



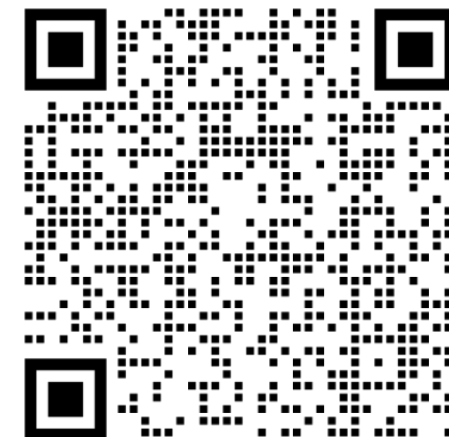
复旦大学大数据学院
School of Data Science, Fudan University

魏忠钰

Reinforcement Learning

Data Intelligence and Social Computing Lab (DISC)

November 16th, 2021



Outline

- Reinforcement Learning

Markov Decision Processes (MDPs)

- S is a (finite) set of Markov states $s \in S$
- A is a (finite) set of actions $a \in A$
- P is dynamics / transition-model for each action,

$$P(s_{t+1} = s' | s_t = s, a_t = a)$$

- R is a reward function

$$R(s_t = s, a_t = a) = E[r_t | s_t = s, a_t = a]$$

- γ is discount factor $\gamma \in [0,1]$
- MDP is a tuple: (S, A, P, R, γ)

Reinforcement Learning

- Assume a Markov decision process (MDP):
 - A set of states $s \in S$
 - A set of actions (per state) A
- Looking for a policy $\pi(s)$ 找到一個策略
- Try actions and states out to learn, i.e. collect trials

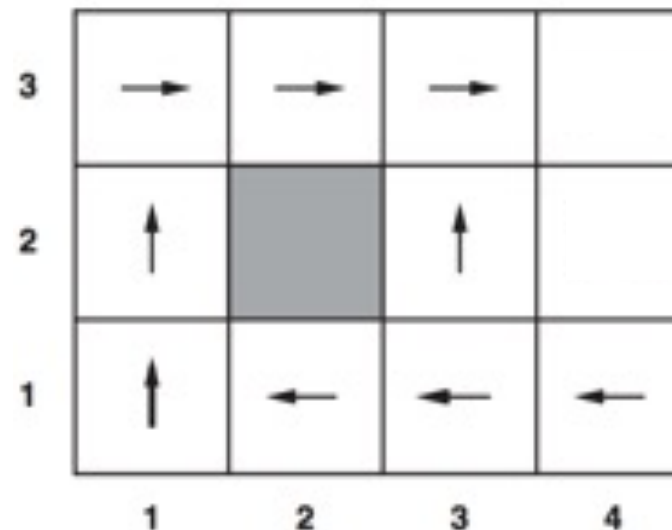
$$\tau = s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T$$

- The return of a trial. $\gamma \in [0,1]$ is the discount factor

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1}$$

Example : Grid World

- States set: All grids $\{(1,1), (1,2)...\}$
- Actions set: north, west, east, south
- No transition model
- No reward model
- Collect trials following the given policy



-0.04的獎勵

$(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) +1$
 $(1, 1) \xrightarrow{-0.04} (1, 2) \xrightarrow{-0.04} (1, 3) \xrightarrow{-0.04} (2, 3) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (3, 3) \xrightarrow{-0.04} (4, 3) +1$
 $(1, 1) \xrightarrow{-0.04} (2, 1) \xrightarrow{-0.04} (3, 1) \xrightarrow{-0.04} (3, 2) \xrightarrow{-0.04} (4, 2) -1$.

Example: Expected Age

Goal: Compute expected age of DATA130008 students

Known $P(A)$

$$E[A] = \sum_a P(a) \cdot a = 0.35 \times 20 + \dots$$

Without $P(A)$, instead collect samples $[a_1, a_2, \dots, a_N]$

Unknown $P(A)$: “Model Based”

$$\hat{P}(a) = \frac{\text{num}(a)}{N} \quad \text{把比例估出來}$$

$$E[A] \approx \sum_a \hat{P}(a) \cdot a$$

Unknown $P(A)$: “Model Free”

$$E[A] \approx \frac{1}{N} \sum_i a_i$$

Outline

- Reinforcement Learning
- Policy Evaluation
 - Model-based

Policy Evaluation

- Input: a fixed policy $\pi(s)$
- You don't know the transitions $P(s' | s, a)$
- You don't know the rewards $R(s, a, s')$
- Goal: learn state values $V^\pi(S)$
- In this case:
 - No choice about what actions to take
 - Just execute the policy and learn from experience

Model-Based Learning

- Model-Based Idea:
 - Learn an approximate model based on trials
 - Solve for values as if the learned model were correct
- Step 1: Learn empirical MDP model
 - Count outcomes s' for each (s, a)
 - Normalize to give an estimate of $P(s_{t+1} = s' | s_t = s, a_t = a)$
 - Discover each $R(s_t = s, a_t = a)$ when we experience (s, a, s')
- Step 2: Solve the learned MDP
 - For example, use value iteration

