

## **L15. Reinforcement Learning (Monte Carlo Methods)**

## Project

1. Define your own problem. It can be anything
2. Formulate the problem using mathematical expressions
3. Describe what type of data is required for the formulation
  - describe how to obtain data
  - describe how to process data
  - describe how to build models that are required for your formulation
4. Discuss what types of decision making methods can be used to solve the formulated problem
  - list more than two methods and discuss their pros and cons
  - discuss limitations in each method
5. Future Plan

Midterm 25%, Final 25%, Project 25%, HW 25% (5% each)

Format: Report (more than 2pages but less than 5 pages)

Evaluations: 25, 24, 23, 22, 21, 20 (Full score is 25)

Bonus points: up to 5 points



### Markov Decision Process (Offline)

- Have mental model of how the world works
- Find policy to collect the maximum rewards

$$\text{Solve } Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$
$$\text{Find } \pi^*(s) = \max_a Q^*(s, a)$$



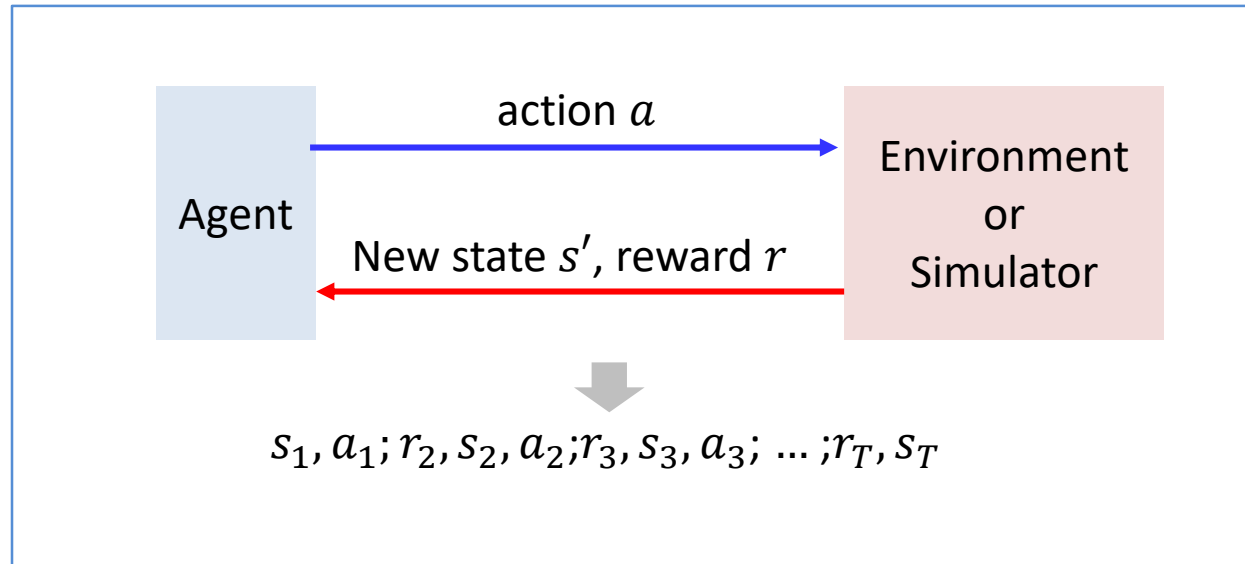
### Reinforcement Learning (Offline & Online)

- Don't know how the world works
- Perform a sequence of actions in the world to maximize the rewards

$$s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T \rightarrow Q^*(s, a) \rightarrow \pi^*(s)$$

- Reinforcement learning is really the way humans work:
  - we go through life, taking various actions, getting feedback.
  - We get rewarded for doing well and learn along the way.

## Reinforcement Learning Template



### Template for Reinforcement Learning

For  $t = 1, 2, 3, \dots$

Choose action  $a_t = \pi(s_t)$  (how?) : Decision making

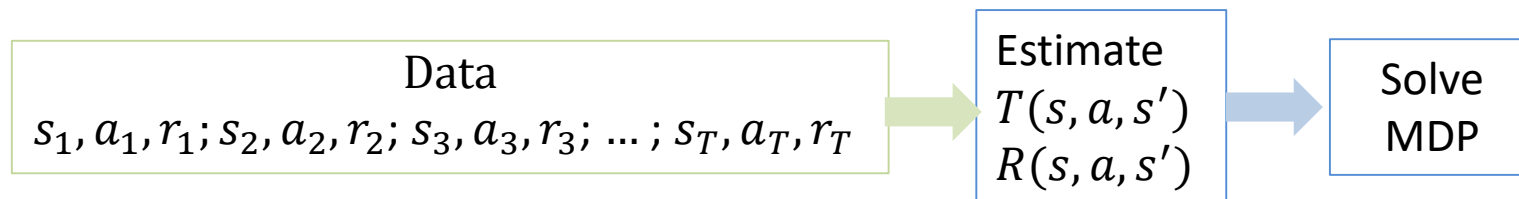
Receive reward  $r_{t+1}$  and observe new state  $s_{t+1}$  (Environment)

Update parameters associated with  $V(t)$ ,  $Q(s, a)$  (how?) : Learning

- **Monte Carlo Method (Sutton & Barto Ch.5)**
  - Model-Based Monte Carlo method
  - Model-free Monte Carlo method
    - Policy Evaluation
    - Policy Improvement
    - Policy Iteration (Monte Carlo control)
      - ✓ On-policy
      - ✓ Off-policy
- **Temporal Difference Learning (Sutton & Barto Ch.6)**
  - SARSA
  - Q-Learning

## Road Map

- Model-Based Reinforcement learning



- Model-FREE Reinforcement learning

How to estimate  $V^*(s)$  and  $Q^*(s, a)$

Monte Carlo method

Temporal Difference methods

How to  
explore ?

		Non-Bootstrap	Bootstrap
How to explore ?	On-policy	On-policy Monte Carlo Control	SARSA
	Off-policy	Off-policy Monte Carlo Control	Q-Learning

- Episodic based

- Single-data-point based

### Key Idea : model-based learning

- Formulate the MDP using the estimated  $T(s, a, s')$  and  $R(s, a, s')$
- Solve the MDP using various solution methods, i.e., DP approach

$$\text{Solve } Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$
$$\text{Find } \pi^*(s) = \max_a Q^*(s, a)$$

- Data following policy  $\pi$

$$s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$$

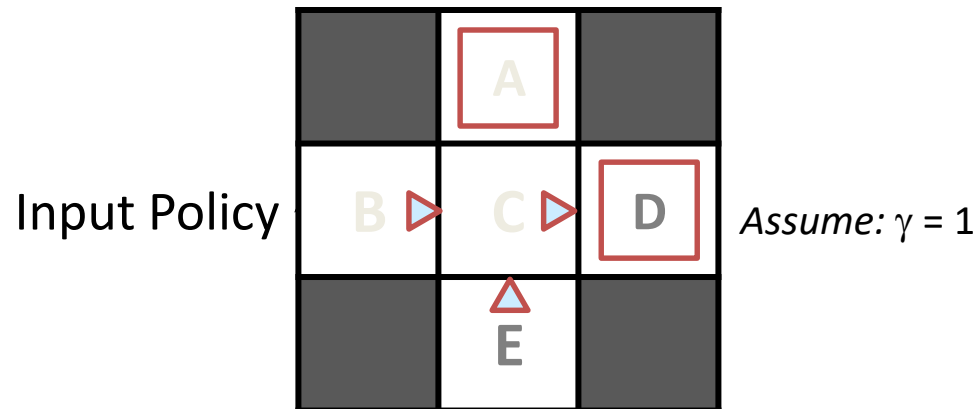
- Transition model

$$\hat{T}(s, a, s') = \frac{\text{\#times } (s, a, s') \text{ occurs}}{\text{\#times } (s, a) \text{ occurs}}$$

