Decision Making and Reinforcement Learning

Module 7: Temporal-Difference Learning

Tony Dear, Ph.D.
Department of Computer Science
School of Engineering and Applied Sciences

Topics

- Temporal difference prediction
- Batch learning methods
- SARSA (on-policy control)
- Q-learning (off-policy control)

Learning Objectives

- **Implement** TD prediction for a given policy
- **Apply** TD methods in batch and/or offline
- **Implement** the SARSA and Q-learning algorithms
- **Compare and contrast** properties and characteristics of on-policy vs off-policy TD learning

Review: Monte Carlo Prediction

- *Prediction*: Estimate state values for given policy π
- MC methods average observed utilities from multiple episodes

$$V^{\pi}(s) = \frac{1}{N}\sum_{i=1}^{N} G_i(s)$$

Review: Monte Carlo Prediction

- *Prediction*: Estimate state values for given policy π
- MC methods average observed utilities from multiple episodes

$$V^{\pi}(s) = \frac{1}{N}\sum_{i=1}^{N} G_i(s)$$

- In our implementation, we used a "moving average" calculation
- Suppose $N(s)$ is the number of visits to state s prior to current episode

$$V^{\pi}(s) \leftarrow \frac{N(s)V^{\pi}(s) + G_i(s)}{N(s) + 1}$$

- Let’s rewrite the state value update expression…

$$V^{\pi}(s) \leftarrow \frac{N(s)V^{\pi}(s) + G_i(s)}{N(s)+1} = V^{\pi}(s) + \frac{1}{N(s)+1}\big(G_i(s) - V^{\pi}(s)\big)$$

- “Old estimate” + “step size” × (“target” – “old estimate”)

- Let's rewrite the state value update expression…

$$V^{\pi}(s) \leftarrow \frac{N(s)V^{\pi}(s) + G_i(s)}{N(s)+1} = V^{\pi}(s) + \frac{1}{N(s)+1}\left(G_i(s) - V^{\pi}(s)\right)$$

- "Old estimate" + "step size" × ("target" – "old estimate")

- We can also choose to give more weight to *recent* returns
- **Constant-α MC** *exponentially* decays the weights on past returns by learning rate α

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \boldsymbol{\alpha}\left(G_i(s) - V^{\pi}(s)\right)$$

Example: Constant-α MC

- Suppose $V(s) = 0$; consider the reward sequence $+3, -2, +1$

- Moving average MC:

- $V(s) = 0 + \frac{1}{1}(3 - 0) = 3$
- $V(s) = 3 + \frac{1}{2}(-2 - 3) = \frac{1}{2}$
- $V(s) = \frac{1}{2} + \frac{1}{3}\left(1 - \frac{1}{2}\right) = \frac{2}{3}$

Example: Constant-α MC

- Suppose $V(s) = 0$; consider the reward sequence $+3, -2, +1$

- Moving average MC:
- $V(s) = 0 + \frac{1}{1}(3 - 0) = 3$
- $V(s) = 3 + \frac{1}{2}(-2 - 3) = \frac{1}{2}$
- $V(s) = \frac{1}{2} + \frac{1}{3}\left(1 - \frac{1}{2}\right) = \frac{2}{3}$

- Constant α MC, $\alpha = 0.5$:
- $V(s) = 0 + \frac{1}{2}(3 - 0) = 1.5$
- $V(s) = 1.5 + \frac{1}{2}(-2 - 1.5) = -0.25$
- $V(s) = -0.25 + \frac{1}{2}(1 + 0.25) = 0.375$

Temporal-Difference Learning

- MC requires episodic structure—what about infinite horizon problems?
- Recall from DP: $V^{\pi}(s)$ depends on values of successors of s

Temporal-Difference Learning

- MC requires episodic structure—what about infinite horizon problems?
- Recall from DP: $V^{\pi}(s)$ depends on values of successors of s

- **One-step TD ($\boldsymbol{TD(0)}$)**: Replace the *target* term with the sum of immediate reward with discounted successor state value!

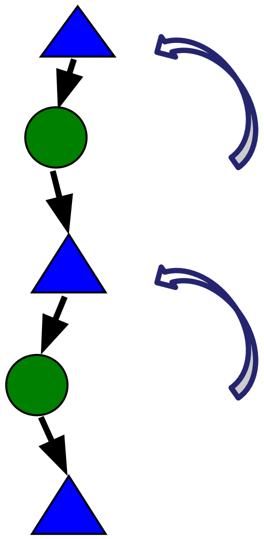
$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \overbrace{\Big(\underbrace{r_{t+1} + \gamma V^{\pi}(s')}_{\text{Target}} - V^{\pi}(s)\Big)}^{\text{TD error } \delta_t}$$

- **Given:** Policy π, *learning rate* α between 0 and 1

- **Initialize** $V^{\pi}(s) \leftarrow 0$

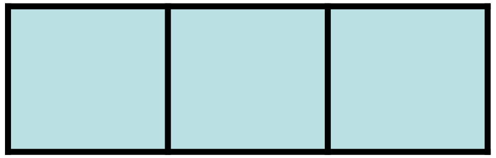
- **Given:** Policy π, *learning rate* α between 0 and 1

- **Initialize** $V^{\pi}(s) \leftarrow 0$
- **Loop**:
  - **Initialize** starting state s if needed
  - **Generate** sequence $(s, \pi(s), r, s')$



