✅ 1. Passive Reinforcement Learning (10 Marks)

🔹 **Definition:**

Passive Reinforcement Learning is a method where the agent **follows a fixed policy** and **learns the utility of each state** without trying to improve the policy. The goal is **policy evaluation**, not policy improvement.

🔹 **Purpose:**

To evaluate how good a fixed policy is by estimating the **utility (expected long-term reward)** of each state under that policy.

🔹 **Working:**

- The agent does **not choose actions**.
- It follows the **given policy π**.
- Records transitions and rewards.
- Estimates utility U(s) using observed returns.

🔹 **Methods Used:**

- **Direct Utility Estimation**
- **Adaptive Dynamic Programming**
- **Temporal Difference Learning**

🔹 **Formula (if using TD(0)):**

🔹 **Formula (if using TD(0)):**

$$U(s) \leftarrow U(s) + \alpha[R(s) + \gamma U(s') - U(s)]$$

Where:

- $U(s)$: utility of state s
- α: learning rate
- γ: discount factor
- $R(s)$: immediate reward
- s': next state

🔹 **Example:**

In a grid world, the robot follows a path (e.g., always move right). Passive RL helps calculate expected rewards of being in each cell based on observed transitions and rewards.

🔹 **Advantages:**

- Simple to implement.
- Suitable for environments where policy is externally fixed.

🔹 **Disadvantages:**

- No exploration or learning of better policies.
- Learning is slow in large or stochastic environments.

✅ **2. Direct Utility Estimation (DUE) (10 Marks)**

🔹 **Definition:**

Direct Utility Estimation is a **model-free technique** to estimate the utility of states by **averaging total returns** from each time a state is visited, under a fixed policy.

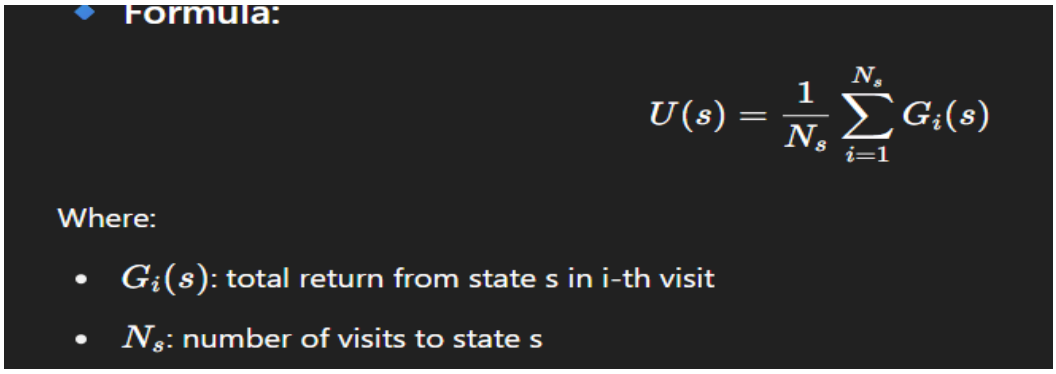
🔹 **Objective:**

To compute utility U(s)directly using sample episodes instead of solving Bellman equations.

🔹 **Steps:**

1. Agent follows a fixed policy π\piπ.
2. Collects total return GGG from each episode.
3. For each state S, keep a list of total returns observed.
4. Compute the average of all returns for each state.

🔹 **Formula:**



🔹 **Characteristics:**

- Purely empirical (based on experience).
- Doesn’t need knowledge of environment dynamics.

🔹 **Example:**

🔹 Example:

If a robot visits state A 3 times and gets total rewards of 5, 8, and 7, then:

$$U(A) = \frac{5+8+7}{3} = 6.67$$

- **Advantages:**
  - Easy to implement.
  - Doesn't require transition or reward model.
- **Disadvantages:**
  - High variance due to randomness.
  - Requires many episodes for accurate estimates.
  - Not efficient for rare or less visited states.

✅ 3. Adaptive Dyna (10 Marks)

- **Definition:**

Adaptive Dyna is a **hybrid RL technique** that learns from both real experiences and simulated experiences generated from a **learned model of the environment**.

- **Objective:**

To improve learning speed by combining **model-free learning** with **model-based planning**.

- **Components:**
  1. **Model Learning**: Learns transition probabilities P(s'|s,a)and rewards R(s,a).
  2. **Real Experience**: Updates value function using real transitions.
  3. **Simulated Experience**: Uses the learned model to simulate extra transitions and perform planning.
- **Algorithm Steps:**
  1. Agent takes action and observes (s, a, r, s').
  2. Updates value function using real data.
  3. Updates model: P and R.
  4. Simulates multiple (s, a) pairs using the model.
  5. Applies value or policy iteration to improve utility/policy.

