AIFA: REINFORCEMENT LEARNING

25/03/2025

Koustav Rudra

The notion of Q-Function

- One approach to RL is then to try to estimate $V^*(\mathrm{s})$
  - We may use the Bellman Equation:
    - $V^*(s) = \max_a [r(s,a) + \gamma V^*(\delta(s,a))]$

- However, this approach requires us to know $r(s,a)$ and $\delta(s,a)$
  - This is unrealistic in many real problems because the state space may not be known

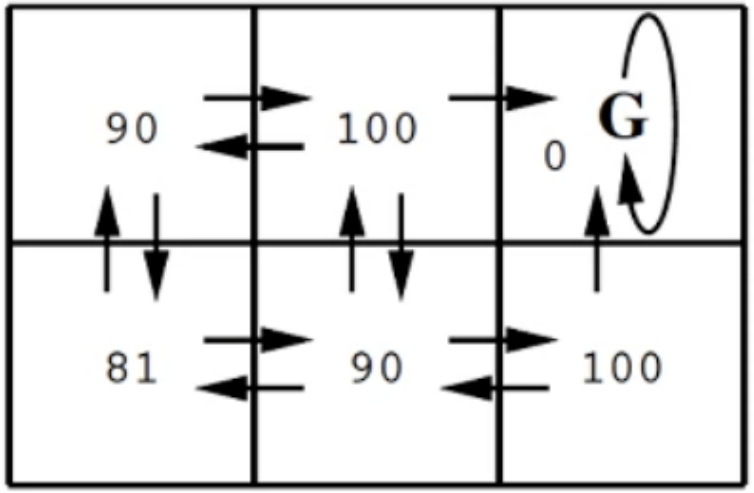
- This problem could be overcome by exploring and experiencing how the world reacts to our actions
  - We need to **learn values** of $r(s,a)$ and $\delta(s,a)$

The notion of Q-Function

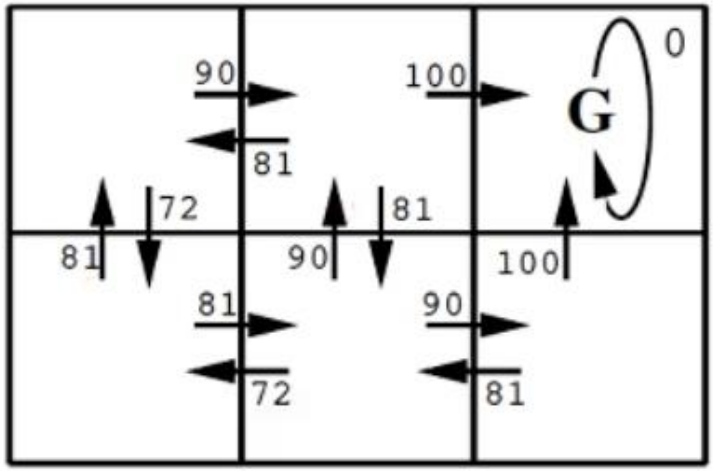
- We want a function that directly learns good state-action pairs, that is, what action should be taken in this state
  - Q(s,a)

- Given Q(s,a), it is trivial to evaluate to execute the optimal policy without knowing $r(s,a)$ and $\delta(s,a)$
  - $\pi^*(s) = argmax_a Q(s,a)$
  - $V^*(s) = max_a Q(s,a)$

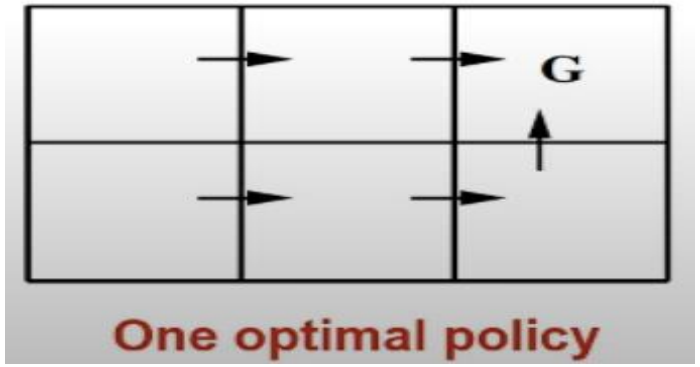
Example 2



V*(s) : values



Q(s,a) : values



- $\pi^*(s) = argmax_a Q(s,a)$
- $V^*(s) = max_a Q(s,a)$

Q-Learning

- $\pi^*(s) = argmax_a Q(s,a)$
- $V^*(s) = max_a Q(s,a)$

- $Q(s,a) = r(s,a) + \gamma V^*(\delta(s,a))$
- $Q(s,a) = r(s,a) + \gamma \max_{a'} Q(\delta(s,a), a')$