Example: Mini-Gridworld

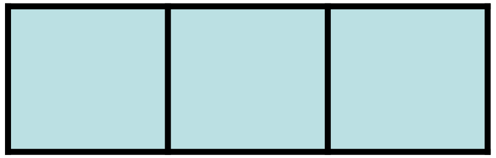
- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states



- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

Example: Mini-Gridworld

- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states



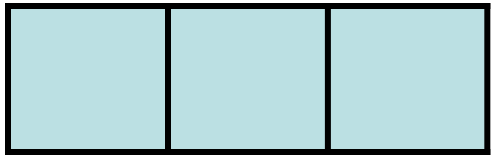
- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

Transition	
	0
	0

$$V^{\pi}(A) \leftarrow V^{\pi}(A) + \alpha\big(r + \gamma V^{\pi}(A) - V^{\pi}(A)\big)$$

$$V^{\pi}(A) \leftarrow 0 + 0.5(3 + 0.8(0) - 0)$$

Example: Mini-Gridworld



- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states
- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

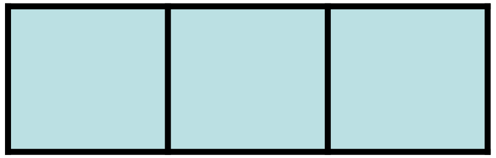
Transition		
	0	0
	0	0

$$V^{\pi}(A) \leftarrow V^{\pi}(A) + \alpha\big(r + \gamma V^{\pi}(B) - V^{\pi}(A)\big)$$

$$V^{\pi}(A) \leftarrow 1.5 + 0.5(-2 + 0.8(0) - 1.5)$$

Example: Mini-Gridworld

- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states

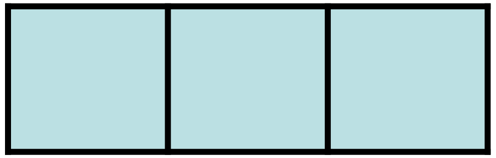


- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

Transition			
	0	0	
	0	0	0

$$V^{\pi}(B) \leftarrow 0 + 0.5(1 + 0.8(0) - 0)$$

Example: Mini-Gridworld



- All values initialized to 0; $\gamma = 0.8, \alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states
- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

Transition					
	0	0			
	0	0	0		

- TD methods perform updates immediately with no episodic structure (MC)
- Useful if problems have long episodes or are continuing tasks

Offline Prediction

- We've presented constant-α MC and $TD(0)$ as online, sample-based algorithms in contrast to dynamic programming
- It is also possible to use these as *offline*, sample-based algorithms

Offline Prediction

- We've presented constant-α MC and $TD(0)$ as online, sample-based algorithms in contrast to dynamic programming
- It is also possible to use these as *offline*, sample-based algorithms

- Suppose we have a fixed set of samples in place of underlying model
- We can run $TD(0)$ on the samples over and over until values converge

Offline Prediction

- We've presented constant-α MC and $TD(0)$ as online, sample-based algorithms in contrast to dynamic programming
- It is also possible to use these as *offline*, sample-based algorithms

- Suppose we have a fixed set of samples in place of underlying model
- We can run $TD(0)$ on the samples over and over until values converge

- We can perform **batch updating**, in which we make a single TD update based on the *sum* of all sample TD errors in sample set

Batch Updating

- **Given**: Training data $s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T$; learning rate α
- **Initialize** $V^{\pi}(s) \leftarrow 0$

Batch Updating

- **Given**: Training data $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$; learning rate α
- **Initialize** $V^\pi(s) \leftarrow 0$

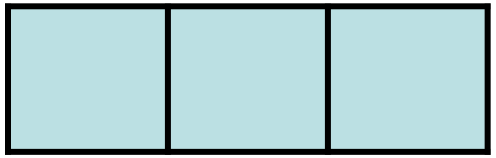
- **Loop** until values converge:
  - $\delta(s) \leftarrow 0\ \forall s$
  - **For** each state s_t in training data:
    - $\delta(s_t) \leftarrow \delta(s_t) + \left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$

Batch Updating

- **Given**: Training data $s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T$; learning rate α
- **Initialize** $V^\pi(s) \leftarrow 0$

- **Loop** until values converge:
  - $\delta(s) \leftarrow 0 \ \forall s$
  - **For** each state s_t in training data:
    - $\delta(s_t) \leftarrow \delta(s_t) + \left(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\right)$
  - **For** each state s:
    - $V^\pi(s) \leftarrow V^\pi(s) + \alpha\delta(s)$

Example: Mini-Gridworld



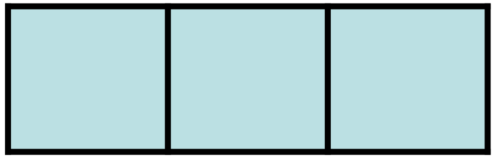
- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states

- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

$$\delta(s_t) \leftarrow r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$

$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha\delta(s_t)$$

Example: Mini-Gridworld



- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states

- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

$$\delta(s_t) \leftarrow r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$

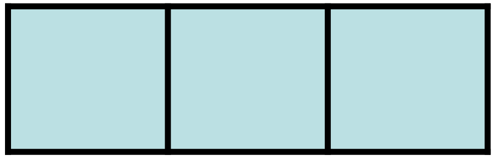
$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha\delta(s_t)$$