🔹 **Key Idea:**

"**Learn while planning and plan while learning**" to improve efficiency.

🔹 **Example:**

A self-driving car learns from driving on real roads and also uses a simulator (built from real data) to practice on virtual roads.

🔹 **Advantages:**

- Much faster learning than pure model-free methods.
- Efficient use of available experience.
- Can adapt quickly to dynamic environments.

🔹 **Disadvantages:**

- Requires memory and computation to store and use the model.
- Complex to implement compared to simple methods.

✅ **Active Reinforcement Learning (10 Marks)**

🔹 **Definition:**

Active Reinforcement Learning is a learning approach where the agent is **not given a fixed policy**, but instead must **learn the best policy** by **interacting with the environment**, **choosing its own actions**, and **balancing exploration with exploitation**.

The agent must discover:

- Which actions yield the highest long-term rewards.
- How to trade off between trying new actions (exploration) and using known good actions (exploitation).

🔹 Goal:

To find an **optimal policy** π^* that maximizes the **expected cumulative reward** over time.

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$$

Where:

- π: policy (mapping from states to actions)
- γ: discount factor (0 < γ < 1)
- r_t: reward at time t

🔹 How it Works:

Unlike passive RL (which follows a given policy), in active RL:

1. The agent starts with **no knowledge** of which actions are best.
2. It **explores** the environment by taking different actions.
3. It **learns** from the consequences (state transitions and rewards).
4. Over time, it builds knowledge about the environment and improves its policy.

🔹 Components of Active RL:

Component	Description
States (S)	All possible situations the agent can be in.
Actions (A)	Choices available to the agent in each state.
Rewards (R)	Feedback signal received after an action.
Policy (π)	Strategy that maps states to actions.
Value Function	Measures expected future rewards from a state or action.
Model (optional)	Some methods use a model of environment transitions and rewards.

🔹 Exploration vs. Exploitation:

One of the biggest challenges in active RL:

- **Exploration**: Try new actions to discover their effects.

- **Exploitation**: Use known best actions to maximize reward.

🔸 Common strategy: **ε-greedy**

- With probability **ε**, explore (random action).
- With probability **1−ε**, exploit (best known action).

🔹 **Algorithms Used in Active RL:**

Algorithm	Type	Description
Q-Learning	Model-free	Learns value of state-action pairs.
SARSA	Model-free	Learns action values using actions taken.
Dyna-Q	Model-based	Combines real and simulated experiences.
Policy Gradient	Model-free	Directly optimizes the policy function.

🔹 **Example:**

In a maze game, the agent must figure out how to reach the goal. It is not told which way to go, but must try different paths. Over time, it learns which paths lead to faster rewards and forms an optimal policy.

🔹 **Advantages:**

- Can **learn optimal policies** from scratch.
- Adapts to **changing environments**.
- Suitable when no model of the environment exists.

🔹 **Disadvantages:**

- **Slower learning** in large or continuous spaces.
- Balancing exploration and exploitation is challenging.
- May require many **interactions (episodes)** for convergence.

🔹 **Real-world Applications:**

- Game playing (e.g., AlphaGo, Chess)
- Robotics (path planning, motion control)
- Autonomous vehicles
- Dynamic pricing in e-commerce

- Smart grid energy optimization

The difference between Q-learning and Temporal Difference Learning based on a few aspects is tabulated below –

Aspect	Temporal Difference (TD) Learning	Q-Learning
Objective	Estimates **state-value function** $V(s)$	Estimates **action-value function** $Q(s, a)$
Type of Algorithm	State values $V(s)$	Action-state values $Q(s, a)$
Policy Type	Model-free, on-policy or off-policy reinforcement learning	Model-free, off-policy reinforcement learning.
Update Rule	Updates based on the next state's value (for state-value)	Update based on maximum future action-value (for Q-function)
Update Formula	$V(s_t) \leftarrow V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]$	$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)]$
Exploration vs Exploitation	Directly follows the exploration-exploitation trade-off of the current policy like epsilon-greedy.	Separates exploration through epsilon-greedy from learning the optimal policy
Type of Learning	**Model-free**, learns from experience and bootstraps off of value estimates	**Model-free**, learns from experience and aims to optimize the policy
Convergence	Converges to a good approximation of the state-value function $V(s)$	Converges to the optimal policy if enough exploration is done
Example Algorithms	TD(0), SARSA	Q-learning

What is Temporal Difference Learning?