Reinforcement Learning

Reinforcement learning (RL) is a field within machine learning intelligence, that introduces a concept of an '*agent*' interacting with its '*environment*,' learning to make decisions through a process of **trial and error**, guided by a defined '*reward*' policy that shapes its behavior **over time**."

Agent: It refers to the entity or system that is designed to observe its environment, take actions, and learn from the consequences of those actions. The agent's primary objective is to **maximize** a cumulative measure of some notion of long-term "*reward*" in its interaction with the *environment*.

Environment: Environment represents an external system that encapsulates the rules and dynamics of the system with which the agents in RL interact. The environment can be anything from a simple simulated game world to a complex real-world system, depending on the application of reinforcement learning.

Reward: In the context of reinforcement learning, a "reward" is feedback that the environment provides to the agent after it takes a certain *action* in a particular *state* of the environment. Rewards can be positive, negative, or zero, and they indicate the desirability or undesirability of the agent's actions. The agent's objective is to learn to take actions that result in the highest possible total reward over the course of its interactions with the environment.

Actions: "Actions" refer to the decisions or moves that an agent can take within its environment. The agent selects actions from a set of possible choices, and the outcome of these actions influences the state of the environment and determines the reward.

State: "State" represents a particular configuration or situation of the environment at a particular time in which the agent is operating. The state encapsulates all the relevant information about the environment for the agent to make decisions about what action to take next.

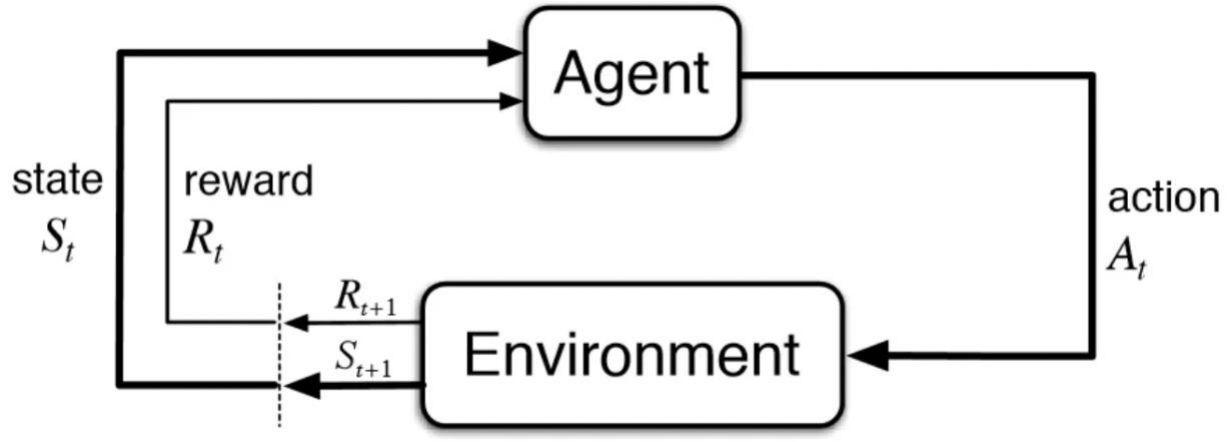

Fig: Illustration of Reinforcement Learning

An Example: A warehouse robot in packaging facility. Imagine a robot that carries object from different aisles and drop off at the packaging station**.** The Goal of RL in this task can be learning the shortest path from the start point to the destination.

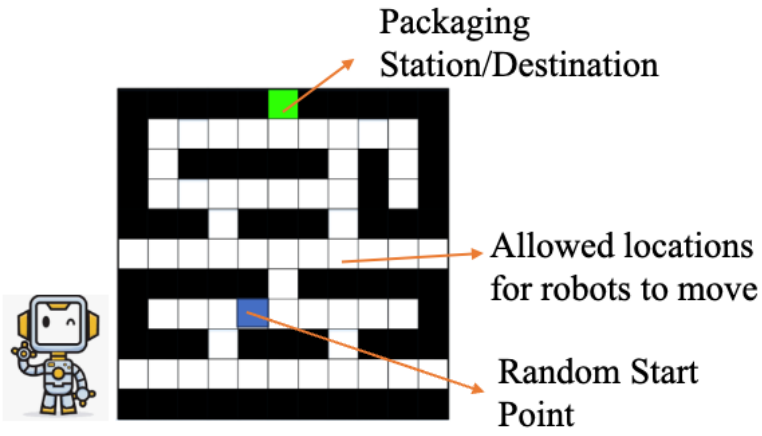

Fig: Warehouse Robot and Shortest Path

In this Warehouse Robot system example:

The *agent* is the Robot. t is responsible for tasks such as picking up items, navigating through the warehouse, and dropping the item at packaging station.