Iteration			

$$\begin{aligned}\delta(A) \leftarrow &\big(3 + \gamma V(A) - V(A)\big) \\ &+\big(-2 + \gamma V(B) - V(A)\big)\end{aligned}$$

$$V(A) \leftarrow V(A) + \alpha\delta(A)$$

Example: Mini-Gridworld



- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states

- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

$$\delta(s_t) \leftarrow r_{t+1} + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$

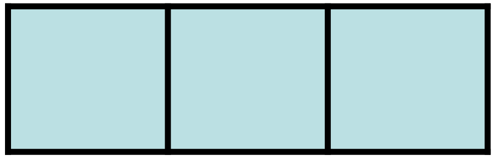
$$V^\pi(s_t) \leftarrow V^\pi(s_t) + \alpha\delta(s_t)$$

Iteration			

$$\begin{aligned}\delta(B) \leftarrow & \big(1 + \gamma V(C) - V(B)\big) \\ & + \big(3 + \gamma V(A) - V(B)\big)\end{aligned}$$

$$V(B) \leftarrow V(B) + \alpha\delta(B)$$

Example: Mini-Gridworld



- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states

- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

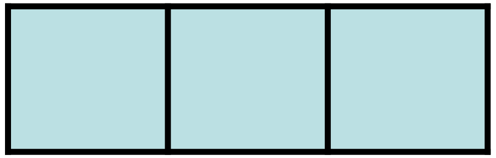
$$\delta(s_t) \leftarrow r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\delta(s_t)$$

Iteration			

$$\delta(C) \leftarrow -2 + \gamma V(B) - V(C)$$
$$V(C) \leftarrow V(C) + \alpha\delta(C)$$

Example: Mini-Gridworld



- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states

- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

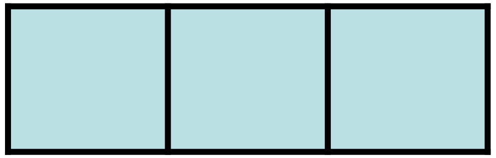
$$\delta(s_t) \leftarrow r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\delta(s_t)$$

Iteration					

Example: Mini-Gridworld

- All values initialized to 0; $\gamma = 0.8$, $\alpha = 0.5$
- Policy to evaluate: $\pi(s) = L$ for all states



- Observed state and reward sequence: $(A, +3, A, -2, B, +1, C, -2, B, +3, A)$

$$\delta(s_t) \leftarrow r_{t+1} + \gamma V^{\pi}(s_{t+1}) - V^{\pi}(s_t)$$

$$V^{\pi}(s_t) \leftarrow V^{\pi}(s_t) + \alpha\delta(s_t)$$

Iteration										

TD Learning for Control

- We can apply TD updates to Q values $Q(s,a)$ rather than state values $V(s)$
- After each transition (s,a,r), we apply the TD update to $Q(s,a)$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$$

TD Learning for Control

- We can apply TD updates to Q values $Q(s,a)$ rather than state values $V(s)$
- After each transition (s,a,r), we apply the TD update to $Q(s,a)$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$$

- Action a in state s has been chosen according to behavior policy
- May allow for exploration to learn about different actions

TD Learning for Control

- We can apply TD updates to Q values $Q(s,a)$ rather than state values $V(s)$
- After each transition (s,a,r), we apply the TD update to $Q(s,a)$

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$$

- Action a in state s has been chosen according to behavior policy
- May allow for exploration to learn about different actions

- Which successor state Q value $Q(s',a')$ to use?
- As with MC control, we need to consider *on-policy* vs *off-policy* learning

On-Policy: SARSA

- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s,a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

On-Policy: SARSA

- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s,a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

- **Loop**:
  - **Initialize** starting state s, action $a = \pi(s)$ *if needed*

On-Policy: SARSA

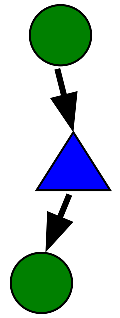
- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s, a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

- **Loop**:
  - **Initialize** starting state s, action $a = \pi(s)$ *if needed*
  - **Generate** sequence (s, a, r, s'), $a' \leftarrow \pi(s')$

On-Policy: SARSA

- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s,a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

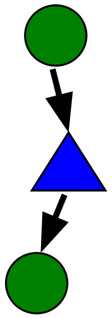
- **Loop**:
  - **Initialize** starting state s, action $a = \pi(s)$ *if needed*
  - **Generate** sequence (s,a,r,s'), $a' \leftarrow \pi(s')$
  - $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$



On-Policy: SARSA

- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s,a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

- **Loop**:
  - **Initialize** starting state s, action $a = \pi(s)$ *if needed*
  - **Generate** sequence (s, a, r, s'), $a' \leftarrow \pi(s')$
  - $Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$
  - $s \leftarrow s', a \leftarrow a'$



Off-Policy: Q-Learning

- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s,a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

