

# SDM5013: Deep Learning and Reinforcement Learning

Zhiyun Lin



南方科技大学  
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY



设计智造学院  
School of  
System Design and  
Intelligent Manufacturing

# Outline

- 1 Introduction
- 2 On-Policy Monte-Carlo Control
- 3 On-Policy Temporal-Difference Learning
- 4 Off-Policy Learning
- 5 Summary

# Model-Free Reinforcement Learning

- Last lecture:
  - Model-free prediction
  - *Estimate* the value function of an *unknown* MDP
- This lecture:
  - Model-free control
  - *Optimise* the value function of an *unknown* MDP

# Uses of Model-Free Control

Some example problems that can be modelled as MDPs

- Elevator
- Parallel Parking
- Ship Steering
- Bioreactor
- Helicopter
- Aeroplane Logistics
- Robocup Soccer
- Quake
- Portfolio management
- Protein Folding
- Robot walking
- Game of Go

For most of these problems, either:

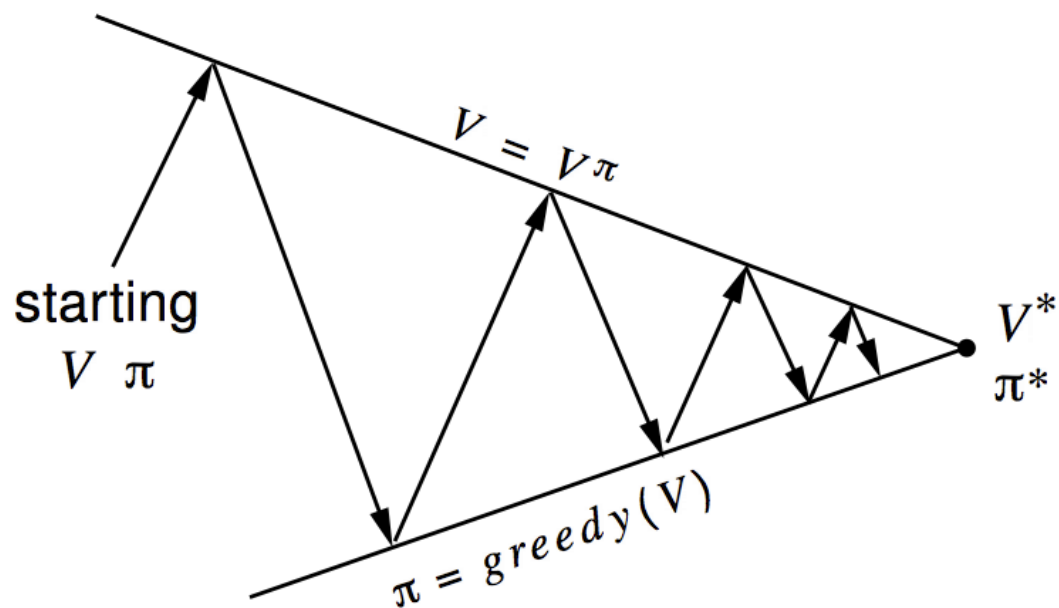
- MDP model is unknown, but experience can be sampled
- MDP model is known, but is too big to use, except by samples

**Model-free control** can solve these problems

# On and Off-Policy Learning

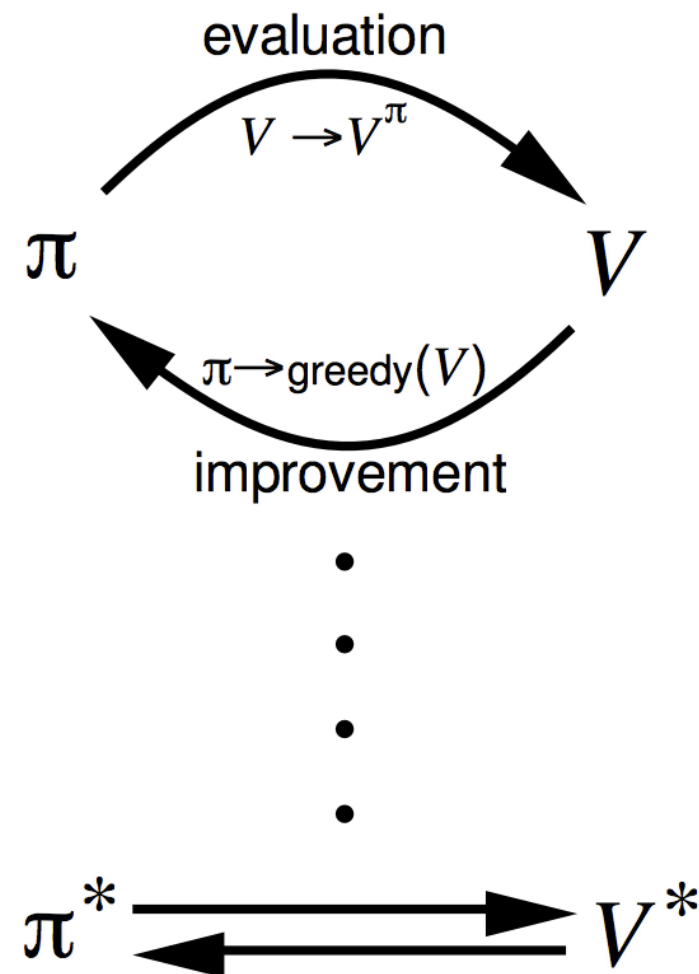
- **On-policy** learning
  - “Learn on the job”
  - Learn about policy  $\pi$  from experience sampled from  $\pi$
- **Off-policy** learning
  - “Look over someone’s shoulder”
  - Learn about policy  $\pi$  from experience sampled from  $\mu$