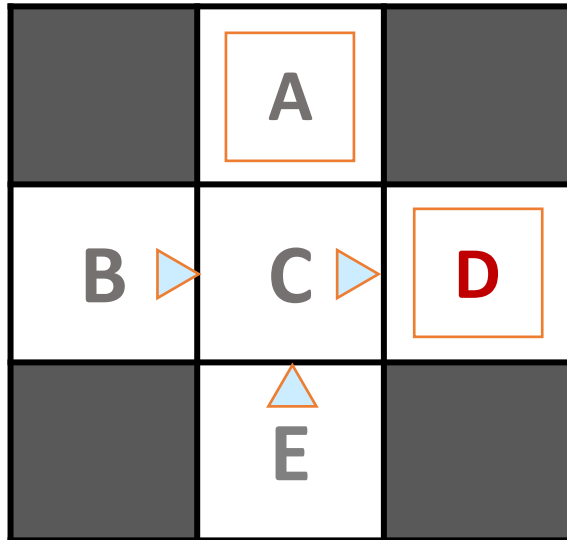
Example: Model-Based Learning

Input Policy π : {(B, east), (C, east), (E, north)}

End states: A and D

Start states: B and E

Assume: $\gamma = 1$



Observed Episodes

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

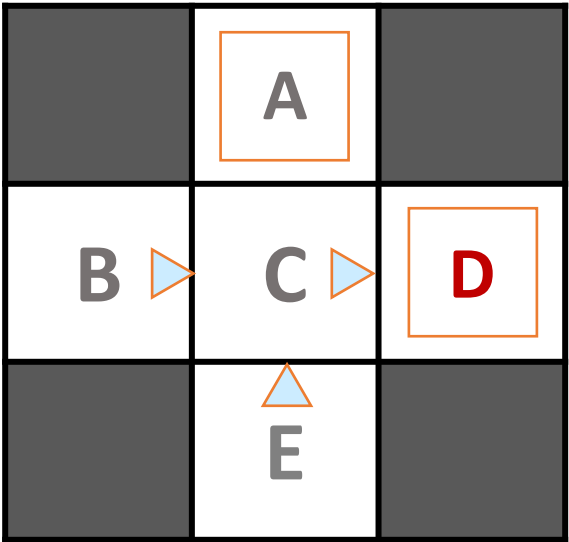
E, north, C, -1
C, east, A, -1
A, exit, x, -10

Example: Model-Based Learning

Input Policy π : {(B, east), (C, east), (E, north)}
End states: A and D
Start states: B and E
Assume: $\gamma = 1$

Observed Episodes (Training)

T1: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10) }
T2: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10) }
T3: { (E, north, C, -1), (C, east, D, -1), (D, exit, x, +10) }
T4: { (E, north, C, -1), (C, east, A, -1), (A, exit, x, -10) }



Transition Model

(s, a)	A	B	C	D	E	x	Total #
(A, exit)							
(B, east)							
(C, east)							
(D, exit)							
(E, north)							

Reward Model

(s, a)	reward	Total #	utility
(A, exit)			
(B, east)			
(C, east)			
(D, exit)			
(E, north)			

Adaptive Dynamic Programming

```
function PASSIVE-ADP-AGENT(percept) returns an action
  inputs: percept, a percept indicating the current state  $s'$  and reward signal  $r'$ 
  persistent:  $\pi$ , a fixed policy
               mdp, an MDP with model  $P$ , rewards  $R$ , discount  $\gamma$ 
                $U$ , a table of utilities, initially empty
                $N_{sa}$ , a table of frequencies for state–action pairs, initially zero
                $N_{s'|sa}$ , a table of outcome frequencies given state–action pairs, initially zero
                $s, a$ , the previous state and action, initially null

  if  $s'$  is new then  $U[s'] \leftarrow r'$ ;  $R[s'] \leftarrow r'$ 
  if  $s$  is not null then
    increment  $N_{sa}[s, a]$  and  $N_{s'|sa}[s', s, a]$ 
    for each  $t$  such that  $N_{s'|sa}[t, s, a]$  is nonzero do
       $P(t | s, a) \leftarrow N_{s'|sa}[t, s, a] / N_{sa}[s, a]$ 
     $U \leftarrow \text{POLICY-EVALUATION}(\pi, U, \textit{mdp})$ 
  if  $s'.\text{TERMINAL?}$  then  $s, a \leftarrow \text{null}$  else  $s, a \leftarrow s', \pi[s']$ 
  return  $a$ 
```

Figure 21.2 A passive reinforcement learning agent based on adaptive dynamic programming. The POLICY-EVALUATION function solves the fixed-policy Bellman equations, as described on page 657.

An **adaptive dynamic programming** (or ADP) agent takes advantage of the constraints among the utilities of states by learning the transition model that connects them and solving the corresponding Markov decision process using a dynamic programming method.

Outline

- Reinforcement Learning
- Policy Evaluation
 - Model-based RL
 - Model-free RL Mont-Carlo

Monte Carlo (MC), i.e., Direct Evaluation

- Given episodes, $s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T$

- Goal: Compute values for each state under π

$$V^\pi(s) = E_\pi[G_t | S_t = s]$$

- Idea: Average together observed sample values

$$V^\pi(s) = \frac{1}{N} \left(\sum_{i=1}^n v_i^\pi(s) \right)$$

$$v_i^\pi(s) = G_t | S_t = s$$

Frist-Visit MC policy evaluation

Initialize $N(s) = 0$, $G(s) = 0 \forall s \in S$

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots + \gamma^{T_i-t} r_{i,T_i}$ as return from time step t onwards in i th episode
- For each state s visited in episode i
 - For **first time** t that state s is visited in episode i
 - Increment counter of total first visits: $N(s) = N(s) + 1$
 - Increment total return $G(s) = G(s) + G_{i,t}$
 - Update estimate $V^\pi(s) = G(s)/N(s)$

First time MC: for an episode, update the state value when it is first visited.

Every time MC: for an episode, update the state value when every time it is visited. Might update multiple times for a single state in a single episode.

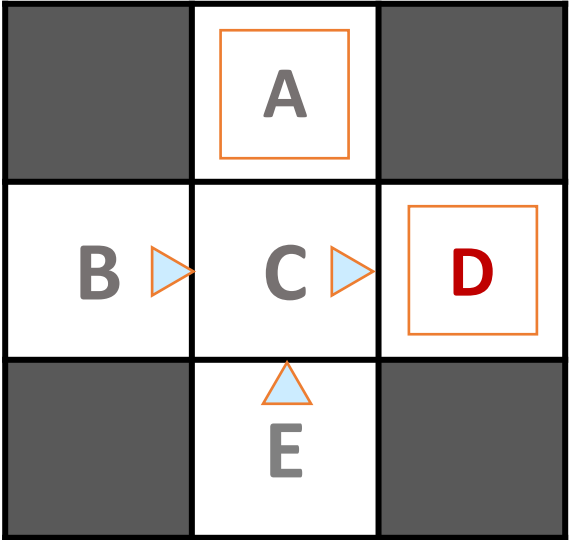
Example: Monte Carlo

Input Policy π : {(B, east), (C, east), (E, north)}
End states: A and D
Start states: B and E
Assume: $\gamma = 1$

