**Reinforcement Learning – Markov Decision Processes (MDPs)**

**1. Computing State Values using Bellman Equations**

In Reinforcement Learning, the value of a state $V_\pi(s)$ means the expected total reward an agent will get if it starts from state s and follows policy $\pi$ in the future.

The Bellman equation gives a recursive way to compute this value. It says that the value of a state depends on the rewards received and the values of the next states.

**Bellman Expectation Equation:**
$V_\pi(s) = \Sigma_a \pi(a|s) \Sigma_{s'} P(s'|s,a) [ R(s,a,s') + \gamma V_\pi(s') ]$

In simple words: from a state, we look at all possible actions, their probabilities, the rewards obtained, and the future values of next states.

**Matrix Form (Fixed Policy):**

When the policy is fixed, the MDP becomes a Markov Reward Process. The Bellman equation can be written as:

$V = R_\pi + \gamma P_\pi V$

Here: V is a vector of state values $R_\pi$ is the expected reward vector $P_\pi$ is the state transition probability matrix $\gamma$ is the discount factor ($0 \leq \gamma < 1$)
To find V, we rearrange the equation:

$(I - \gamma P_\pi)V = R_\pi$
$V = (I - \gamma P_\pi)^{-1} R_\pi$

This gives the exact state values for the given policy.

**2. Identifying Optimal Policies**

An optimal policy $\pi^*$ is one that gives the highest value for every state compared to any other policy.

**Step 1: Compute Optimal State Values**

We use the Bellman Optimality Equation:

$V^*(s) = \max_a \Sigma_{s'} P(s'|s,a) [ R(s,a,s') + \gamma V^*(s') ]$

This equation checks all actions and selects the best one. It is usually solved using Value Iteration.

**Step 2: Extract the Optimal Policy**

Once $V^*$ is known, the best action for each state is chosen as:

$\pi^*(s) = \arg\max_a \Sigma_{s'} P(s'|s,a) [ R(s,a,s') + \gamma V^*(s') ]$

This means we pick the action that gives the highest immediate reward plus future value.

**3. Defining MDPs for Common Tasks**

An MDP is defined using five components:
$(S, A, P, R, \gamma)$

**A. Taxi Driver Problem**

States: Taxi position on a 5×5 grid, passenger location, and destination. Total states are 500.
Actions: Move in four directions, pick up passenger, drop passenger.
Transitions: Mostly deterministic; hitting a wall keeps the taxi in the same position.
Rewards: -1 per step, +20 for correct drop, -10 for illegal actions.
Discount factor: 0.9 or 0.99.

**B. Game of Dice**

States: In game, End state.
Actions: Stay or Quit.
Transitions: Rolling the dice can end the game or continue.
Rewards: +10 for quitting, +4 for staying safely.
Discount factor: 1.0 or slightly less.

**C. Robot Navigation**

States: Grid positions (x, y).
Actions: Up, Down, Left, Right.
Transitions: 80% move as intended, 10% slip left, 10% slip right.
Rewards: +1 for goal, -1 for trap, -0.04 per step.
Discount factor: 0.9.

**Conclusion**

MDPs provide a mathematical way to model decision-making problems. Bellman equations help compute state values and identify optimal policies. These ideas are the foundation of Reinforcement Learning.