- Reward model

$$\hat{R}(s, a, s') = \text{average of } r \text{ in } (s, a, r, s')$$

## Example



- Observed Episodes (Training)

### Episode 1

B, east, C, -1  
C, east, D, -1  
D, exit, x, +10

### Episode 2

B, east, C, -1  
C, east, D, -1  
D, exit, x, +10

### Episode 3

E, north, C, -1  
C, east, D, -1  
D, exit, x, +10

### Episode 4

E, north, C, -1  
C, east, A, -1  
A, exit, x, -10

- Learned Model

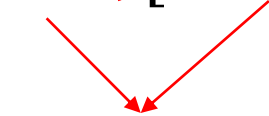
$$T(s, a, s')$$

$T(B, \text{east}, C) = 1.00$   
 $T(C, \text{east}, D) = 0.75$   
 $T(C, \text{east}, A) = 0.25$

$$R(s, a, s')$$

$R(B, \text{east}, C) = -1$   
 $R(C, \text{east}, D) = -1$   
 $R(D, \text{exit}, x) = +10$



$$Q^*(s, a) = \sum_{s'} \hat{T}(s, a, s') [\hat{R}(s, a, s') + \gamma V^*(s')]$$


Estimation

Why not estimating  $Q^*(s, a)$  directly  
instead of separately estimating  $\hat{T}(s, a, s')$  and  $\hat{R}(s, a, s')$  ?

$$\pi^*(s) = \operatorname{argmax}_a Q^*(s, a)$$

## Model-Free Monte Carlo Based Methods

How to estimate  $V^*(s)$  and  $Q^*(s, a)$

Monte Carlo method

Temporal Difference methods

How to  
explore ?

		Non-Bootstrap	Bootstrap
How to explore ?	On-policy	On-policy Monte Carlo Control	SARSA
	Off-policy	Off-policy Monte Carlo Control	Q-Learning

- Episodic based

- Single-data-point based

## Model-Free Monte Carlo Based Methods

- Monte Carlo methods require only experience-sample sequences of states, actions, and rewards from on-line or simulated interaction with an environment
- Monte Carlo methods are ways of solving the reinforcement learning problem based on **averaging sample returns (Monte Carlo)**

$$R(s, a) = \text{average of } r \text{ in } (s, a, r)$$

- Monte Carlo methods are incremental in an **episode-by-episode sense**, but not in a step-by-step sense  
→ after a final state has reached, reward appears, e.g., Go

### Dynamic Programming

- Policy Evaluation
- Policy Iteration
- Policy Improvement

Key ideas

### Monte Carlo Methods

- Policy Evaluation
- Policy Iteration
- Policy Improvement

### Key Idea : Monte Carlo Policy Evaluation

- Learn the state-value function  $V^\pi(s)$  for a given policy  $\pi$
- The value of state is the expected utility - **expected accumulative future reward** starting from  $s$  and following the policy  $\pi$

Data (following policy  $\pi$ ):

$$s_t \in \mathcal{S} = \{s^1, s^2, s^3\}, a_t \in \mathcal{A} = \{a^1, a^2, a^3\}$$

Episode 1:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

Episode 2:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

Episode 3:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

$\vdots$

Utility  $u_t$ :

$$u_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

$$u_t = r_t + r_{t+2} + \dots + r_T$$

Estimate  $V^\pi(s)$  or  $Q^\pi(s, a)$  :

$$V^\pi(s) = \text{average of } u_t \text{ where } s_t = s \text{ and following } \pi$$

Because the value being computed is dependent on the policy used to generate the data, we call this an **on-policy** algorithm.

### Key Idea : Monte Carlo Policy Evaluation

- Learn the state-value function  $V^\pi(s)$  for a given policy  $\pi$
- The value of state is the expected utility - **expected accumulative future reward** starting from  $s$  and following the policy  $\pi$

$$s_t \in \mathcal{S} = \{s^1, s^2, s^3\}, a_t \in \mathcal{A} = \{a^1, a^2, a^3\}$$

(State, Action, Reward ) pairs generated by policy  $\pi$

Episode 1: ( $s^1, a^2, 1$ ); ( $s^3, a^1, 5$ ); ( $s^2, a^3, 3$ ), ( $s^1, a^3, 10$ ), ( $s^2, a^2, 2$ )

Episode 2: ( $s^3, a^1, 5$ ); ( $s^2, a^2, 2$ ); ( $s^1, a^2, 1$ ); ( $s^2, a^3, 3$ ), ( $s^1, a^3, 10$ )

Episode 3: ( $s^2, a^3, 3$ ); ( $s^1, a^2, 1$ ); ( $s^3, a^1, 5$ ); ( $s^1, a^3, 10$ ), ( $s^2, a^2, 2$ )

( $\gamma = 1$ )

Episode 1:  $V^\pi(s^1) = 1 + 5 + 3 + 10 + 2 = 21$

Episode 2:  $V^\pi(s^1) = 1 + 3 + 10 = 14$

Episode 3:  $V^\pi(s^1) = 1 + 5 + 10 + 2 = 19$

First visit to  $s$

$V^\pi(s^1) =$  average of accumulated reward over all episodes

$$= \frac{21+14+19}{3} = 14.6$$

### Key Idea : Monte Carlo Policy Evaluation

- Learn the state-value function  $V^\pi(s)$  for a given policy  $\pi$
- The value of state is the expected utility - **expected accumulative future reward** starting from  $s$  and following the policy  $\pi$

$$s_t \in \mathcal{S} = \{s^1, s^2, s^3\}, a_t \in \mathcal{A} = \{a^1, a^2, a^3\}$$

(State, Action, Reward ) pairs generated by policy  $\pi$

Episode 1: ( $s^1, a^2, 1$ ); ( $s^3, a^1, 5$ ); ( $s^2, a^3, 3$ ), ( $s^1, a^3, 10$ ), ( $s^2, a^2, 2$ )

Episode 2: ( $s^3, a^1, 5$ ); ( $s^2, a^2, 2$ ); ( $s^1, a^2, 1$ ); ( $s^2, a^3, 3$ ), ( $s^1, a^3, 10$ )

Episode 3: ( $s^2, a^3, 3$ ); ( $s^1, a^2, 1$ ); ( $s^3, a^1, 5$ ); ( $s^1, a^3, 10$ ), ( $s^2, a^2, 2$ )

( $\gamma = 1$ )