Observed Episodes
T1: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10)}
T2: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10)}
T3: { (E, north, C, -1), (C, east, D, -1), (D, exit, x, +10)}
T4: { (E, north, C, -1), (C, east, A, -1), (A, exit, x, -10)}

T1:

$$v^\pi(B) = G_{1,1} = (-1) + \gamma(-1) + \gamma * \gamma * (+10) = 8$$



	$V^\pi(B)$	$V^\pi(C)$	$V^\pi(D)$	$V^\pi(E)$	$V^\pi(A)$
T1 updating					
T2 updating					
T3 updating					
T4 updating					
Total #					

Incremental Monto Carlo (MC)

- After each episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots$ as return from time step t onwards in i_{th} episoder
- For state s visited at time step t in episode i
 - $G_i(s)$ is the total return obtained for state s after episode i .
 - Increment counter of total first visits: $N_i(s) = N_{i-1}(s) + 1$
 - Update estimate:

$$V_i^\pi(s) = V_{i-1}^\pi(s) \left(\frac{N_i(s) - 1}{N_i(s)} + \frac{G_{i,t}}{N_i(s)} \right) = V_{i-1}^\pi(s) + \frac{1}{N_i(s)} (G_{i,t} - V_{i-1}^\pi(s))$$

Incremental Monto Carlo (MC)

- After each episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots$ as return from time step t onwards in i_{th} episoder
- For state s visited at time step t in episode i
 - Increment counter of total first visits: $N(s) = N(s) + 1$
 - Update estimate:

$$V^\pi(s) = V^\pi(s) + \alpha (G_{i,t} - V^\pi(s))$$

$$\alpha = \frac{1}{N(s)}: \text{identical to first visit MC}$$

$$\alpha > \frac{1}{N(s)}: \text{forget older data}$$

Incremental Monto Carlo (MC), Running Mean

Initialize $N(s) = 0$, $G(s) = 0 \forall s \in S$

Loop

- Sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$
- Define $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots \gamma^{T_i-1} r_{i,T_i}$ as return from time step t onwards in i th episode
- For state s visited at time step t in episode i
 - Increment counter of total first visits: $N(s) = N(s) + 1$
 - Update estimate

$$V^\pi(s) = V^\pi(s) + \alpha(G_{i,t} - V^\pi(s))$$

- $\alpha = \frac{1}{N(s)}$: identical to every visit MC
- $\alpha > \frac{1}{N(s)}$: forget older data, helpful for non-stationary domains

Outline

- Reinforcement Learning
- Policy Evaluation
 - Model-based RL
 - Model-free RL Mont-Carlo
 - Model-free RL Time Difference

Problems with MC

- What's good about Mont Carlo?
 - It follows the definition of accumulated rewards
 - It is easy to understand
- What's bad about it?
 - States are learned separately.
 - Information about state connections are ignored.
 - The estimation is not stable because it can be influenced by sample easily.

Trial 1: B, C, D	# of updates:
Trial 2: B, C, D	C: 4 times
Trial 3: E, C, D	E: 2 times
Trial 4: E, C, A	B: 2 times

Why not use the value of C to help estimate the value of E and B?

Output Values

	<div>-10 A</div>	
<div>+8 B</div>	<div>+4 C</div>	<div>+10 D</div>
	<div>-2 E</div>	

B and E both go to C under this policy. But their values are quite different. Why?

Transition-Based Policy Evaluation

- Improve our estimate of V by policy evaluation:

$$V^\pi(s) \leftarrow R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^\pi(s')$$

- Idea: average and value of s' in all transitions
 - Sample transitions, (s, a, r_1, s_1') , (s, a, r_2, s_2') , (s, a, r_3, s_3')

$$sample_1 = r_1 + \gamma V^\pi(s_1')$$

$$sample_2 = r_2 + \gamma V^\pi(s_2')$$

$$sample_3 = r_3 + \gamma V^\pi(s_3')$$

$$V^\pi(s) = \frac{1}{N} \left(\sum_{i=1}^n sample_i \right)$$

Temporal Difference

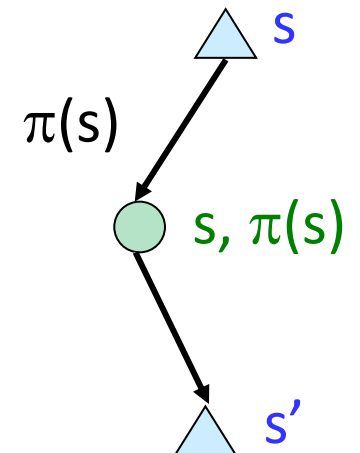
- Update $V^\pi(s)$ each time we experience a transition (s, a, s', r)
- Likely outcomes s' will contribute updates more often

很小概率的採集到在小樣本影響大

- Temporal difference Policy Evaluation
 - Move values **toward value of successor occurs**

Sample of $V^\pi(s)$: $sample = r_1 + \gamma V^\pi(s')$

Update to $V^\pi(s)$: $V^\pi(s) = V^\pi(s) + \alpha(sample - V^\pi(s))$



Temporal Difference Learning

- Aim: estimate $V^\pi(s)$ given episodes generated under policy π
 - $s_1, a_1, r_1, s_2, a_2, r_2, \dots$ where the actions are sampled from π
- Simplest TD learning: update value towards estimated value

$$V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$$

- TD error:

$$\delta_t = r_t + \gamma V^\pi(s_{t+1}) - V^\pi(s_t)$$

- Can immediately update value estimate after (s, a, r, s') tuple

“If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning.” – Sutton and Barto 2017

Temporal Difference Learning Algorithm

Input: α

Initialize $V^\pi(s) = 0, \forall s \in S$

Loop

- Sample **tuple** (s_t, a_t, r_t, s_{t+1})
- $V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$

Example: Temporal Difference Policy Evaluation

Input Policy π : {(B, east), (C, east), (E, north)}

End states: A and D

Start states: B and E

Assume: $\gamma = 1$

Learning rate: alpha = 0.5

States value initialized as 0 (including x)

