

# CS5180 Reinforcement Learning

## Exercise 3: Dynamic Programming

Anway Shirgaonkar

### Q1 Action Value Function

- (a) Give an equation for  $v_\pi$  in terms of  $q_\pi$  and  $\pi$

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

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

- (b) Give an equation for  $q_\pi$  in terms of  $v_\pi$  and the four-argument  $p$

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

- (c) What is the Bellman equation for action values, that is, for  $q_\pi$ ? It must give the action value  $q_\pi(s, a)$  in terms of the action values,  $q_\pi(s', a')$ , of possible successors to the state–action pair  $(s, a)$ .

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

$$q_\pi(s, a) = \mathbb{E}_\pi(R_{t+1} + \gamma G_{t+1} | S_t = s, A_t = a)$$

$$q_\pi(s, a) = \mathbb{E}_\pi(R_{t+1} | s, a) + \gamma \mathbb{E}_\pi(G_{t+1} | s, a)$$

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

## Q2 Fun with Bellman

- (a) Give an equation for  $v_*$  in terms of  $q_*$

$$v_*(s) = \max_a (q_*(s, a))$$

- (b) Give an equation for  $q_*$  in terms of  $v_*$  and the four-argument  $p$

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

- (c) Give an equation for  $\pi_*$  in terms of  $q_*$

$$\pi_*(s) = \operatorname{argmax}_a q_*(s, a)$$

- (d) Give an equation for  $\pi_*$  in terms of  $v_*$  and the four-argument  $p$

$$\pi_*(s) = \operatorname{argmax}_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')]$$

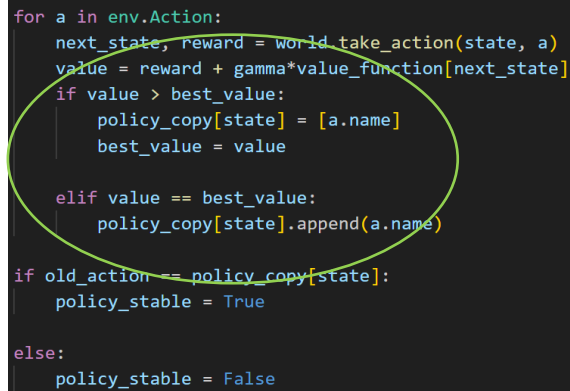
- (e) Rewrite the four Bellman equations for the four value functions ( $v_\pi, v_*, q_\pi, q_*$ ) in terms of the three-argument function  $p$

$$\begin{aligned} v_\pi(s) &= \sum_a \pi(a|s) \sum_{s'} p(s'|s, a) [r(s, a) + \gamma v_\pi(s')] \\ v_*(s) &= \max_a \sum_{s'} p(s'|s, a) [r(s, a) + \gamma v_*(s')] \\ q_\pi(s, a) &= \sum_{s'} p(s'|s, a) [r(s, a) + \gamma \sum_{a'} \pi(a'|s') q_\pi(s', a')] \\ q_*(s, a) &= \sum_{s'} p(s'|s, a) [r(s, a) + \gamma \max_{a'} q_*(s', a')] \end{aligned}$$

### Q3 Fixing Policy Iteration

(a) Yes, the algorithm won't terminate if the policy keeps switching between two or more policies which are equally good. There are a couple of ways to tackle this issue.

- How I have implemented policy iteration for the 5x5 world is that the algorithm stores **ALL** the actions which are optimal corresponding to a state. In this way, we can guarantee that the algorithm terminates. It is demonstrated in the picture below



```
for a in env.Action:
    next_state, reward = world.take_action(state, a)
    value = reward + gamma*value_function[next_state]
    if value > best_value:
        policy_copy[state] = [a.name]
        best_value = value

    elif value == best_value:
        policy_copy[state].append(a.name)

if old_action == policy_copy[state]:
    policy_stable = True
else:
    policy_stable = False
```

- Secondly, we can always select the first optimal action while iterating through the loop of actions. Even if there is another optimal action, it would be ignored unless its value is greater than the previous action

(b) No, such a bug does not exist in value iteration.

- For each state, we assign its value by selecting the action which maximizes the return.
- Even if, for a state, we have two actions which will maximize the return, the value associated with that state would be equal
- Since the algorithm terminates based on  $\Delta$ , which measures the change in the value of a state after one sweep, it does not matter which optimal action we choose.
- Hence the value iteration would always terminate

## Q4 Policy Iteration for Action Values

(a) Pseudocode for policy iteration on action values.

**First, arbitrarily initialize the action value function and the policy, and set the threshold**

$Q(s, a) \in R$  and  $\pi(s) \in A(s) \forall s \in S, a \in A$

$\theta \leftarrow \text{threshold}$

**For policy evaluation:**

*Loop:*

$\Delta \leftarrow 0$

*Loop for each  $s \in S$  and  $a \in A$ :*

$q = Q(s, a)$

$Q(s, a) \leftarrow \sum_{s', r} p(s', r | s, a) [r + \gamma \sum_{a'} \pi(a' | s') Q(s', a')]$

$\Delta \leftarrow \max(\Delta, |q - Q(s, a)|)$

*until  $\Delta < \theta$*

**For policy Improvement:**

$\text{policy\_stable} \leftarrow \text{true}$

*for each  $s \in S$  and  $a \in A$ :*

$\text{old\_action} \leftarrow \pi(s)$

$\pi(s) \leftarrow \underset{a}{\operatorname{argmax}} Q(s, a)$

*if  $\text{old\_action} \neq \pi(s)$ , then  $\text{policy\_stable} = \text{false}$*

*if  $\text{policy\_stable}$ , then stop and return  $Q \approx q_*$  and  $\pi \approx \pi_*$ ; else go to policy evaluation*

(b) Analog of the value iteration update for  $q_{k+1}(s, a)$ :

$$q_{k+1}(s, a) = \sum_{s'} p(s', r | s, a) [r + \gamma \max_{a'} q_k(s', a')]$$