# Generalised Policy Iteration (Refresher)

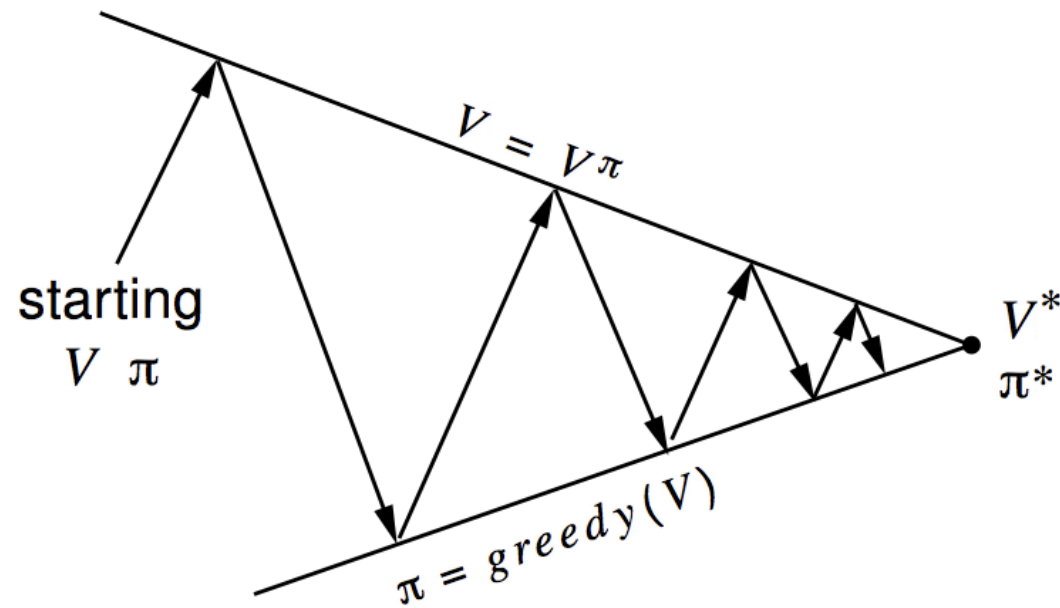


**Policy evaluation** Estimate  $v_\pi$   
e.g. Iterative policy evaluation

**Policy improvement** Generate  $\pi' \geq \pi$   
e.g. Greedy policy improvement



# Generalised Policy Iteration With Monte-Carlo Evaluation



**Policy evaluation** Monte-Carlo policy evaluation,  $V = v_\pi$ ?

**Policy improvement** Greedy policy improvement?

# Model-Free Policy Iteration Using Action-Value Function

- Greedy policy improvement over  $V(s)$  requires model of MDP

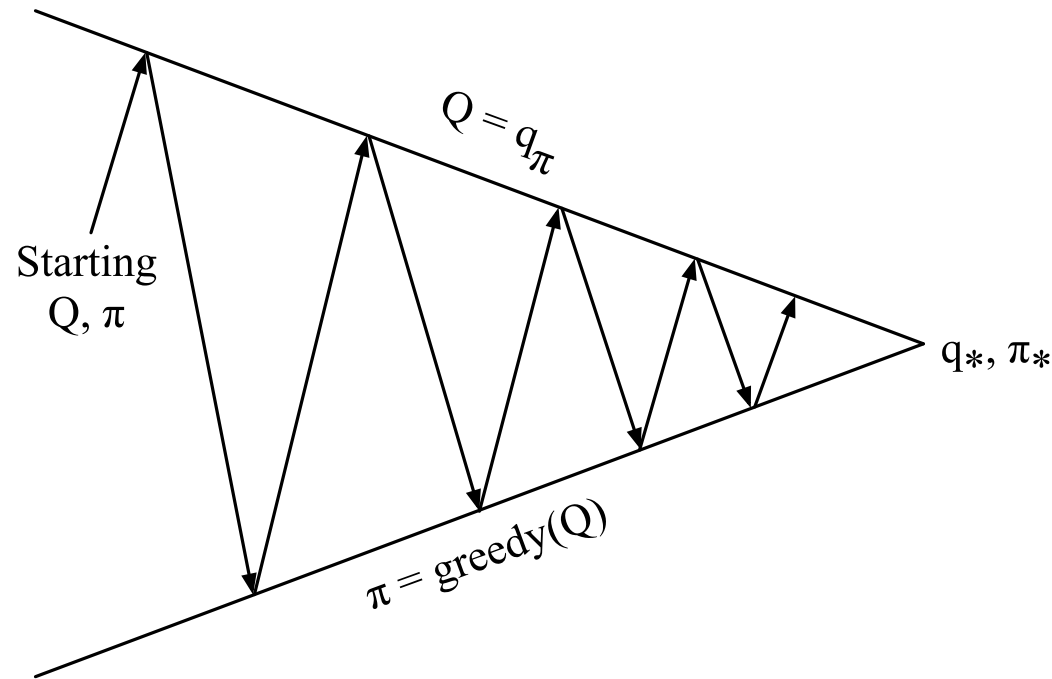
$$\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$

- Greedy policy improvement over  $Q(s, a)$  is model-free

$$\pi'(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q(s, a)$$



# Generalised Policy Iteration with Action-Value Function



Policy evaluation Monte-Carlo policy evaluation,  $Q = q_\pi$

Policy improvement Greedy policy improvement?

# Example of Greedy Action Selection



- There are two doors in front of you.
- You open the left door and get reward 0  
 $V(\text{left}) = 0$
- You open the right door and get reward +1  
 $V(\text{right}) = +1$
- You open the right door and get reward +3  
 $V(\text{right}) = +2$
- You open the right door and get reward +2  
 $V(\text{right}) = +2$
- $\vdots$
- Are you sure you've chosen the best door?

