

# ***Introduction to Reinforcement Learning***

*2025. 1<sup>st</sup> semester*

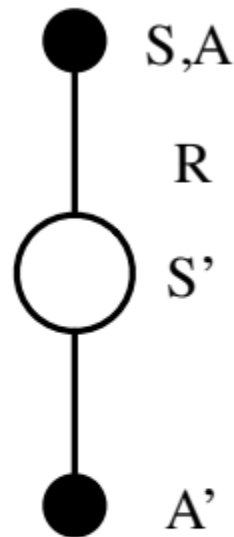
# Evaluation to Control

- *Last time: how good is a specific policy?*
  - *Given no access to the decision process model parameters*
  - *Instead have to estimate from data / experience*
- *Today: how can we learn a good policy?*

# MC vs. TD Control

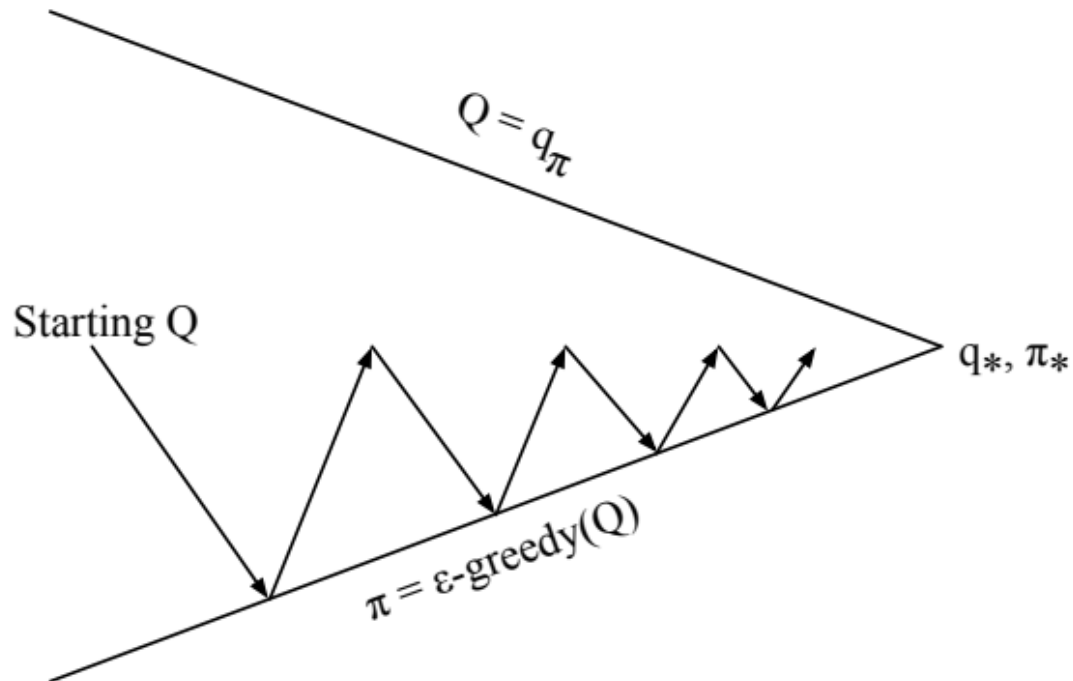
- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to  $Q(S, A)$
  - Use  $\epsilon$ -greedy policy improvement
  - Update every time-step

# Updating Action-Value Functions with Sarsa



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

# On-Policy Control With Sarsa



Every **time-step**:

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

Policy improvement  $\epsilon$ -greedy policy improvement

# Sarsa Algorithm 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

# Convergence of Sarsa

## Theorem

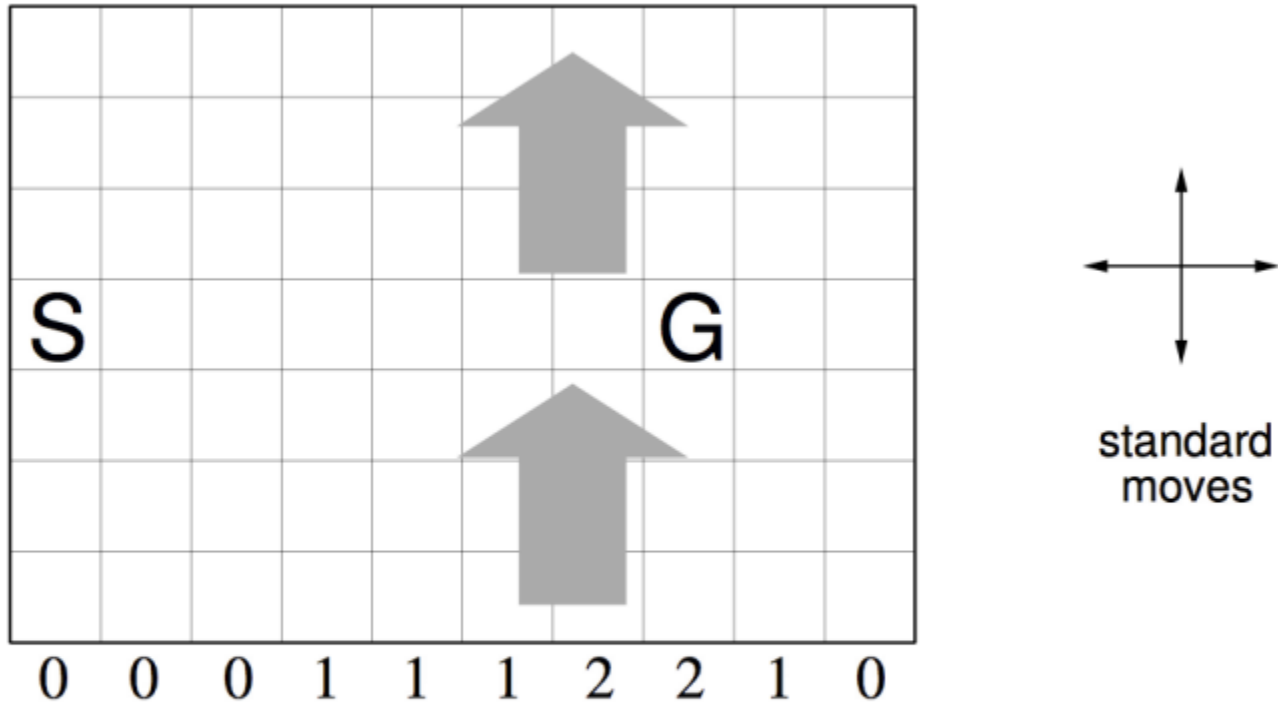
*Sarsa converges to the optimal action-value function,  $Q(s, a) \rightarrow q_*(s, a)$ , under the following conditions:*

- *GLIE sequence of policies  $\pi_t(a|s)$*
- *Robbins-Monro sequence of step-sizes  $\alpha_t$*