Episode 1:  $V^\pi(s^1) = 1 + 5 + 3 + 10 + 2 = 21$ ;  $V^\pi(s^1) = 10 + 2 = 12$

Episode 2:  $V^\pi(s^1) = 1 + 3 + 10 = 14$ ;  $V^\pi(s^1) = 10 = 10$

Episode 3:  $V^\pi(s^1) = 1 + 5 + 10 + 2 = 19$ ;  $V^\pi(s^1) = 10 + 2 = 12$

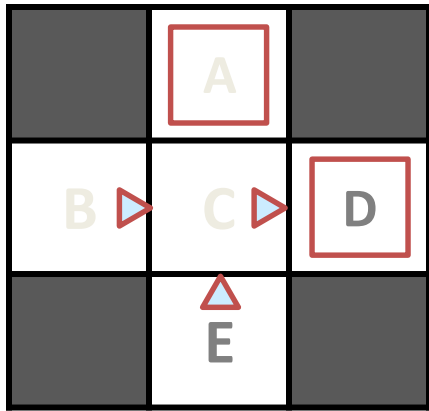
Every visit to  $s$

$V^\pi(s^1)$  = average of accumulated reward over all episodes

$$= \frac{21+12+14+10+19+12}{6} = 15$$

## Example

Input Policy  $\pi$



Assume:  $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1  
C, east, D, -1  
D, exit, x, +10

Episode 2

B, east, C, -1  
C, east, D, -1  
D, exit, x, +10

Episode 3

E, north, C, -1  
C, east, D, -1  
D, exit, x, +10

Episode 4

E, north, C, -1  
C, east, A, -1  
A, exit, x, -10

Output Values

	-10	
+8	+4	+10
	-2	

### Example

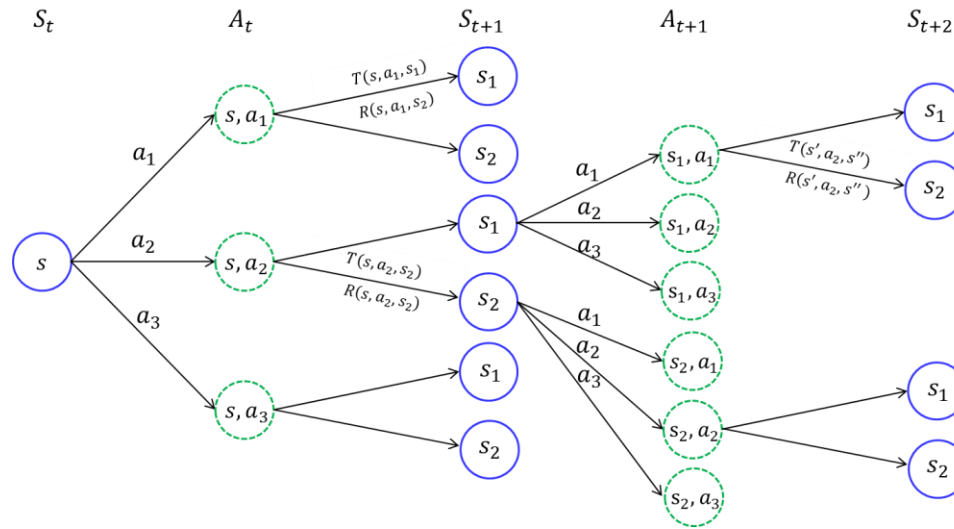
- What's good about direct evaluation?
  - It's easy to understand
  - It doesn't require any knowledge of  $T$ ,  $R$
  - It eventually computes the correct average values, using just sample transitions
- What bad about it?
  - It wastes information about state connections
  - Each state must be learned separately
  - So, it takes a long time to learn

### Output Values

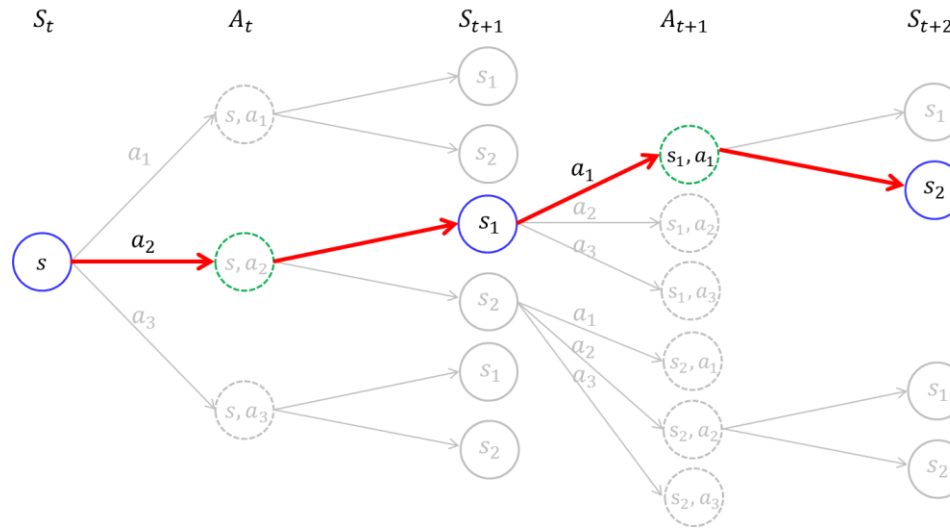
	-10 A	
+8 B	+4 C	+10 D
	-2 E	



# Monte Carlo Policy Evaluation



## Monte Carlo Policy Evaluation



- Estimates for each state are independent
  - ✓ The estimate for one state does not build upon the estimate of any other state (No bootstrap)
- The computational expense of estimating the value of a single state is independent of the number of states
  - ✓ Attractive when one requires the value of only a subset of the states

## Monte Carlo Estimation of Action Values

- Without a model, state value function  $V^\pi(s)$  is not enough
  - ✓ We need  $T(s, a, s')$  and  $R(s, a, s')$  to compute the optimum policy:

$$Q^*(s, a) = \sum_{s'} T(s, a, s') \{R(s, a, s') + \gamma V^*(s')\}$$

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

- One of primary goals for Monte Carlo method is to estimate  $Q^*(s, a)$

## Monte Carlo Estimation of Action Values

### Key Idea : Monte Carlo Policy Evaluation

- Learn the action-value function  $Q^\pi(s, a)$  for a given policy  $\pi$
- The value of action-state is the expected utility - **expected accumulative future reward** starting from  $s$  and following the policy  $\pi$

Data (following policy  $\pi$ ):

$$s_t \in \mathcal{S} = \{s^1, s^2, s^3\}, a_t \in \mathcal{A} = \{a^1, a^2, a^3\}$$

Episode 1:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

Episode 2:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

Episode 3:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

$\vdots$

Utility  $u_t$ :

$$u_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots$$

$$u_t = r_t + r_{t+2} + \dots + r_T$$

Estimate  $Q^\pi(s, a)$  :

$$Q^\pi(s, a) = \text{average of } u_t \text{ where } s_t = s, a_t = a \text{ and following } \pi$$

### Key Idea : Monte Carlo Policy Evaluation

- Learn the state-value function  $Q^\pi(s, a)$  for a given policy  $\pi$
- The value of action-state is the expected utility - **expected accumulative future reward** starting from  $s$  and following the policy  $\pi$

$$s_t \in \mathcal{S} = \{s^1, s^2, s^3\}, a_t \in \mathcal{A} = \{a^1, a^2, a^3\}$$

(State, Action, Reward ) pairs generated by policy  $\pi$

Episode 1: ( $s^1, a^2, 1$ ); ( $s^3, a^1, 5$ ); ( $s^2, a^3, 3$ ), ( $s^1, a^3, 10$ ), ( $s^2, a^2, 2$ )

Episode 2: ( $s^1, a^1, 5$ ); ( $s^2, a^2, 2$ ); ( $s^1, a^2, 1$ ); ( $s^2, a^3, 3$ ), ( $s^1, a^3, 10$ )