# $\epsilon$ -Greedy Exploration

- Simplest idea for ensuring continual exploration
- All  $m$  actions are tried with non-zero probability
- With probability  $1 - \epsilon$  choose the greedy action
- With probability  $\epsilon$  choose an action at random

$$\pi(a|s) = \begin{cases} \epsilon/m + 1 - \epsilon & \text{if } a^* = \operatorname{argmax}_{a \in \mathcal{A}} Q(s, a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

# $\epsilon$ -Greedy Policy Improvement

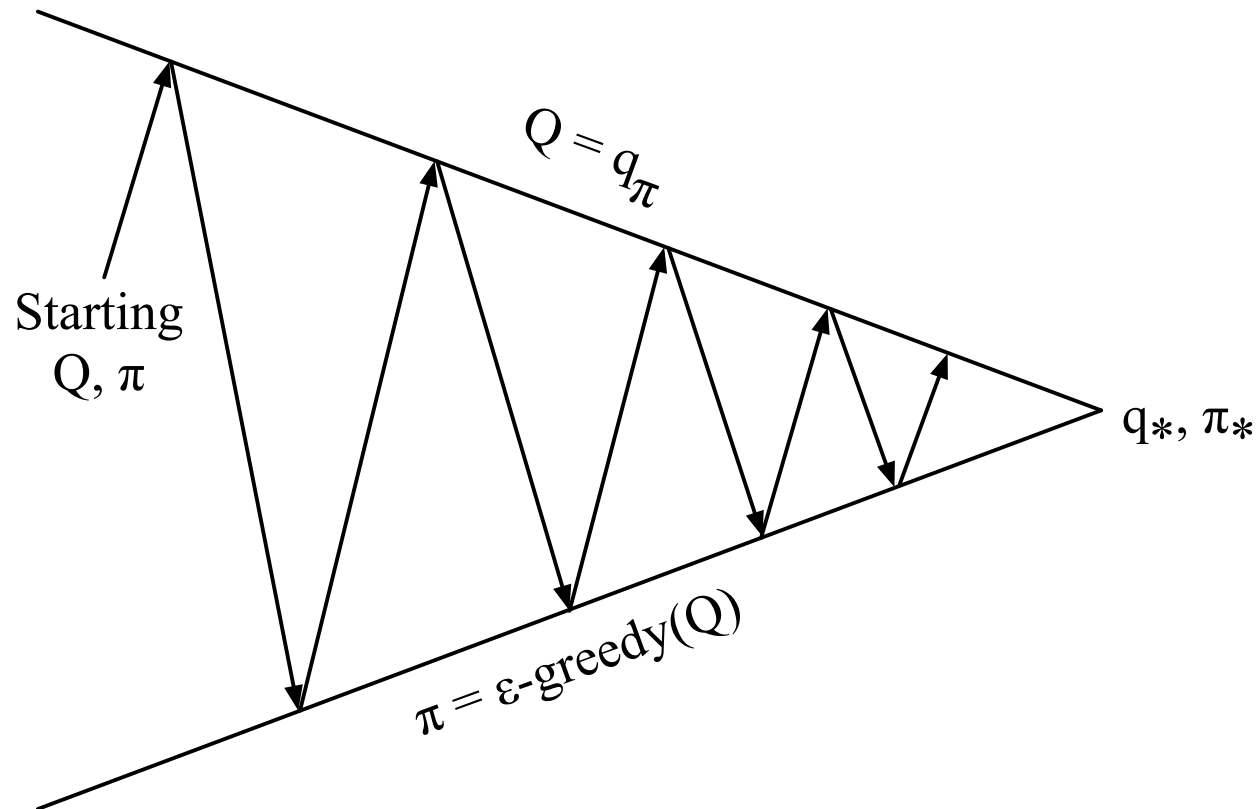
## Theorem

*For any  $\epsilon$ -greedy policy  $\pi$ , the  $\epsilon$ -greedy policy  $\pi'$  with respect to  $q_\pi$  is an improvement,  $v_{\pi'}(s) \geq v_\pi(s)$*

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s, a) \\ &= \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \max_{a \in \mathcal{A}} q_\pi(s, a) \\ &\geq \epsilon/m \sum_{a \in \mathcal{A}} q_\pi(s, a) + (1 - \epsilon) \sum_{a \in \mathcal{A}} \frac{\pi(a|s) - \epsilon/m}{1 - \epsilon} q_\pi(s, a) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) = v_\pi(s) \end{aligned}$$

