Model-free reinforcement learning: Q-learning and SARSA

Model-free 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

Model-based vs model-free

MDP:

- Set of states *S*
- Transition probabilities *P_a(s' | s)*
- Reward function *r(s, a, s')* in real
- Discount factor γ

Q-learning

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) using *Q(s,a)*
        ii. Execute action $\boxed{a}$ and observe the reward $\boxed{r}$ and new state $\boxed{s'}$
        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

estimated reward

estimated future reward

(1-α)

reward

$$Q(s,a) := Q(s,a) + \alpha[r + \boxed{\gamma} \max_{a'} Q(s',a') - \cancel{Q(s,a)}]$$

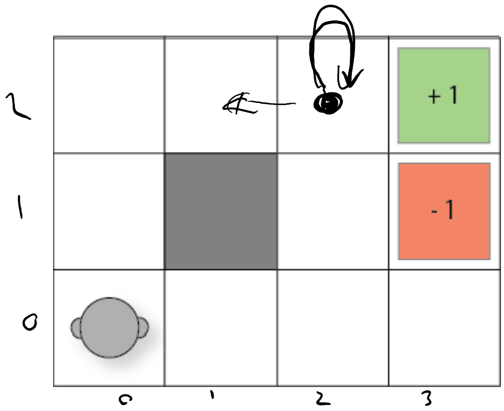
learning rate $0 < \alpha < 1$

discount factor

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	0.72	0.90	0.72
(2,3)	0.90	0.78	0.99	0.81

SARSA example



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	0.72	0.90	0.72
(2,3)	~~0.90~~	~~0.78~~	~~0.99~~	~~0.81~~

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

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

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

SARSA, with assumption that a' is West (on-policy action)
Q((2,2), North) = 0.79 + 0.1* (0 + 0.9*0.72 -0.79)=0.7758

If a' was 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)*
      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

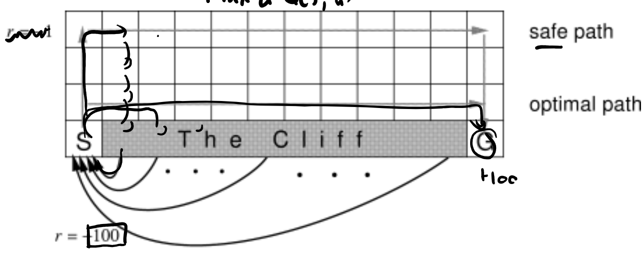
SARSA: On-policy learning

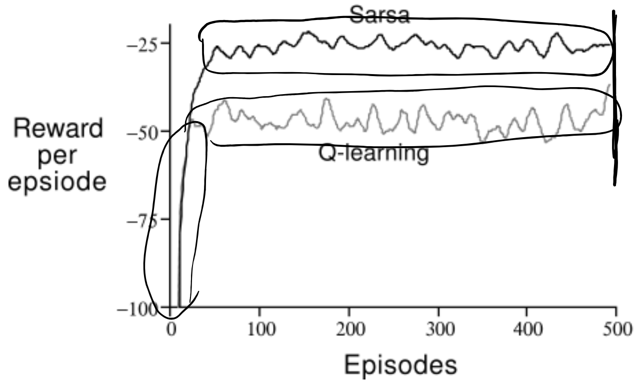
1. Initialise *Q(s,a)* arbitrarily
2. For each episode:
   a. Initialise *s* (go to the initial state)
   b. Select the next action *a* to apply from *s* (using e.g. epsilon greedy, UCT)
   c. Repeat for each step in the episode
      i. Execute action *a* and observe the reward *r* and new state *s'*
      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

$\epsilon = 0.2$

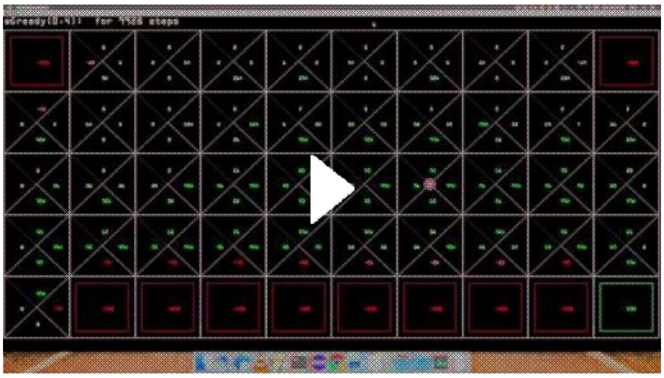
Off-policy vs. on-policy learning

$\max_{a'} Q(s', a')$





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



[Learning to Play Freeway, using Reinforcement Learning]()