UNIT NO-5
Reinforcement Learning

Mr. R. S. Mirajkar

DYPCET, Kolhapur

Reinforcement Learning

- Feedback-based Machine learning technique
- Agent learns to behave in an **environment by performing the actions and seeing the results of actions**.
- For each good action, the **agent gets positive feedback**
- For each bad action, the agent gets **negative feedback or penalty.**

Reinforcement Learning

- In Reinforcement Learning, the agent learns automatically using feedbacks without any labeled data, unlike supervised learning.

- Since there is no labeled data, so the agent is bound to learn by its experience only.

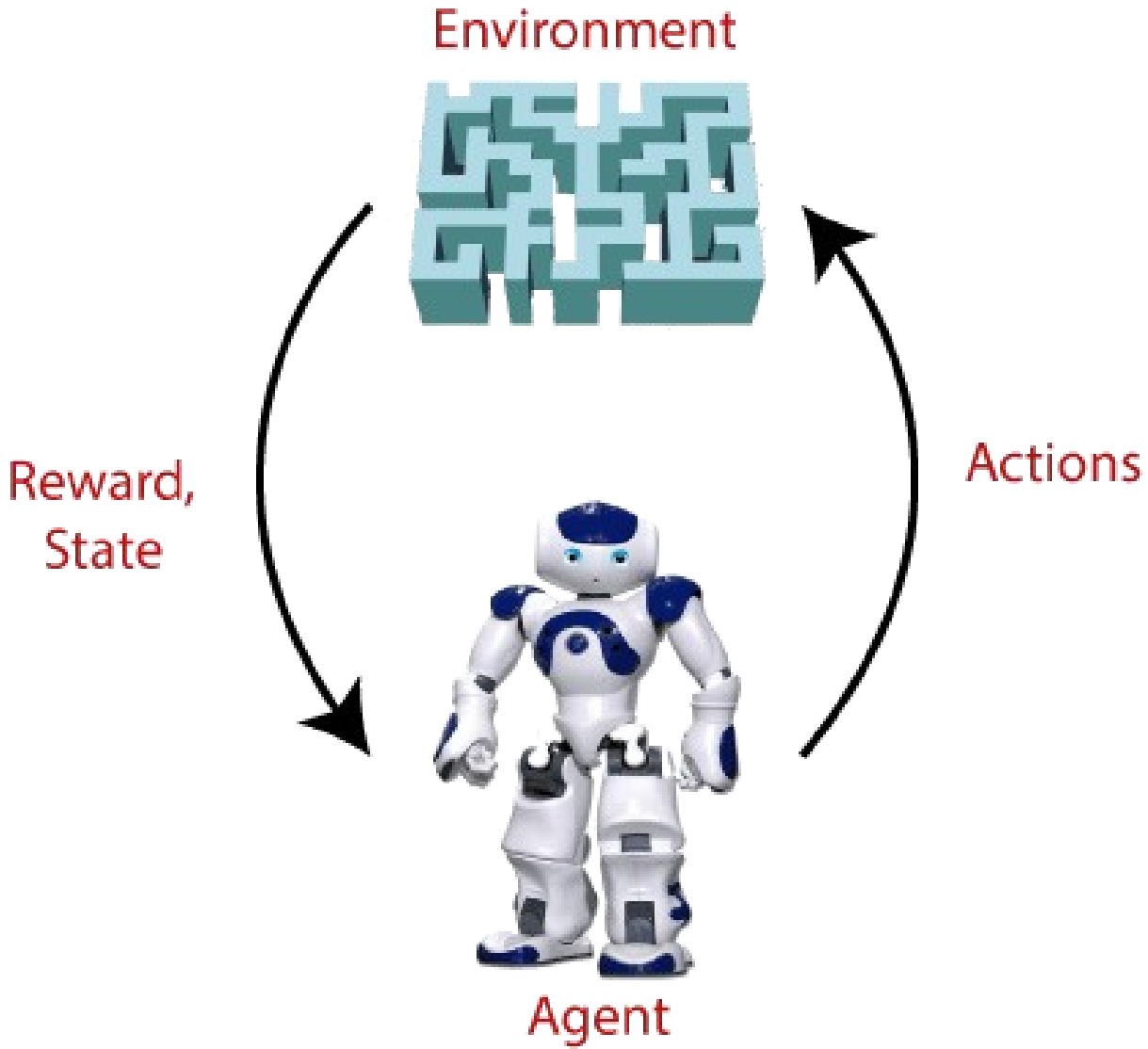
Reinforcement Learning?