Off-Policy: Q-Learning

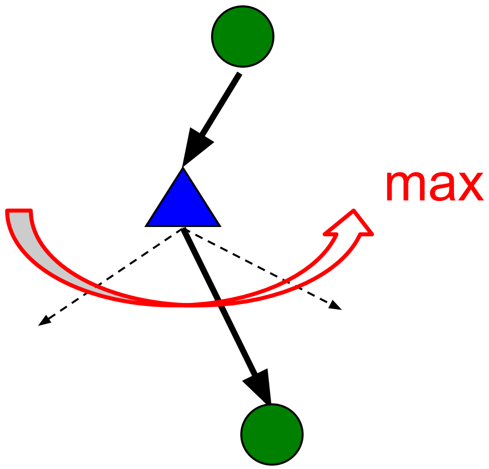
- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s,a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

- **Loop**:
  - **Initialize** starting state s *if needed*, action $a = \pi(s)$
  - **Generate** sequence (s, a, r, s')

Off-Policy: Q-Learning

- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s,a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

- **Loop**:
  - **Initialize** starting state s *if needed*, action $a = \pi(s)$
  - **Generate** sequence (s, a, r, s')
  - $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s', a') - Q(s,a)\right)$

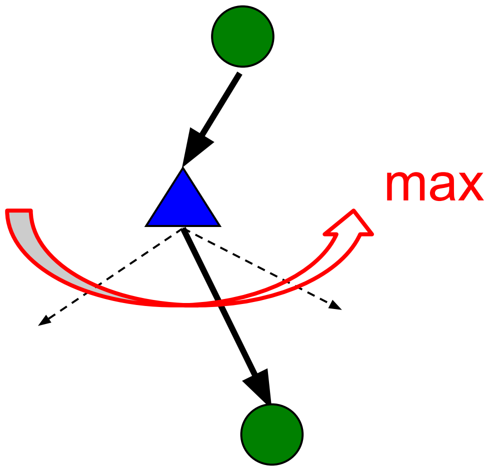


Off-Policy: Q-Learning

- **Given:** Step size α, exploration rate ε
- **Initialize** $Q(s,a) \leftarrow 0$, behavior policy π (e.g., ε-greedy)

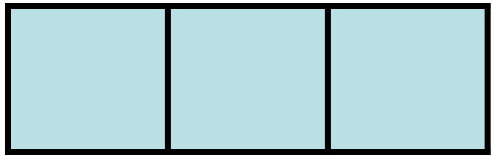
- **Loop**:
  - **Initialize** starting state s *if needed*, action $a = \pi(s)$
  - **Generate** sequence (s, a, r, s')
  - $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$
  - $s \leftarrow s'$

Action a' selected according to target policy



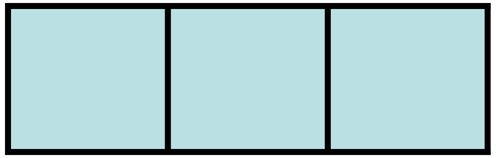
Example: Mini-Gridworld

- Suppose we currently have $Q(A, L) = 1.5, Q(A, R) = 1$
- Behavior policy is ε-greedy; $\alpha = 0.5, \gamma = 0.8$



Example: Mini-Gridworld

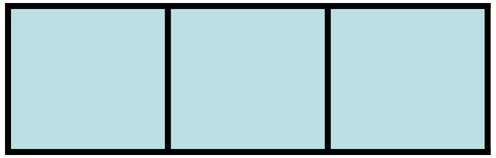
- Suppose we currently have $Q(A, L) = 1.5, Q(A, R) = 1$
- Behavior policy is ε-greedy; $\alpha = 0.5, \gamma = 0.8$



- Observed (s, a, r, s') sequence: $A, L, +3, A$
- Suppose behavior policy generates $a' = R$ (*explore*)

Example: Mini-Gridworld

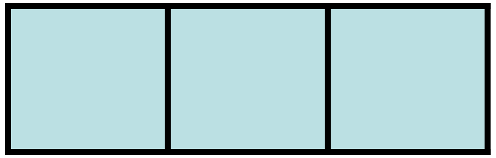
- Suppose we currently have $Q(A, L) = 1.5$, $Q(A, R) = 1$
- Behavior policy is ε-greedy; $\alpha = 0.5$, $\gamma = 0.8$



- Observed (s, a, r, s') sequence: $A, L, +3, A$
- Suppose behavior policy generates $a' = R$ (*explore*)
- SARSA target: $r + \boldsymbol{\gamma Q(A, R)} = 3 + 0.8(1) = 3.8$
- New Q-value: $Q(A, L) \leftarrow 1 + 0.5(3.8 - 1) = 2.4$

Example: Mini-Gridworld

- Suppose we currently have $Q(A, L) = 1.5$, $Q(A, R) = 1$
- Behavior policy is ε-greedy; $\alpha = 0.5$, $\gamma = 0.8$



- Observed (s, a, r, s') sequence: $A, L, +3, A$
- Suppose behavior policy generates $a' = R$ (*explore*)
- Q-learning target: $r + \gamma \max_{\boldsymbol{a}} \boldsymbol{Q}(\boldsymbol{A}, \boldsymbol{a}) = r + \gamma \boldsymbol{Q}(\boldsymbol{A}, \boldsymbol{L}) = 3 + 0.8(1.5) = 4.2$
- New Q-value: $Q(A, L) \leftarrow 1 + 0.5(4.2 - 1) = 2.6$

SARSA vs Q-Learning

SARSA	Q-learning
• On-policy: learn values of behavior policy	• Off-policy: learn values of target policy
•	•
•	

