

Q Learning

COMP4211



THE DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

At state s , take action a , then follow policy π

- assume **deterministic** rewards and actions
- discounted cumulative reward:

$$r(s, a) + \gamma V^\pi(\delta(s, a)) \equiv Q^\pi(s, a)$$

- **action value function** for policy π

For the **optimal** policy, $Q^*(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$

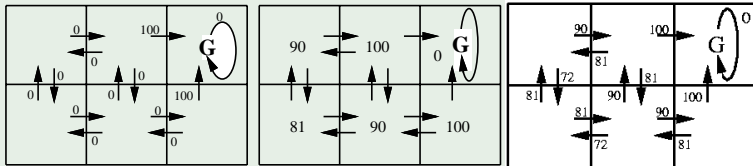
- **maximum** discounted cumulative reward

Optimal policy:

$$\pi^*(s) = \arg \max_a [r(s, a) + \gamma V^*(\delta(s, a))] = \arg \max_a Q^*(s, a)$$

Example

problem; state value function; action value function ($\gamma = 0.9$)

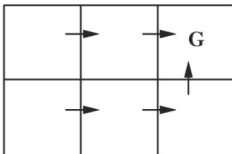


e.g., $(s, a) = (\text{bottom left state; move up})$

- $Q: 0 + \gamma \times 90 = 81$

e.g., $(s, a) = (\text{bottom right state; move up})$

- $Q: 100 + \gamma \times 0 = 100$



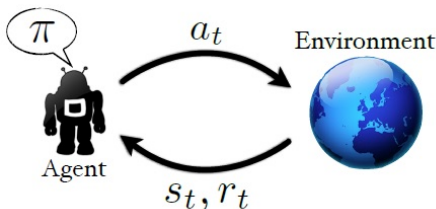
One optimal policy

Unknown Environment

For the optimal policy

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

What if the agent does **not** know δ and r ?



learn an **approximation** of Q^*

- if the agent knows Q^* , it can choose optimal action even **without** knowing r and δ !

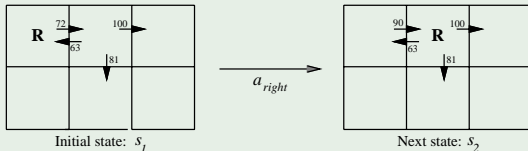
How to Approximate Q^* ?

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

Learn directly from **experience**:

- observe the current state s
- choose and execute an action a
- **observe** the resulting reward $r = r(s, a)$ and the new state $s' = \delta(s, a)$
 - does **not** need to know $\delta(s, a)$ and $r(s, a)$ **explicitly**
- update $\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$

Example (\hat{Q})



$$\begin{aligned}\hat{Q}(s_1, a_{right}) &\leftarrow r + \gamma \max_{a'} \hat{Q}(s_2, a') \\ &\leftarrow 0 + 0.9 \max \{63, 81, 100\} = 90\end{aligned}$$

begin

for each s, a ; initialize table entry $\hat{Q}(s, a) \leftarrow 0$
observe current state s ;

repeat

select an action a and execute it;

receive immediate reward r ;

observe the new state s' ;

update the table entry for $\hat{Q}(s, a)$ as:

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

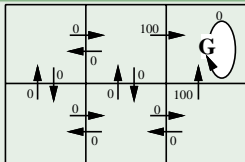
$s \leftarrow s'$;

until;

end

\hat{Q} converges to Q in the limit

Example (numbers are the rewards)



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

With all the \hat{Q} values initialized to zero

- the agent will make **no** changes to any \hat{Q} entry until it happens to reach the goal state and receive a **nonzero** reward
- refine the \hat{Q} value for the single transition leading into the goal state

On the next **episode**

- if the agent passes through this state adjacent to the goal state, its nonzero \hat{Q} value will allow refining the value for some transition two steps from the goal, and so on

Given a sufficient number of training episodes, the information will propagate through the entire state-action space

Non-Deterministic Rewards and Actions

Reward function $r(s, a)$ and action transition function $\delta(s, a)$ may have **probabilistic** outcomes

$$\begin{aligned} V^\pi(s) &\equiv E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots] \\ Q^\pi(s, a) &\equiv E[r(s, a) + \gamma V^\pi(\delta(s, a))] \\ Q^*(s, a) &= E[r(s, a) + \gamma V^*(\delta(s, a))] \\ &= E[r(s, a)] + \gamma \sum_{s'} P(s, s', a) V^*(s') \\ &= E[r(s, a)] + \gamma \sum_{s'} P(s, s', a) \max_{a'} Q^*(s', a') \end{aligned}$$

$$Q^*(s, a) = E[r(s, a)] + \gamma \sum_{s'} P(s, s', a) \max_{a'} Q^*(s', a')$$

Deterministic case:

$$Q(s_t, a_t) \leftarrow r_t + \gamma \max_a Q(s_{t+1}, a)$$

Non-deterministic case:

- a **nondeterministic** reward function will produce different rewards each time the transition (s_t, a_t) is repeated

$$Q(s_t, a_t) \leftarrow (1 - \alpha) Q(s_t, a_t) + \alpha [r_t + \gamma \max_a Q(s_{t+1}, a)]$$

- cf. perceptron learning
- the new $Q(s_t, a_t)$ is an average between $Q(s_t, a_t)$ and $r_t + \gamma \max_a Q(s_{t+1}, a)$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha [r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

- can still prove **convergence** to Q^*

How to set α ?

- as training progresses, reduce α

Example

$$\alpha_n = \frac{1}{1 + \text{visits}_n(s, a)}$$

- $\text{visits}_n(s, a)$: total number of times this state-action pair has been visited up to and including the n th iteration

Action Selection

begin

for each s, a ; initialize table entry $\hat{Q}(s, a) \leftarrow 0$

observe current state s ;

repeat

select an action a and execute it;

receive immediate reward r ;

observe the new state s' ;

update the table entry for $\hat{Q}(s, a)$ as;

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

$s \leftarrow s'$;

until convergence;

end

How to select actions when the agent is in state s ?

Greedy Action Selection

Obvious strategy:

- always select the action that looks **best**

$$\pi(s) = \arg \max_a \hat{Q}(s, a)$$

- **greedy** policy

Problem

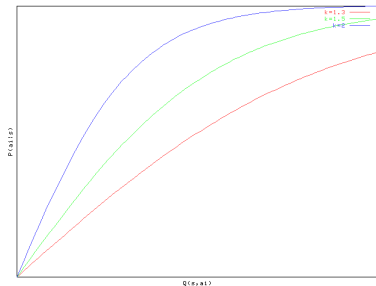
- over-commit to actions that are found during early training to have high \hat{Q} values
- fail to **explore** other actions that may have even higher values

Be less greedy!

Exploit and Explore

Select action a_i with probability

- $\pi(s, a_i) = k \hat{Q}(s, a_i) / \sum_j k \hat{Q}(s, a_j), \quad k > 0$



- exploit** actions that currently have high \hat{Q} values
- explore** actions that currently do not have high \hat{Q} values

ϵ -Greedy Policy

Alternatively

- be greedy most of the time (exploit)
- **occasionally** (with probability ϵ), take a **random** action (explore)
- surprising, this is the **state of the art**