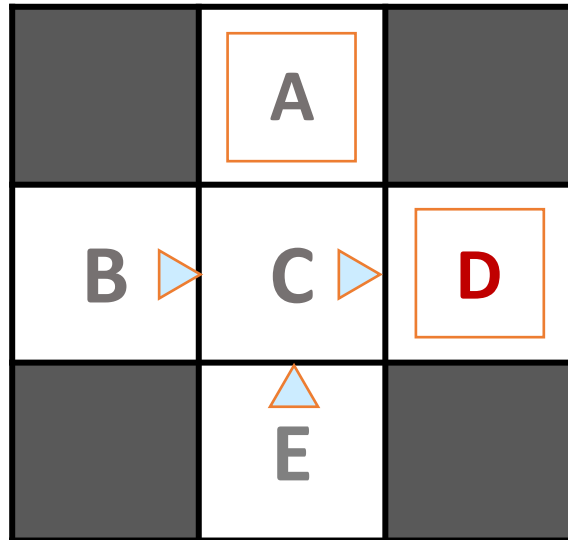
Observed Episodes (Training)

T1: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10) }

T2: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10) }

T3: { (E, north, C, -1), (C, east, D, -1), (D, exit, x, +10) }

T4: { (E, north, C, -1), (C, east, A, -1), (A, exit, x, -10) }



Update for T1

	$V^\pi(B)$	$V^\pi(C)$	$V^\pi(D)$	$V^\pi(E)$	$V^\pi(A)$
Initial	0	0	0	0	0
(B, east, C, -1)					
(C, east, D, -1)					
(D, exit, x, +10)					

$$V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$$

Example: Temporal Difference Policy Evaluation

Input Policy π : {(B, east), (C, east), (E, north)}

End states: A and D

Start states: B and E

Assume: $\gamma = 1$

Learning rate: alpha = 0.5

States value initialized as 0 (including x)

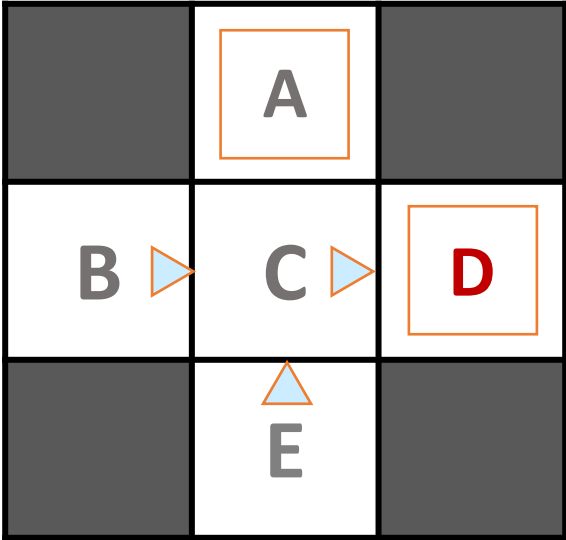
Observed Episodes (Training)

T1: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10)}

T2: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10)}

T3: { (E, north, C, -1), (C, east, D, -1), (D, exit, x, +10)}

T4: { (E, north, C, -1), (C, east, A, -1), (A, exit, x, -10)}



Update for T2

	$V^\pi(B)$	$V^\pi(C)$	$V^\pi(D)$	$V^\pi(E)$	$V^\pi(A)$
After T1					
(B, east, C, -1)					
(C, east, D, -1)					
(D, exit, x, +10)					

$$V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$$

Example: Temporal Difference Policy Evaluation

Input Policy π : {(B, east), (C, east), (E, north)}

End states: A and D

Start states: B and E

Assume: $\gamma = 1$

Learning rate: alpha = 0.5

States value initialized as 0 (including x)

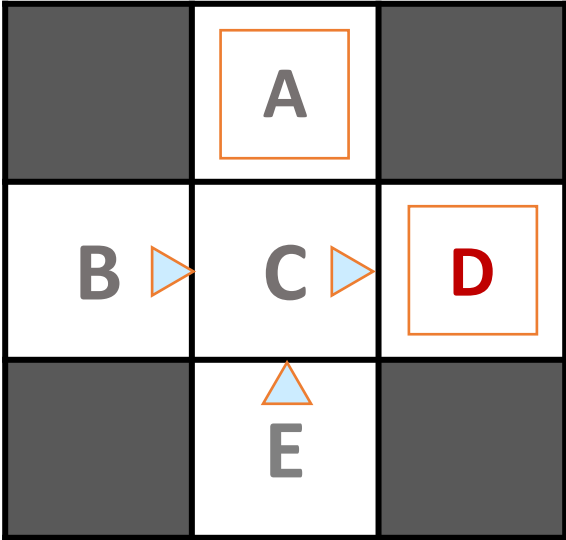
Observed Episodes (Training)

T1: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10)}

T2: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10)}

T3: { (E, north, C, -1), (C, east, D, -1), (D, exit, x, +10)}

T4: { (E, north, C, -1), (C, east, A, -1), (A, exit, x, -10)}



Update for T3

	$V^\pi(B)$	$V^\pi(C)$	$V^\pi(D)$	$V^\pi(E)$	$V^\pi(A)$
After T2					
(E, north, C, -1)					
(C, east, D, -1)					
(D, exit, x, +10)					

$$V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$$

Example: Temporal Difference Policy Evaluation

Input Policy π : {(B, east), (C, east), (E, north)}

End states: A and D

Start states: B and E

Assume: $\gamma = 1$

Learning rate: $\alpha = 0.5$

States value initialized as 0 (including x)

