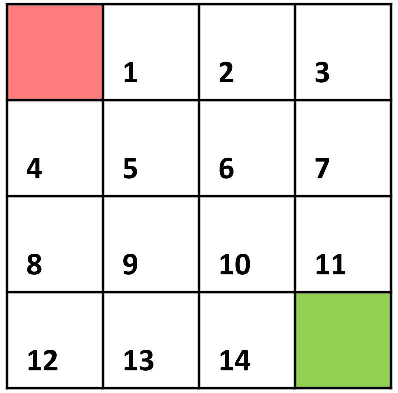
CS 540 Introduction to Artificial Intelligence

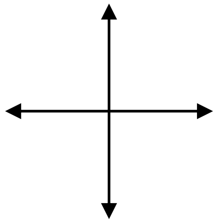
Reinforcement Learning

Asmit Nayak
UW-Madison
Based on slides by David Silver, Fred Sala, Yingyu Liang and R. Sutton

Value Iterations Example



Grid World



Actions

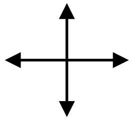
$R_t = -1$
on all transitions

$\gamma = 0.9$
$\alpha = 1$

$\pi_0 \Rightarrow \mathbb{P}[a] = 0.25 \; \forall \; a$

For terminal states $s' = s$

Value Iterations Example



$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 2$

0	-1.675	-1.9	-1.9
-1.675	-1.9	-1.9	-1.9
-1.9	-1.9	-1.9	-1.675
-1.9	-1.9	-1.675	0

$$V_k = 3\big(0.25 * (-1 + 0.9 * (-1))\big) + 0.25 * (-1 + 0) = -1.675$$

$$V_k = 4\big(0.25 * (-1 + 0.9 * (-1))\big) = -1.9$$

Value Iterations Example

$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 2$

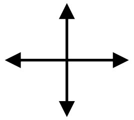
0	-1.575	-1.8	-1.8
-1.575	-1.8	-1.8	-1.8
-1.8	-1.8	-1.8	-1.575
-1.8	-1.8	-1.575	0

$k = 3$

0	-2.23	-2.67	-2.71
-2.23	-2.61	-2.71	-2.66
-1.8	-2.71	-2.61	-2.23
-2.71	-2.66	-2.23	0

$$
\begin{aligned}
V_k = {} & 0.25 * \left(-1 + 0.9 * (-1.675)\right) \\
& +0.25 * \left(-1 + 0.9 * (-1.9)\right) \\
& +0.25 * \left(-1 + 0.9 * (-1.9)\right) \\
& +0.25 * (-1 + 0) = -2.23
\end{aligned}
$$

Value Iterations Example



$R_t = -1$

V_k for π_0

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	

$k = 0$

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

$k = 1$

0	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	0

$k = 3$

0	-2.23	-2.67	-2.71
-2.23	-2.61	-2.71	-2.66
-1.8	-2.71	-2.61	-2.23
-2.71	-2.66	-2.23	0

$k = \infty$

0	-5.28	-7.13	-7.65
-5.28	-6.60	-7.18	-7.13
-7.13	-7.17	-6.60	-5.28
-7.65	-7.13	-5.28	0

Q-Learning

For the previous value iteration, we knew $\mathbb{P}[s'|s,a]$. What if we didn't?

We will use Q-Learning!

Q-Learning tells us the value of doing a in state s

$$Q(s_t, a_t) = \mathbb{E}[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots \mid s, a]$$

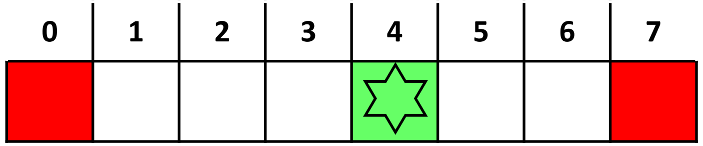
We follow a similar iterative approach as value iterations and get:

$$Q(s_t', a_t') \leftarrow Q(s_t, a_t) + \alpha \left[r(s_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

This is off-policy!

