

# Motivation of Dynamic Programming Chapter

Everything in the chapter serves to answer this question:

given the full knowledge of an MDP (the entire table of  $p(s', r | s, a)$ ),  
how to find the optimal policy  $\pi$  that maximizes the cumulative reward?

# Understand the connection between state-value function $v(s)$ and action-value function $q(s, a)$

explore  
their  
relationships

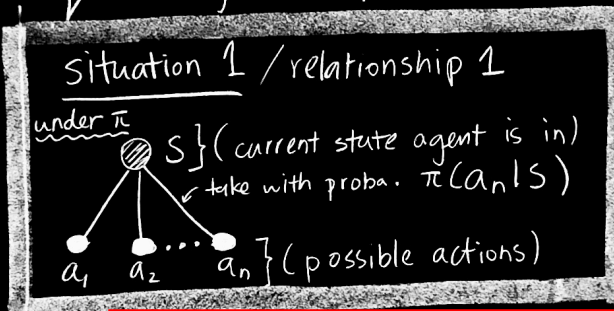
using "backup" diagrams

State-value function  
(for policy  $\pi$ )

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s]$$

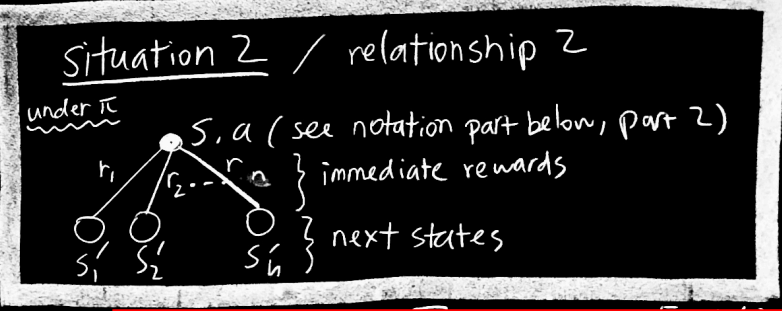
State-action-value function  
(for policy  $\pi$ )

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$$



$$\Rightarrow v_{\pi}(s) = \sum_{a \in A(s)} \pi(a|s) q_{\pi}(s, a)$$

(by going from top to bottom of the above diagram "actions possible in state S")



$$\Rightarrow q_{\pi}(s, a) = \sum_{s'} \sum_r p(s', r | s, a) [v_{\pi}(s') + r]$$

(by going from top to bottom of the above diagram)

Since one  $s'$  can be paired with multiple different  $r$ 's and vice versa

future cumulative reward (points to  $v_{\pi}(s')$ )

immediate reward (points to  $r$ )

Notations in the 2 above diagrams:

- 1)  $\textcircled{\bullet}$ , a state (current) /  $\bigcirc$ , future state
- 2)  $\bullet$ , an action (after an action is taken in a state, we have a state-action pair)

We have found the true value of all states or all state-action pairs when the two equations in red boxes are true for all states / all state-action pairs. Both of these come from the definition of "value", the cumulative reward onwards. For example, the equation in the left red box describes that the cumulative reward from  $s$  onwards should equal the weighted sum of the values of possible actions (when the agent is in that state). This definition is justified by our daily experience as agents.

# Policy Evaluation Algorithm For Full-MDP

later on.

① Policy Evaluation  
 Problem Statement [given a  $\pi$   $\xrightarrow{\text{find}}$   $V_\pi: S \rightarrow \mathbb{R}$ ], we'll see why this is useful  
(we don't know if this is optimal)

Can be done using the iterative algorithm (proven to converge)

$$S^+ = \{S, s_{\text{terminal}}\}$$

- input: a policy,  $\pi$
- initialize  $\theta > 0$ ,  $V(s)$  for all  $s \in S^+$  ( $V(s_{\text{terminal}}) = 0$ )
- while True:  $\Delta = 0$   $\rightarrow$  precision parameter

for  $s$  in  $S^+$ :

$$v = V(s)$$

$$V(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma V(s')]$$

$$\Delta \leftarrow \max(\Delta, |v - V(s)|)$$

if  $\Delta < \theta$ :  
break

value of  
state  $s$

value of  
state  $s$   
by one-step  
look-ahead  
(weight each  $(s', r)$   
by probability  $p(s', r|s, a)$ )

By definition, these 2 quantities  
should be exactly equal when  
 $V$  is the true  $V_\pi$ .

$\rightarrow$  Bellman's equation for  $V_\pi(s)$

$$\underline{\underline{V_\pi(s)}} = \sum_a \pi(a|s) \sum_{s', r} p(s', r|s, a) [r + \gamma \underline{\underline{V_\pi(s')}}]$$

# Policy Improvement

---

Sutton 2018, Chapter 4, Dynamic Programming

Zhihan Yang

## Policy improvement algorithm

---

The policy improvement algorithm happens after each iteration of policy iteration and can be summed up very concisely:

1. For each accessible state, sum up the immediate reward of arriving in that state and the value of that state (expected cumulative reward of being in that state and onwards).
2. Choose the action that take yourself to the accessible state with the highest sum. This is called **greedy action selection**.

However, the most important question is why this simple algorithm works; we'll answer this by proving the convergence of this algorithm (to the optimal value function and the optimal policy) when used in combination with policy evaluation (which we discussed in the previous session).

## Proof

---

The second step of the algorithm updates the old policy such that now the new policy  $\pi(a|s)$  gives the highest probability to the action  $a$  that maximizes  $q_\pi(s, a)$  - the highest value of  $\pi(a|s)$  and the highest value of  $q_\pi(s, a)$  now occur for the same  $a$ . By understanding this alignment and inspecting equation 1 (expressing  $v_\pi$  in terms of  $q_\pi$ ), it becomes clear why  $V_{\text{new}}(s) \geq V_{\text{old}}(s)$ .

(Recall that the value of a state  $v_\pi$  is related to the value of its accessible states  $q_\pi(s, a)$  by the following relationship.)

$$v_\pi(s) = \sum_a \pi(a|s) q_\pi(s, a) \quad (1)$$

There are two cases that allow  $V_{\text{new}}(s) \geq V_{\text{old}}(s)$ :

1.  $V_{\text{new}}(s) = V_{\text{old}}(s)$ 
  - This is the Bellman's optimality equation, which indicates that both  $V$  are already optimal.
2.  $V_{\text{new}}(s) > V_{\text{old}}(s)$ 
  - This is what happens otherwise.

Therefore, we see that  $V$  improves unless it is already optimal.