Week 9 T8

Wednesday, 21 September 2022 9:20 AM

Model Free Methods

Ways to assign values to states and actions when transition function and action rewards are unknown

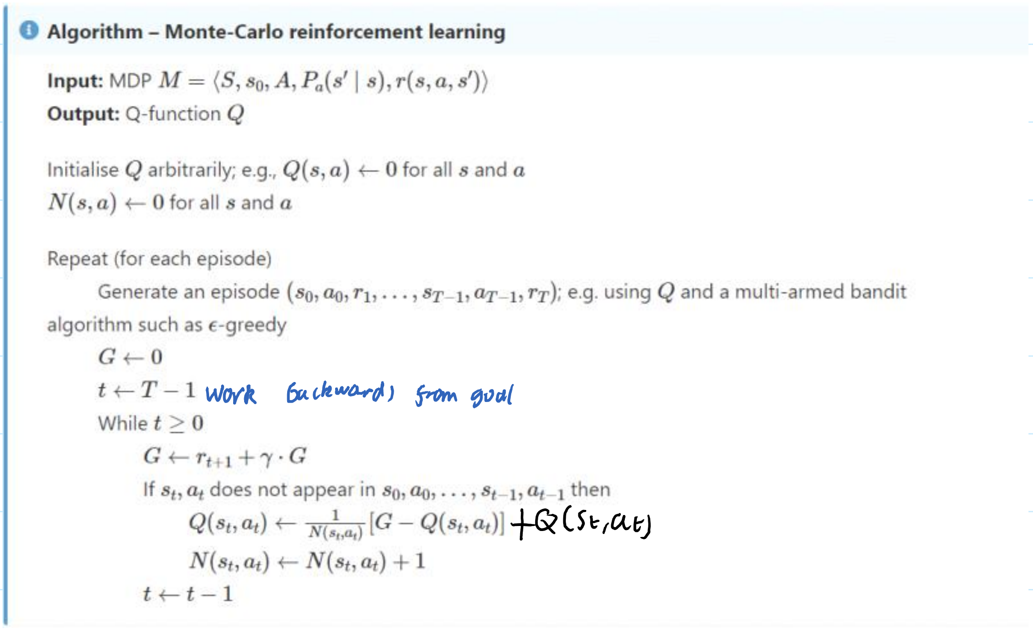
Episode: A 'playthrough' of the situation from initial state to goal state

Monte-Carlo Reinforcement Learning

Learns Q-values for each state, action pair over multiple episodes without knowledge of transitions, probabilities or rewards.

Idea: During an episode, we choose an action to take from the current state, which results in a new state and reward. Keep track of the order of states, actions, and rewards. At the end of the episode, update $Q(s,a)$ values to be the average accumulated future discounted reward (G) observed from taking action a in state s.

Note: The algorithm updates Q values backwards. However, we only update the Q value for each state, action pair once per episode; the first time we see $Q(s,a)$ during the episode i.e the last time we see $Q(s,t)$ during the algorithm.



Algorithm – Monte-Carlo reinforcement learning

```
Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s,a,s') \rangle$
Output: Q-function $Q$

Initialise $Q$ arbitrarily; e.g., $Q(s,a) \leftarrow 0$ for all $s$ and $a$
$N(s,a) \leftarrow 0$ for all $s$ and $a$

Repeat (for each episode)
    Generate an episode $(s_0, a_0, r_1, \ldots, s_{T-1}, a_{T-1}, r_T)$; e.g. using $Q$ and a multi-armed bandit algorithm such as $\epsilon$-greedy
    $G \leftarrow 0$
    $t \leftarrow T-1$ work backwards from goal
    While $t \geq 0$
        $G \leftarrow r_{t+1} + \gamma \cdot G$
        If $s_t, a_t$ does not appear in $s_0, a_0, \ldots, s_{t-1}, a_{t-1}$ then
            $Q(s_t,a_t) \leftarrow \frac{1}{N(s_t,a_t)}[G - Q(s_t,a_t)] + Q(s_t,a_t)$
            $N(s_t,a_t) \leftarrow N(s_t,a_t) + 1$
        $t \leftarrow t-1$
```

