→ Markov Decision Process :-

It is one of the RL algo.

A MDP is a mathematical frame-work used in RL to model decision making problems.

→ It's like a game where an agent makes decisions & receives rewards & penalties based on those decisions.

→ The MDP consists of states, action, transition, & rewards.

→ The agent starts in a state position takes an action/decisions & transition to a new state & receiving a reward based on the action taken.

→ The goal is to find the optimal policy. (best solution).

→ The optimal policy is often denoted by $\pi^*(s)$.

Function that takes current state 's' as i/p and returns optimal action 'a' to take in that state.

→ Policy & Value function :-

In the context of Markov Decision Process (MDP), policy and value function are 2 concepts that help guide decision-making.

→ Policy :-

It is a simple strategy or a set of rule that tells you what actions to take in any given state.

It can be deterministic and stochastic.

Deterministic says for each state there is exactly one action to take.

Stochastic says for each state, there could be probability distribution over multiple possible actions.

For example :-

In a game, a policy might tell you to always jump when there's an obstacle.

In short policy answer the question "what should I do in this situation".

→ Value function :-

It estimates the expected return or value of taking an action in a state.

It evaluate the goodness or badness of a state or action.

- common types of value function :-

1) State value function :- (V_s)

It estimates the value of being in a state.

Ex: If you start at a particular position in a game, the value function gives you an estimated reward you can expect from that position.

2) Action-value function :- $Q(s,a)$

It estimates the value of taking an action in a state.

It is similar to state value function but focuses on specific action 'a' taken in a specific state 's'.

In simple policy determines the actions taken in each state & value function evaluates good / bad of those actions.

The goal is to learn a policy the maximize the value function.

Example :-
chess game.

→ ~~A policy might map the state opponent's king in check. to the action move my~~

→ The policy is your plan for what move to make next, depending on where you are on the board.

Ex: If your opponent's king is in danger, your policy might be to attack it. If you are losing pieces, your policy might be to defend.

- Value function is how good or bad your position is on the board. If you are close to checkmating your opponent your position has high value because you are likely to win. else you are at low value.

In general The policy is your move strategy
The value function is how good your current position is.

Reward models :-

In RL, reward models is a key part of how an agent learns from its interactions with the environment.

- There are several types of Reward models.

1) Infinite discounted reward model :-

It is a mathematical framework used in RL to evaluate total reward an agent can expect to receive over an infinite horizon.

- The agent will get rewards over time but the reward in the future are considered less valuable than received sooner. This is done using a technique called discounting.
- The agent maximize the sum of discounted reward over time. The agent values immediate rewards more than delayed rewards.
- Rewards in future are discounted with a factor called (gamma) (γ). a number blw 0 & 1.

- It helps agent to balance short term rewards with long term goal.

← Formula to calculate present value of a future reward is

$$P_{reward} = F_{Future\ Reward} \times \gamma^{t(time)}$$
(gamma)

2) Total reward Model:-

In RL, total reward refers to the sum of all rewards an agent received while interacting with the environment.

- For a given episode total reward is calculated by adding up all the rewards at each time step.
- Total reward equally values immediate & future rewards.
- In many RL problems future reward are discounted i,e agent values less important for future rewards.
- The discounted total reward is the sum of all rewards where each future reward is multiplied by a discount factor.

Formula:

$$R = \sum_{t=0}^{T} \gamma^t \cdot \gamma_t$$

γ^* → discount factor b/w 0 & 1.

γ_t → Reward at time

3) Finite Horizon Reward Model

In this model the agent aims to maximize the sum of rewards over a fixed finite-time horizon.

- The RL is a framework used in decision-making problems in RL.

Formula

$$\text{Total Reward} = \sum_{t=1}^{T} R_t$$

T → Total no. of time steps

R_t → reward received by an agent

4) Average reward model :-

It evaluate the avg reward an agent receives over an infinite horizon

the long-term avg reward received by the agent.

This process doesn't have clear end of fixed Horizon. The agent interacts with an environment continuously.

Formula :-

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} R_t$$

Date .....

Episodic & continuing tasks:-

In RL, tasks can be classified into 2 types.

1) Episodic 2) continuing.

Episodic task :-

These tasks have clear starting point and end point. I,e these tasks have clear begining and end.

- once the agent reaches the end, the task resets and a new episode starts.
- Think of a game where you start, play and finish the game. After finishing you "restart" the game for another round. . . . .

→ Its characteristics are : Finite horizon, clear terminal state & Agent learn's from episode to episode.

Ex: Playing a chess game where the episode ends when the game is won & lost.

Continuing tasks :-

These are the tasks that have no clear end or terminal state. It has no natural end point.

The agent just keep going forever & for as long as it can. There is no "reset" after each action. It just keeps trying to maximize its reward continuously.

Its characteristics are as follows.