- RL solves a specific type of problem where **decision making is sequential, and the goal is long-term**
- **Ex. game-playing, robotics**, etc.
- The agent interacts with the environment and explores it by itself.
- The primary goal of an agent in reinforcement learning is to improve the performance by getting the maximum positive rewards.

Reinforcement Learning

- The agent learns with the process of **hit and trial**, and based on the experience, it learns to perform the task in a better way.

- ***Reinforcement learning is a type of machine learning method where an intelligent agent (computer program) interacts with the environment and learns to act within that."***


Environment
Reward,
State
Actions
Agent

Terms used in Reinforcement Learning

- **Agent():** An entity that can perceive/explore the environment and act upon it.
- **Environment():** A situation in which an agent is present or surrounded by.
- In RL, we assume the stochastic environment, which means it is random in nature.
- **Action():** Actions are the moves taken by an agent within the environment.
- **State():** State is a situation returned by the environment after each action taken by the agent.

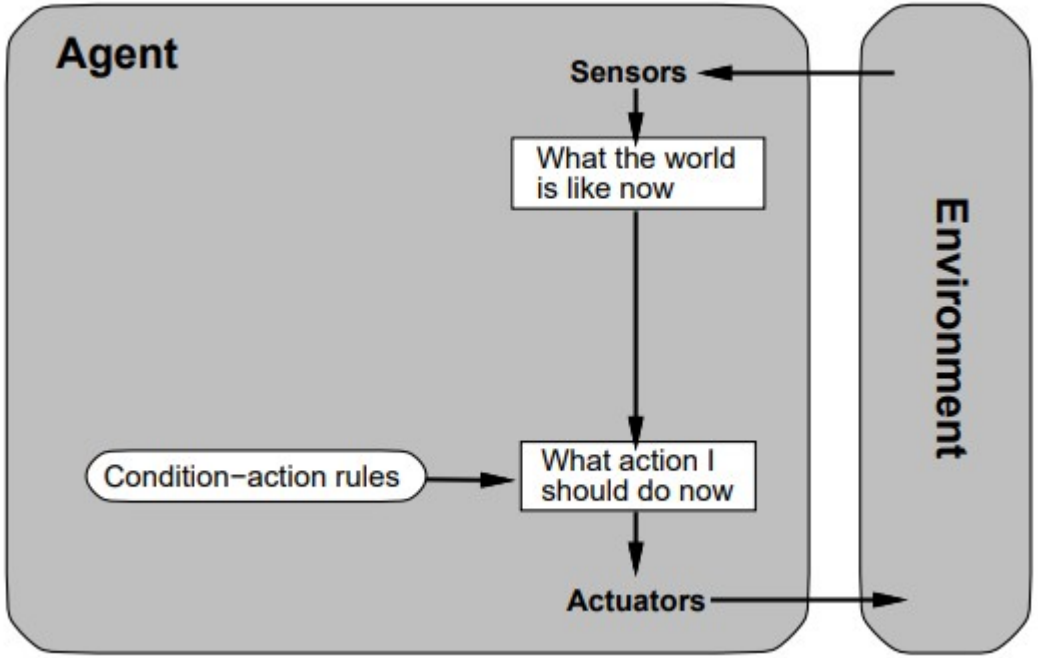
Terms used in Reinforcement Learning

- **Reward():** A feedback returned to the agent from the environment to evaluate the action of the agent.
- **Policy():** Policy is a strategy applied by the agent for the next action based on the current state.
- **Value():** It is expected **long-term retuned with the discount factor** and opposite to the short-term reward.