Cons of MCRL

- Only update values after entire episode completed
- High variance because each update depends on all future actions in episode. (Action choices are a random process)

Solution? Update Q values faster by basing them on the value of the next state

Temporal Differencing

Update Q value based on future state value

$$Q(s,a) \leftarrow \underbrace{Q(s,a)}_{\text{old value}} + \overbrace{\alpha}^{\text{learning rate}} \cdot \underbrace{[\overbrace{r}^{\text{reward}} + \overbrace{\gamma}^{\text{discount factor}} \cdot V(s')}_{\text{TD target}} - \overbrace{Q(s,a)}^{\text{do not count extra } Q(s,a)}]$$

2 approaches: Q-learning and SARSA

TD Algorithm

1) Initialise Q values arbitrarily

For each episode:

1) S = current state

2) Select action a and perform action

3) Observe new state s' and reward r

4) update value of $Q(s,a)$

5) repeat until end of episode

Update function

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

$\alpha \in [0,1]$: learning rate

$\gamma \in [0,1]$: discount factor

Q-learning: a' = action maximising $Q(s',a')$

SARSA : a' = next action chosen for state s' during step 2)

Q-Learning	SARSA
• Off-policy	• On policy
• Assumes next action chosen is best action	• Uses actual next action chosen
• Tends to result in more optimal policies	• Tends to result in safer policies
• Useful when possible to train in dummy environment	• Useful when have to train on deployment environment

N-Step Reinforcement Learning

Updates Q-value based on actions taken up to N steps in the future

- 1-Step RL same as Q-learning / SARSA
- ∞-Step RL same as Monte Carlo RL

Update function:

$$Q(s,a) = Q(s,a) + \alpha \times [R_n + \gamma^n \times Q(s_n, a_n) - Q(s,a)]$$

$$R_n = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \dots + \gamma^{n-1} r_{n-1}$$

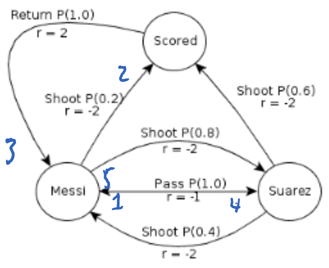
S_n : State reached n steps in the future

a_n : Max action (Q-learning) or policy action (SARSA) for state S_n

Increasing n increases speed that Q-values are learnt for more states.

However, increases variance

Eg) 3-step SARSA



Q-table

State	Pass	Shoot	Return
Messi	-0.4	-0.8	--
Suarez	-0.7	-0.2	--
Scored	--	--	1.2

$\alpha = 0.4 \quad \gamma = 0.9$

Given the following trace from a historical game feed from last season: "Suarez passes the ball to Messi, Messi dribbles around all of his opponents, shoots and scores yet another goal! Barcelona F.C 10 - 0 Real Madrid! The ball is returned to Messi for kickoff. After he passes the ball to Suarez, the referee blew the final whistle. End of the game, the ball is taken by Messi to remember the match forever."

Show the 3-step SARSA update for the above feed. Do you think the 1-step update is more accurate or the 3-step update? Does it indicate more steps is always better? Explain why.

For 3-step SARSA:

$$Q(s,a) = Q(s,a) + \alpha \times [R_3 + \gamma^3 Q(s_3, a_3) - Q(s,a)]$$

$$R_3 = r_0 + \gamma r_1 + \gamma^2 r_2$$

$$R_3 = -1 + 0.9 \times -2 + 0.9^2 \times 2 = -1.18$$

$$Q(s_3, a_3) = Q(\text{Messi}, \text{Pass}) = -0.4$$

$$Q(\text{Suarez}, \text{Pass}) = -0.7 + 0.4 \times [-1.18 + 0.9^3 \times -0.4 - (-0.7)]$$

$$= -0.7 + 0.4 \times -0.7716$$

$$= -1.00864$$