Q-Learning Example

Setup:



Actions: $\rightarrow$ $\leftarrow$

Rewards:

+1 if I reach S_4

-1 if I reach S_0 or S_7

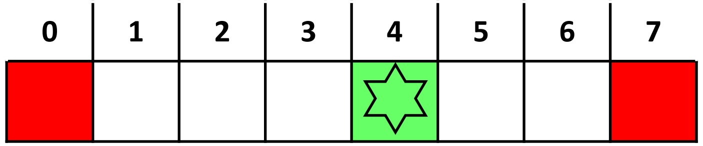
+0 otherwise

$$\alpha = 1,\ \gamma = 1$$

$\boldsymbol{Q_0}$	**Action**	
State	Left	Right
S_0		
S_1		
S_2		
S_3		
S_4		
S_5		
S_6		
S_7		

Q-Learning Example

Setup:



Actions: $\rightarrow$ $\leftarrow$

Rewards:

+1 if I reach S_4

-1 if I reach S_0 or S_7

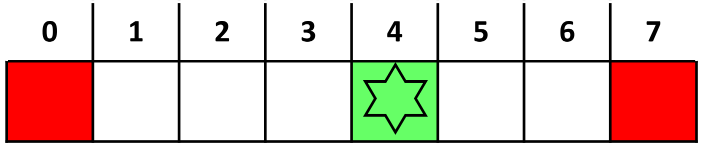
+0 otherwise

$$\alpha = 1, \ \gamma = 1$$

Q_0	**Action**	
State	Left	Right
S_0	0	0
S_1	0	0
S_2	0	0
S_3	0	0
S_4	0	0
S_5	0	0
S_6	0	0
S_7	0	0

Q-Learning Example

Setup:



Actions: $\rightarrow$ $\leftarrow$

Rewards:

+1 if I reach S_4

-1 if I reach S_0 or S_7

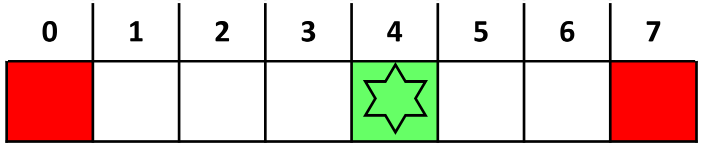
+0 otherwise

$$\alpha = 1, \ \gamma = 1$$

Q_1	**Action**	
State	Left	Right
S_0		
S_1		
S_2		
S_3		
S_4		
S_5		
S_6		
S_7		

Q-Learning Example

Setup:



Actions: $\rightarrow$ $\leftarrow$

Rewards:

+1 if I reach S_4

-1 if I reach S_0 or S_7

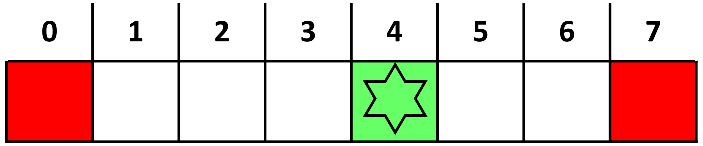
+0 otherwise

$$\alpha = 1, \ \gamma = 1$$

Q_1	**Action**	
State	Left	Right
S_0	0	0
S_1	-1	0
S_2	0	0
S_3	0	1
S_4	0	0
S_5	1	0
S_6	0	-1
S_7	0	0

Q-Learning Example

Setup:



Actions: $\rightarrow$ $\leftarrow$

Rewards:

+1 if I reach S_4

-1 if I reach S_0 or S_7

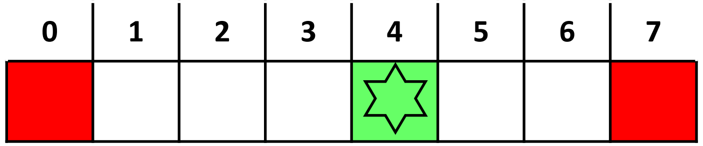
+0 otherwise

$$\alpha = 1,\ \gamma = 1$$

Q_2	**Action**	
State	Left	Right
S_0	0	0
S_1		
S_2		
S_3		
S_4		
S_5		
S_6		
S_7	0	0

Q-Learning Example

Setup:



Actions: $\rightarrow \; \leftarrow$

Rewards:

+1 if I reach S_4

-1 if I reach S_0 or S_7

+0 otherwise

$$\alpha = 1, \; \gamma = 1$$

Q_2	**Action**	
State	Left	Right
S_0	0	0
S_1	-1	0
S_2	0	1
S_3	0	1
S_4	0	0
S_5	1	0
S_6	1	-1
S_7	0	0

Exploration in Q-Learning

With some $0 \leq \epsilon \leq 1$ probability we choose to either take a random action at any given state or go with the highest $Q(s, a)$ value

$$a = \begin{cases} \text{argmax}_{a \in A} Q(s, a) & \epsilon \\ \text{random action } a \in A & 1 - \epsilon \end{cases}$$