Episode 3: ( $s^2, a^3, 3$ ); ( $s^1, a^2, 1$ ); ( $s^3, a^1, 5$ ); ( $s^1, a^3, 10$ ), ( $s^2, a^2, 2$ )

( $\gamma = 1$ )

Episode 1:  $Q^\pi(s^1, a^2) = 1 + 5 + 3 + 10 + 2 = 21$

Episode 2:  $Q^\pi(s^1, a^2) = 1 + 3 + 10 = 14$

Episode 3:  $Q^\pi(s^1, a^2) = 1 + 5 + 10 + 2 = 19$

First visit to  $s$

$Q^\pi(s^1, a^2) =$  average of accumulated reward over all episodes

$$= \frac{21+14+19}{3} = 14.6$$

## Incremental Formulation

$s_1, a_1, r_1; s_2 = s, a_2 = a, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$   
 $s_1 = s, a_1 = a, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$   
 $s_1, a_1, r_1; s_2, a_2, r_2; s_3 = s, a_3 = a, r_3; \dots; s_T, a_T, r_T$   
 $\vdots$   
 $s_1, a_1, r_1; s_2 = s, a_2 = a, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

Episode 1

Episode 2

Episode 3

Episode  $\infty$

For each episode,

We can get state and action pair

$(s, a)$  and the following accumulated  
reward (utility)  $u$ :  $(s, a) \rightarrow u$

### Incremental formulation (convex combination):

On each  $(s, a, u)$  for a single episode:

$$Q^\pi(s, a) \leftarrow (1 - \eta) \underbrace{Q^\pi(s, a)}_{\text{Current estimate}} + \eta \underbrace{u}_{\text{New data}}$$

where  $\eta = \frac{1}{1 + (\text{\#updates to } (s, a))}$

Current  
estimate

New data

### Stochastic gradient form :

On each  $(s, a, u)$  for a single episode:

$$Q^\pi(s, a) \leftarrow Q^\pi(s, a) - \eta(Q^\pi(s, a) - u)$$

The incremental formulations can be thought of as a way to solve the following least square estimation in an online manner

$$\min_{Q^\pi} \sum_{(s, a, u)} (Q^\pi(s, a) - u)^2$$

## Issue of computing $Q^\pi(s, a)$

- If  $\pi$  is a deterministic policy, many relevant state –action pairs may never be visited
  - ✓ then in following  $\pi$ , one will observe returns only for one of the actions from each state, i.e., we won't even see  $(s, a)$  if a  $a \neq \pi(s)$
  - ✓ To improve the policy  $\pi$ , we need to **compare** the state-action values of all the possible actions,  $Q^\pi(s, a_1), Q^\pi(s, a_2), \dots$
- To estimate  $Q^\pi(s, a)$  reliably, we need a large number of episodes
  - ✓ *Infinite* number of episodes are required for accurate estimations

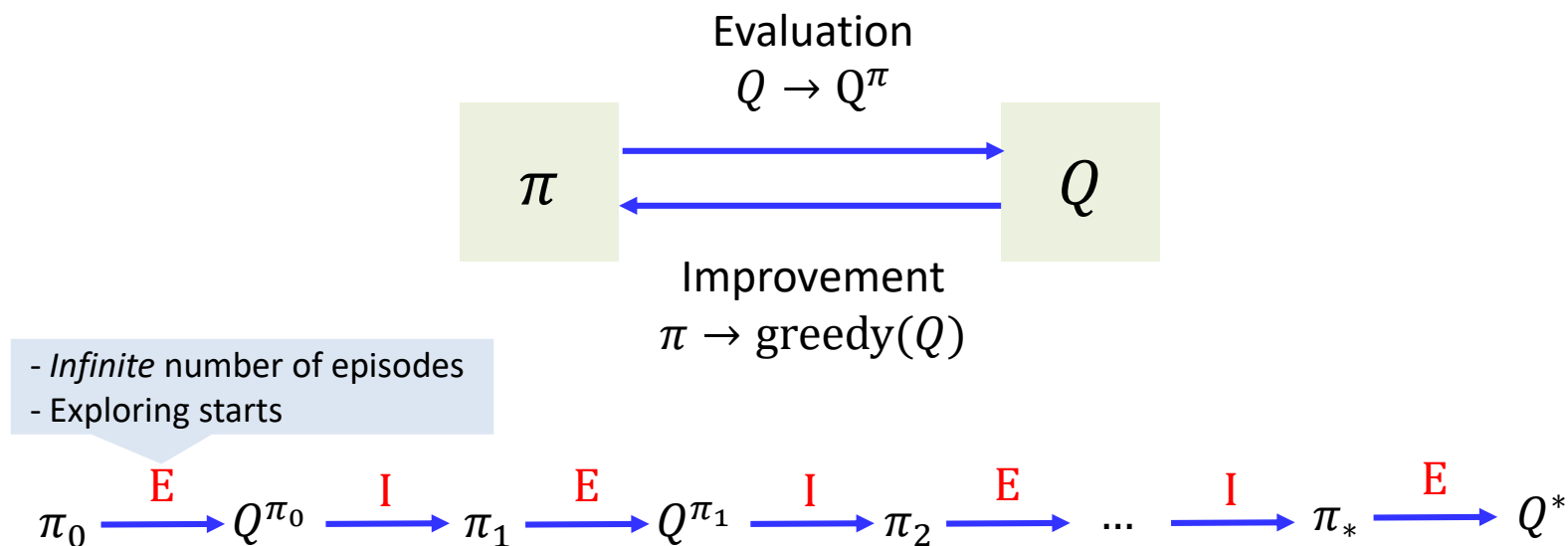
### How to resolve?

- **Exploring starts + infinite number of episodes**
  - ✓ Start from an arbitrary state-action pair  $(s, a)$  with a non-zero probability
  - ✓ Guarantees **that all state-action pair will be visited an infinite number of times in the limit of an infinite number of episodes**
  - ✓ Cannot be used when learning directly from experience
- **Stochastic Policy + infinite number of episodes**
  - ✓ Policy with **a nonzero probability of selecting all actions**
  - ✓ Guarantees that all state-action pair will be visited an infinite number of times in the limit of an infinite number of episodes

We need an efficient exploration strategy!