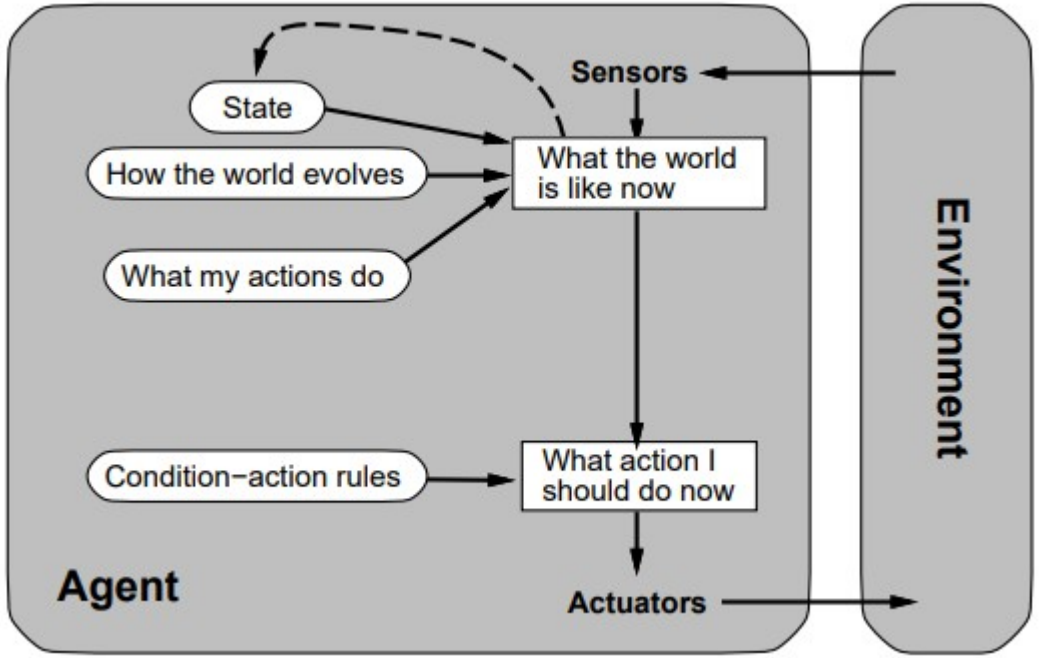
Key Features of RL

- In RL, the agent is not instructed about the environment and what actions need to be taken.
- It is based on the **hit and trial process.**
- The agent takes the **next action and changes states according to the feedback of the previous action.**
- The agent may get a **delayed reward.**
- The agent needs to explore to reach to get the maximum positive rewards.

