

# Model Free Control

岩延 [yany@ucas.ac.cn](mailto:yany@ucas.ac.cn)

# Model Free Reinforcement Learning

## Introduction

- ▶ **Last Chapter**
  - Model Free prediction
    - Estimate the value function of an **unknown** MDP
- ▶ **Model Free control**
  - Optimize the value function of an **unknown** MDP

# Monte Carlo Methods

## Monte Carlo Control

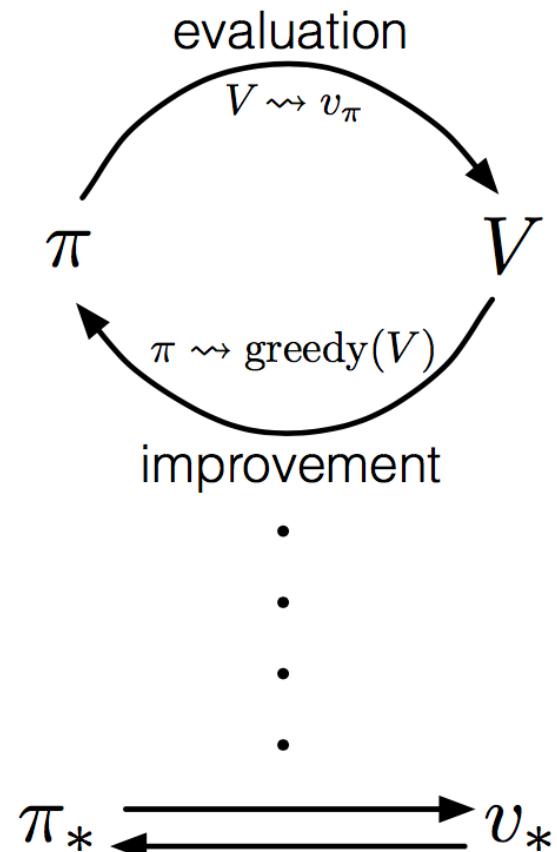
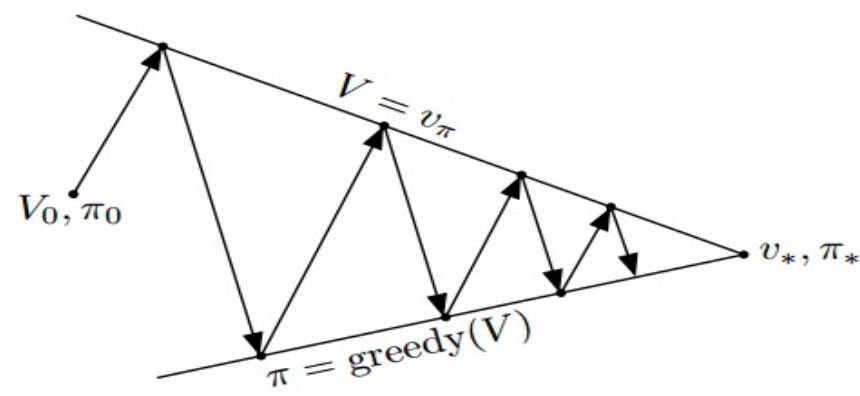
- ▶ 模型不可用时，估计**Action value**  $q_{\pi}(s, a)$ 更加有用
- ▶ 与**State Value**方法基本相同
- ▶ 可能会有许多状态-动作对（**state-action pairs**）从未访问到
  - 无法求平均
  - 必须保证持续的探索
- ▶ **Exploring starts**
  - 从特定的状态动作对出发，对每种动作都有大于零的概率选择到。这能够保证经历无限个回合后，所有的状态-动作对（**state-action pair**）都会被访问到无限次
  - 不具备普遍意义
  - 采用随机策略更为普遍

# Monte Carlo Methods

## Monte Carlo Control

### ► 广义策略迭代 (GPI)

- Policy evaluation和Policy improvement任意交互
- 几乎所有的强化学习方法都可以被描述为GPI



# Monte Carlo Methods

## Monte Carlo Control

- ▶ 使用蒙特卡洛估计来解决控制问题

$$\pi_0 \xrightarrow{E} q_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} q_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} q_*,$$

- ▶ Policy evaluation
  - MC prediction
- ▶ Policy improvement
  - 可使用贪心Greedy方法
  - 基于 $v(s)$ 的Greedy Policy improvement需要MDP的模型
    - $\pi'(s) = \operatorname{argmax}_a v_\pi(s)$
    - $= \operatorname{argmax}_a r_s^a + \gamma \sum_{s'} p_{ss'}^a v_\pi(s')$
  - 直接选择使得Action value最大的动作
    - $\pi'(s) = \operatorname{argmax}_a q_\pi(s, a)$

# Monte Carlo Methods

## Monte Carlo Control

### ► Policy improvement

- 针对  $q_{\pi_k}$  构建下一个贪心策略  $\pi_{k+1}$
- 可应用 Policy Improvement Theorem
  - $q_{\pi_k}(s, \pi_{k+1}(s)) = q_{\pi_k}\left(s, \operatorname{argmax}_a q_{\pi_k}(s, a)\right)$
  - $= \max_a q_{\pi_k}(s, a)$
  - $\geq q_{\pi_k}(s, \pi_k(s))$
  - $\geq v_{\pi_k}(s)$
- 上述推导基于两个假设
  - Exploring starts
  - Episode 足够多
- 如何去除上述假设?
  - Exploring starts
    - $\varepsilon$  - greedy
  - Episode 足够多
    - 采用与上一章相同思想

# Monte Carlo Methods

## Monte Carlo Control

### ► Monte Carlo with Exploring Starts

Monte Carlo ES (Exploring Starts), for estimating  $\pi \approx \pi_*$

Initialize:

$\pi(s) \in \mathcal{A}(s)$  (arbitrarily), for all  $s \in \mathcal{S}$

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

$Returns(s, a) \leftarrow$  empty list, for all  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$

Loop forever (for each episode):

Choose  $S_0 \in \mathcal{S}$ ,  $A_0 \in \mathcal{A}(S_0)$  randomly such that all pairs have probability  $> 0$

Generate an episode from  $S_0, A_0$ , following  $\pi$ :  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode,  $t = T-1, T-2, \dots, 0$ :

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair  $S_t, A_t$  appears in  $S_0, A_0, S_1, A_1, \dots, S_{t-1}, A_{t-1}$ :

Append  $G$  to  $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$\pi(S_t) \leftarrow \arg \max_a Q(S_t, a)$

# Monte Carlo Methods

## On and Off-Policy Learning

- ▶ Greedy方法的问题
  - 策略迭代可能会陷入僵局
- ▶ 如何保证每个动作都能被选择?
  - On-policy
    - Learn on the job
    - Learn about policy  $\pi$  from experience sampled from  $\pi$
  - Off-policy
    - Learn about policy  $\pi$  from experience sampled from  $\mu$



# Monte Carlo Methods

## On-policy Learning

### ► Soft on-policy control

- 对于所有的 $s, a$ , 概率 $\pi(a|s) > 0$ , 并逐渐接近确定最优策略

### ► $\epsilon$ -greedy Exploration

- 最简单的保持持续Exploration的方法
- 对于所有的 $s, a$ , 概率 $\pi(s|a) > 0$
- 以 $1-\epsilon$ 的概率选择Greedy Action
- 以 $\epsilon$ 的概率随机选择其他action

$$\bullet \quad \pi(a|s) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|}, & \text{if } a = \underset{a}{\operatorname{argmax}} Q(s, a) \\ \frac{\epsilon}{|A(s)|}, & \text{otherwise} \end{cases}$$

# Monte Carlo Methods

## $\varepsilon$ -greedy Monte Carlo Control

### ► On-policy first-visit MC control

On-policy first-visit MC control (for  $\varepsilon$ -soft policies), estimates  $\pi \approx \pi_*$

Algorithm parameter: small  $\varepsilon > 0$

Initialize:

$\pi \leftarrow$  an arbitrary  $\varepsilon$ -soft policy

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

$Returns(s, a) \leftarrow$  empty list, for all  $s \in \mathcal{S}$ ,  $a \in \mathcal{A}(s)$

Repeat forever (for each episode):

Generate an episode following  $\pi$ :  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$

Loop for each step of episode,  $t = T-1, T-2, \dots, 0$ :