SARSA vs Q-Learning

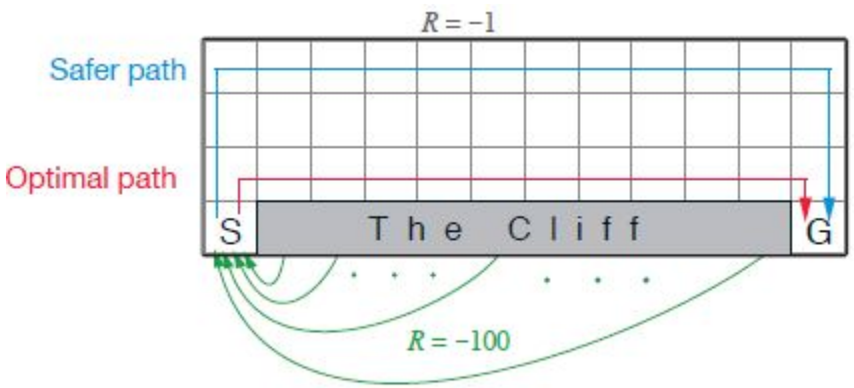
SARSA	Q-learning
• On-policy: learn values of behavior policy	• Off-policy: learn values of target policy
• Agents are more "cautious" as they worry about low rewards from exploration	• Agents are more "optimistic" as only the best (greedy) actions at each state matter
•	

SARSA vs Q-Learning

SARSA	Q-learning
• On-policy: learn values of behavior policy	• Off-policy: learn values of target policy
• Agents are more "cautious" as they worry about low rewards from exploration	• Agents are more "optimistic" as only the best (greedy) actions at each state matter
• SARSA and Q-learning are identical if behavior and target policy are the same, or if exploration is removed!	

Cliff Walking (Sutton and Barto, Ex. 6.6)

- Deterministic gridworld with one terminal (goal) state
- Living reward is -1 in all states except for "cliff", which rewards -100

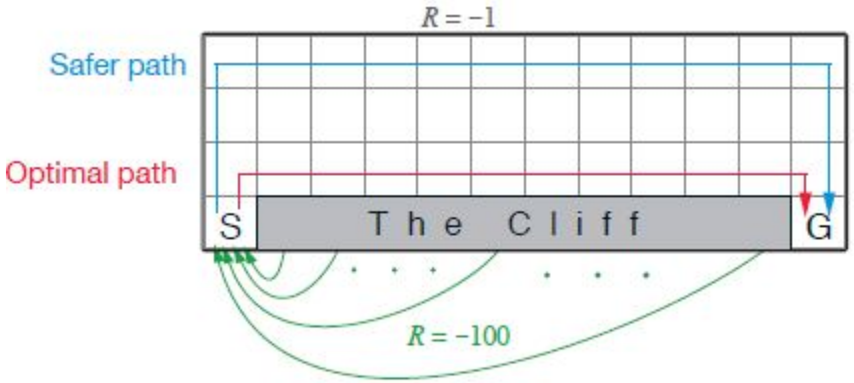


Cliff Walking (Sutton and Barto, Ex. 6.6)

- Deterministic gridworld with one terminal (goal) state
- Living reward is -1 in all states except for "cliff", which rewards -100

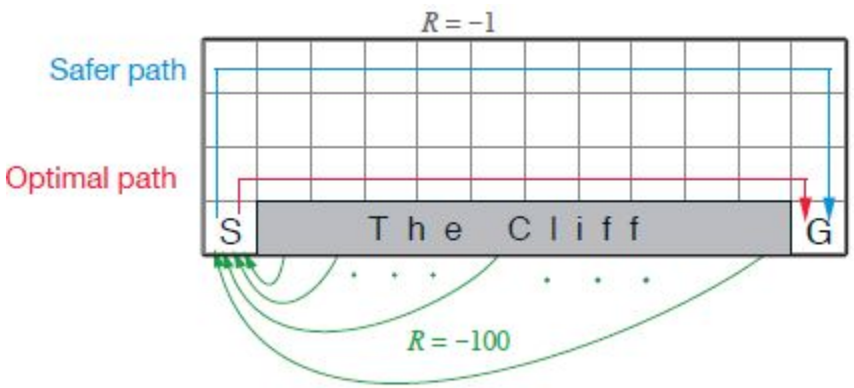
- Since transitions are deterministic, the optimal action at each state is to head in goal direction while ignoring the cliff

- We can use TD control to learn Q values and the corresponding policy



Cliff Walking (Sutton and Barto, Ex. 6.6)

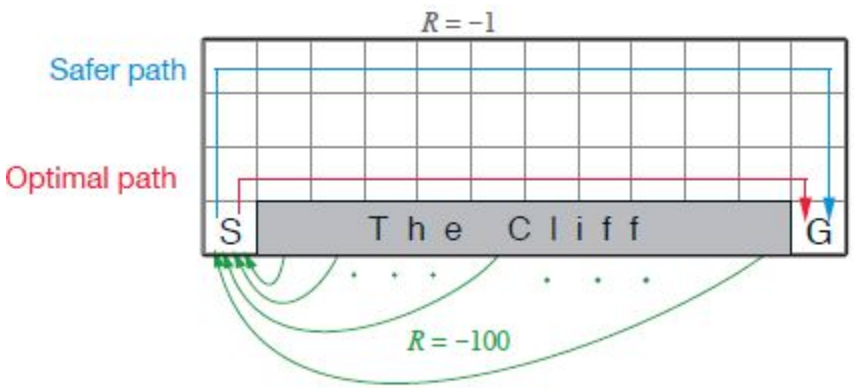
- Suppose we use SARSA (on-policy) with exploratory behavior policy
- States nearer cliff will have lower Q values than those further away!