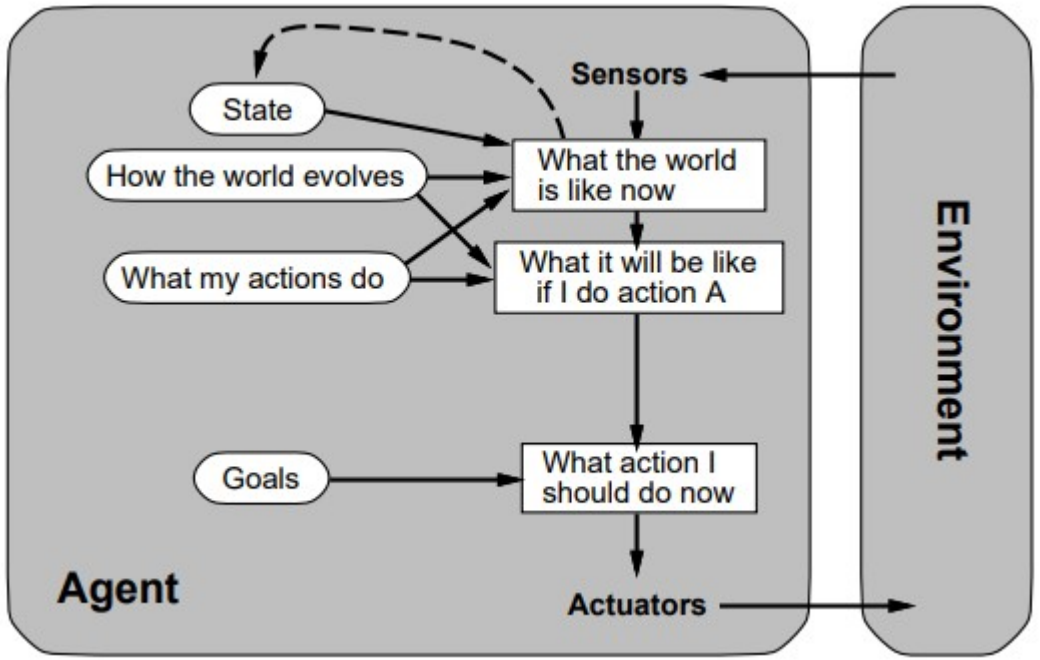
Simple Agent



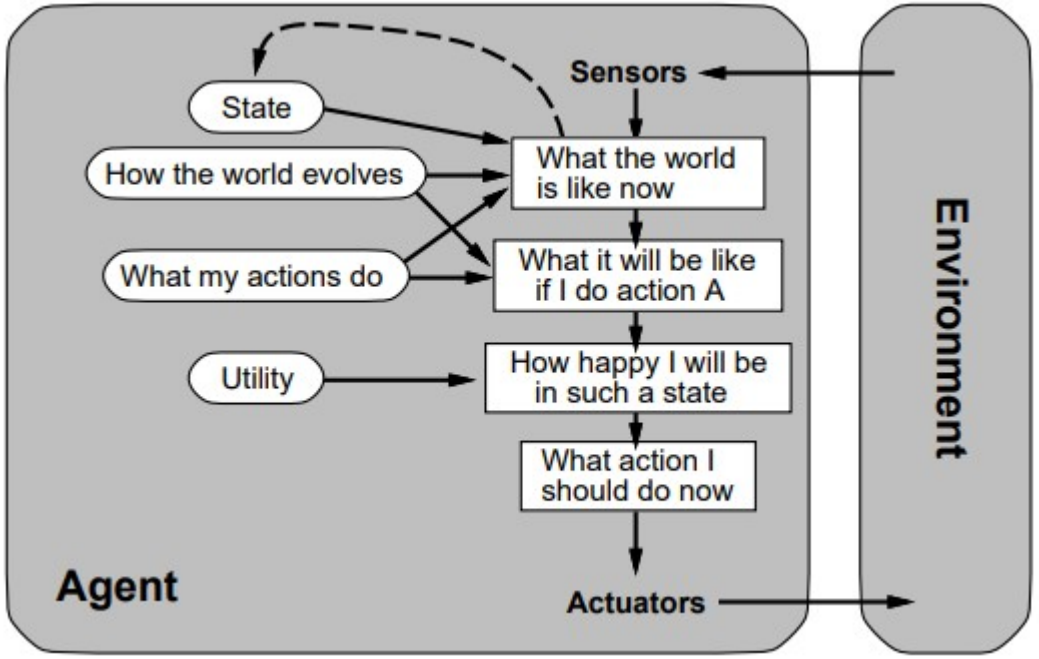
Model based Agent



Goal based Agent



Utility based Agent



Approaches to implement RL

- There are mainly three ways to implement reinforcement-learning in ML, which are:

- **Value-based:** pick the action with the best value

- To find the **optimal value function**, which is the maximum value at a state under any policy.
- Therefore, the agent expects the long-term return at any state(s) under some policy.

Approaches to implement RL

- **Policy-based:**

- **Explicitly build a representation of a policy and keep it in memory during learning.**

- Policy-based approach is to **find the optimal policy** for the maximum future rewards without using the value function.
- Agent tries to apply such a policy that the action performed in each step helps to **maximize the future reward**.

Approaches to implement RL

- **Model-based:**

- In **Model-based** approach you either have an access to the model (environment) so you know the probability distribution over states that you end up in, or you first try to build a model (often - approximation) yourself.
- This might be useful because it allows you to do *planning* (you can "think" about making moves ahead without actually performing any actions).