$G \leftarrow \gamma G + R_{t+1}$

Unless the pair  $S_t, A_t$  appears in  $S_0, A_0, S_1, A_1, \dots, S_{t-1}, A_{t-1}$ :

Append  $G$  to  $Returns(S_t, A_t)$

$Q(S_t, A_t) \leftarrow \text{average}(Returns(S_t, A_t))$

$A^* \leftarrow \arg \max_a Q(S_t, a)$  (with ties broken arbitrarily)

For all  $a \in \mathcal{A}(S_t)$ :

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$$

# Monte Carlo Methods

## $\varepsilon$ -greedy Monte Carlo Control

▶ 可应用 Policy improvement theorem

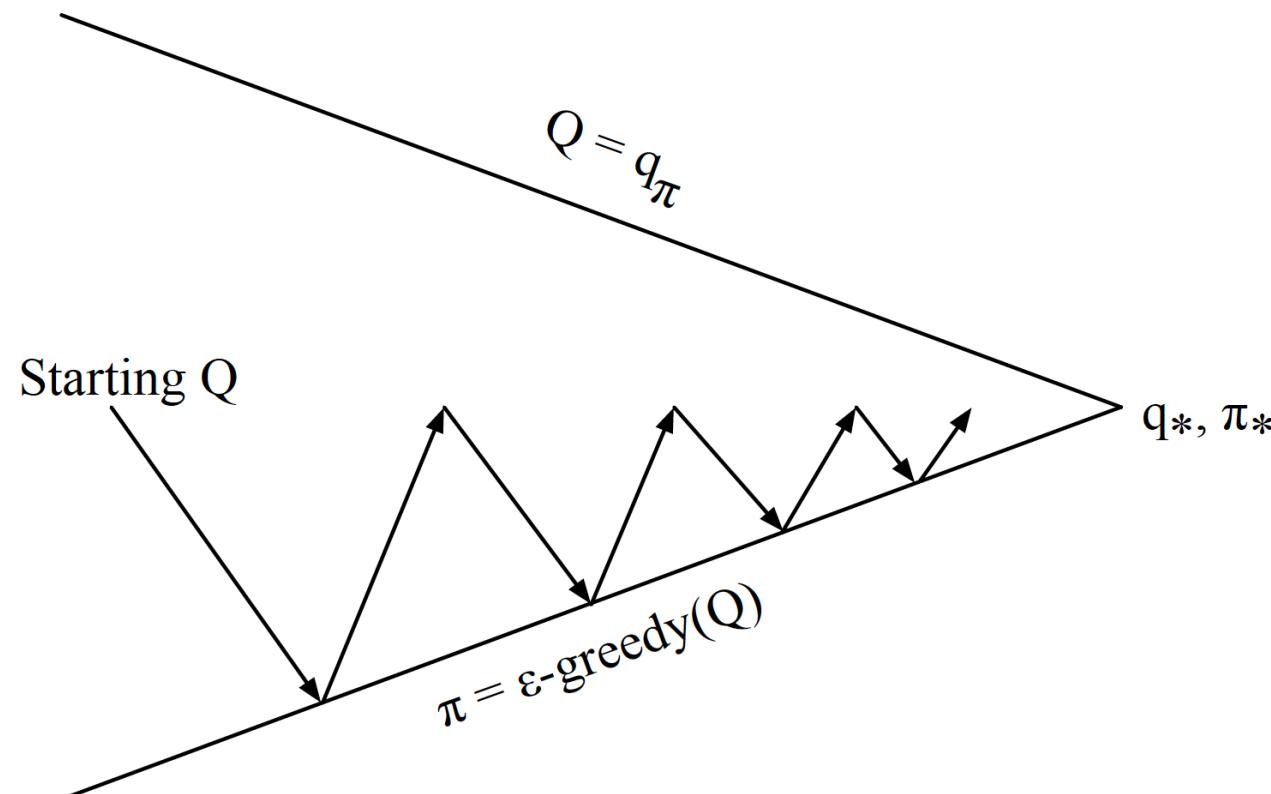
- $q_{\pi}(s, \pi'(s)) = \sum_a \pi'(a|s)q_{\pi}(s, a)$
- $= \frac{\varepsilon}{|A(s)|} \sum_a q_{\pi}(s, a) + (1 - \varepsilon) \max_a q_{\pi}(s, a)$
- $\geq \frac{\varepsilon}{|A(s)|} \sum_a q_{\pi}(s, a) + (1 - \varepsilon) \sum_a \frac{\pi(a|s) - \frac{\varepsilon}{|A(s)|}}{1 - \varepsilon} q_{\pi}(s, a)$
- $= \frac{\varepsilon}{|A(s)|} \sum_a q_{\pi}(s, a) - \frac{\varepsilon}{|A(s)|} \sum_a q_{\pi}(s, a) + \sum_a \pi(a|s)q_{\pi}(s, a)$
- $= v_{\pi}(s)$
- 因此  $\pi' \geq \pi$

# Monte Carlo Methods

## $\varepsilon$ -greedy Monte Carlo Control

- ▶ Every episode:

- Policy evaluation: Monte-Carlo policy evaluation  $Q \approx q_\pi$
- Policy improvement:  $\varepsilon$ -greedy policy improvement



# Monte Carlo Methods

## GLIE

### ► GLIE: Greedy in the Limit with Infinite Exploration

- 所有的State-Action均被探索无限次
  - $\lim_{k \rightarrow \infty} N_k(s, a) = \infty$
- Policy收敛至贪心Policy
  - $\lim_{k \rightarrow \infty} \pi_k(a|s) = 1(a = \operatorname{argmax}_{a' \in A} Q_k(s, a'))$

### ► $\varepsilon$ -greedy满足GLIE如果 $\varepsilon_k = \frac{1}{K}$

# Monte Carlo Methods

## GLIE Monte Carlo Control

► **GLIE Monte Carlo Control**收敛到最优的Action-value function  $Q(s, a) \rightarrow q_*(s, a)$

- 基于策略 $\pi$ 采样第 $k$ 个episode $\{S_1, A_1, R_2, \dots, S_T\} \sim \pi$
- 对于episode中每个 $S_t, A_t$ 
  - $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$
- 使用新的action-value function提升policy
  - $\varepsilon \leftarrow \frac{1}{K}$
  - $\pi \leftarrow \varepsilon - \text{greedy}(Q)$

# Monte Carlo Methods

## Off-policy Learning

### ▶ 探索的困境

- 需要通过最优 (optimal) 的行为来学习动作价值
- 需要表现地非最优来探索所有的动作

### ▶ On-policy

- 学习的并非最优策略，而是仍然探索的近似-最优策略

### ▶ Off-policy

- 使用两个策略
- Target policy: 要学习的并变为最优的策略
- Behavior policy: 用于生成行为(behavior)的探索策略

# Monte Carlo Methods

## Off-policy Learning

### ► Off-policy Learning

- 学习Target Policy  $\pi$ :  $v_\pi$  or  $q_\pi$
- 使用Behaviour Policy  $\mu \neq \pi$
- Coverage假设
  - 要求任何 $\pi$ 采取的行动至少偶尔被 $\mu$ 采用
    - $\pi(a|s) > 0 \Rightarrow \mu(a|s) > 0$

### ► Importance Sampling

- 一种通过给定其他分布样本来估计另一种分布下期望的通用技巧
- $E_{X \sim p}[f(X)] = \sum P(X)f(X)$
- $= \sum Q(X) \frac{P(X)}{Q(X)}f(X)$
- $= E_{X \sim Q} \left[ \frac{P(X)}{Q(X)}f(X) \right]$