Infinite horizon
No clear terminal state.
Agent learns continuously.

Ex: A robot that continuously move around a warehouse picking up items without a defined end to its task. It keeps learning and acting as long as its working.

⇒ Bellman's optimality operator (or) Bellman's Equation.

- It is a RL algo. It solves Markov Decision problem or process.
- It's a long-term reward in a given action is equal to the reward from the ~~combined~~ current action combined with expected reward for the future action.

Bellman eqn is a mathematical eqn that represent the relship btw the value of a state and the value of its possible next states.

Formula

$$V(s) = \max \left[R(s,a) + \gamma * V(s') \right]$$

(s: state)

$V(s)$ → Value of a state 's'

$R(s,a)$ → reward received for taking action a in state 's'.

γ → the discount factor. (0.9).

$V(s')$ → Value of a next state s'

Max → maximum value overall possible actions 'a'.

Bellman's eqn is a way of breaking down a big, complicated decision-making problem into smaller, simpler piece, where you can focus on the current decision and think abt how it leads to future rewards.

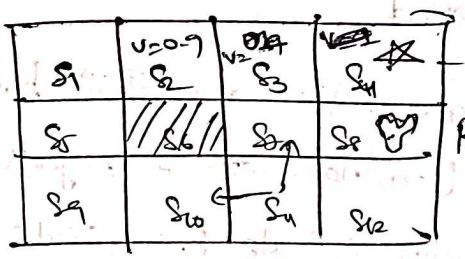
→ Bellman's optimality operator is a mathematical operator that helps to find the optimal solution to a problem by breaking it down into smaller sub problems.

→ Both Bellman's eqn & optimality operator are core concepts in RL & Dynamic programming.

Ex: Imagine you are playing a game where you can move left or right. you want to find the optimal path to the goal.

— Bellman eqn helps you calculate the value of each state. By considering this eqn repeatedly you can find the optimal path to the goal.

→ Consider a problem.



S6 → blocked, S8 → fire (-1)

S4 → Goal (Diamond). State (+1)

The agent can be any where from remaining states.

→ closest state is considered is S_3.

∴ Agent is in S_3. To achieve goal state agent can move towards right side.

$\therefore V(S_3) = \max[1 + 0.9 \times 0]$

$V(S_3) = \max[1 + 0.9]$

$\boxed{V(S_3) = 1.}$

→ $V(S_2) = \max[0 + 0.9 \times 1]$

$\boxed{V(S_2) = 0.9}$

→ $V(S_1) = \max[0 + 0.9 \times 0.9] = 0.81$

$V(S_5) = \max[0 + 0.9 \times 0.81] = 0.73$

$V(S_9) = \max[0 + 0.9 \times 0.73] = 0.66.$

$V(S_{10}) = \max[0 + 0.9 \times 0.66] = 0.59$

$V(S_{11}) = \max[0 + 0.9 \times 0.59] = 0.53.$

$V(S_4) = \max[0 + 0.9 \times 0.53] = 0.48.$

$V(S_7) = \max[0 + 0.9 \times 1] = 0.9$

$V(S_{11}) = \max[0 + 0.9 \times 0.9] = 0.81$ ← replace

process continues & maximum values are updated.

Using this formula we choose optimal path to reach goal state

⟹ Policy & Value iteration :-

Both are popular methods used to find optimal (best) policy for an agent to act in an environment.

- Both aims to solve RL problems but they work in slightly different way.

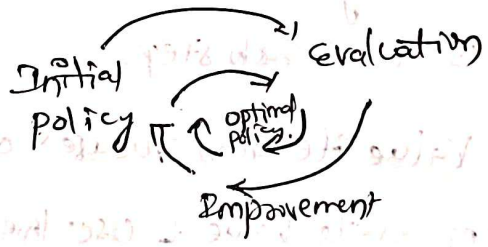
⟹ Policy iteration is a method to find optimal way in MDP.

This method work by improving the agent's current policy step by step until it becomes optimal. Just like in Bellman's optimization operator.

It starts with random policy & evaluate it & improve it & repeats until it becomes optimal.

Policy iteration = Policy Evaluation + Policy Improvement

∴ An initial policy can be repeatedly evaluated & improved until optimal policy is found.



→ Value Iteration :-

It is also finds optimal value function in an MDP.

This method works by refining the value of each state until you can figure out the best policy.

- It only does a single iteration for each state. It takes max action

Value - to be estimated state value.

→ once state values are converged to optimal state values then optimal policy can be achieved.

→ It start with random ~~policy~~ values & update the values & Repeats.

→ Both methods eventually find the best policy, but value iteration takes fewer steps & is more efficient.

→ The key difference is policy iteration focus on improving the policy directly and checks how good is it at each step.

while value iteration focuses on updating state value & use them to derive the optimal policy.