Therefore from policy improvement theorem,  $v_{\pi'}(s) \geq v_\pi(s)$

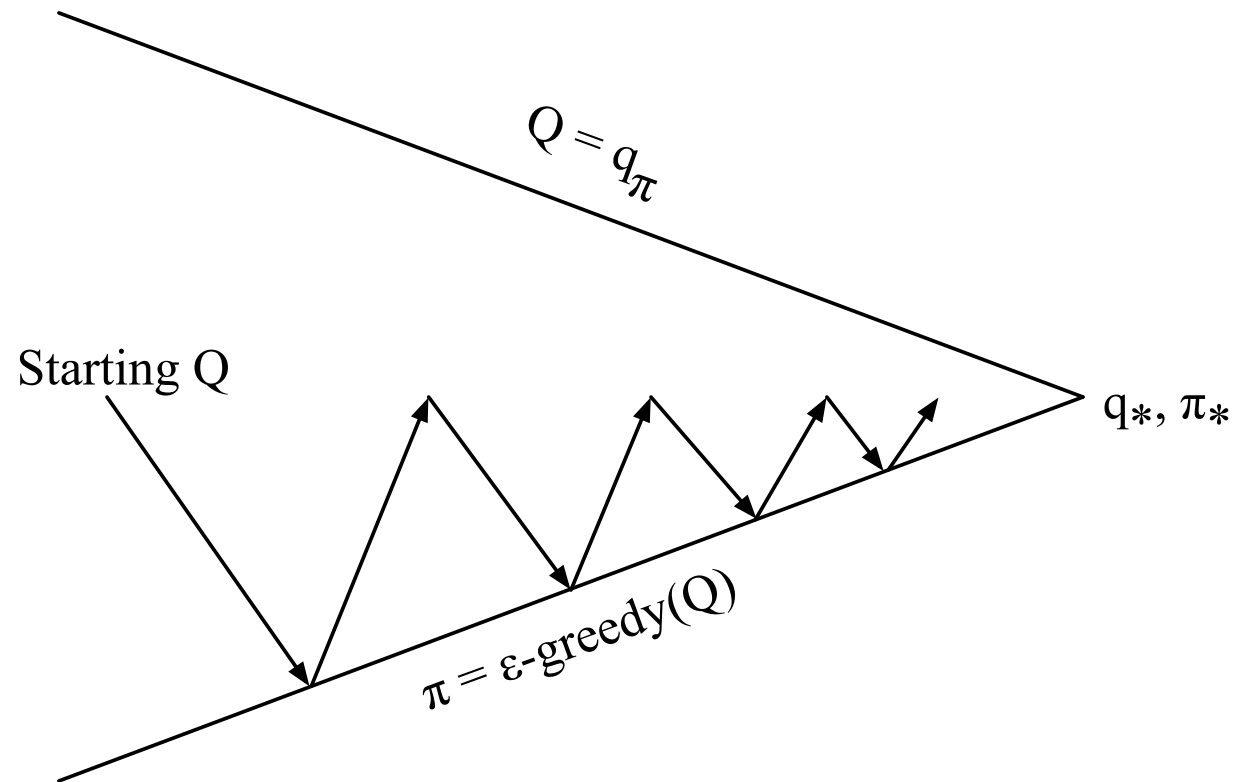
# Monte-Carlo Policy Iteration



**Policy evaluation** Monte-Carlo policy evaluation,  $Q = q_\pi$

**Policy improvement**  $\epsilon$ -greedy policy improvement

# Monte-Carlo Control



Every episode:

Policy evaluation Monte-Carlo policy evaluation,  $Q \approx q_\pi$

Policy improvement  $\epsilon$ -greedy policy improvement

# GLIE

## Definition

*Greedy in the Limit with Infinite Exploration (GLIE)*

- All state-action pairs are explored infinitely many times,

$$\lim_{k \rightarrow \infty} N_k(s, a) = \infty$$

- The policy converges on a greedy policy,

$$\lim_{k \rightarrow \infty} \pi_k(a|s) = \mathbf{1}(a = \operatorname{argmax}_{a' \in \mathcal{A}} Q_k(s, a'))$$

- For example,  $\epsilon$ -greedy is GLIE if  $\epsilon$  reduces to zero at  $\epsilon_k = \frac{1}{k}$

# GLIE Monte-Carlo Control

- Sample  $k$ th episode using  $\pi$ :  $\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
- For each state  $S_t$  and action  $A_t$  in the episode,

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

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} (G_t - Q(S_t, A_t))$$

- Improve policy based on new action-value function

$$\epsilon \leftarrow 1/k$$

$$\pi \leftarrow \epsilon\text{-greedy}(Q)$$

## Theorem

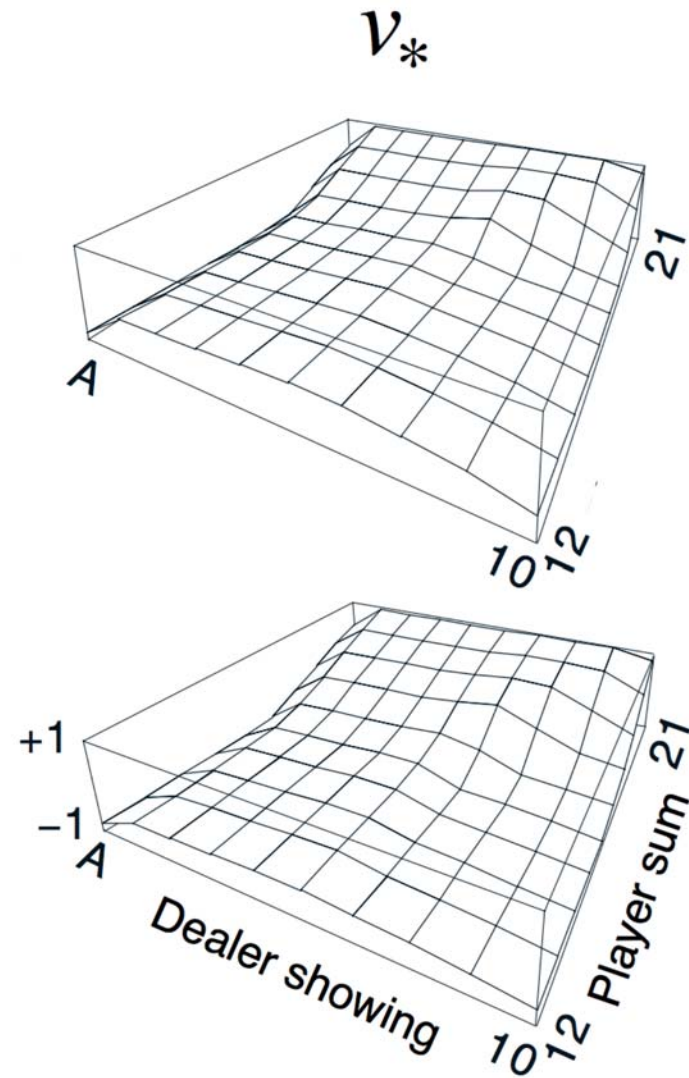
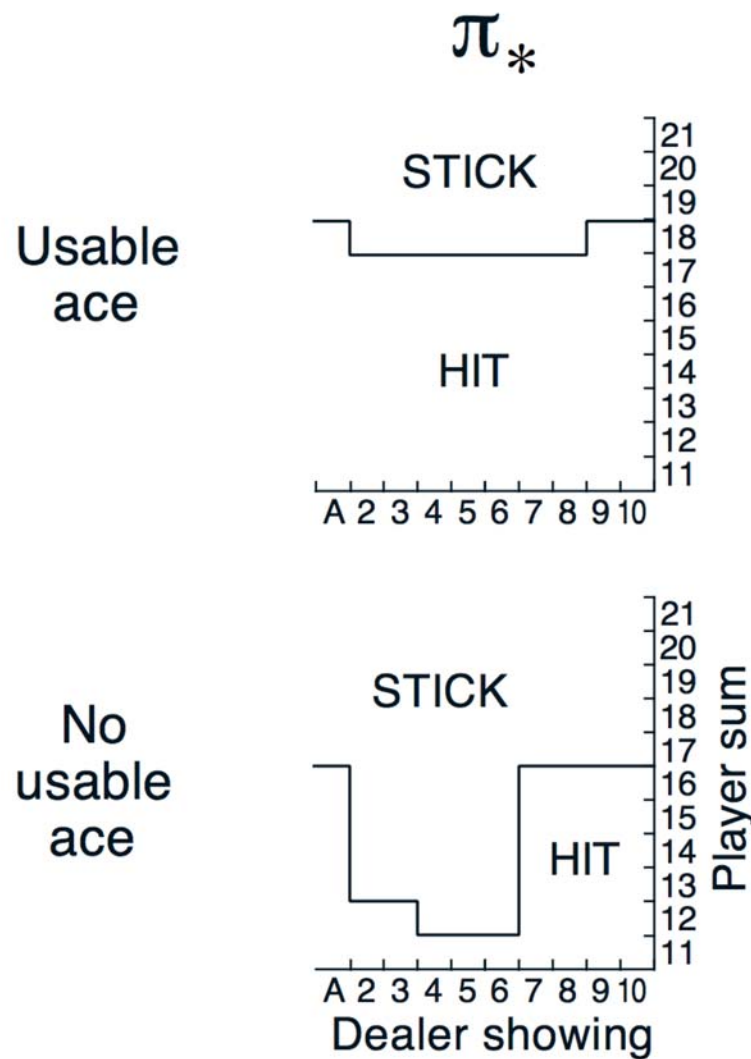
*GLIE Monte-Carlo control converges to the optimal action-value function,  $Q(s, a) \rightarrow q_*(s, a)$*