### Key Idea : Monte Carlo Control

- Idea of generalized policy iteration (GPI)
- Monte Carlo Policy Evaluation + Policy improvement



- **Policy Evaluation**
  - ✓ The value function is repeatedly altered to more closely approximate the value function for the current policy  $\pi$
- **Policy Improvement**
  - ✓ The policy is repeatedly improved with respect to the current action value function  $Q$



## Monte Carlo Control

$$\pi_0 \xrightarrow{E} Q^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} Q^{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} Q^*$$

### E Policy evaluation:

- The value function is repeatedly altered to more closely approximate the value function for the current policy  $\pi$
  - Infinite number of episodes under policy  $\pi$
  - The episodes are generated with exploring starts
- } Converge to exact  $Q^\pi(s, a)$

exploring starts	$S_1, a_1, r_1; S_2, a_2, r_2; S_3, a_3, r_3; \dots; S_T, a_T, r_T \rightarrow u$	Episode 1
	$S_1, a_1, r_1; S_2, a_2, r_2; S_3, a_3, r_3; \dots; S_T, a_T, r_T \rightarrow u$	Episode 2
	$S_1, a_1, r_1; S_2, a_2, r_2; S_3, a_3, r_3; \dots; S_T, a_T, r_T \rightarrow u$	Episode 3
	$S_1, a_1, r_1; S_2, a_2, r_2; S_3, \ddots, r_3; \dots; S_T, a_T, r_T \rightarrow u$	Episode $\infty$

Average expected reward

### I Policy Improvement:

- Policy improvement can be done by constructing each  $\pi_{k+1}$  as the greedy policy with respect to  $Q^{\pi_k}$

$$\pi_{k+1}(s) = \operatorname{argmax}_a Q^{\pi_k}(s, a)$$

$$\begin{aligned} Q^{\pi_k}(s, \pi_{k+1}(s)) &= Q^{\pi_k}\left(s, \operatorname{argmax}_a Q^{\pi_k}(s, a)\right) \\ &= \max_a Q^{\pi_k}(s, a) \\ &\geq Q^{\pi_k}(s, \pi_k(s)) \\ &= V^{\pi_k}(s) \end{aligned}$$

$V^{\pi_{k+1}}(s) \geq V^{\pi_k}(s)$  for all  $s$

(Policy improvement theorem)

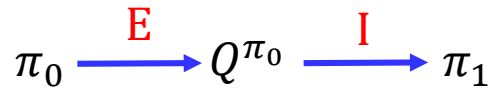
Overall process converges to an optimal policy and the optimal value function

## Make Monte Carlo Control Practical

### Assumptions:

- *Infinite* number of episodes
- Exploring starts

➡ **Relax** these two assumptions



- **Relaxing “Infinite number of episodes”**
  - ✓ Policy improvement with an early stop (Generalized policy improvement concept)
  - ✓ The “in place” version of value iteration
- Relaxing “exploring starts”
  - ✓ Discussed later

### Algorithm : Monte Carlo ES Control

Initialize, for all  $s \in S, a \in A(s)$

$Q(s, a) \leftarrow$  arbitrary

$\pi(s) \leftarrow$  arbitrary

$U(s, a) \leftarrow$  empty list

Repeat forever:

(a) Generate *an episode* using *exploring starts* and  $\pi$

(b) For each pair  $(s, a)$  appearing in the episode:

$U \leftarrow$  utility following the first occurrence of  $s, a$

Append  $U$  to  $U(s, a)$

$Q(s, a) \leftarrow \text{average}(U(s, a))$

} Policy evaluation

(c) For each  $s$  in the episode:

$\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} Q(s, a)$

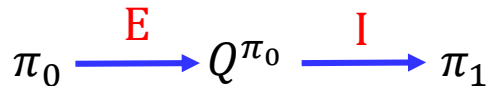
} Policy improvement

In an single episode, both Policy evaluation and policy improvement proceeds together

### Assumptions:

- *Infinite* number of episodes
- Exploring starts

➡ **Relax** these two assumptions



- Relaxing “Infinite number of episodes”
  - ✓ Policy improvement with an early stop (Generalized policy improvement concept)
  - ✓ The “in place” version of value iteration
- **Relaxing “exploring starts”**
  - ✓ The only general way to ensure that all actions are selected infinitely often is for the agent to continue to select them
    - ❖ On-policy methods
    - ❖ Off-policy methods

## On policy algorithm



- On-policy methods attempt to evaluate or improve the policy that is used to make decisions (generate data)
  - ✓ Policy is soft:  $\pi(s, a) > 0$  for all  $s \in S$  and  $a \in A(s)$

## Off policy algorithm



- Behavioral policy is used to generate behavior
- Estimation policy is evaluated and improved
  - ✓ Estimation policy may be deterministic, while the behavior policy can continue to sample all possible actions

## On-policy Monte Carlo Control

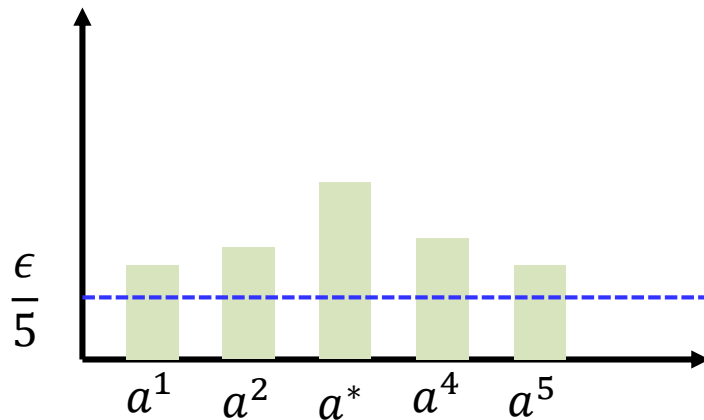
### Key Idea : On-policy methods: Soft policies

- attempt to evaluate or improve the policy that is used to make decisions
- $\pi(s, a) > 0$  for all  $s \in S$  and  $a \in A(s)$

#### $\epsilon$ – soft policy

$$\pi(s, a) \geq \frac{\epsilon}{|A(s)|} \text{ for all } a$$

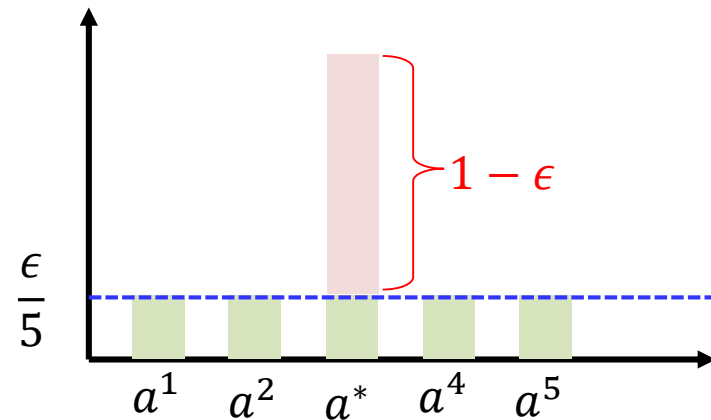
$$\sum_a \pi(s, a) = 1$$



#### $\epsilon$ – greedy policy

$$\pi(s, a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A(s)|} & \text{If } a = a^* \\ \frac{\epsilon}{|A(s)|} & \text{If } a \neq a^* \end{cases}$$

$$\text{Total probability} = \left(1 - \epsilon + \frac{\epsilon}{|A(s)|}\right) 1 + \frac{\epsilon}{|A(s)|} (|A(s)| - 1) = 1$$



## On-policy Monte Carlo Control

Any  $\epsilon$  – *greedy* policy  $\pi'$  respect to  $Q^\pi$  is an improvement over any  $\epsilon$  – *soft* policy  $\pi$

Let  $\pi'$  be the  $\epsilon$  – *greedy* policy,

$$\begin{aligned} Q^\pi(s, \pi'(s, a)) &= \sum_a \pi'(s, a) Q^\pi(s, a) \\ &= \frac{\epsilon}{|A(s)|} \sum_a Q^\pi(s, a) + (1 - \epsilon) \max_a Q^\pi(s, a) \\ &\geq \frac{\epsilon}{|A(s)|} \sum_a Q^\pi(s, a) + (1 - \epsilon) \sum_a \frac{\pi(s, a) - \frac{\epsilon}{|A(s)|}}{1 - \epsilon} Q^\pi(s, a) \\ &= \frac{\epsilon}{|A(s)|} \sum_a Q^\pi(s, a) - \frac{\epsilon}{|A(s)|} \sum_a Q^\pi(s, a) + \sum_a \pi(s, a) Q^\pi(s, a) \\ &= V^\pi(s) \end{aligned}$$