$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

$$\sum_{t=1}^{\infty} \alpha_t^2 < \infty$$

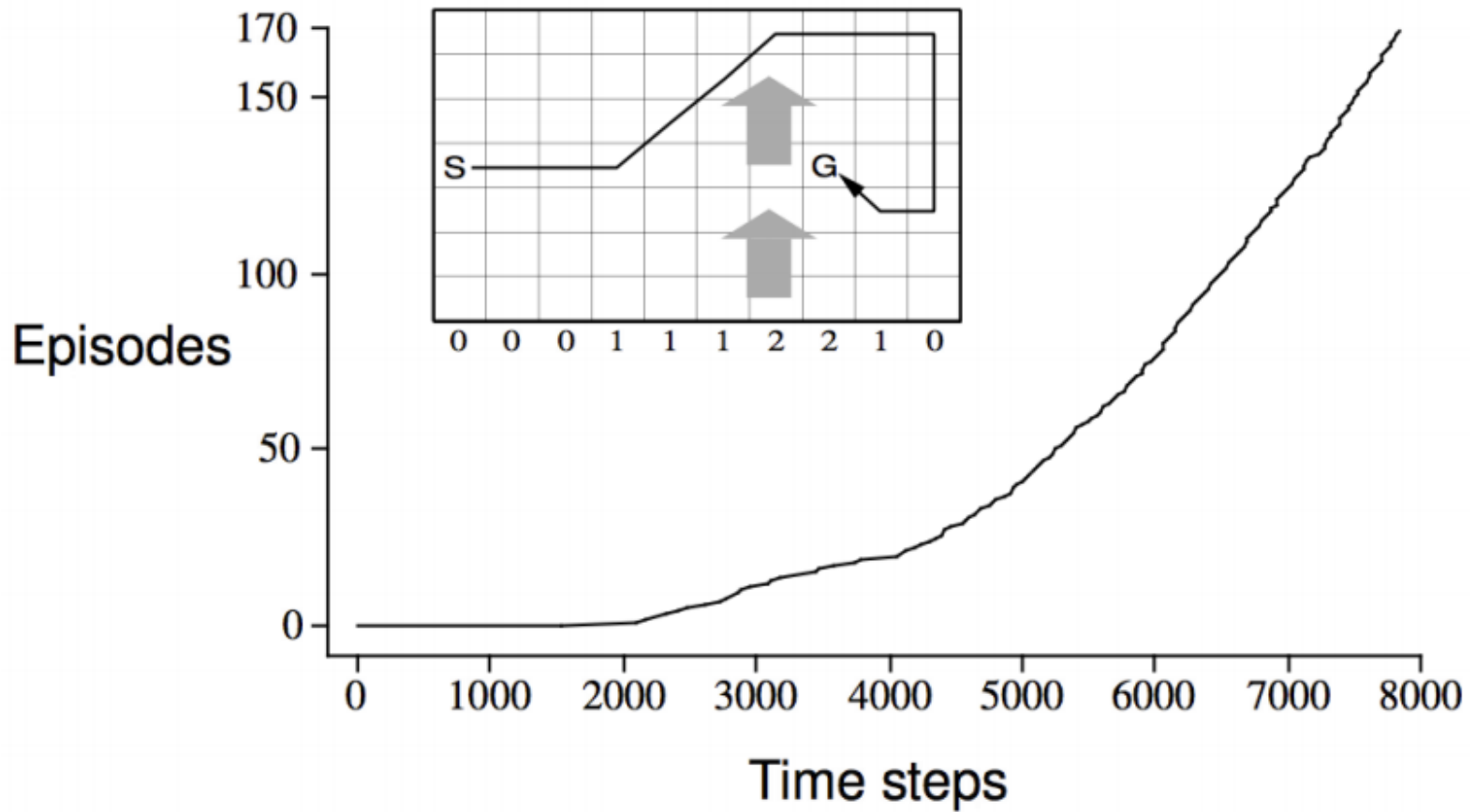
# Windy Gridworld Example



- Reward = -1 per time-step until reaching goal
- Undiscounted



# *Sarsa on the Windy Gridworld*



## ***n-Step Sarsa***

- Consider the following  $n$ -step returns for  $n = 1, 2, \infty$ :

$$n = 1 \quad (\text{Sarsa}) \quad q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1})$$

$$n = 2 \quad q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2})$$

$$\vdots$$

$$n = \infty \quad (\text{MC}) \quad q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