SARSA (***S**tate – **A**ction – **R**eward – **S**tate – **A**ction)*

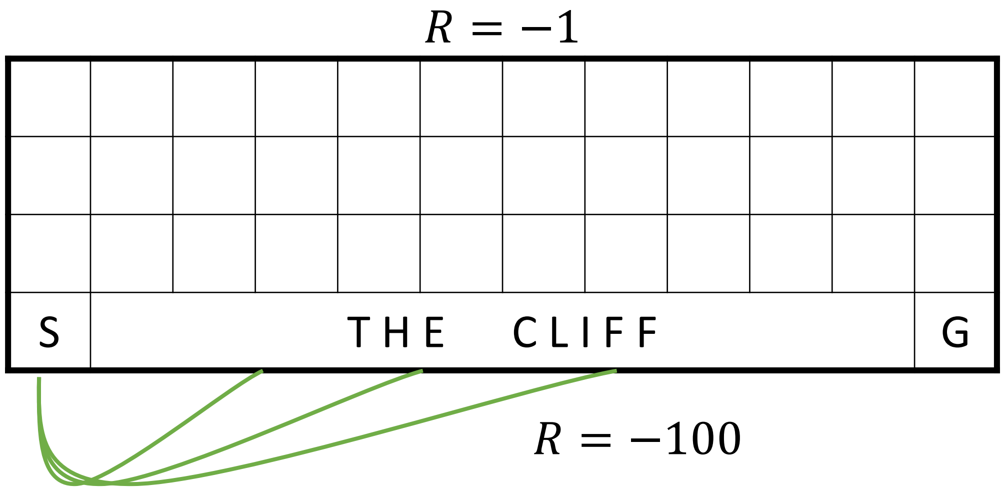
Alternative to Q-Learning, instead of choosing the best possible action we chose the next action according to the policy

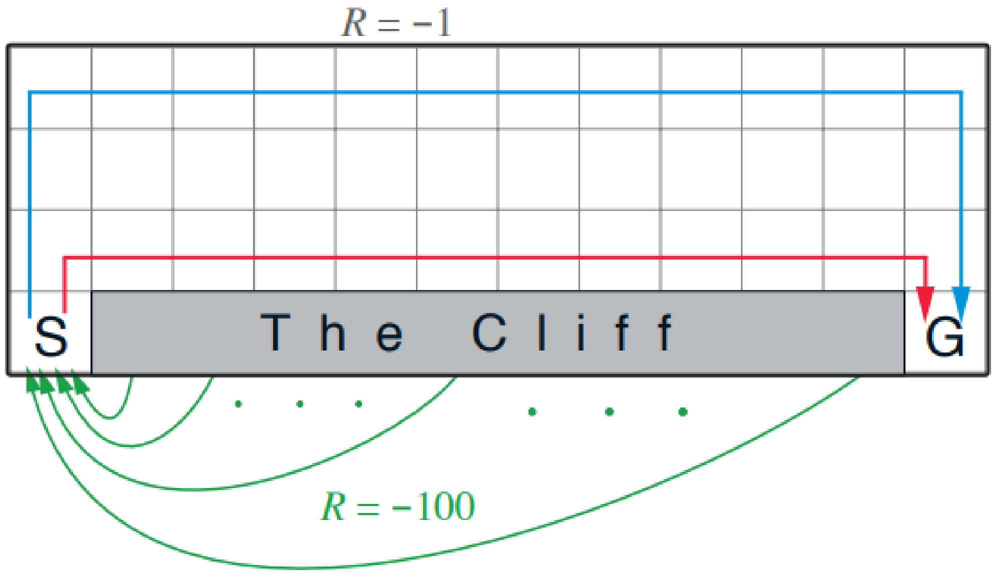
$$Q(s_t', a_t') \leftarrow Q(s_t, a_t) + \alpha[r(s_t) + \gamma \sum_a \pi(a|s_{t+1}) Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$$

This is on-policy!

SARSA vs Q-Learning

Cliff World:





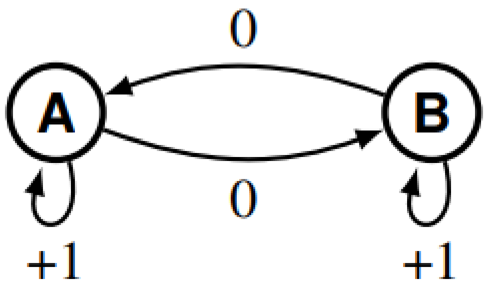
Discussions

Q1. Q-learning vs Value iterations?

Discussions

Q2. Q-Learning with no exploration?

Consider the following Markov Decision Process. It has two states s. It has two actions a: move and stay. The state transition is deterministic: “move” moves to the other state, while “stay” stays at the current state. The reward r is 0 for move, 1 for stay. The agent starts at state A. In case of tie move.



Discussions

Q3. Why Q-Learning is Off-Policy and SARSA On-Policy?