# Back to the Blackjack Example



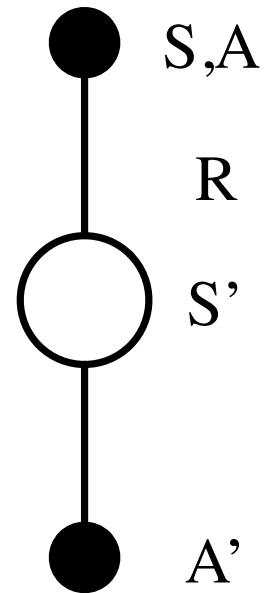
# Monte-Carlo Control in Blackjack



# MC vs. TD Control

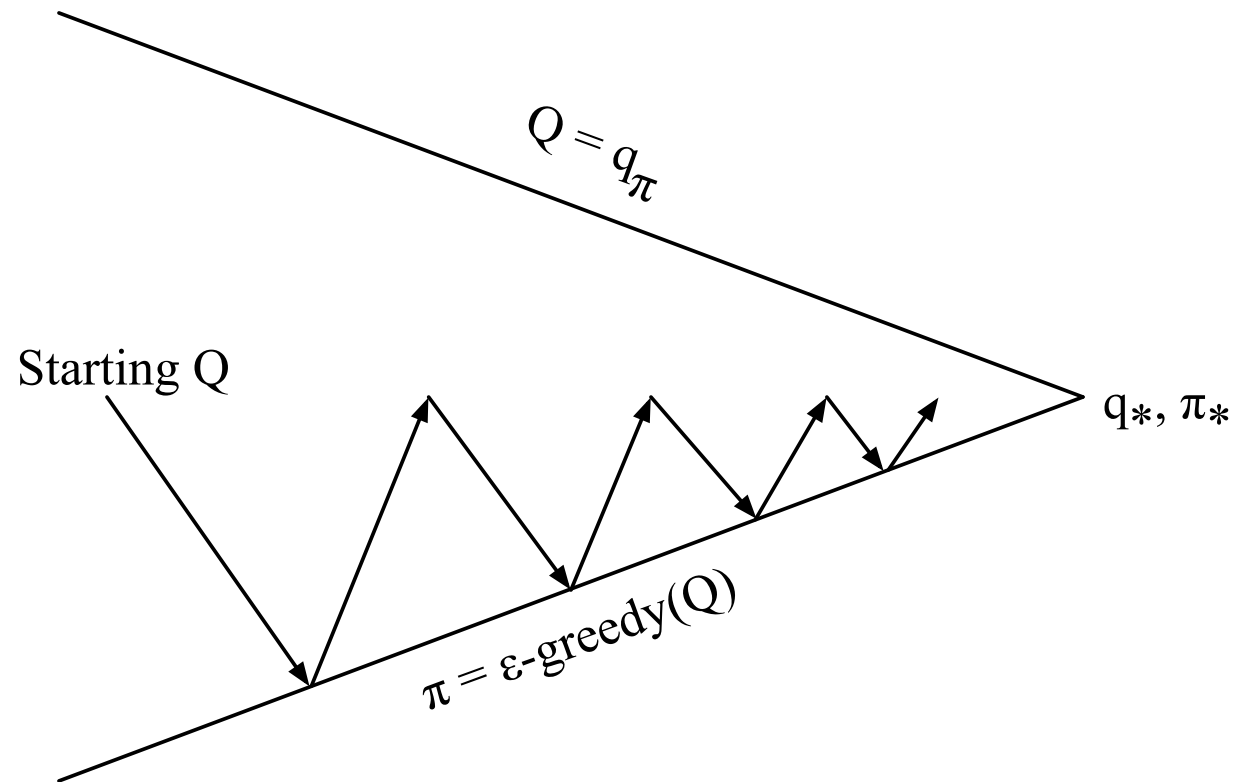
- Temporal-difference (TD) learning has several advantages over Monte-Carlo (MC)
  - Lower variance
  - Online
  - Incomplete sequences
- Natural idea: use TD instead of MC in our control loop
  - Apply TD to  $Q(S, A)$
  - Use  $\epsilon$ -greedy policy improvement
  - Update every time-step

# 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.,  $\varepsilon$ -greedy)

Repeat (for each step of episode):

Take action  $A$ , observe  $R, S'$

Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\varepsilon$ -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