# Monte Carlo Methods

## Importance Sampling for Off-Policy MC

- ▶ 给定初始状态  $S_t$ , 在策略  $\pi$  下, 接下来的状态动作轨迹  $A_t, S_{t+1}, A_{t+1}, \dots, S_T$  发生的概率为
  - $\Pr\{A_t, S_{t+1}, A_{t+1}, \dots, S_T | S_t, A_{t:T-1} \sim \pi\}$
  - $= \pi(A_t | S_t) p(S_{t+1} | S_t, A_t) \pi(A_{t+1} | S_{t+1}) \dots p(S_T | S_{T-1}, A_{T-1})$
  - $= \prod_{k=t}^T \pi(A_k | S_k) p(S_{k+1} | S_k, A_k)$
- ▶ Importance-sampling 为在 Target policy 和 Behavior policy 下的该轨迹发生的相对概率
  - $\rho_t^T = \frac{\prod_{k=t}^T \pi(A_k | S_k) p(S_{k+1} | S_k, A_k)}{\prod_{k=t}^T \mu(A_k | S_k) p(S_{k+1} | S_k, A_k)} = \prod_{k=t}^T \frac{\pi(A_k | S_k)}{\mu(A_k | S_k)}$
- ▶ 可从 Behavior policy 的 return 得到 Target policy 的 return
  - $G_t^\pi = \rho_t^T G_t^\mu = \prod_{k=t}^T \frac{\pi(A_k | S_k)}{\mu(A_k | S_k)} G_t^\mu$
- ▶ Off-policy value
  - $q_\pi(s, t) = \mathbb{E}[G_t^\pi | S_t = s, A_t = a] = \mathbb{E}[\rho_t^T G_t^\mu | S_t = s, A_t = a]$
- ▶ Off-Policy MC
  - $Q_\pi(S_t, A_t) \leftarrow Q_\pi(S_t, A_t) + \alpha(G_t^\pi - Q_\pi(S_t, A_t))$

# Monte Carlo Methods

## Off-policy MC

### ▶ 使用MC方法计算Value function

- 两种方式
- Ordinary importance sampling
  - $$\frac{\sum_{t \in J(s,a)} \rho_t^T G_t^\mu}{|J(s,a)|}$$
  - Unbiased 估计
  - 方差较大
- Weighted importance sampling
  - $$\frac{\sum_{t \in J(s,a)} \rho_t^T G_t^\mu}{\sum_{t \in J(s,a)} \rho_t^T}$$
  - Biased 估计
  - 方差趋近于0

# Monte Carlo Methods

## Off-Policy Incremental Monte-Carlo Updates

### ▶ Ordinary importance sampling

- $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left( \frac{\sum_{t \in J(s,a)} \rho_t^T G_t^\mu}{|J(s,a)|} - Q(S_t, A_t) \right)$$

### ▶ Weighted importance sampling

- 对于从同一状态开始的Return序列  $G_1, G_2, \dots, G_{n-1}$ , 每个都有权重  $W_i = \rho_{t_i}^{T_i}$
- 希望估计

- $$Q_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k} \quad n \geq 2$$
- 增量更新规则为
- $$Q_{n+1} \leftarrow V_{n+1} + \frac{W_n}{C_n} (G_n - Q_n)$$
- $$C_{n+1} \leftarrow C_n + W_{n+1}$$

# Monte Carlo Methods

## Off-Policy Incremental Monte-Carlo Updates

Off-policy MC control, for estimating  $\pi \approx \pi_*$

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

$$Q(s, a) \in \mathbb{R} \text{ (arbitrarily)}$$

$$C(s, a) \leftarrow 0$$

$$\pi(s) \leftarrow \arg \max_a Q(s, a) \quad (\text{with ties broken consistently})$$

Loop forever (for each episode):

$b \leftarrow$  any soft policy

Generate an episode using  $b$ :  $S_0, A_0, R_1, \dots, S_{T-1}, A_{T-1}, R_T$

$$G \leftarrow 0$$

$$W \leftarrow 1$$

Loop for each step of episode,  $t = T-1, T-2, \dots, 0$ :

$$G \leftarrow \gamma G + R_{t+1}$$

$$C(S_t, A_t) \leftarrow C(S_t, A_t) + W$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{W}{C(S_t, A_t)} [G - Q(S_t, A_t)]$$

$$\pi(S_t) \leftarrow \arg \max_a Q(S_t, a) \quad (\text{with ties broken consistently})$$

If  $A_t \neq \pi(S_t)$  then exit inner Loop (proceed to next episode)

$$W \leftarrow W \frac{1}{b(A_t | S_t)}$$

# TD Control

## Sarsa: On-policy TD Control

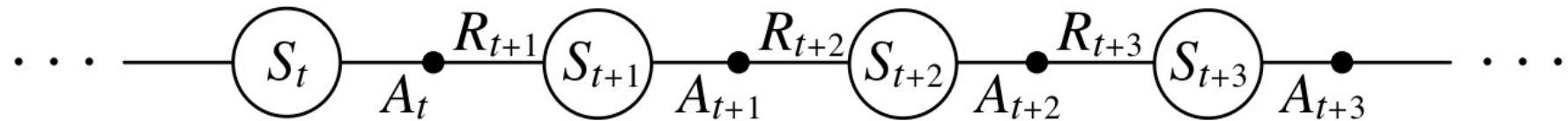
- ▶ TD learning has several advantages over MC
  - Lower variance
  - Online
  - Incomplete sequences
- ▶ 使用TD进行Control
  - Apply TD to  $Q(S, A)$
  - Use  $\varepsilon$ -greedy policy improvement
  - Update every time-step

# TD Control

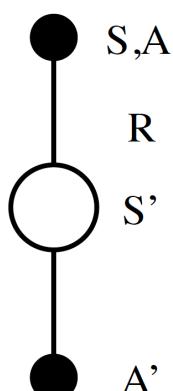
## Sarsa: On-policy TD Control

### ▶ Learning Action-value function on policy

- Estimate  $q_{\pi}(s, a)$  for current policy  $\pi$



- 在  $q_{\pi}(s, a)$  上应用 TD(0)
  - TD(0)
    - $V(S_t) \leftarrow V(S_t) + \alpha(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$
  - Sarsa  $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ 
    - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t))$

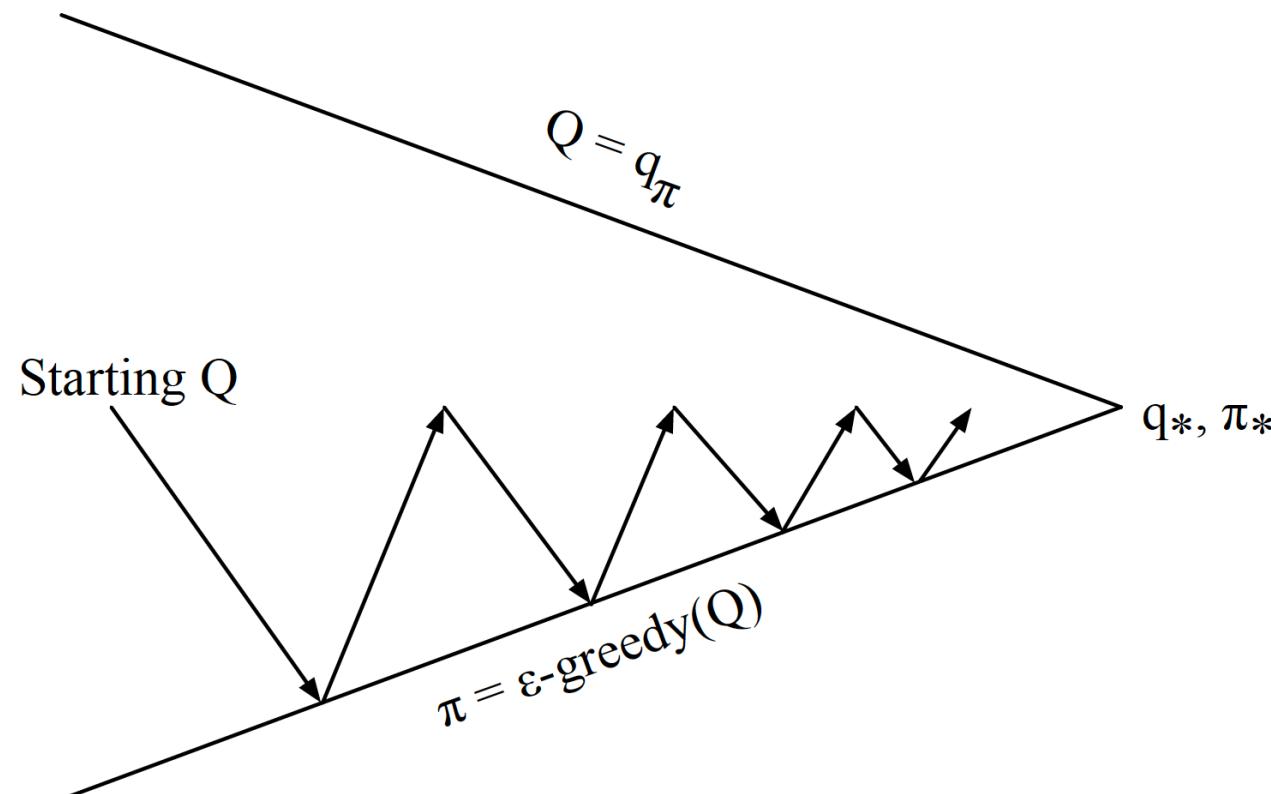


# TD Control

## Sarsa: On-policy TD Control

- ▶ Every **time-step**:

- Policy evaluation: Sarsa  $Q \approx q_\pi$
- Policy improvement:  $\epsilon$ -greedy policy improvement



# TD Control

## Sarsa: On-policy TD Control

- ▶ 在满足GLIE的条件下可以概率1收敛到最优策略和Action-value function

Sarsa (on-policy TD control) for estimating  $Q \approx q_*$

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\varepsilon > 0$

Initialize  $Q(s, a)$ , for all  $s \in \mathcal{S}^+$ ,  $a \in \mathcal{A}(s)$ , arbitrarily except that  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $S$

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

    Loop 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

# TD Control

## Sarsa: On-policy TD Control

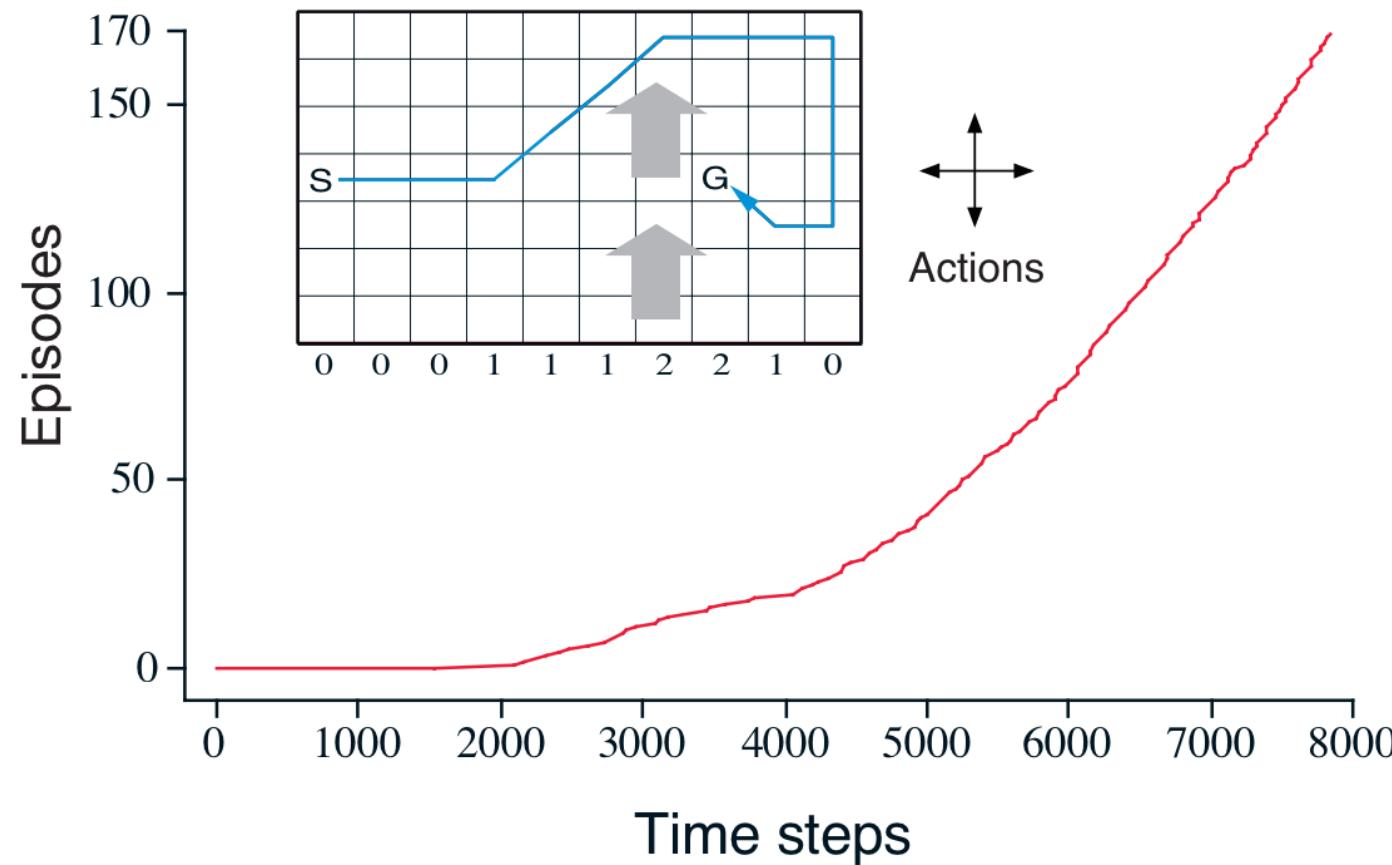
### ► Convergence of Sarsa

- Sarsa在满足下列条件的情况下可以收敛到最优的Action-value function  $Q(s, a) \rightarrow q_*(s, a)$ 
  - GLIE 策略序列  $\pi_t(a|s)$
  - 步长  $\alpha_t$  满足
    - $\sum_{t=1}^{\infty} \alpha_t = \infty$
    - $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$

# TD Control

## Sarsa: On-policy TD Control

- ▶ Example: Windy Gridworld



# TD Control

## Importance Sampling for Off-Policy TD Prediction

### ▶ Importance-sampling ratio for Off-Policy MC

— 从Behavior policy的return得到Target policy的return

- $\rho_t^T = \frac{\prod_{k=t}^T \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^T \mu(A_k|S_k) p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^T \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$
- $G_t^\pi = \rho_t^T G_t^\mu = \prod_{k=t}^T \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)} G_t^\mu$

### ▶ Importance-sampling ratio for Off-Policy TD prediction

— 使用由 $\mu$ 生成的TD target来评估 $\pi$

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

— 只需单步权重

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

# TD Control

## Q-learning: Off-policy TD Control

- ▶ 采用 off policy 方式学习 Action-value function  $q_{\pi}(s, a)$ 
  - 但无需 importance sampling
- ▶ 下一个动作是由 Behavior policy 选择
  - $A_{t+1} \sim \mu(\cdot | S_t)$
- ▶ 但是考虑 Target policy 的 action
  - $A' \sim \pi(\cdot | S_t)$
- ▶ 使用  $A'$  来更新  $q_{\pi}(s, a)$ 
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$

# TD Control

## Q-learning: Off-policy TD Control

- ▶ Behavior policy和Target policy同时提升
- ▶ Target policy是贪心策略
  - $\pi(S_{t+1}) = \operatorname{argmax}_{a'} Q(S_{t+1}, a')$
- ▶ Behavior policy是 $\varepsilon$ -greedy策略
- ▶ Q-learning Target为
  - $R_{t+1} + \gamma Q(S_{t+1}, A')$
  - $= R_{t+1} + \gamma Q(S_{t+1}, \operatorname{argmax}_{a'} Q(S_{t+1}, a'))$
  - $= R_{t+1} + \gamma \max_{a'} Q(S_{t+1}, a')$

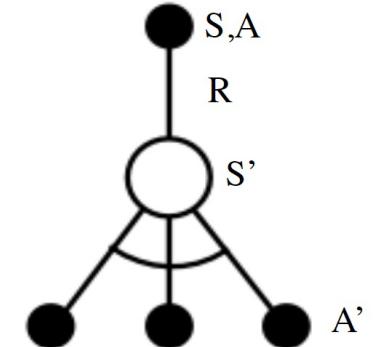
# TD Control

## Q-learning: Off-policy TD Control

### ► Q-learning

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

### ► Q-learning 以概率1收敛至最优 Action-value function $Q(s, a) \rightarrow q_*(s, a)$



Q-learning (off-policy TD control) for estimating  $\pi \approx \pi_*$

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\varepsilon > 0$

Initialize  $Q(s, a)$ , for all  $s \in \mathcal{S}^+$ ,  $a \in \mathcal{A}(s)$ , arbitrarily except that  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $S$

    Loop for each step of episode:

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

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

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

$$S \leftarrow S'$$

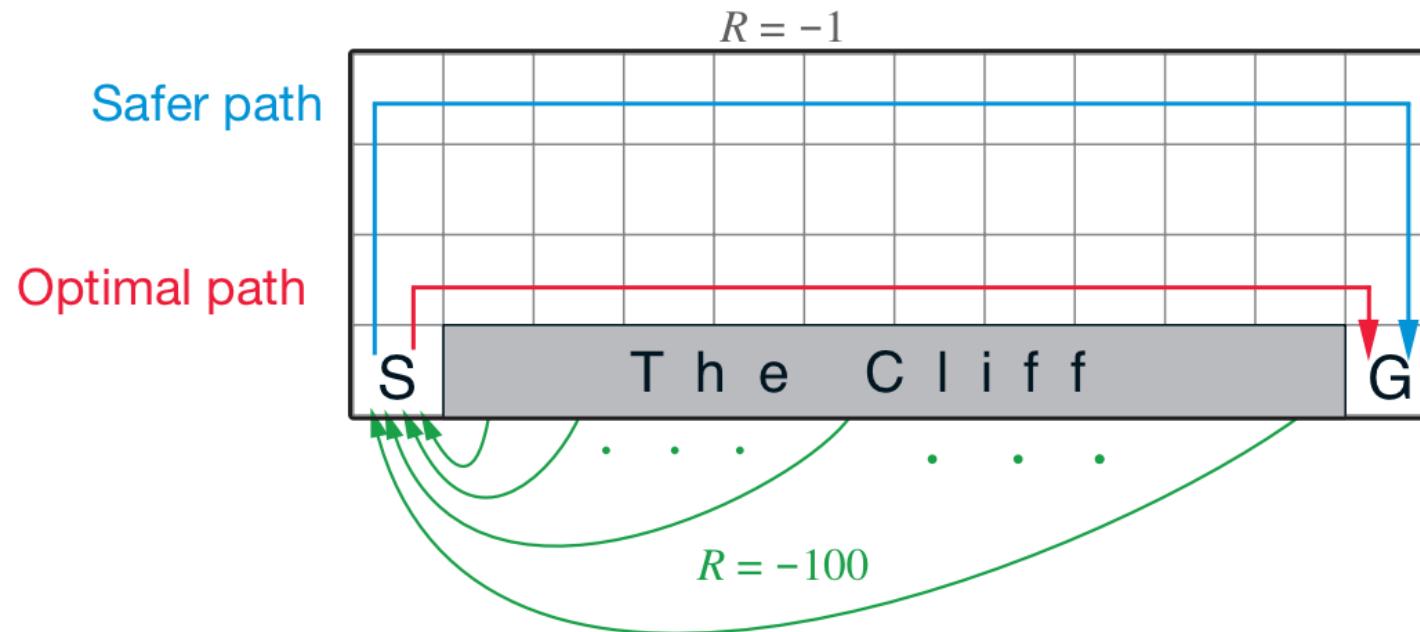
    until  $S$  is terminal

# TD Control

## Q-learning: Off-policy TD Control

### ► Example: Cliff Walking

- Gridworld with “cliff” with high negative reward
- $\epsilon$ -greedy (behavior) policy for both Sarsa and Q-learning ( $\epsilon = 0.1$ )

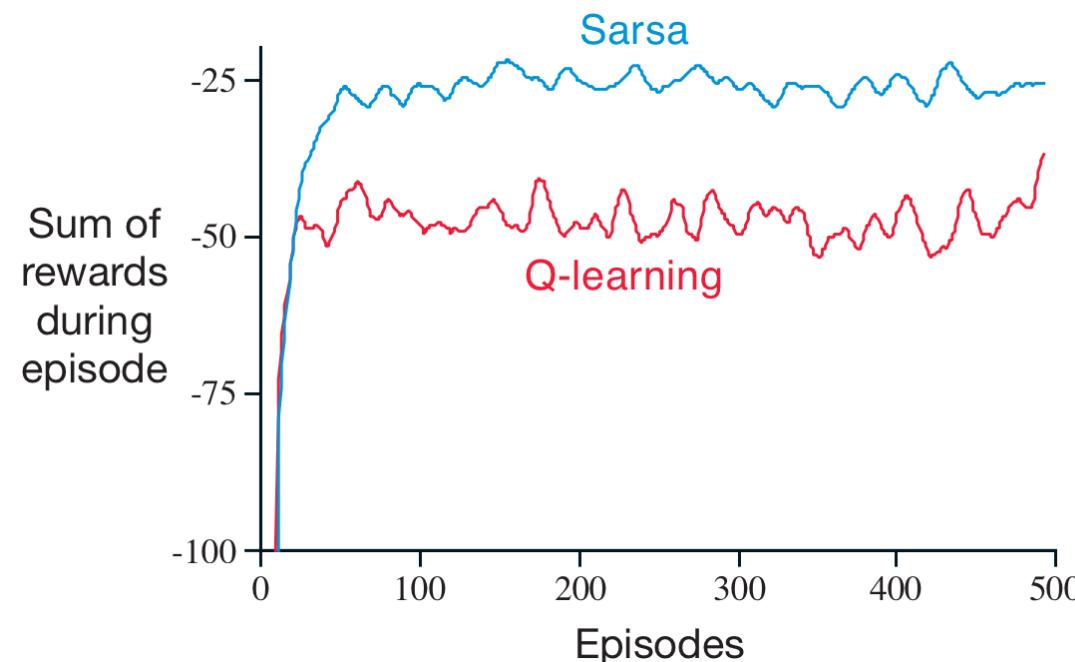


# TD Control

## Q-learning: Off-policy TD Control

### ► Example: Cliff Walking

- Q-learning learns optimal policy
- Sarsa learns safe policy
- Q-learning has worse online performance
- Both reach optimal policy with  $\varepsilon$ -decay



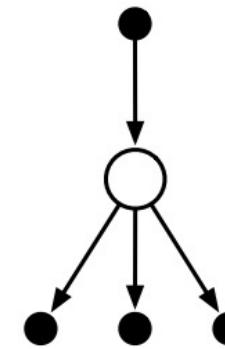
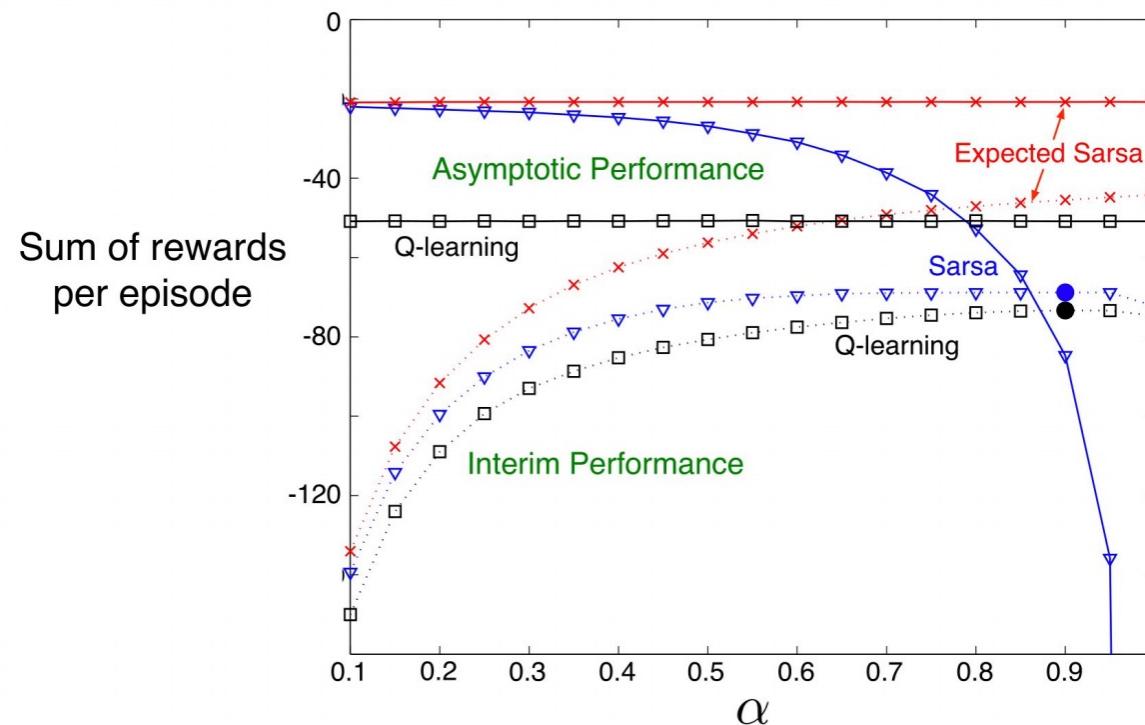
# TD Control