Cliff Walking (Sutton and Barto, Ex. 6.6)

- Suppose we use SARSA (on-policy) with exploratory behavior policy
- States nearer cliff will have lower Q values than those further away!

- Exploration introduces stochasticity into the problem
- Agent learns that it is more likely to “fall” into the cliff if closer to it

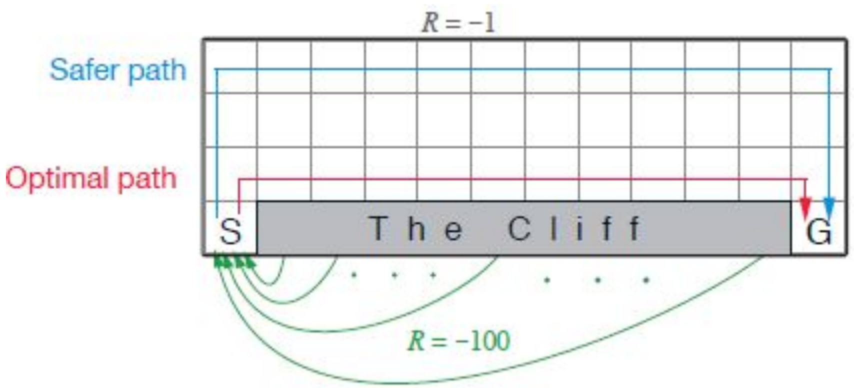


Cliff Walking (Sutton and Barto, Ex. 6.6)

- Suppose we use SARSA (on-policy) with exploratory behavior policy
- States nearer cliff will have lower Q values than those further away!

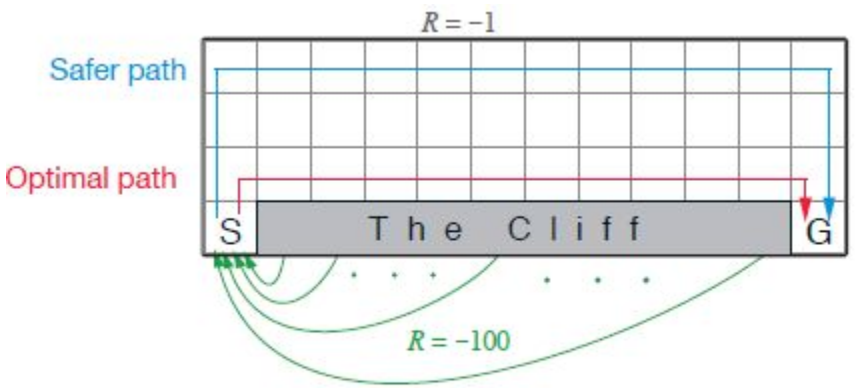
- Exploration introduces stochasticity into the problem
- Agent learns that it is more likely to “fall” into the cliff if closer to it

- SARSA ends up learning the “safer path”
- Get to the goal but minimize risk of falling



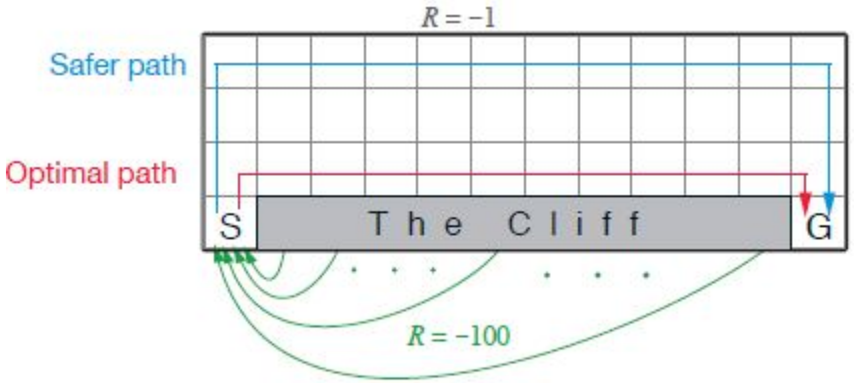
Cliff Walking (Sutton and Barto, Ex. 6.6)

- Suppose we use Q-learning (off-policy)
- Converged Q-values will not depend on proximity to cliff



Cliff Walking (Sutton and Barto, Ex. 6.6)

- Suppose we use Q-learning (off-policy)
- Converged Q-values will not depend on proximity to cliff
- Even if behavior policy is exploratory, the Q-value update is "optimistic" and assumes that the agent acts greedily at all times