### Recall Policy improvement Theorem

Policy improvement must give us a strictly better policy  $\pi'(s)$  than the older policy  $\pi(s)$  except when the original policy is already optimal  $\pi(s, a) = \pi^*(s, a)$

$$Q^\pi(s, \pi'(s, a)) \geq V^\pi(s) \rightarrow V^{\pi'}(s) \geq V^\pi(s)$$

Thus, by the policy improvement theorem,  $\pi' \geq \pi$  (i.e.,  $V^{\pi'}(s) \geq V^\pi(s)$  for all  $s \in S$ )

### Algorithm : $\epsilon$ – *soft* On-Policy Monte Carlo Control

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

$Q(s, a) \leftarrow$  arbitrary

$\pi \leftarrow$  an arbitrary  $\epsilon$  – *soft policy*

$U(s, a) \leftarrow$  empty list

Repeat forever:

(a) Generate *an episode* using  $\pi$

(b) For each pair  $(s, a)$  appearing in the episode:

$U \leftarrow$  utility following the first occurrence of  $s, a$

Append  $U$  to  $U(s, a)$

$Q(s, a) \leftarrow \text{average}(U(s, a))$

} Policy evaluation

(c) For each  $s$  in the episode:

$a^* \leftarrow \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} Q(s, a)$

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

$$\pi(s, a) = \begin{cases} 1 - \epsilon + \epsilon/|A(s)| & \text{If } a = a^* \\ \epsilon/|A(s)| & \text{If } a \neq a^* \end{cases}$$

} Policy improvement



## Monte Carlo control algorithm

Black Jack Example

## Model-Free Monte Carlo Based Methods

How to estimate  $V^*(s)$  and  $Q^*(s, a)$

Monte Carlo method

Temporal Difference methods

How to  
explore ?

		Non-Bootstrap	Bootstrap
How to explore ?	On-policy	On-policy Monte Carlo Control	SARSA
	Off-policy	Off-policy Monte Carlo Control	Q-Learning

- Episodic based

- Single-data-point based

## Off-policy Monte Carlo Control

### Evaluate One Policy While Following Another

Episodes generated by  $\pi'$ :

$$s_t \in \mathcal{S} = \{s^1, s^2, s^3\}, a_t \in \mathcal{A} = \{a^1, a^2, a^3\}$$

Episode 1:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

Episode 2:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

Episode 3:  $s_1, a_1, r_1; s_2, a_2, r_2; s_3, a_3, r_3; \dots; s_T, a_T, r_T$

$\vdots$

Estimate  $Q^\pi(s, a)$ :

$Q^\pi(s, a) = \text{average of } u_t \text{ where } s_t = s, a_t = a \text{ and following } \pi$

Is it possible?

Yes, if  $\pi'(s, a) > 0 \rightarrow \pi(s, a) > 0$ :

In order to use episodes from  $\pi'$  to estimate values for  $\pi$ , we require that every action taken under  $\pi$  is also taken, at least occasionally, under  $\pi'$

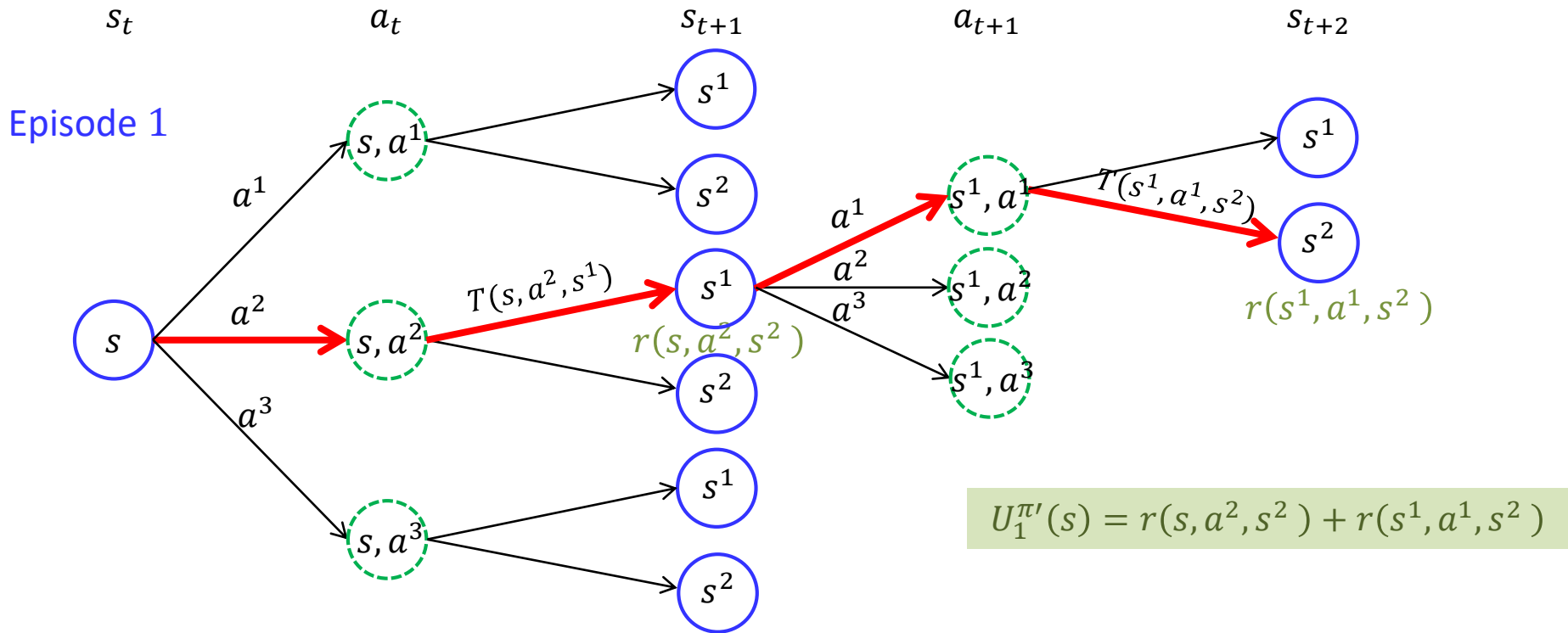
### Key Idea : Off-policy algorithm

- Follows the behavior policy while learning about and improving the estimation policy
- *Behavior* policy  $\pi'(s, a)$ 
  - ✓ The policy used to generate behavior
  - ✓ Requires that the behavior policy have a nonzero probability of selecting all actions that might be selected by the estimation policy (e.g.,  $\epsilon$  – *soft* policy )
- *Estimation* policy  $\pi(s, a)$ 
  - ✓ The policy that is evaluated and improved
  - ✓  $\pi$  can be deterministic
  - ✓  $\pi$  can be the greedy policy with respect to  $Q$  (an estimation  $Q^\pi$ )

