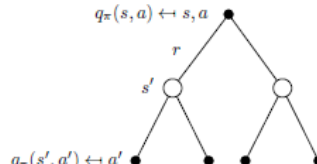
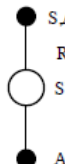
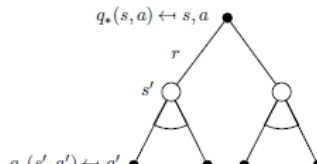
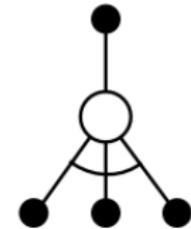
---

- Expected Sarsa

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

- Off-policy or on-policy?
- Better than Sarsa or not?
- See “A Theoretical and Empirical Analysis of Expected Sarsa” (Seijen 09’)

# Summary: DP vs. TD

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

# Summary: DP vs. TD

---

<i>Full Backup (DP)</i>	<i>Sample Backup (TD)</i>
Iterative Policy Evaluation $V(s) \leftarrow \mathbb{E}[R + \gamma V(S') \mid s]$	TD Learning $V(S) \stackrel{\alpha}{\leftarrow} R + \gamma V(S')$
Q-Policy Iteration $Q(s, a) \leftarrow \mathbb{E}[R + \gamma Q(S', A') \mid s, a]$	Sarsa $Q(S, A) \stackrel{\alpha}{\leftarrow} R + \gamma Q(S', A')$
Q-Value Iteration $Q(s, a) \leftarrow \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S', a') \mid s, a\right]$	Q-Learning $Q(S, A) \stackrel{\alpha}{\leftarrow} R + \gamma \max_{a' \in \mathcal{A}} Q(S', a')$

where  $x \stackrel{\alpha}{\leftarrow} y \equiv x \leftarrow x + \alpha(y - x)$