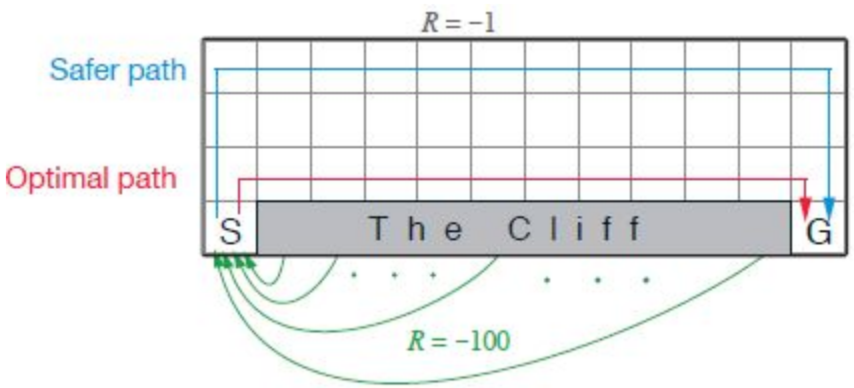


Cliff Walking (Sutton and Barto, Ex. 6.6)

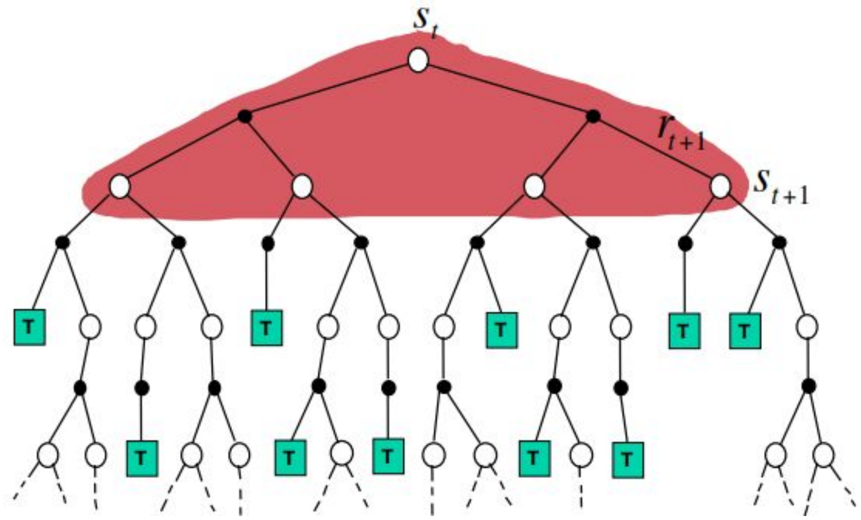
- Suppose we use Q-learning (off-policy)
- Converged Q-values will not depend on proximity to cliff

- Even if behavior policy is exploratory, the Q-value update is "optimistic" and assumes that the agent acts greedily at all times

- Learned Q-values will be optimal
- Shortest path will result from learned policy