The *environment* in this example includes the physical layout of the warehouse, the locations of products, potential obstacles, location of the packaging station, and any other relevant factors that affect the robot's movement and tasks.

The *reward:* The agent receives a positive reward (some positive numeric value) when it successfully picks up from the source and drops at the packaging location, and it might receive a negative reward if it collides with an obstacle, reaches the end zone or if it fails to do the task.

Actions for the warehouse robot include moving to a specific location such as move left, right, up, down, move around obstacles, and dropping items in packaging areas.

State: The state of the warehouse robot includes information such as its current location, the status of its gripper (whether it's holding an item or not), and the presence of any obstacles in its vicinity.

Formulating a Reinforcement Learning Problem:

RL problems can be formulated as a model-based or model-free approach. RL problems can be formulated as a model-based or model-free process but have their own advantages and disadvantages. In Model-based approach, the agent builds an internal mathematical model of the problem. The model is used to predict the outcomes of different actions of agent at various state. During the learning process, the agent does not interact directly with the environment where in model-free approach the agent interacts directly with the environment. Typically, Model-based RL problems are formulated as a *Markov's Decision Process (MDP),* whereas Q-Learning is an example of model-free approach. Note, Q-Learning and Markov's decision process are not exclusive. Often time, Q Learning approach are also used along with MDP to solve the optimization problem of MDP.

Markov's Decision Process (MDP):

A Markov Decision Process (MDP) is a mathematical framework used in reinforcement learning to model decision-making scenarios. It consists of states, actions, transition probabilities, rewards, and a discount factor. The **important property of Markov process is that the future state depends only on the current state and action.** In an MDP, an agent interacts with an environment, taking actions that lead to state transitions and immediate rewards. The goal is for the agent to learn an optimal policy, a strategy that maximizes cumulative expected rewards. The discount factor balances short-term and long-term rewards. Solving an MDP involves finding the optimal policy or value function, often using dynamic programming or reinforcement learning algorithms like Q-learning.

MDP contains a tuple of four elements (S, A, P_a, R_a):

- A set of finite **states** S often called State Space
- A set of finite **Actions** A. A_s is set of actions available from state s.
- **Reward** $R_a(s, s')$ is the immediate reward (or expected immediate reward) received after transitioning from state s to state s' due to action a
- **Probability** $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ is the probability that action a in state s at time t will lead to state s' at time $t + 1$

The objective is to choose an optimal policy π^* that maximizes the cumulative reward. Optimal Policy is simply the collection of actions a chosen along the time at each state s that maximizes the cumulative reward. The optimality is defined as the maximum of expected rewards:

$$E\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1})\right]$$

Where γ is called a discount factor such that $0 \leq \gamma \leq 1$ and controls when the decision is made. It is helpful in situation when rewards changes over time. A smaller γ value typically encourages the agent to make decisions earlier than later.

Q-Learning:

'Q' in the Q-Learning stands for Quality. Q-value is some function of the state 's' and action 'a' that measures how good an action 'a' would be at state 's'. Q-learning involves computing Quality value of each action at step s and updating a Q-table using some rule such as:

$$Q^{new}(s_t, a_t) \leftarrow (1 - \underbrace{\alpha}_{\text{learning rate}}) \cdot \underbrace{Q(s_t, a_t)}_{\text{current value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \underbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max_a Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} \right)}_{\text{new value (temporal difference target)}}$$

Where, $0 \leq \alpha \leq 1$.

Q-Learning typically involves following steps:

1. Initialize Q table: Q-table is a table that consists of reward for each state-action pair.
2. Select and action to perform.
3. Measure the reward.
4. Update Q -table
5. Repeat 2-4 until the objective is met.

We will look at a very simple example of Reinforcement Learning application using Q-Learning for warehouse robot. Please refer to the **07_Q_LearningExample.ipynb** in your code folder in OneDrive.