

Policies

In reinforcement learning, the main goal is to maximize the return value. The agent's task is to select an action which produces a reward and the choice of this action may influence the reward both immediately and also in the long run.

In an RL system, the actions are selected by following a policy. A policy maps a state to an action. This kind of policy is called the deterministic policy and is denoted by the symbol π .
i.e. one action

$$\text{ie } \pi(a|s)$$

This gives the probability of selecting action a given the state s .

If multiple actions can be selected with ≥ 0 probability the policy is called a stochastic policy. In this case, since there can be actions with multiple probabilities

$$\sum_{a \in A} \pi(a|s) = 1.$$

For policies also, the choice of action for MDP policies must only depend on the current state and not on any other state before. This shouldn't be thought of as a limitation of an MDP policy but more of a condition to be satisfied by the current state.

Value Functions

The received rewards capture how good an action was in that state. Focussing on maximizing this immediate reward may not be ideal and it's better to take all future rewards into account.

To measure future rewards, we have 2 measures:-

- ① Value Function or State Value Functions
- ② Action Value Function

① State Value Function :- State value function is the expected reward the agent will receive in all future states if it follows the policy π .
↳ after the current state

It is denoted as :-

$$v_{\pi}(s) = \sum_{\pi} [G_t | S_t = s]$$

② Action value function :- This is the total expected reward the agent is expected to receive after state s_t if it takes action a_t at s_t and follows the policy π after that.

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

We can use both these to evaluate how good or bad a policy or action is at a state s .

For example :- we can find the action value functions for various actions at time t and following various policies after that. This way we can select the action and policy with the maximum expected return.

Note however that this is very computationally intensive.

Bellman Equations

If the agent learns only by observing rewards, the problem of RL wouldn't be very generalizable. The agent will not be able to generalize its learnings from one area to other areas. For example, a person may fall while walking and hitting a rock and then falling. The agent will learn from this but the learning will be very specific to this situation. If in future, the agent is travelling on a bike and again encounters a rock, this will be a like new situation for the agent as it wouldn't have received any reward for this in the past. So it would need to be explicitly trained on every situation. This is clearly not ideal.

Bellman's =ns solve this problem. They enable the agent to learn from the current state and relate it to the value of future states without actually observing them.

We have Bellmann =ns for State value functions and also for action value functions.

① Bellman's =n for State value function.

We know that the state value is given as :-

$$v_{\pi}(s) = E_{\pi}[G_t | s_t = s]$$

$$= E_{\pi}[r_{t+1} + \gamma G_{t+1}]$$

This can be written as :-

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma E_{\pi}[G_{t+1} | s_{t+1} = s']]$$

expand over sum
of probabilities
of choosing actions
which was
chosen → a green state = s

$$= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r | s, a) [r + \gamma \left[\sum_a \pi(a'|s) \sum_{s''} \sum_{r'} p(s'', r' | s', a') \right. \\ \left. [r' + \gamma E_{\pi}[G_{t+2} | s_{t+2} = s'']] \right]$$

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

② Bellman's =ns for action value functions

We can //ly also derive =n for action value function.

We know that

$$q_{\pi}(s, a) = E_{\pi}(G_t | s_t = s, a_t = a)$$

$$\begin{aligned}
 &= \sum_{s'} \sum_a p(s', r | s, a) \left[r + \gamma E_{\pi}[G_{t+1} | S_{t+1} = s'] \right] \\
 \text{Need only this} \\
 \text{as the action is fixed} \\
 &= \sum_{s'} \sum_a p(s', r | s, a) \left[r + \gamma \sum_{a'} \pi(a' | s') E_{\pi}[G_{t+1} | S_{t+1} = s' \right. \\
 &\quad \left. A_{t+1} = a'] \right]
 \end{aligned}$$

We need to change above =n to make it of the format of q_{π} .

So we weigh it with all actions and the probability of selecting action a' so that current state is s' to make it reducible to $q_{\pi}(s, a)$

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

So how are they useful?

Bellman =ns reduce the problem of computing state or state action values to a simple algebraic problem of finding values satisfying =ns. [See video on how] \rightarrow rather than computing them infinitely in future

Without Bellmann =ns, one will have to compute an infinite sum to find state values or state action values. Bellman =ns thus provide a simple generalizable way to find state or state action values.

As the problem starts getting more complex, the number of =ns to be solved rises very quickly. For chess for eg $2^{45} = ns$ need to be solved. They are not only hard to solve but even hard to write. We now see methods to use Bellman =ns to estimate state or state action values.

Optimal Policies

Optimal policies are those which follow the max values for value functions. For an RL problem, we can have

Optimal policies are those which if followed return the max values for value functions. For an RL problem, we can have many policies and one of our job is to find the optimal policy.

For every problem, optimal policy exists. We can have a policy a which is better at some times and a policy b which is better at others. If we create a new policy c , which behaves like a when a is best and like b when it is, we get a new policy which is better everytime.

We can use the brute force approach to find the best policy however it can only be done for simple problems. For more complex problems, we need a better method.

$$\pi_1 \geq \pi_2 \text{ iff } v_{\pi_1}(s) \geq v_{\pi_2}(s) \quad \forall s \in S$$

Thus the optimal policy is one in which the value function is the greatest for all states.

We can write this as

$$v_{\pi^*}(s) = E_{\pi^*} [G_t | s_t = s] = \max_{\pi} v_{\pi}(s) \quad \forall s \in S$$

which means optimal policy has the max value function for every state for all the policies.

The optimal policy (s) have the same state value function denoted by v^* . The shared action value function is given by q^* .

Now we know that

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

This also holds for optimal value function.

\therefore We get

$$v^*(s) = \sum_a \pi^*(a|s) \sum_{s'} \sum_{r_e} p(s', r_e | s, a) [r_e + \gamma v^*(s')]$$

a s' r

Because it's the optimal policy, we can write it as

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

proved below

Notice this doesn't contain π_* . We can use v_* to find π_* . This is the Bellman optimality equation for Value functions.

In a similar way for state action value functions, we can derive :-

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

We can use $v_*(s)$ and $q_*(s)$ to find the optimal policy.

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

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

↑

Select the action with the highest value obtained in the above = a^* .