### Disadvantages

→ Learning can be slow

# Off-policy Monte Carlo Control :Evaluate One Policy While Following Another



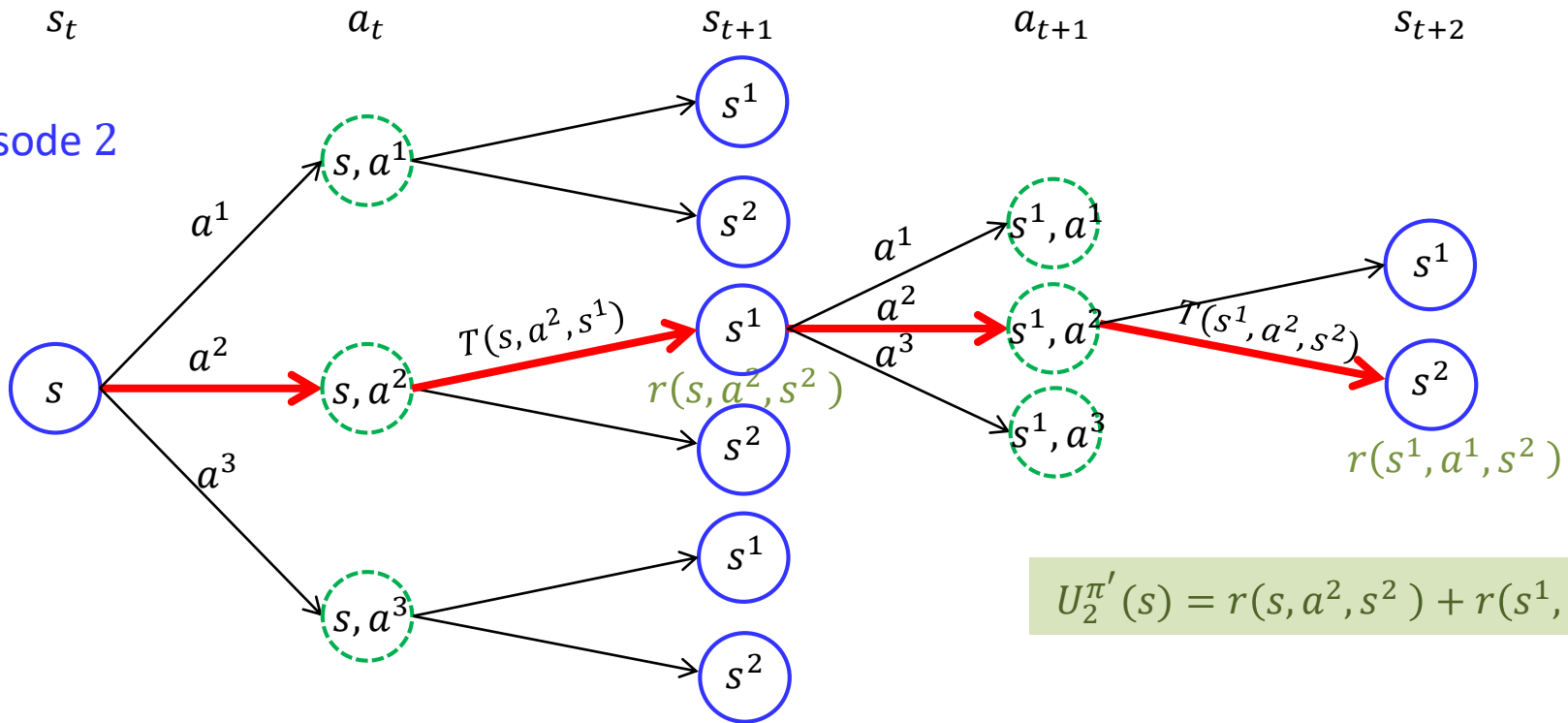
$$p_1(s_t = s) = \prod_{k=t}^{T_1(s)-1} \pi(s_k, a_k) T(s_k, a_k, s_{k+1}) = \pi(s, a^2) T(s, a^2, s^1) \pi(s^1, a^1) T(s^1, a^1, s^2)$$

$$p'_1(s_t = s) = \prod_{k=t}^{T_1(s)-1} \pi'(s_k, a_k) T(s_k, a_k, s_{k+1}) = \pi'(s, a^2) T(s, a^2, s^1) \pi'(s^1, a^1) T(s^1, a^1, s^2)$$

$$\frac{p_1(s)}{p'_1(s)} = \prod_{k=t}^{T_1(s)-1} \frac{\pi(s_k, a_k) T(s_k, a_k, s_{k+1})}{\pi'(s_k, a_k) T(s_k, a_k, s_{k+1})} = \prod_{k=t}^{T_1(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)} = \frac{\pi(s, a^2) \pi(s^1, a^1)}{\pi'(s, a^2) \pi'(s^1, a^1)}$$

# Off-policy Monte Carlo Control :Evaluate One Policy While Following Another

Episode 2

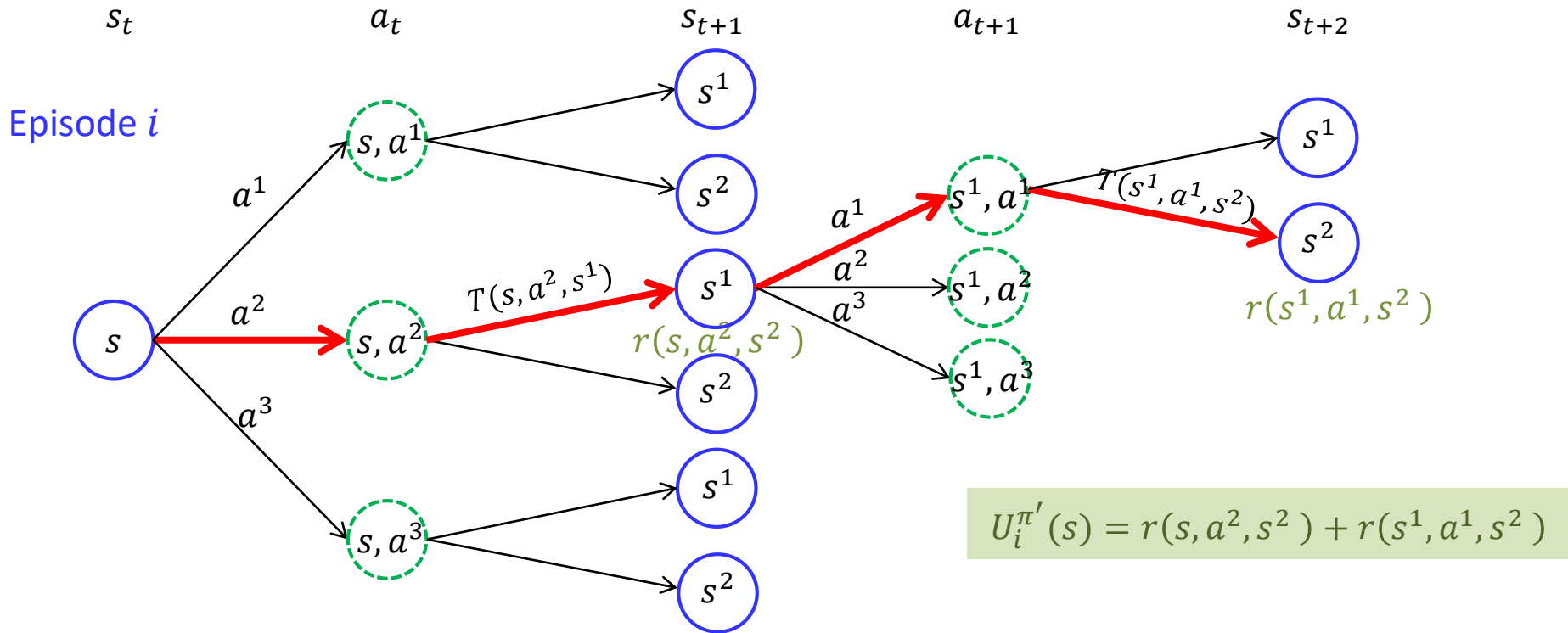


$$p_2(s_t = s) = \prod_{k=t}^{T_2(s)-1} \pi(s_k, a_k) T(s_k, a_k, s_{k+1}) = \pi(s, a^2) T(s, a^2, s^1) \pi(s^1, a^2) T(s^1, a^2, s^2)$$