- **Model-free -**

- In **Model-free** you're not given a model and you're not trying to explicitly figure out how it works.
- You just collect some experience and then derive (hopefully) optimal policy.

Type	Description	Example Algorithms
Policy-Based	Learns the policy directly	REINFORCE, PPO, A2C
Value-Based	Learns the value function and derives the policy from it	Q-Learning, DQN, SARSA
Model-Based	Learns a model of the environment and uses it for planning	Dyna-Q, MPC, AlphaZero

- **Passive RL -** Agent’s policy is fixed which means that it is ***told what to do***.
- Therefore, the goal of a passive RL agent is to execute a fixed policy (sequence of actions) and evaluate it

- **Active RL** - An agent ***needs to decide what to do*** as there’s no fixed policy that it can act on.
- Active RL agent is to act and learn an optimal policy.

Passive Learning

- In passive reinforcement learning, an **agent learns by observing data collected from an external source**, rather than actively interacting with the environment.
- It involves **data collection, learning from that data, and then testing the learned policy or value function.**
- It's useful when direct interaction with the environment is costly or risky, and existing data can be leveraged for learning.

Passive Learning

- Imagine a **robot** that is placed in a simulated environment, and its goal is to **learn a policy for navigating through a maze**.
- In passive reinforcement learning, the robot doesn't take actions itself, but it observes the behavior of other agents or a human demonstrator navigating the maze.

Active Learning

- In active reinforcement learning, an agent, like a robot, actively interacts with an environment to learn a specific task.
- It explores, takes actions, receives rewards, and learns from the consequences to improve its performance over time.
- An example is a **robot learning to play table tennis by practicing and refining its skills through trial and error.**

1. Direct Utility Estimation

- In this method, the agent executes a **sequence of trials or runs** (sequences of states-actions transitions that continue until the agent reaches the terminal state).
- Each trial gives **a sample value** and the agent estimates the utility based on the samples values.
- Can be calculated as **running averages of sample values**.

Direct Utility Estimation

- In a **healthcare** context, Direct Utility Estimation is a method where a reinforcement learning agent estimates the expected benefits of different treatment strategies for a patient without explicitly modeling the patient's health dynamics.
- It directly learns from data, estimates the utility of treatments, and makes personalized treatment recommendations based on the estimated utilities.

2. **Adaptive Dynamic Programming(ADP)**

- ADP is a smarter method than **Direct Utility Estimation** as it runs trials to learn the model of the environment by estimating the utility of a state as a sum of reward for being in that state and the expected discounted reward of being in the next state.

Adaptive Dynamic Programming (ADP)

- In the context of **autonomous drone navigation**, ADP allows the drone to learn how to navigate through unknown, complex environments by iteratively exploring, estimating the value of different states and actions, and adapting its policy to make optimal decisions while avoiding obstacles and reachin.g its destination

3. **Temporal Difference Learning *(TD)***

- TD learning does not require the agent to learn the transition model.
- The update occurs between successive states and Agent only updates states that are directly affected.

- ***While ADP adjusts the utility of s with all its successor states,***
- ***TD learning adjusts it with that of a single successor state s'.***
- TD is slower in convergence but much simpler in terms of computation.

Temporal Difference Learning

- Temporal Difference (TD) learning is used in the context of playing chess.
- The program learns by playing games, updating state values using TD learning rules, and making moves based on the estimated values.
- Over time, it becomes a better chess player by iteratively improving its gameplay strategy.

Active Learning

- *Adaptive Dynamic Programming(**ADP) with exploration function***
- As the goal of an active agent is to learn an **optimal policy**, the agent needs to learn the expected utility of each state and update its policy.

- ***Hence, we use an approach that gives higher weights to unexplored actions and lower weights to actions with lower utilities.***

Q-Learning

- Q-learning is a TD learning method which does not require the agent to learn the transitional model, instead learns Q-value functions *Q(s, a)* .

	Fixed Policy(Active)	Policy not fixed(Passive)
Model-free (real world)	Temporal Difference Learning (TD)	Q-learning
Model-based (simulation)	Adaptive Dynamic Programming(ADP)	ADP with proper exploration function

Thank You