MDP Method Comparison

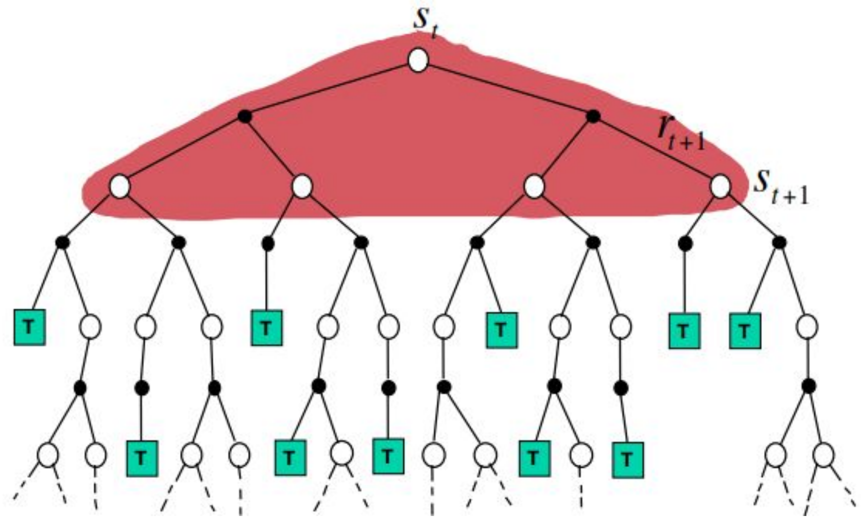


https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf

Dynamic programming

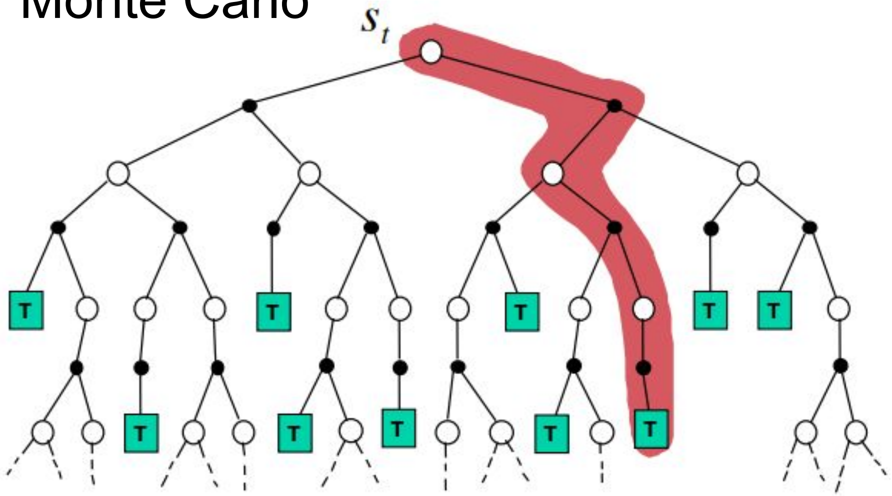
MDP Method Comparison

https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf



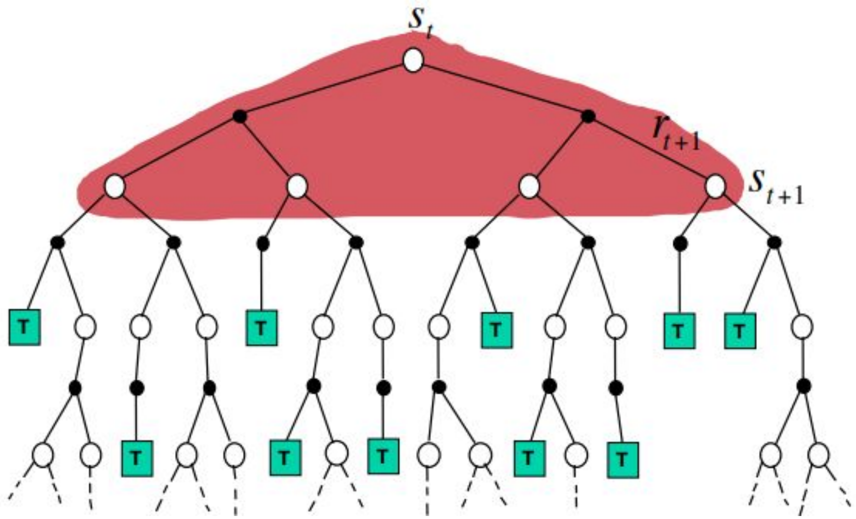
Dynamic programming

Monte Carlo



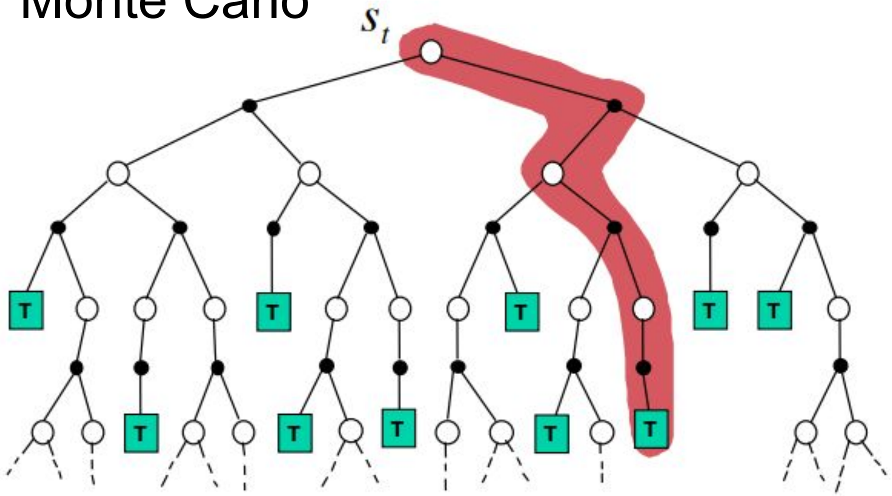
MDP Method Comparison

https://www.davidsilver.uk/wp-content/uploads/2020/03/MC-TD.pdf

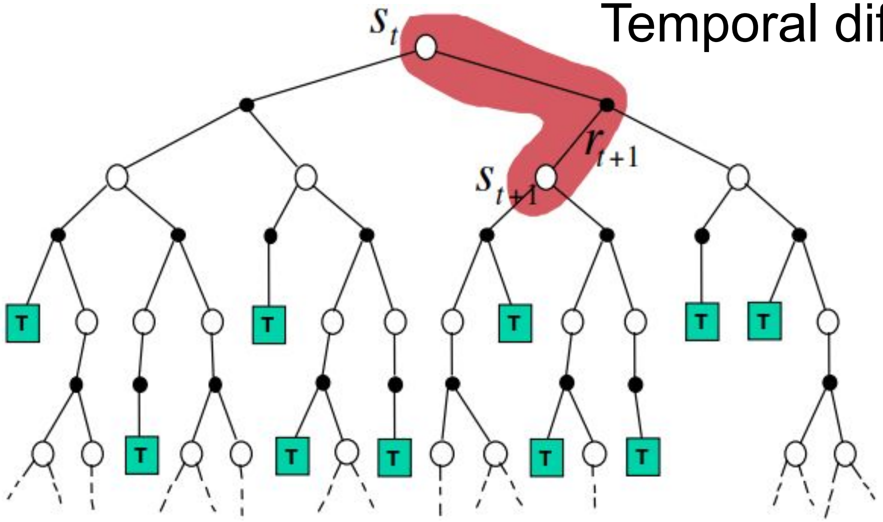


Dynamic programming

Monte Carlo



Temporal difference



Summary

- Temporal-difference methods update values after each transition
- Uses samples like MC, but bootstraps like DP

- Prediction: One-step ($TD(0)$) for online updates
- Batch updating can be done offline for fixed set of samples

- Control: On-policy (SARSA) vs off-policy (Q-learning)
- SARSA learns values of behavior policy, utilities reflect exploration
- Q-learning learns values of target policy, producing optimal policy