$$p'_2(s_t = s) = \prod_{k=t}^{T_2(s)-1} \pi'(s_k, a_k) T(s_k, a_k, s_{k+1}) = \pi'(s, a^2) T(s, a^2, s^1) \pi'(s^1, a^2) T(s^1, a^2, s^2)$$

$$\frac{p_2(s)}{p'_2(s)} = \prod_{k=t}^{T_2(s)-1} \frac{\pi(s_k, a_k) T(s_k, a_k, s_{k+1})}{\pi'(s_k, a_k) T(s_k, a_k, s_{k+1})} = \prod_{k=t}^{T_2(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)} = \frac{\pi(s, a^2) \pi(s^1, a^2)}{\pi'(s, a^2) \pi'(s^1, a^2)}$$

# Off-policy Monte Carlo Control :Evaluate One Policy While Following Another



$$p_i(s_t = s) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) T(s_k, a_k, s_{k+1}) = \pi(s, a^2) T(s, a^2, s^1) \pi(s^1, a^1) T(s^1, a^1, s^2)$$

$$p'_i(s_t = s) = \prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) T(s_k, a_k, s_{k+1}) = \pi'(s, a^2) T(s, a^2, s^1) \pi'(s^1, a^1) T(s^1, a^1, s^2)$$

$$\frac{p_i(s)}{p'_i(s)} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k) T(s_k, a_k, s_{k+1})}{\pi'(s_k, a_k) T(s_k, a_k, s_{k+1})} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)} = \frac{\pi(s, a^2) \pi(s^1, a^1)}{\pi'(s, a^2) \pi'(s^1, a^1)}$$

## Off-policy Monte Carlo Control :Evaluate One Policy While Following Another

Compute the value function  $V^\pi(s)$  using the data generated by  $\pi'$

For  $i = 1:n_s$  ( $n_s$  episodes) with  $s_t = s$

$$\left\{ \begin{array}{l} p_i(s_t = s) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) T(s_k, a_k, s_{k+1}) \\ p'_i(s_t = s) = \prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) T(s_k, a_k, s_{k+1}) \end{array} \right. \quad \frac{p_i(s_t)}{p'_i(s_t)} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)} \quad U_i^{\pi'}(s)$$

The desired Monte Carlo estimate after observing  $n_s$  utilities from state  $s_t = s$  is then

$$V^\pi(s) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} U_i^{\pi'}(s)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}}$$

To compute the weights  $\frac{p_i(s)}{p'_i(s)}$  and estimate the value, we only need two policies  $\pi(s, a)$  and  $\pi'(s, a)$



## Off-policy Monte Carlo Control :Evaluate One Policy While Following Another

Compute the value function  $Q^\pi(s, a)$  using the data generated by  $\pi'$

For  $i = 1:n_s$  ( $n_s$  episodes) with  $s_t = s$

$$\left\{ \begin{array}{l} p_i(s_t = s, a_t = a) = \prod_{k=t}^{T_i(s)-1} \pi(s_k, a_k) T(s_k, a_k, s_{k+1}) \\ p'_i(s_t = s, a_t = a) = \prod_{k=t}^{T_i(s)-1} \pi'(s_k, a_k) T(s_k, a_k, s_{k+1}) \end{array} \right.$$

$$\frac{p_i(s_t)}{p'_i(s_t)} = \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)} \quad U_i^{\pi'}(s, a)$$

The desired Monte Carlo estimate after observing  $n_s$  utilities from state  $s_t = s$  is then

$$Q^\pi(s, a) = \frac{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)} U_i^{\pi'}(s, a)}{\sum_{i=1}^{n_s} \frac{p_i(s)}{p'_i(s)}} = \frac{\sum_{i=1}^{n_s} \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)} U_i^{\pi'}(s, a)}{\sum_{i=1}^{n_s} \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}}$$

To compute the weights  $\frac{p_i(s)}{p'_i(s)}$  and estimate the value, we only need two policies  $\pi(s, a)$  and  $\pi'(s, a)$

### Key Idea : Off-policy algorithm

- Follows the behavior policy while learning about and improving the estimation policy
- *Behavior* policy  $\pi'(s, a)$ 
  - ✓ The policy used to generate behavior
  - ✓ Requires that the behavior policy have a nonzero probability of selecting all actions that might be selected by the estimation policy (e.g.,  $\epsilon$  – *soft* policy )
- *Estimation* policy  $\pi(s, a)$ 
  - ✓ The policy that is evaluated and improved
  - ✓  $\pi$  can be deterministic
  - ✓  $\pi$  can be the greedy policy with respect to  $Q$  (an estimation  $Q^\pi$ )

### Disadvantages

→ Learning can be slow

### Algorithm : An off-policy Monte Carlo Control

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

$Q(s, a) \leftarrow$  arbitrary

$N(s, a) \leftarrow 0$

$D(s, a) \leftarrow 0$

$\pi \leftarrow$  arbitrary deterministic policy

Repeat forever:

(a) Select a policy  $\pi'$  and use it to generate an episode:

$s_0, a_0, r_1, s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T, s_T$

(b) Find  $\tau \leftarrow$  the latest time at which  $a_\tau \neq \pi(s_\tau)$

(c) For each pair  $(s, a)$  appearing in the episodes at time  $\tau$  or later

$t \leftarrow$  the time of first occurrence of  $(s, a)$  such that  $t \geq \tau$

$$w \leftarrow \prod_{k=t+1}^{T-1} \frac{1}{\pi'(s_k, a_k)}$$


$$Q(s, a) \leftarrow Q(s, a) + wU_t$$

$$D(s, a) \leftarrow D(s, a) + w$$

$$Q(s, a) \leftarrow \frac{N(s, a)}{D(s, a)}$$

(c) For each  $s$  in the episode:

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


$$Q^\pi(s, a) = \frac{\sum_{i=1}^{n_s} \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)} U_i^{\pi'}(s, a)}{\sum_{i=1}^{n_s} \prod_{k=t}^{T_i(s)-1} \frac{\pi(s_k, a_k)}{\pi'(s_k, a_k)}}$$

## Summary

- The Monte Carlo methods learn value functions and optimal policies from experience in the form of sample episodes
- Advantages are:
  - ✓ Can be used to learn optimal behavior directly from interaction with the environment with no models of environment dynamics
  - ✓ Can be used with simulation or sample models
  - ✓ It is efficient to focus Monte Carlo methods on a small subset of the states
  - ✓ Less harmed by violations of the Markov property. This is because they do not update their value estimates on the basis of the value estimates of successor states (do not use bootstrap)