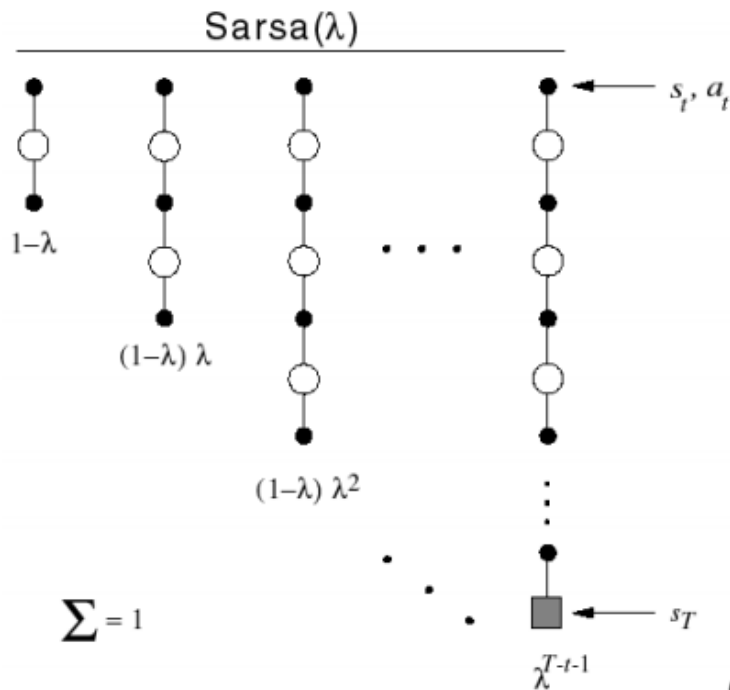
- Define the  $n$ -step Q-return

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n Q(S_{t+n})$$

- $n$ -step Sarsa updates  $Q(s, a)$  towards the  $n$ -step Q-return

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left( q_t^{(n)} - Q(S_t, A_t) \right)$$

# Forward View Sarsa( $\lambda$ )



- The  $q^\lambda$  return combines all  $n$ -step Q-returns  $q_t^{(n)}$
- Using weight  $(1 - \lambda)\lambda^{n-1}$

$$q_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} q_t^{(n)}$$

- Forward-view Sarsa( $\lambda$ )

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left( q_t^\lambda - Q(S_t, A_t) \right)$$

## Backward View Sarsa( $\lambda$ )

- Just like TD( $\lambda$ ), we use **eligibility traces** in an online algorithm
- But Sarsa( $\lambda$ ) has one eligibility trace for each state-action pair

$$E_0(s, a) = 0$$

$$E_t(s, a) = \gamma\lambda E_{t-1}(s, a) + \mathbf{1}(S_t = s, A_t = a)$$

- $Q(s, a)$  is updated for every state  $s$  and action  $a$
- In proportion to TD-error  $\delta_t$  and eligibility trace  $E_t(s, a)$

$$\delta_t = R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$$

$$Q(s, a) \leftarrow Q(s, a) + \alpha \delta_t E_t(s, a)$$

# Sarsa( $\lambda$ ) Algorithm

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

Repeat (for each episode):

$E(s, a) = 0$ , for all  $s \in \mathcal{S}, a \in \mathcal{A}(s)$

Initialize  $S, A$

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)

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

$E(S, A) \leftarrow E(S, A) + \delta$

For all  $s \in \mathcal{S}, a \in \mathcal{A}(s)$ :

$Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$

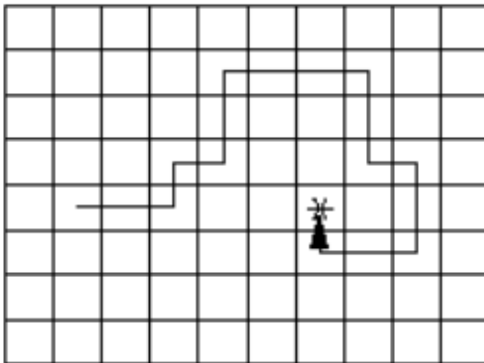
$E(s, a) \leftarrow \gamma \lambda E(s, a) + \delta$