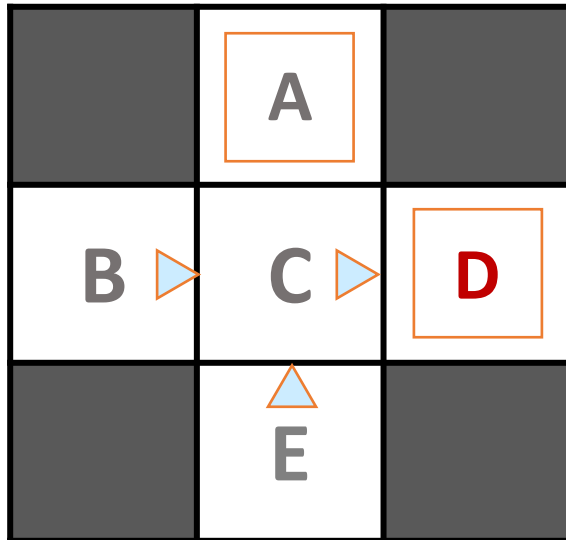
Observed Episodes (Training)

T1: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10) }

T2: { (B, east, C, -1), (C, east, D, -1), (D, exit, x, +10) }

T3: { (E, north, C, -1), (C, east, D, -1), (D, exit, x, +10) }

T4: { (E, north, C, -1), (C, east, A, -1), (A, exit, x, -10) }



Update for T4

	$V^\pi(B)$	$V^\pi(C)$	$V^\pi(D)$	$V^\pi(E)$	$V^\pi(A)$
After T3					
(E, north, C, -1)					
(C, east, A, -1)					
(A, exit, x, -10)					

$$V^\pi(s_t) = V^\pi(s_t) + \alpha(\underbrace{[r_t + \gamma V^\pi(s_{t+1})]}_{\text{TD target}} - V^\pi(s_t))$$

Outline

- Reinforcement Learning
- Policy Evaluation
 - Model-based RL
 - Model-free RL Mont-Carlo
 - Model-free RL Time Difference
 - Comparison of different approaches

Difference of ADP and TD

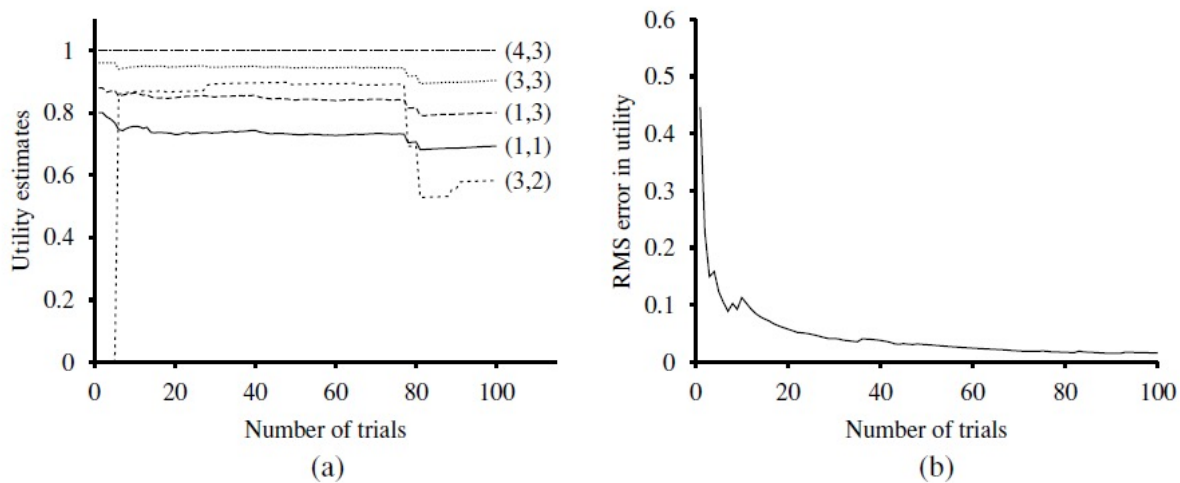


Figure 21.3 The passive ADP learning curves for the 4×3 world, given the optimal policy shown in Figure 21.1. (a) The utility estimates for a selected subset of states, as a function of the number of trials. Notice the large changes occurring around the 78th trial—this is the first time that the agent falls into the -1 terminal state at $(4,2)$. (b) The root-mean-square error (see Appendix A) in the estimate for $U(1,1)$, averaged over 20 runs of 100 trials each.

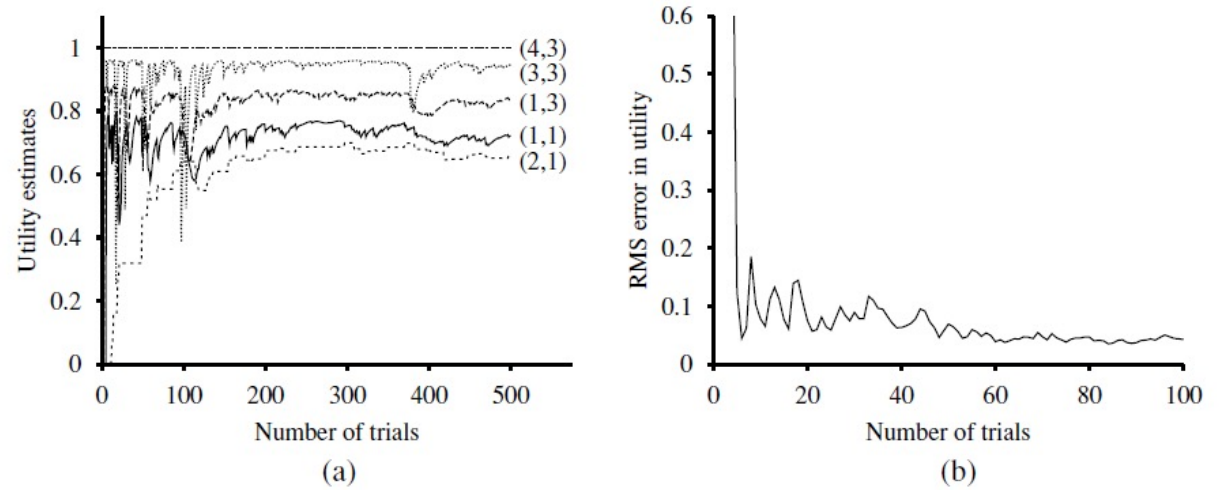


Figure 21.5 The TD learning curves for the 4×3 world. (a) The utility estimates for a selected subset of states, as a function of the number of trials. (b) The root-mean-square error in the estimate for $U(1,1)$, averaged over 20 runs of 500 trials each. Only the first 100 trials are shown to enable comparison with Figure 21.3.