## Expected Sarsa

► 不使用 $Q$ 的最大值(Q-learning), 而是使用 $Q$ 的期望值

- $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \mathbb{E}_\pi [Q(S_{t+1}, A_{t+1}) | S_{t+1}] - Q(S_t, A_t))$$
- $$\leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \sum_{a'} \pi(a' | S_{t+1}) Q(S_{t+1}, a') - Q(S_t, A_t))$$

► 排除了因随机选择 $A_{t+1}$ 造成的方差



# TD Control

## Maximization Bias

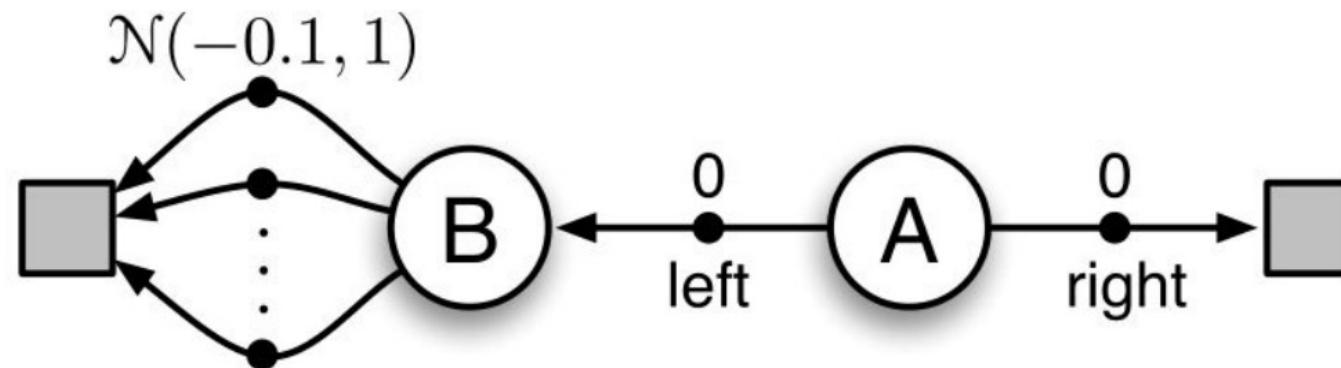
- ▶ 目前的算法都包括了**maximization**操作
  - Sarsa:  $\varepsilon$ -greedy
  - Q-learning: Greedy target policy
- ▶ 可带来显著的**positive bias**
  - 称为maximization bias

# TD Control

## Maximization Bias

### ▶ Example

- Action and Reward
  - left/right in A, reward 0
  - 10 action in B, each gives reward from  $\mathcal{N}(-0.1, 1)$
- 最优策略是在A总是选择right

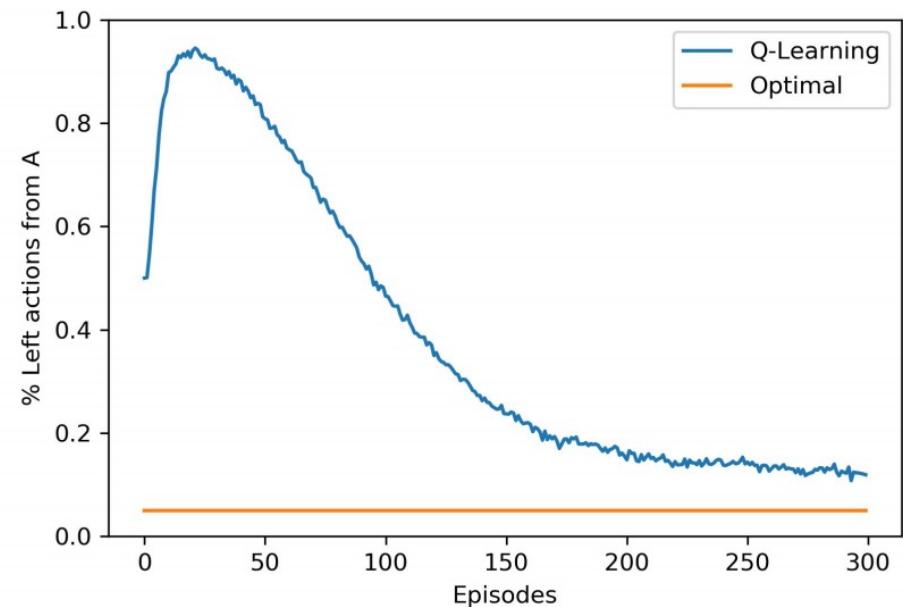
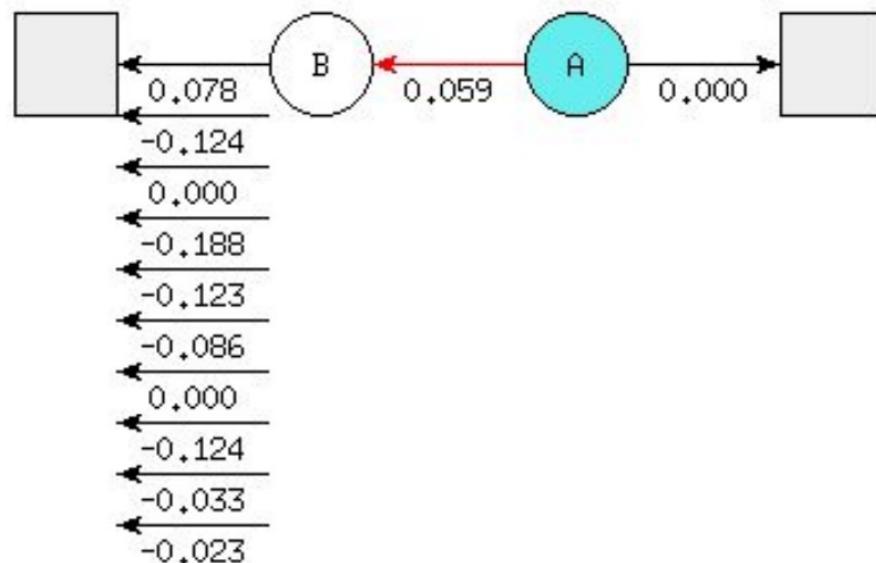


# TD Control

## Maximization Bias

### ▶ Example

- One positive action value causes maximization bias



# TD Control

## Double Q-Learning

### ► Maximization Bias产生的原因

- 使用同样的sample做两件事
  - Determining the maximizing action
  - Estimating action value

### ► 解决方法

- 使用两个Action-value的估计值 $Q_1$ ,  $Q_2$ 
  - 每次更新一个

$$\triangleright Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha(R_{t+1} + \gamma \underset{a}{\operatorname{argmax}} Q_1(S_{t+1}, a)) - Q_1(S_t, A_t))$$

$$\triangleright Q_2(S_t, A_t) \leftarrow Q_2(S_t, A_t) + \alpha(R_{t+1} + \gamma \underset{a}{\operatorname{argmax}} Q_2(S_{t+1}, a)) - Q_2(S_t, A_t))$$

# TD Control

## Double Q-Learning

Double Q-learning, for estimating  $Q_1 \approx Q_2 \approx q_*$

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\varepsilon > 0$

Initialize  $Q_1(s, a)$  and  $Q_2(s, a)$ , for all  $s \in \mathcal{S}^+$ ,  $a \in \mathcal{A}(s)$ , such that  $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

    Initialize  $S$

    Loop for each step of episode:

        Choose  $A$  from  $S$  using the policy  $\varepsilon$ -greedy in  $Q_1 + Q_2$

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

        With 0.5 probability:

$$Q_1(S, A) \leftarrow Q_1(S, A) + \alpha \left( R + \gamma Q_2(S', \arg \max_a Q_1(S', a)) - Q_1(S, A) \right)$$

        else:

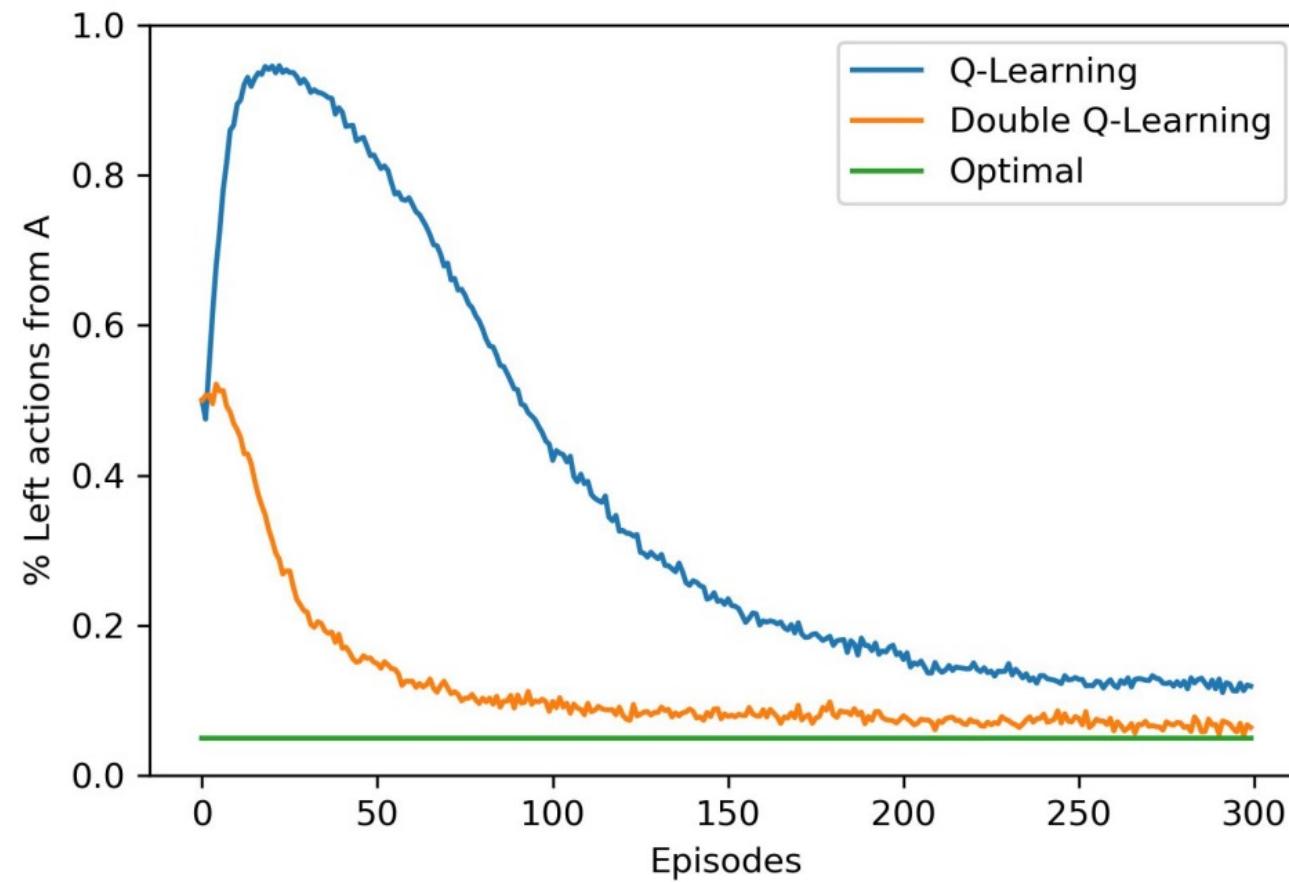
$$Q_2(S, A) \leftarrow Q_2(S, A) + \alpha \left( R + \gamma Q_1(S', \arg \max_a Q_2(S', a)) - Q_2(S, A) \right)$$

$S \leftarrow S'$

    until  $S$  is terminal

# TD Control

## Double Q-Learning

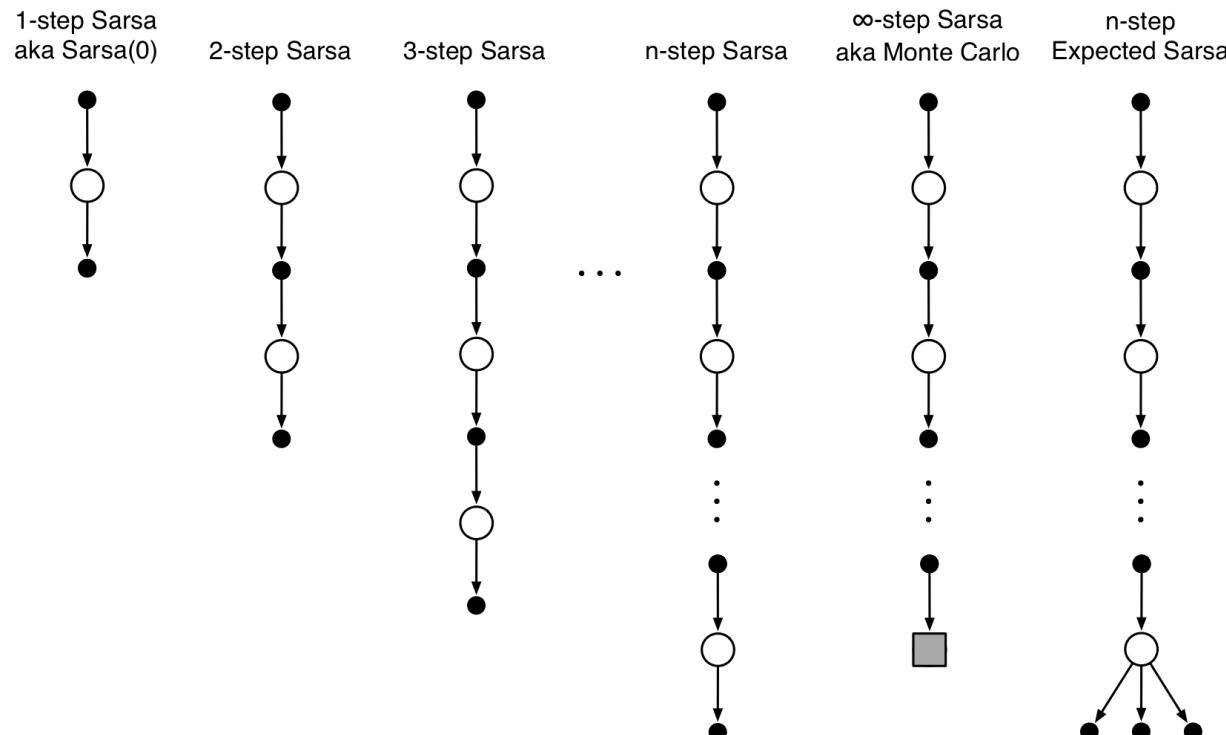


# n-step Control

## n-step Sarsa

### ► Extend n-step TD prediction to Control (Sarsa)

- Use  $Q$  instead of  $V$
- Use  $\varepsilon$ -greedy policy



# n-step Control

## n-step Sarsa

### ► Redefine n-step return with $Q$

- MC: complete return
  - $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-1} R_T$
- Sarsa: one-step Q-return
  - $G_{t:t+1} = R_{t+1} + \gamma Q_t(S_{t+1}, A_{t+1})$
- two-step Q-return
  - $G_{t:t+2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 Q_{t+1}(S_{t+2}, A_{t+2})$
- n-step Q-return
  - $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n})$

### ► n-step Sarsa update $Q(s, a)$ towards the n-step Q-return

- $Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha(G_{t:t+n} - Q_{t+n-1}(S_t, A_t))$

# n-step Control

## n-step Sarsa

### n-step Sarsa for estimating $Q \approx q_*$ or $q_\pi$

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

Initialize  $\pi$  to be  $\varepsilon$ -greedy with respect to  $Q$ , or to a fixed given policy

Algorithm parameters: step size  $\alpha \in (0, 1]$ , small  $\varepsilon > 0$ , a positive integer  $n$

All store and access operations (for  $S_t$ ,  $A_t$ , and  $R_t$ ) can take their index mod  $n + 1$

Loop for each episode:

    Initialize and store  $S_0 \neq$  terminal

    Select and store an action  $A_0 \sim \pi(\cdot | S_0)$

$T \leftarrow \infty$

    Loop for  $t = 0, 1, 2, \dots$  :