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

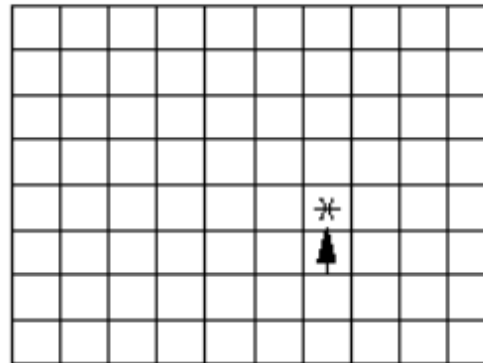
until  $S$  is terminal

# *Sarsa( $\lambda$ ) Gridworld Example*

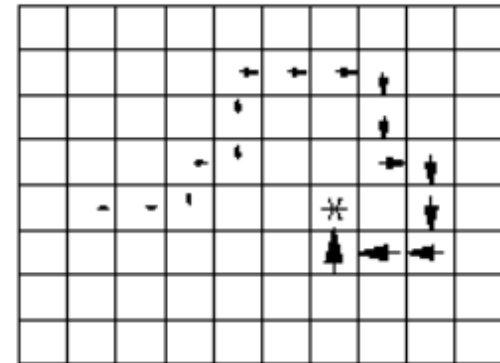
Path taken



Action values increased  
by one-step Sarsa



Action values increased  
by Sarsa( $\lambda$ ) with  $\lambda=0.9$



# Off-Policy Learning

- Evaluate target policy  $\pi(a|s)$  to compute  $v_\pi(s)$  or  $q_\pi(s, a)$
- While following behaviour policy  $\mu(a|s)$

$$\{S_1, A_1, R_2, \dots, S_T\} \sim \mu$$

- Why is this important?
- Learn from observing humans or other agents
- Re-use experience generated from old policies  $\pi_1, \pi_2, \dots, \pi_{t-1}$
- Learn about *optimal* policy while following *exploratory* policy
- Learn about *multiple* policies while following *one* policy

# Importance Sampling

- Estimate the expectation of a different distribution

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



# Importance Sampling for Off-Policy Monte-Carlo

- Use returns generated from  $\mu$  to evaluate  $\pi$
- Weight return  $G_t$  according to similarity between policies
- Multiply importance sampling corrections along whole episode

$$G_t^{\pi/\mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} \frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})} \cdots \frac{\pi(A_T|S_T)}{\mu(A_T|S_T)} G_t$$

- Update value towards *corrected* return

$$V(S_t) \leftarrow V(S_t) + \alpha \left( G_t^{\pi/\mu} - V(S_t) \right)$$

- Cannot use if  $\mu$  is zero when  $\pi$  is non-zero
- Importance sampling can dramatically increase variance

# Importance Sampling for Off-Policy TD

- Use TD targets generated from  $\mu$  to evaluate  $\pi$
- Weight TD target  $R + \gamma V(S')$  by importance sampling
- Only need a single importance sampling correction

$$V(S_t) \leftarrow V(S_t) + \alpha \left( \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)} (R_{t+1} + \gamma V(S_{t+1})) - V(S_t) \right)$$

- Much lower variance than Monte-Carlo importance sampling
- Policies only need to be similar over a single step

# Q-Learning

- We now consider off-policy learning of action-values  $Q(s, a)$
- **No** importance sampling is required
- Next action is chosen using behaviour policy  $A_{t+1} \sim \mu(\cdot|S_t)$
- But we consider alternative successor action  $A' \sim \pi(\cdot|S_t)$
- And update  $Q(S_t, A_t)$  towards value of alternative action

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha (R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

# Off-Policy Control with Q-Learning

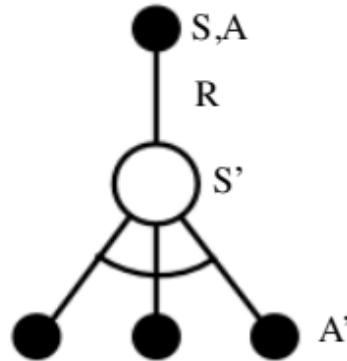
- We now allow both behaviour and target policies to **improve**
- The target policy  $\pi$  is **greedy** w.r.t.  $Q(s, a)$

$$\pi(S_{t+1}) = \operatorname{argmax}_{a'} Q(S_{t+1}, a')$$

- The behaviour policy  $\mu$  is e.g.  **$\epsilon$ -greedy** w.r.t.  $Q(s, a)$
- The Q-learning target then simplifies:

$$\begin{aligned} & R_{t+1} + \gamma Q(S_{t+1}, A') \\ &= R_{t+1} + \gamma Q(S_{t+1}, \operatorname{argmax}_{a'} Q(S_{t+1}, a')) \\ &= R_{t+1} + \max_{a'} \gamma Q(S_{t+1}, a') \end{aligned}$$

# Q-Learning Control Algorithm



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

## Theorem

*Q-learning control converges to the optimal action-value function,  
 $Q(s, a) \rightarrow q_*(s, a)$*