- ADP (chapter 21.3): model-based, **left figure**
- TD: model free, **right figure**
- ADP is more stable and faster to converge.
- TD is easier to implement and flexible for problem with large state space.

Temporal Difference VS Monte Carlo

- With episode $s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T$

- **Both update the state value by samples**

$$V^\pi(s) = V^\pi(s) + \alpha(v^\pi(s) - V^\pi(s))$$

- **Monte Carlo**

$$v^\pi(s) = \sum_{k=0}^{T-1} \gamma^k r_{t+k} | S_t = s$$

- **Temporal Difference**

$$v^\pi(s) = r_t + \gamma V^\pi(s_{t+1}) | S_t = s$$

- In TD, use (s, a, r, s') to update $V(s)$
 - $O(1)$ operation per update; in an episode of length L , $O(L)$
- In MC have to wait till episode finishes, then also $O(L)$
- TD exploits Markov structure

Comparison of DP, MC and TD

- **DP:** Dynamic Programming, i.e., value iteration, policy iteration
- **MC:** model free approach, monte carlo
- **TD:** model free approach, time difference

Criteria	DP	MC	TD
Rely on model (transition, reward) of the world?	Y	N	N
Able to handle continuing (non-episodic) domains	Y	N	Y
Able to handles Non-Markovian domains	N	Y	N
Converges to true value in limit	Y	Y	Y

Yes or No

1. episodic: game will end in a certain step. You will have a lot of episodes.
2. Continuing: game will last for ever. You will only have one sample.
3. Markovian domains: transitions of states follow Markov property

Outline

- Reinforcement Learning
- Policy Evaluation
 - Model-based RL
 - Model-free RL Mont-Carlo
 - Model-free RL Time Difference
 - Comparison of different approaches
- Policy Control

Model-free Control

- You don't know the transitions $P(s' | s, a)$
- You don't know the rewards $R(s, a)$
- You choose the actions now
- Goal: learn the optimal policy / values: | S | X | A | values
- In this case:
 - **On-policy** training VS **off-policy** training
 - Fundamental tradeoff: **exploration** vs. **exploitation**

勘探和开发

Policy Iteration with Model

- Initialize policy π
- Repeat:
 - Policy evaluation: compute $Q^\pi(s, a)$
 - Policy improvement: update π

$$Q^\pi(s, a) \leftarrow R(s, a) + \gamma \sum_{s'} P(s'|s, a) Q^\pi(s', a')$$

$$\pi[s] \leftarrow \operatorname{argmax}_{a \in A(s)} \sum_{s'} P(s'|s, a) V(s')$$

Policy Iteration without model

- Initialize policy π
- Repeat:
 - **Policy evaluation**: compute $Q^{\pi}(s, a)$
 - **Policy improvement**: update π
- How to perform policy iteration with samples?
- **Sample-based policy evaluation + policy improvement**

Outline

- Reinforcement Learning
- Policy Evaluation
 - Model-based RL
 - Model-free RL Mont-Carlo
 - Model-free RL Time Difference
 - Comparison of different approaches
- Policy Control
 - Monto Carlo Policy Iteration
 - Monto Carlo Policy Evaluation (On VS Off Policy)

Sample-based Policy Evaluation

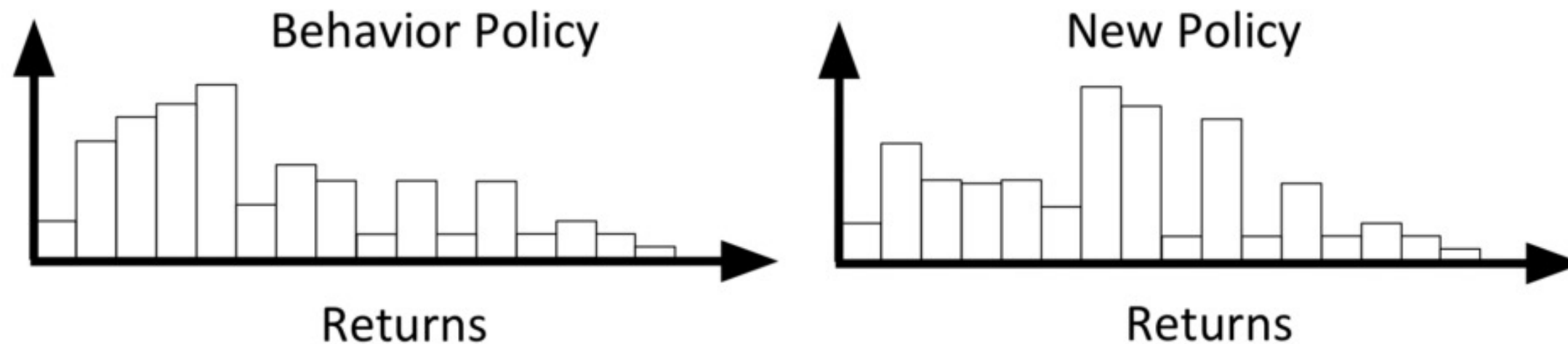
- Sample Episodes following a policy π_2
 - $s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T$
- Compute values for each state under π_1
 - $Q^{\pi_1}(s, a)$
- **Behavior policy π_2** : gather experience from
- **Estimated policy π_1** : the one you want to estimate the value of

On and Off-Policy Learning

- **Behavior policy π_2** : gather experience from
- **Estimated policy π_1** : the one you want to estimate the value of
- On-policy learning ($\pi_1 = \pi_2$) 代價太高
 - Learn to estimate a policy from experience obtained from following that policy
- Off-policy learning ($\pi_1 \neq \pi_2$) 主要是使用異策略 樣本量多
 - Learn to estimate a policy using experience gathered from following a different policy
 - ✓ Sometimes trying actions out is costly
 - ✓ Would like to use historical data from old policies

