1. Define Full Reinforcement Learning

Full reinforcement learning refers to the scenario in which an agent learns and makes decisions in a completely unknown environment without any prior knowledge or guidance. In full reinforcement learning, the agent starts with no information about the environment's dynamics, rewards, or optimal policies. It must explore and interact with the environment to learn through trial and error.

2. Compare Full RL and Immediate RL

Full RL involves learning a policy that maximizes the expected cumulative reward over a long-term horizon. This approach requires the agent to have a long-term view and consider the consequences of its actions far into the future. Full RL algorithms typically involve estimating the value function or the Q-function for each state-action pair and using this estimate to determine the optimal policy.

Immediate RL, on the other hand, focuses on learning a policy that maximizes the immediate reward at each time step, without considering the long-term consequences of the actions. This approach is useful when the rewards are sparse and occur only at specific time steps, rather than being spread out over time.

In terms of their strengths and weaknesses, Full RL is more powerful than Immediate RL since it can handle more complex problems with long-term dependencies. However, it requires more computational resources and can be slower to converge. Immediate RL, on the other hand, is more computationally efficient and faster to converge, but it may not be suitable for problems with long-term dependencies or delayed rewards.

3. What is the thumb rule followed for drawing boundary between agent and environment?

The general rule we follow is that anything that cannot be changed arbitrarily by the agent is considered to be outside of it and thus part of its environment.

4. Identify agent and environment in “A person driving a car”

The agent is the person who is driving the car, and the environment is the surrounding road, traffic, weather, and other external factors that can influence the driving experience.

5. Explain the relationship between goal and reward with an example

GOALS AND REWARDS

- In reinforcement learning, the purpose or goal of the agent is formalized in terms of a special reward signal passing from the environment to the agent. At each time step, the reward is a simple number, $R_t \in R$.
- Informally, the agent's goal is to maximize the total amount of reward it receives. This means maximizing not immediate reward, but **cumulative reward in the long run.**
- We can clearly state this informal idea as the reward hypothesis:

➢ *That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward).*

- The use of a reward signal to formalize the idea of a goal is one of the most distinctive features of reinforcement learning.
- Although formulating goals in terms of reward signals might at first appear limiting, in practice it has proved to be flexible and widely applicable.
- The best way to see this is to consider examples of how it has been, or could be, used.
- **Examples**

1. To make a robot learn to walk, researchers have provided reward on each time step proportional to the robot's forward motion.
2. In making a robot learn how to escape from a maze, the reward is often −1 for every time step that passes prior to escape; this encourages the agent to escape as quickly as possible.
3. To make a robot learn to find and collect empty soda cans for recycling, one might give it a reward of zero most of the time, and then a reward of +1 for each can collected

6. Explain the relationship between goal and reward with an example. How rewards will be formulated for given scenarios.

I) Making a robot learn to walk

Provide a reward on each time step proportional to the robots' forward motion

II) Making a robot learn how to escape from a maze

The reward is -1 for every time step that passes prior to escape; this encourages the agent to escape as quickly as possible

III) Making a robot learn to find and collect empty soda cans for recycling

Give a reward of +1 for each can that is collected, else 0

7. Consider “training a robot learn how to escape from a maze”, and how rewards will be formulated to encourage the agent to escape as quickly as possible.

The reward is -1 for every time step that passes prior to escape; this encourages the agent to escape as quickly as possible

8. Discuss the importance of having scalar rewards

- It gels well in some sense with how we do decision making and that’s why it also seems natural. Example: whether I should go to college or not? Whether I should book the car now? etc.
- Having a single scalar value makes optimization of the solution easy.

9. Distinguish between Episodic and Continuous tasks in RL and give one example of each

An episodic task lasts a finite amount of time. For example, playing a single game is an episodic task, which you win or lose. In an episodic task, there might be only a single reward, at the end of the task, and one option is to distribute the reward evenly across all actions taken in that episode.
Episodic tasks are tasks that have a terminal state (end). Episodes are considered agent-environment interactions from initial to final states. *For example, in a car racing video game, you start the game (initial state) and play the game until it is over (final state). This is called an episode. Once the game is over, you start the next episode by restarting the game, and you will begin the initial state irrespective of the position you were in the previous game. So each episode is independent of the other.*