        If  $t < T$ , then:

            Take action  $A_t$

            Observe and store the next reward as  $R_{t+1}$  and the next state as  $S_{t+1}$

            If  $S_{t+1}$  is terminal, then:

$T \leftarrow t + 1$

            else:

                Select and store an action  $A_{t+1} \sim \pi(\cdot | S_{t+1})$

$\tau \leftarrow t - n + 1$  ( $\tau$  is the time whose estimate is being updated)

                If  $\tau \geq 0$ :

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$

                    If  $\tau + n < T$ , then  $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$   $(G_{\tau:\tau+n})$

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$

                    If  $\pi$  is being learned, then ensure that  $\pi(\cdot | S_\tau)$  is  $\varepsilon$ -greedy wrt  $Q$

    Until  $\tau = T - 1$

# n-step Control

## n-step Expected Sarsa

- ▶ **Same update as Sarsa except the last element**

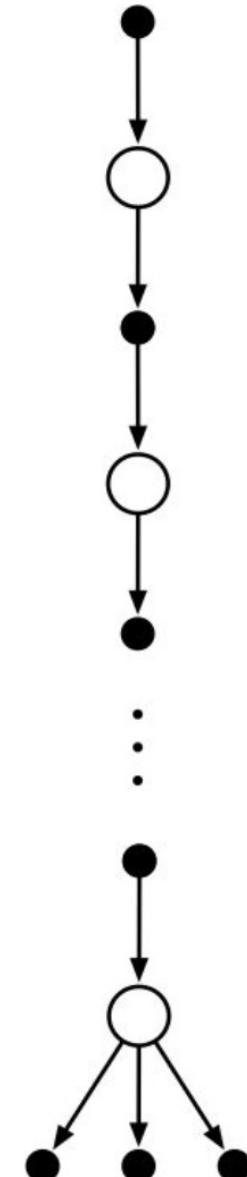
- Consider all possible actions in the last step

- ▶ **Same n-step return as Sarsa except the last step**

- $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n \bar{V}_{t+n-1}(S_{t+n})$
  - $\bar{V}_t(s) = \sum_a \pi(a|s) Q_t(s, a)$

- ▶ **Same update as Sarsa**

- $Q_{t+n}(S_t) \leftarrow Q_{t+n-1}(S_t) + \alpha(G_{t:t+n} - Q_{t+n-1}(S_t))$



# n-step Control

## n-step Off-policy Learning

### ► Require importance sampling

- Importance sampling 为在 Target policy 和 Behavior policy 下的该轨迹发生的相对概率

$$\bullet \quad \rho_t^T = \frac{\prod_{k=t}^T \pi(A_k|S_k) p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^T \mu(A_k|S_k) p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^T \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$$

- Redefine importance sampling

$$\bullet \quad \rho_{t:h} = \prod_{k=t}^{\min(h, T-1)} \frac{\pi(A_k|S_k)}{\mu(A_k|S_k)}$$

### ► Update target policy's values with behavior policy's return

- $V_{t+n}(S_t) \leftarrow V_{t+n-1}(S_t) + \alpha \rho_{t:t+n-1} (G_{t:t+n} - V_{t+n-1}(S_t))$

- On-policy 是 off-policy 的特例

- If  $\pi = \mu$ , then  $\rho = 1$

# n-step Control

## Off-policy n-step Sarsa

- ▶ Update  $Q$  instead of  $V$
- ▶ Importance sampling ratio starts one step later for  $Q$  values
  - $A_t$  is already chosen
- ▶ Off-policy n-step Sarsa
  - $$Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1:t+n} (G_{t:t+n} - Q_{t+n-1}(S_t, A_t))$$

# n-step Control

# Off-policy n-step Sarsa

## Off-policy $n$ -step Sarsa for estimating $Q \approx q_*$ or $q_\pi$

Input: an arbitrary behavior policy  $b$  such that  $b(a|s) > 0$ , for all  $s \in \mathcal{S}, a \in \mathcal{A}$

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

Initialize  $\pi$  to be greedy with respect to  $Q$ , or as a fixed given policy

Algorithm parameters: step size  $\alpha \in (0, 1]$ , a positive integer  $n$

All store and access operations (for  $S_t$ ,  $A_t$ , and  $R_t$ ) can take their index mod  $n + 1$

Loop for each episode:

Initialize and store  $S_0 \neq \text{terminal}$

Select and store an action  $A_0 \sim b(\cdot | S_0)$

$$T \leftarrow \infty$$

Loop for  $t = 0, 1, 2, \dots$ :

If  $t < T$ , then:

Take action  $A_t$

Observe and store the ne

$S_{t+1}$  is term

7

se:  $\text{Gal}(\mathbb{Q}_p/\mathbb{Q}) \cong \text{Gal}(\mathbb{Q}_p/\mathbb{Q}) \times \text{Gal}(\mathbb{Q}_p/\mathbb{Q})$

Select and store an action  $A_{t+1} \sim b(\cdot|S_{t+1})$

$\tau \leftarrow t - \tau$

$$\tau \geq 0: \quad \rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1, T-1)} \frac{\pi(A_i | S_i)}{b(A_i | S_i)} \quad (\rho_{\tau+1:t+n-1})$$

$$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$$

If  $\tau + n < T$ , then:  $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$

$$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha \rho [G - Q(S_\tau, A_\tau)]$$

If  $\pi$  is being learned, then ensure that  $\pi(\cdot|S_\tau)$  is greedy wrt  $Q$

Until  $\tau \equiv T - 1$

# n-step Control

## Off-policy n-step Expected Sarsa

- ▶ **Importance sampling ratio ends one step earlier for Expected Sarsa**

- Off-policy n-step Sarsa
    - $Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1:t+n} (G_{t:t+n} - Q_{t+n-1}(S_t, A_t))$
  - Off-policy n-step Expected Sarsa
    - $Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha \rho_{t+1:t+n-1} (G_{t:t+n} - Q_{t+n-1}(S_t, A_t))$
    - Use expected n-step return
      - $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n \bar{V}_{t+n-1}(S_{t+n})$
      - $\bar{V}_t(s) = \sum_a \pi(a|s) Q_t(s, a)$

# Sarsa( $\lambda$ )

## Forward-View Sarsa( $\lambda$ )

### ► n-step Q-return

- $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n})$

### ► The $\lambda$ Q-return

- $Q_t^\lambda = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}$

### ► Forward-view Sarsa( $\lambda$ )

- $Q_{t+n}(S_t, A_t) \leftarrow Q_{t+n-1}(S_t, A_t) + \alpha(Q_t^\lambda - Q_{t+n-1}(S_t, A_t))$

# Sarsa( $\lambda$ )

## Backward-View Sarsa( $\lambda$ )

### ► Eligibility Traces

- $E_0(s, a) = 0$
- $E_t(s, a) = \gamma \lambda E_{t-1}(s, a) + \mathbf{1}(S_t = s, A_t = a)$

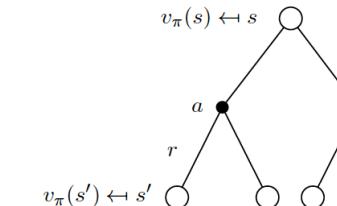
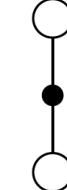
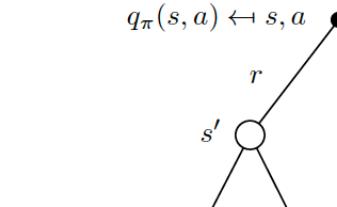
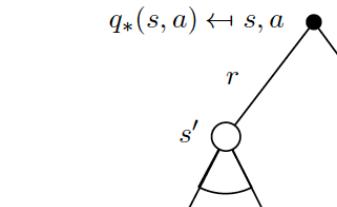
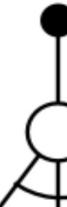
### ► Backward-view Sarsa( $\lambda$ )

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



# Summary

## Relationship Between DP and TD

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

# Summary

## Relationship Between DP and TD

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

# Model Free Control

## Suggested reading

- ▶ **H Van Hassel, Deep reinforcement learning with double q-learning**
- ▶ [http://www-anw.cs.umass.edu/~barto/courses/cs687/importance\\_sampling.pdf](http://www-anw.cs.umass.edu/~barto/courses/cs687/importance_sampling.pdf)