Returns Mismatch of Two Policies

- Distribution of episodes & resulting returns differs between policies



相似權重可增加；不相似則減少

- Use importance sampling to weight different policies

Monte Carlo (MC) On Policy Evaluation

Initialize $N(s, a) = 0$, $G(s, a) = 0$, $Q^\pi(s, a) = 0$, $\forall s \in S$, $\forall a \in A$

Loop

- Using policy π sample episode $i = s_{i,1}, a_{i,1}, r_{i,1}, s_{i,2}, a_{i,2}, r_{i,2}, \dots, s_{i,T_i}$
- $G_{i,t} = r_{i,t} + \gamma r_{i,t+1} + \gamma^2 r_{i,t+2} + \dots + \gamma^{T_i-1} r_{i,T_i}$
- For each **state,action** (s, a) visited in episode i
 - For **first or every** time t that (s, a) is visited in episode i
 - $N(s, a) = N(s, a) + 1$, $G(s, a) = G(s, a) + G_{i,t}$
 - Update estimate $Q^\pi(s, a) = G(s, a) / N(s, a)$

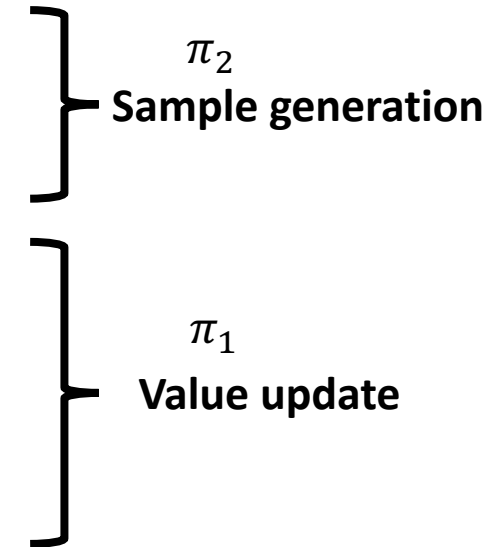
π_1
Sample generation

π_1
Value update

- Both behavior and estimated policies are π_1

Monte Carlo (MC) Off Policy Evaluation

- Aim: estimate value of policy π_1 , $V^{\pi_1}(s)$, given episodes generated under behavior policy π_2
 - $s_1, a_1, r_1, s_2, a_2, r_2, \dots$ where the actions are sampled from π_2
- $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$ in MDP M under policy π
- $V^\pi(s) = \mathbb{E}_\pi[G_t | s_t = s]$
- Have data from a different policy, behavior policy π_2



- Note that, not all G_t can be used in this case.

Outline

- Reinforcement Learning
- Policy Evaluation
 - Model-based RL
 - Model-free RL Mont-Carlo
 - Model-free RL Time Difference
 - Comparison of different approaches
- Policy Control
 - Monto Carlo Policy Iteration
 - Monto Carlo Policy Evaluation (On VS Off Policy)
 - Monto Carlo Policy Improvement (Exploration VS Exploitation)

Policy Improvement

- Given an estimate $Q^{\pi_i}(s, a) \forall s, a$
- Update new policy

$$\pi_{i+1}(s) = \arg \max_a Q^{\pi_i}(s, a)$$

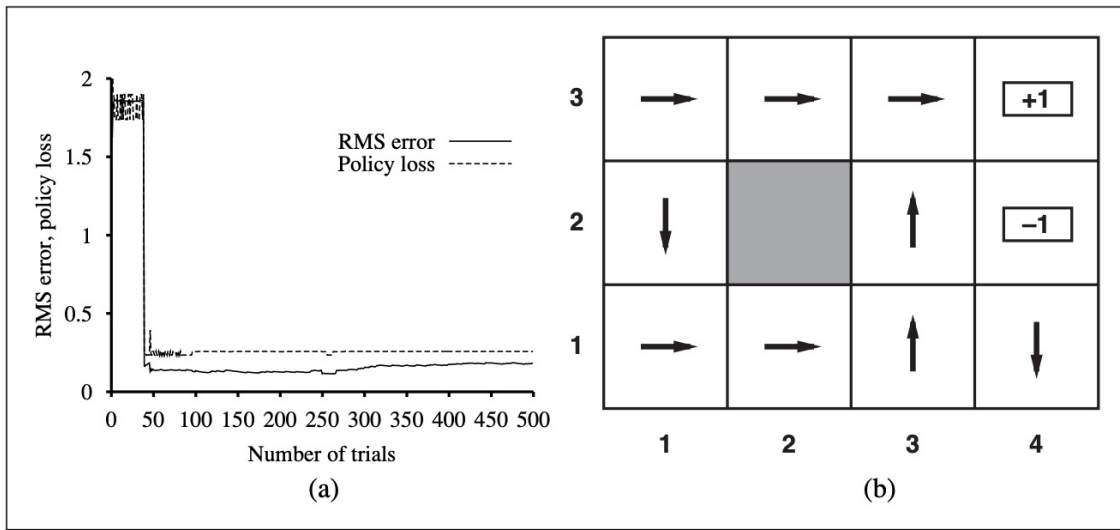


Figure 21.6 Performance of a greedy ADP agent that executes the action recommended by the optimal policy for the learned model. (a) RMS error in the utility estimates averaged over the nine nonterminal squares. (b) The suboptimal policy to which the greedy agent converges in this particular sequence of trials.

How to cover more policies?