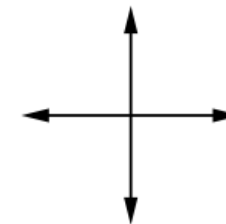
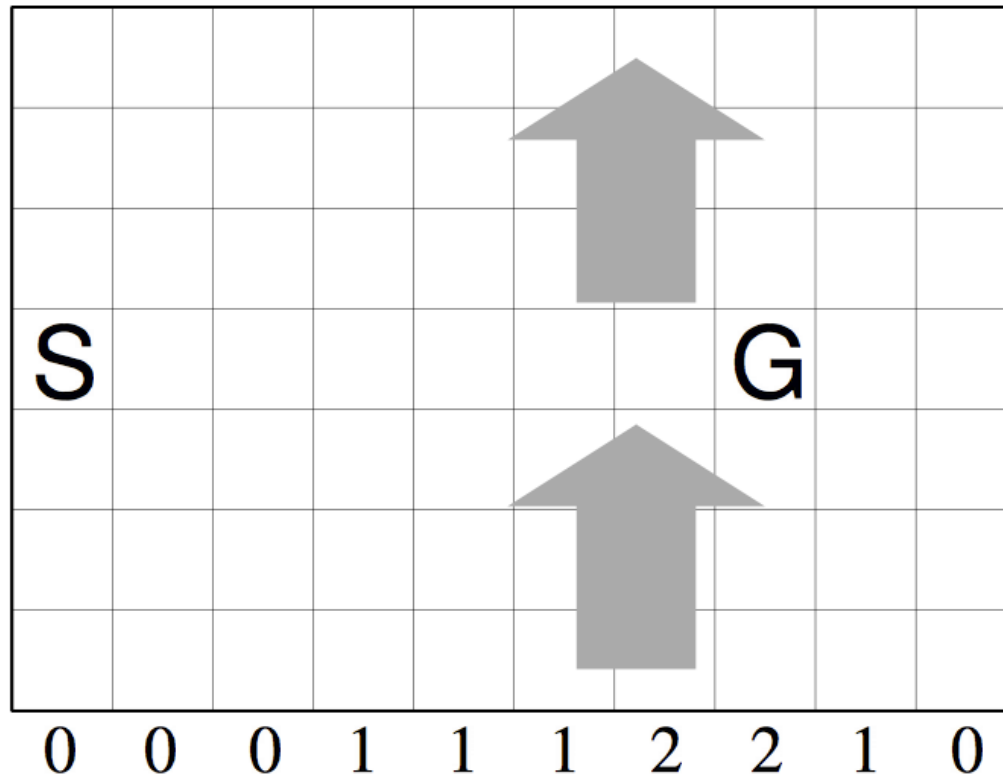
*Sarsa converges to the optimal action-value function,  $Q(s, a) \rightarrow q_*(s, a)$ , under the following conditions:*

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

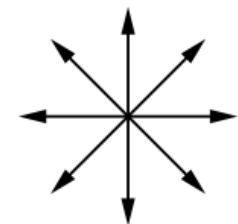
$$\sum_{t=1}^{\infty} \alpha_t = \infty$$

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

# Windy Gridworld Example



standard  
moves

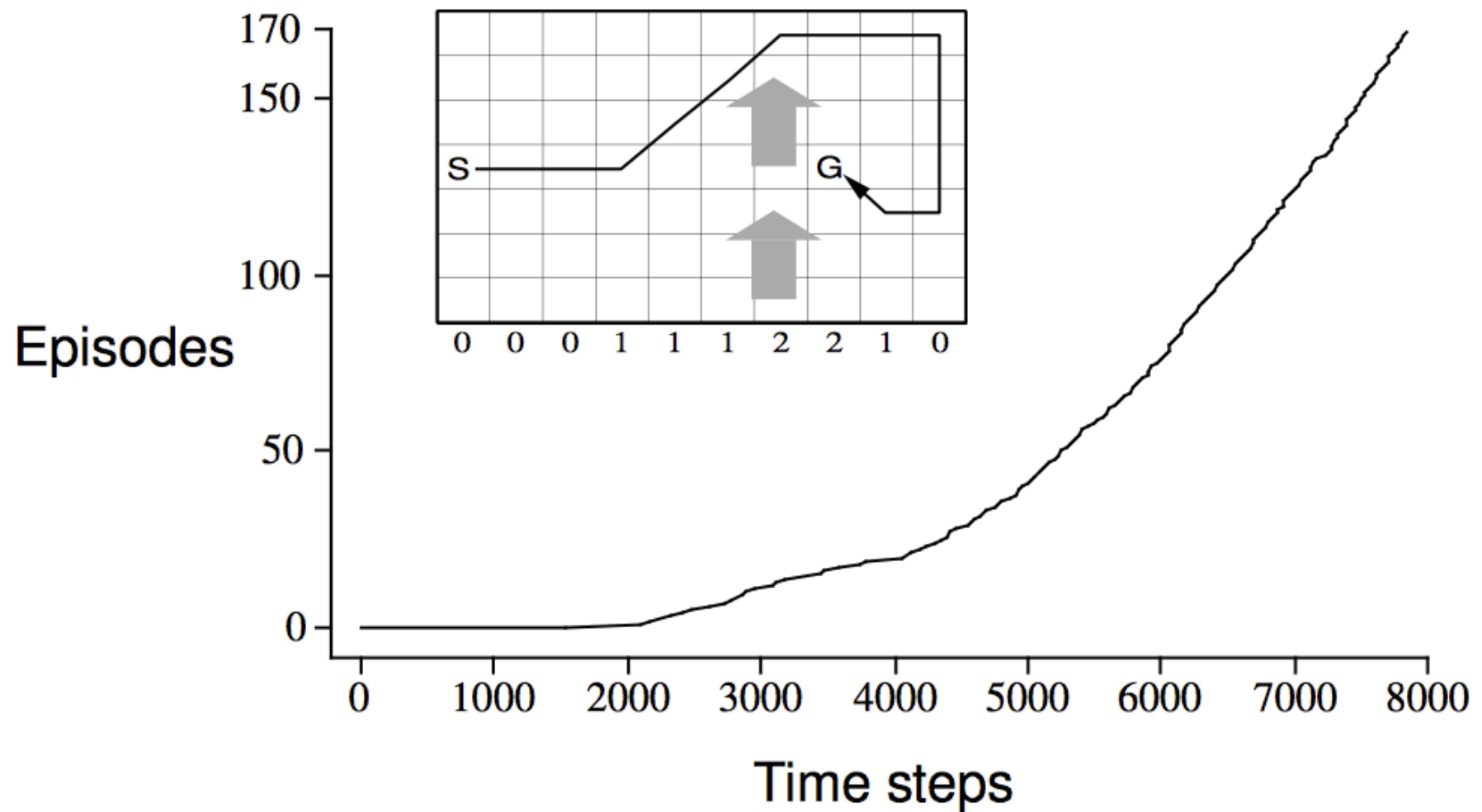


king's  
moves

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



# Sarsa on the Windy Gridworld



# $n$ -Step Sarsa

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

$$\begin{array}{ll} n = 1 & \text{(Sarsa)} \quad q_t^{(1)} = R_{t+1} + \gamma Q(S_{t+1}) \\ n = 2 & q_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q(S_{t+2}) \\ & \vdots \\ n = \infty & \text{(MC)} \quad q_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T \end{array}$$

