Reinforcement Learning

Reinforcement learning: what if we do not know transitions *P* and reward function *r* of an MDP?

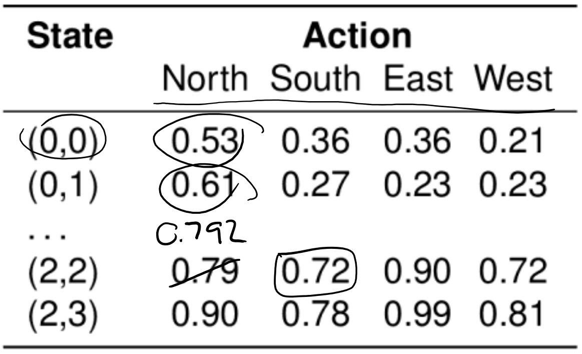
The Mystery Game:
https://programmingheroes.blogspot.com/2016/02/udacity-reinforcement-learning-mystery-game.html

Q-learning

$Q(s,a)$

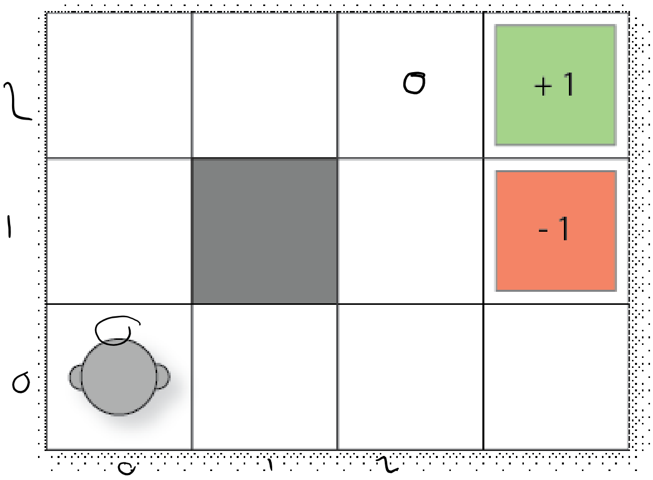
```
1. Initialise $Q(s,a)$ arbitrarily
2. For each episode:
    a. Initialise $s$ (go to the initial state)
    b. Repeat for each step in the episode
        i. Select the next action $a$ to apply from $s$ (using e.g. epsilon greedy, UCT) use $Q(s,a)$
        ii. Execute action $a$ and observe the reward $\boxed{r}$ and new state $s'$
        iii. $Q(s,a) := Q(s,a) + \underset{\text{learning rate}}{\alpha}[r + \gamma \max a' Q(s',a') - Q(s,a)]$   update
        iv. $s := s'$
    c. Until $s$ is terminal
```

Q-Tables



State	Action			
	North	South	East	West
(0,0)	0.53	0.36	0.36	0.21
(0,1)	0.61	0.27	0.23	0.23
...				
(2,2)	~~0.79~~ 0.792	0.72	0.90	0.72
(2,3)	0.90	0.78	0.99	0.81

$Q(s_{(2,2)}, South) = 0.72$



$Q(s,a) := Q(s,a) + \alpha[r + \gamma \max a' Q(s',a') - Q(s,a)]$

Learning rate $\alpha = 0.1$
Discount reward factor $\gamma = 0.9$

Q-learning:
Q((2,2), North) = 0.79 + 0.1*(0 + 0.9*0.9 - 0.79) = 0.792

SARSA, with assumption that a' is West
Q((2,2), North) =

If a' is East is for SARSA, the update would be just the same as for Q-learning because East is the max action from (2,2)

Q-learning: Off-policy

```
1. Initialise $Q(s,a)$ arbitrarily
2. For each episode:
    a. Initialise $s$ (go to the initial state)
    b. Repeat for each step in the episode
        i. Select the next action $a$ to apply from $s$ (using e.g. epsilon greedy, UCT) use $Q(s,a)$
```

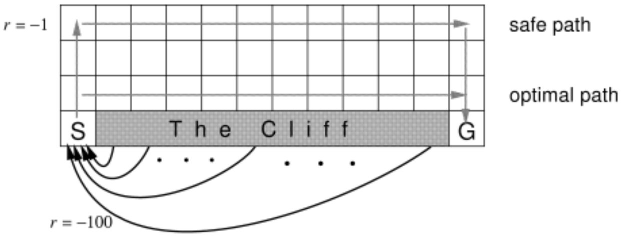
SARSA: On-policy learning

```
1. Initialise $Q(s,a)$ arbitrarily
2. For each episode:
    a. Initialise $s$ (go to the initial state)
    b. Select the next action $a$ to apply from $s$ (using e.g. epsilon greedy, UCT)
    c. Repeat for each step in the episode
```

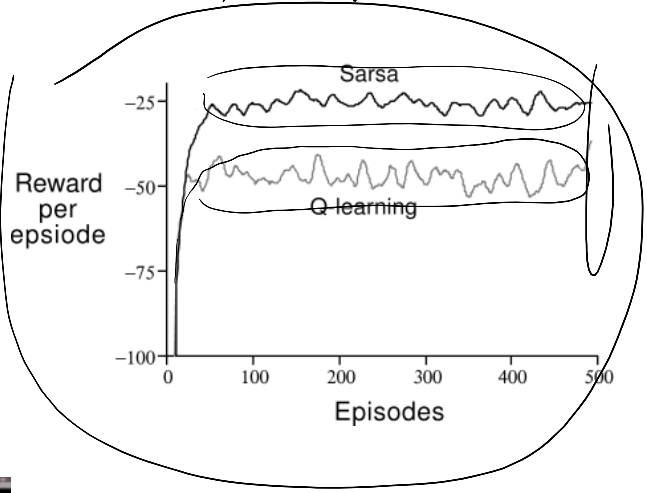
```
b. Repeat for each step in the episode
    i. Select the next action $a$ to apply from $s$ (using e.g. epsilon greedy, UCT) use $Q(s,a)$
    ii. Execute action $a$ and observe the reward $r$ and new state $s'$
    iii. $Q(s,a) := Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    iv. $s := s'$
c. Until $s$ is terminal
```

```
b. Select the next action $a$ to apply from $s$ (using e.g. epsilon greedy, UCT)
c. Repeat for each step in the episode
    i. Execute action $a$ and observe the reward $r$ and new state $s'$
    ii. Select the next action $a$ to apply from $s'$ (using e.g. epsilon greedy, UCT)
    iii. $Q(s,a) := Q(s,a) + \alpha [r + \gamma Q(s',a') - Q(s,a)]$
    iv. $s := s'$; $a := a'$;
d. Until $s$ is terminal
```

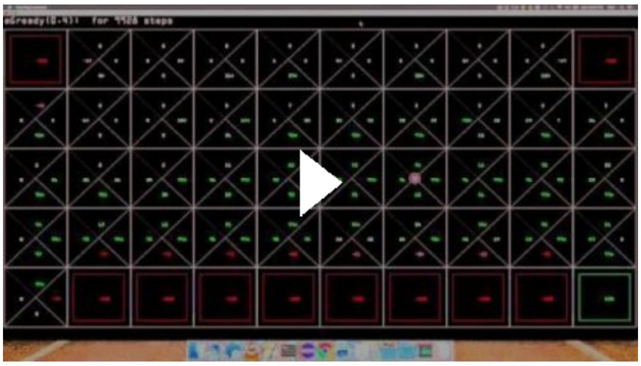
Off-policy vs. on policy learning



during learning



[Gridworld Q-Learning - Example 3 - The Cliff]



[Learning to Play Freeway, using Reinforcement Learning]