# ***Q-Learning Algorithm 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  $R, S'$

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

$S \leftarrow S'$ ;

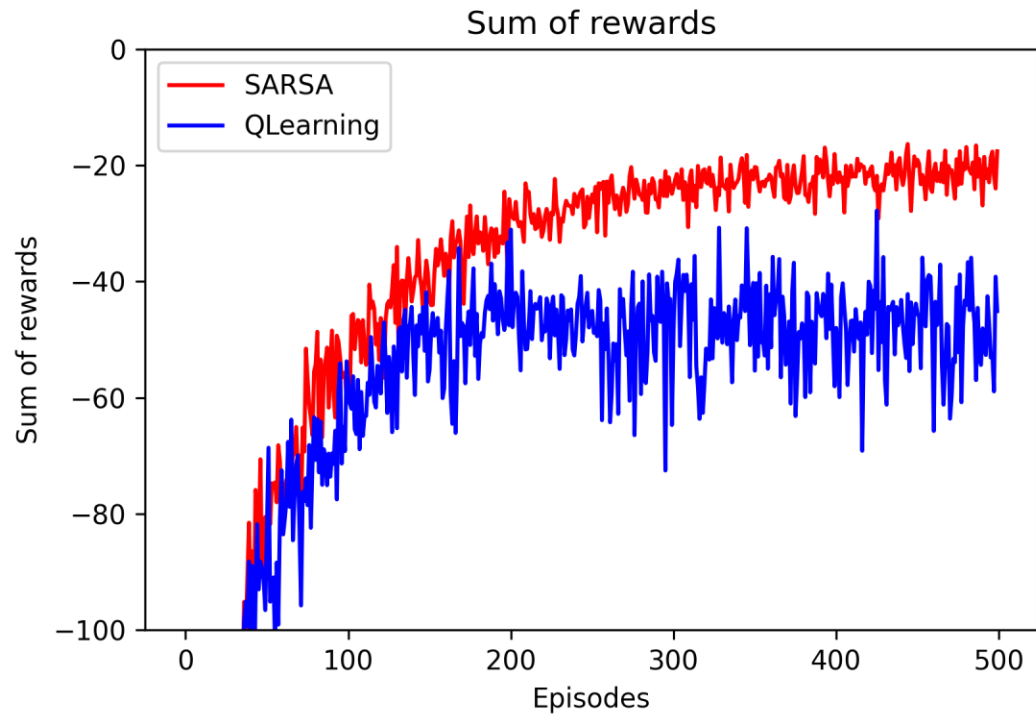
until  $S$  is terminal

# Cliff Walking Example

- Start in the lower left corner of the S column. The goal is to reach square G in the lower right corner.
- At each square you can move by one square in four directions: up, down, left, right.
- You get a -1 reward for each move you make. However, if you enter the Cliff area below (because you fell from the cliff), you will receive a -100 reward and move immediately to the starting point (S) (rescued).
- When you reach G, you get a reward of 0. If you move outside the border (eg, if you try to move down from S), the position does not change, only the reward is -1.

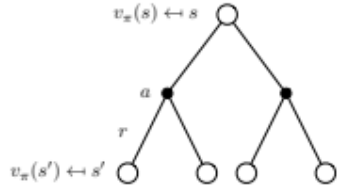

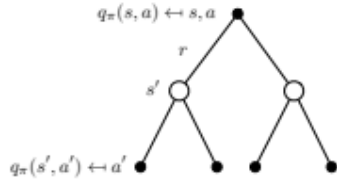
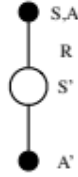
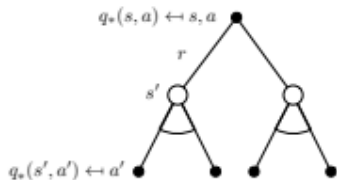
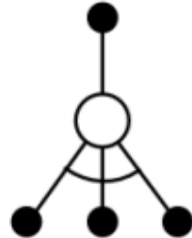
S	Cliff										G

# Cliff Walking Example





# Relationship Between DP and TD

	Full Backup (DP)	Sample Backup (TD)
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>

## Relationship Between DP and TD (2)

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