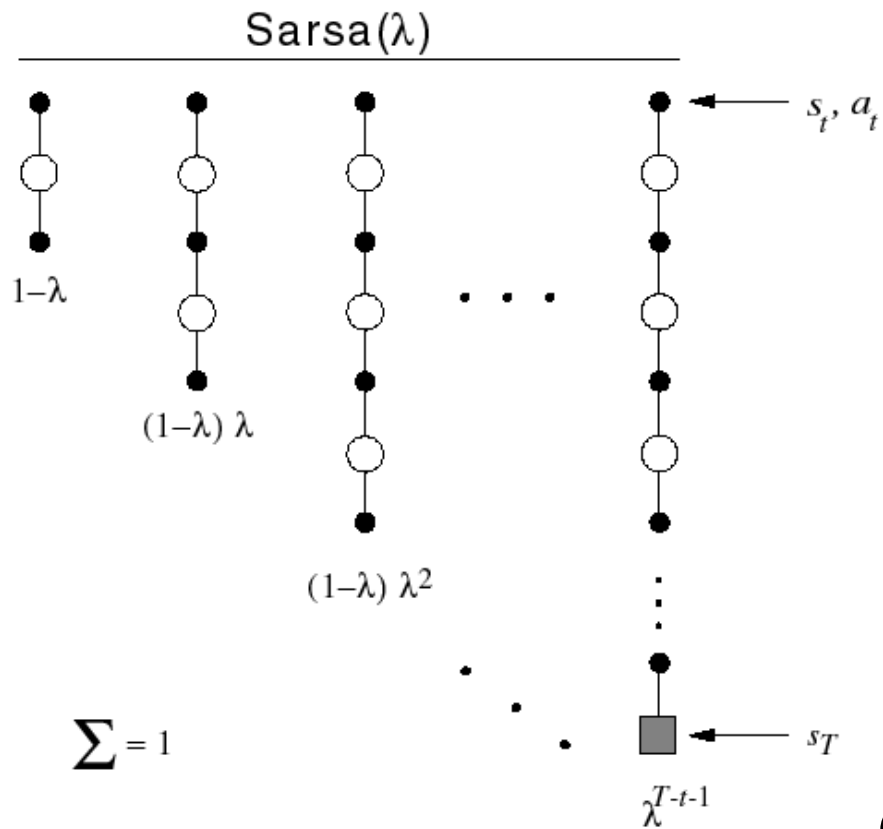
- Define the  $n$ -step Q-return

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

- $n$ -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  $s \in \mathcal{S}, a \in \mathcal{A}(s)$ :

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

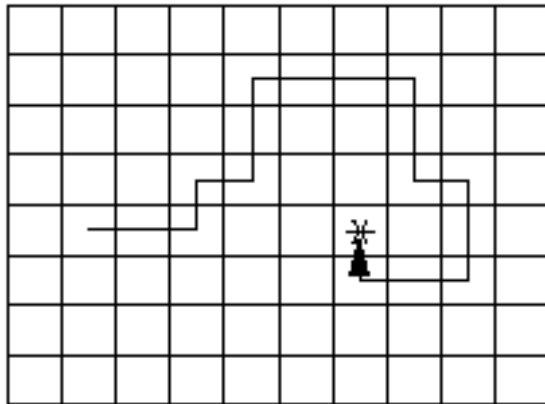
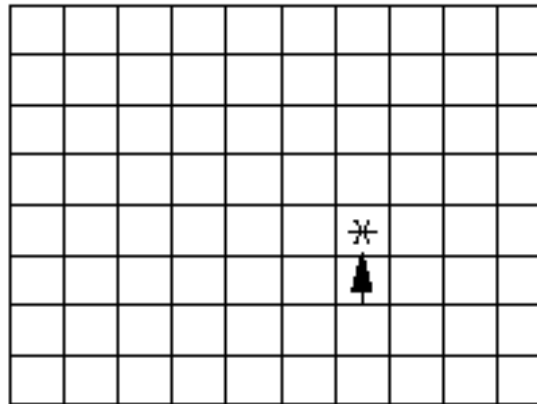
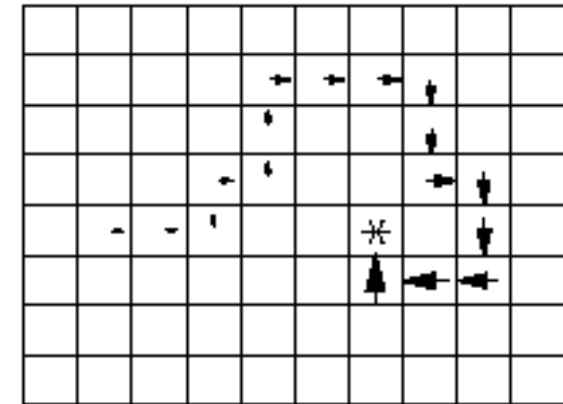
$E(s, a) \leftarrow \gamma \lambda E(s, a)$

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

    until  $S$  is terminal

# Sarsa( $\lambda$ ) Gridworld Example

Path taken

Action values increased  
by one-step SarsaAction 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: a general technique for estimating expected value under one distribution given samples from another