- Still depends on $r(s,a)$ and $\delta(s,a)$

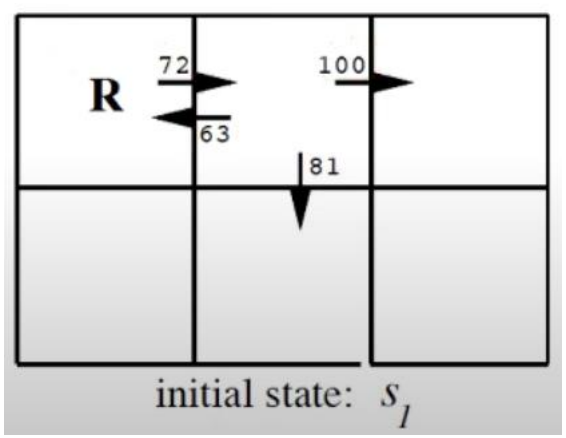
- $Imagine\ robot\ is\ exploring\ environment$
- At every step it receives reward "r" and it observes the environment changes to a new state s' for action a
- How can we use this observations (s, a, s', r) to learn a model?

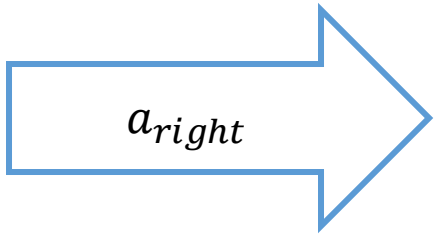
- $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$
- $s' = \delta(s,a)$

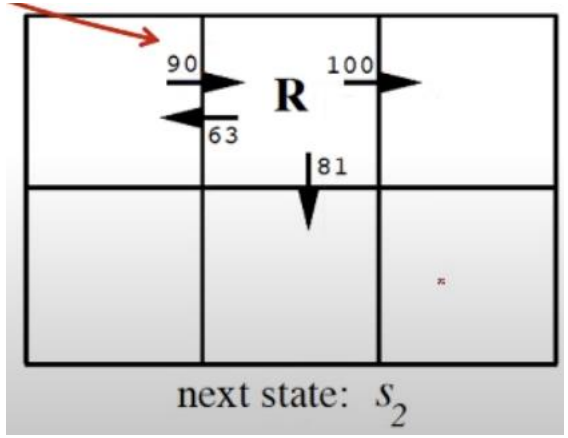
Q-Learning

- $\hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$
- This equation continually estimates Q at state s consistent with an estimate of Q at state s', one step in the future
- Note that s' is closer to the goal, and hence more reliable, but still an estimate itself
- Updating estimate based on other heuristics is called bootstrapping
- We do an update after each state-action pair, i.e., we are learning online
- We are learning useful things about explored state-action pairs
  - Useful because they are likely to be encountered again
- Under suitable conditions, these updates will converge to real value

Example: One step Q Learning







$$\hat{Q}(S_1, a_{right}) \leftarrow r + \gamma \max_{a'} \hat{Q}(S_2, a')$$

$$\hat{Q}(S_1, a_{right}) \leftarrow 0 + 0.9 \max_{a'} \{63,81,100\}$$

Limitations: Getting stuck in local optima

- It is very important that the agent does not simply follow the current policy when learning Q (off-policy learning)
  - The reason is that you may get stuck in a suboptimal solution
  - There may be other solutions out there that you have never seen

- Hence, it is good to try new things so now and then
  - For example, we could use something like:
  - $P(a|s) \propto e^{\frac{\widehat{Q}(s,a)}{T}}$
  - If T is large, lots of exploring
  - If T is small, follow current policy
  - We can decrease T over time

Improvements

- One can trade-off memory and computation by cashing (s, a, s', r) $for\ observed\ transitions$
  - After a while, as $Q(s', a') has\ changed$, you can "replay" the update
  - $\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$

Thank You