Policy Improvement with Exploration

- Intuitive approach: ~~Need to try all (s, a) pairs~~, collect enough trails to evaluate all possible policies.
 - **This is impossible in practice.**
- Start from an initial policy and explore more policy spaces step by step.
- **Exploration VS Exploitation**
 - When to try new action ? **Exploration.**
 - When to follow the **current optimal π** ? **Exploitation.**

Exploration with ε -greedy policy

- Let $|A|$ be the number of actions
- Then an **ε -greedy policy** w.r.t. a state-value $Q^\pi(s, a)$ is

$$\pi^\varepsilon(s) = \begin{cases} a, & \text{with probability of } \frac{\varepsilon}{|A|} \\ \operatorname{argmax}_a Q^\pi(s, a), & \text{with probability of } 1 - \varepsilon \end{cases}$$

Monte Carlo for On-policy Policy Iteration

```
1: Initialize  $Q(s, a) = 0, N(s, a) = 0 \forall (s, a)$ , Set  $\epsilon = 1, k = 1$ 
2:  $\pi_k = \epsilon$ -greedy( $Q$ ) // Create initial  $\epsilon$ -greedy policy
3: loop
4:   Sample  $k$ -th episode  $(s_{k,1}, a_{k,1}, r_{k,1}, s_{k,2}, \dots, s_{k,T})$  given  $\pi_k$ 
4:    $G_{k,t} = r_{k,t} + \gamma r_{k,t+1} + \gamma^2 r_{k,t+2} + \dots + \gamma^{T-t} r_{k,T}$ 
5:   for  $t = 1, \dots, T$  do
6:     if First visit to  $(s, a)$  in episode  $k$  then
7:        $N(s, a) = N(s, a) + 1$ 
8:        $Q(s_t, a_t) = Q(s_t, a_t) + \frac{1}{N(s,a)} (G_{k,t} - Q(s_t, a_t))$ 
9:     end if
10:  end for
11:   $k = k + 1, \epsilon = 1/k$ 
12:   $\pi_k = \epsilon$ -greedy( $Q$ ) // Policy improvement
13: end loop
```

π_k
Sample generation

π_k
Value update

π_{k+1}
Policy Improvement

Outline

- Reinforcement Learning
- Policy Evaluation
 - Model-based RL
 - Model-free RL Mont-Carlo
 - Model-free RL Time Difference
 - Comparison of different approaches
- Policy Control
 - Monto Carlo Policy Iteration
 - Monto Carlo Policy Evaluation (On VS Off Policy)
 - Monto Carlo Policy Improvement (Exploration VS Exploitation)
 - Time Difference Policy Iteration

Policy Iteration without model using TD

MC:一次轉到底

- Use Temporal difference methods for policy iteration
- Initialize policy π
- Repeat:
 - **Policy evaluation**: compute $Q^\pi(s, a)$ using TD updating with ε -greedy policy
 - **Policy improvement**: update π same as MC method, set π to ε -greedy policy

每次變化轉

Policy Evaluation without model using TD

- MC: Sample Episodes

- $s_1, a_1, r_1, s_2, a_2, r_2, \dots, s_T$

- TD: Sample Transitions

- $(s_1, a_1, r_1, s_2, a_2), (s_2, a_2, r_2, s_2, a_2), \dots, (s_{T-1}, a_{T-1}, r_{T-1}, s_T, a_T)$

- Compute values for each state

- $Q^{\pi}(s, a)$

- TD based policy evaluation updates values with transition samples.

SARSA Algorithm

- 1: Set initial ϵ -greedy policy π , $t = 0$, initial state $s_t = s_0$
 - 2: Take $a_t \sim \pi(s_t)$ // Sample action from policy
 - 3: Observe (r_t, s_{t+1})
 - 4: **loop**
 - 5: Take action $a_{t+1} \sim \pi(s_{t+1})$
 - 6: Observe (r_{t+1}, s_{t+2})
 - 7: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$
 - 8: $\pi(s_t) = \arg \max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
 - 9: $t = t + 1$
 - 10: **end loop**
-

Q-Learning: Learning towards Optimal $Q(s,a)$

- Key idea: Maintain state-action Q estimates and use the value of the best future action for bootstrap update

- SARSA: $(s_1, a_1, r_1, s_2, a_2)$

同策略 $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma Q(s_{t+1}, \underline{a_{t+1}})) - Q(s_t, a_t))$

- Q-learning: $(s_1, a_1, r_1, s_2, \text{argmax}(s_2))$

偏高

異策略 $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t))$

Important elements for Policy Control (active RL)

- Behavior Policy: choose the action to generate samples
- Estimated policy: the policy we are evaluating

- On policy strategy

- when the two policies are the same: SARSA

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma Q(s_{t+1}, \underline{a_{t+1}})) - Q(s_t, a_t))$$

- Off policy strategy

- When the two policies are different: Q-learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha((r_t + \gamma \max_{a'} Q(s_{t+1}, a')) - Q(s_t, a_t))$$

- We use important sampling for off policy strategies

Q-Learning: Learning towards Optimal $Q(s,a)$

-
- 1: Initialize $Q(s, a), \forall s \in S, a \in A$ $t = 0$, initial state $s_t = s_0$
 - 2: Set π_b to be ϵ -greedy w.r.t. Q
 - 3: **loop**
 - 4: Take $a_t \sim \pi_b(s_t)$ // Sample action from policy
 - 5: Observe (r_t, s_{t+1})
 - 6: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \arg \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$
 - 7: $\pi(s_t) = \arg \max_a Q(s_t, a)$ w.prob $1 - \epsilon$, else random
 - 8: $t = t + 1$
 - 9: **end loop**
-


Example: Monte Carlo for On-policy Policy Iteration

7 discrete states (location of rover)

2 deterministic actions: left (a_1) or right (a_2)

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
a_1	1	0	0	0	0	0	+10
a_2	0	0	0	0	0	0	+5

Reward model $R(s, a)$

s_1	s_2	s_3	s_4	s_5	s_6	s_7
						

Trajectory =

$(s_3, a_1, 0, s_2, a_2, 0, s_3, a_1, 0, s_2, a_2, 0, s_1, a_1, 1, terminal)$

$\epsilon = 0.5, \gamma = 1$

First visit MC estimate of Q of each (s, a) pair

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
a_1	1	0	1	0	0	0	0
a_2	0	1	0	0	0	0	0

$Q^{\epsilon-\pi}(s, a)$

what is $\pi(s) = \operatorname{argmax}_a Q^{\epsilon-\pi}(s, a)$ for $\forall s$?

$[a_1, a_2, a_1, a_1, a_1, a_1, a_1]$

what is the new ϵ – *greedy* policy, if $k = 3$, $\epsilon = \frac{1}{k}$?

with probability $2/3$ choose $\pi = [1 \ 2 \ 1 \ \text{tie} \ \text{tie} \ \text{tie} \ \text{tie}]$, else choose actions randomly.

Summary of Tabular RL learning