# 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

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

# Off-Policy Control with Q-Learning

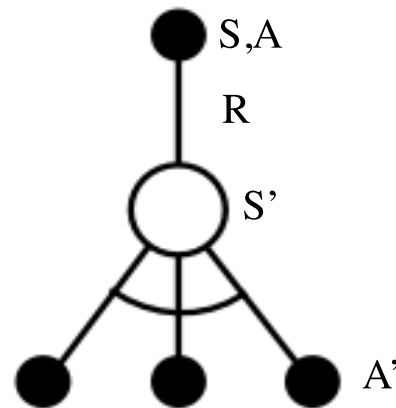
- We now allow both behaviour and target policies to **improve**
- The target policy  $\pi$  is **greedy** w.r.t.  $Q(s, a)$

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

- 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)

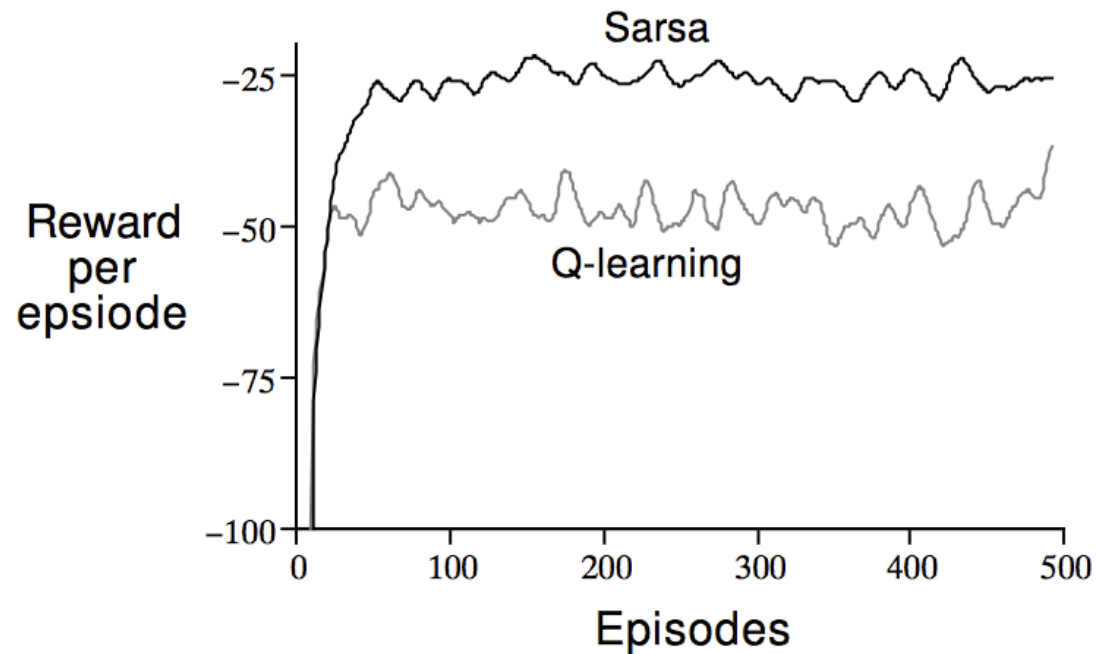
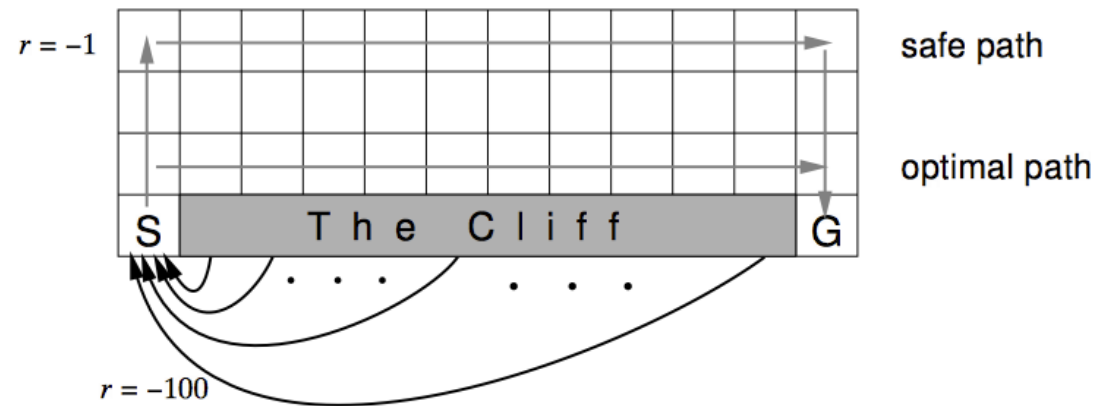
Take action  $A$ , observe  $R, S'$

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

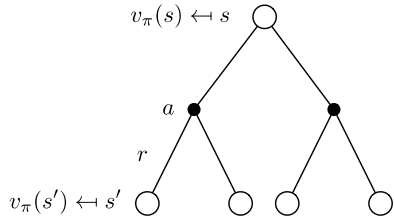
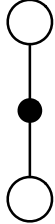
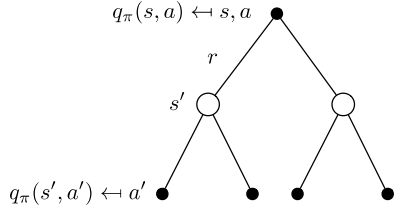
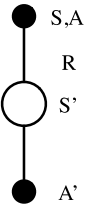
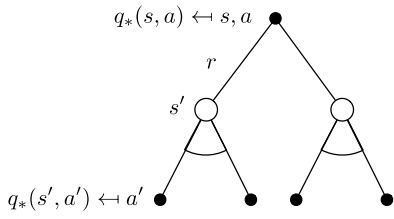
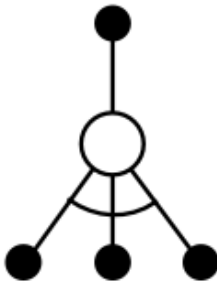
$S \leftarrow S'$ ;

until  $S$  is terminal

# Cliff Walking Example



# Relationship Between DP and 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>



# Relationship Between DP and TD (2)

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

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