In a continuous task, rewards might be assigned with discounting, so more recent reactions receive a greater reward, and actions a long time in the past receive a vanishingly small reward.
In a continuous task, there is no terminal state. Continuous tasks will never end. *For example, a personal assistance robot does not have a terminal state.*

10. Explain the significance of discount factor

- **The discount factor** (a number between 0-1) is a clever way to scale down the rewards more and more after each step so that, the total sum remains bounded. The discounted sum of rewards is called return (G_t) in RL.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where γ is a parameter, $0 \leq \gamma \leq 1$, called the discount rate

- According to this discounting approach, the agent tries to select actions so that the sum of the discounted rewards it receives over the future is maximized. In particular, it chooses A_t to maximize the expected discounted return.

11. State and explain unified notation for Episodic and Continuous tasks

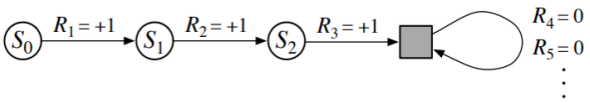
UNIFIED NOTATION FOR EPISODIC AND CONTINUOUS TASKS

- Episodic tasks requires some additional notation. Rather than one long sequence of time steps, we need to consider a series of episodes, each of which consists of a finite sequence of time steps.
- We number the time steps of each episode starting a new from zero. Therefore, we have to refer not just to S_t , the state representation at time t, but to $S_{t,i}$, the state representation at time t of episode i (and similarly for $A_{t,i}$, $R_{t,i}$, $\pi_{t,i}$, T_i , etc.).

- We need one convention to obtain a single notation that covers both episodic and continuing tasks
- $G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$, where T is a final time step.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- These can be unified by considering episode termination to be the entering of a special absorbing state that transitions only to itself and that generates only rewards of zero



- Thus, we can define return using the convention of omitting episode numbers when they are not needed and including the possibility that γ = 1 if the sum remains defined (e.g., because all episodes terminate). Alternatively, we can also write the return as

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$

Including the possibility that $T = \infty$ or $\gamma = 1$ (but not both)

12. Give formal definition of Markov Property and explain with an example

A state signal that succeeds in retaining all relevant information is said to be Markov or to have the Markov property.

THE MARKOV PROPERTY

- Let us formally define the Markov property for the reinforcement learning problem.
- To keep the mathematics simple, we assume here that there are a finite number of states and reward values. This enables us to work in terms of sums and probabilities rather than integrals and probability densities, but the argument can easily be extended to include continuous states and rewards.
- Consider how a general environment might respond at time t+ 1 to the action taken at time t. In the most general, causal case this response may depend on everything that has happened earlier. In this case the dynamics can be defined only by specifying the complete probability distribution:

$$\Pr\{R_{t+1} = r, S_{t+1} = s' \mid S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\}$$

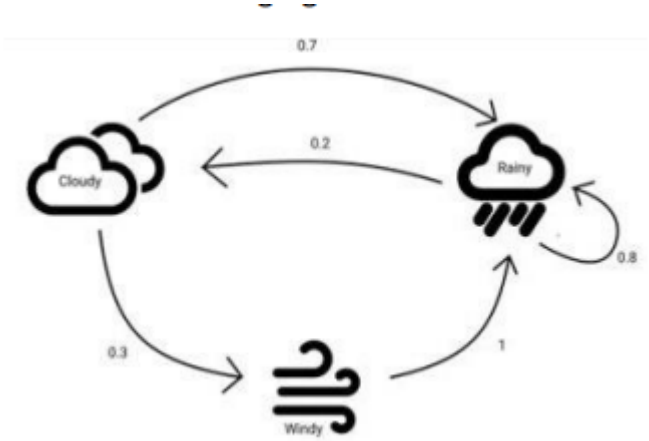
- If the state signal has the Markov property, on the other hand, then the environment's response at t + 1 depends only on the state and action representations at t, in which case the environment's dynamics can be defined by specifying only

$$p(s', r|s, a) = \Pr\{R_{t+1} = r, S_{t+1} = s' \mid S_t, A_t\} \qquad \text{Eq. 2}$$

In other words, a state signal has the Markov property, and is in a Markov state, if and only if equation 2 is equal to equation 1 for all s', r and histories, S0, A0, R1, ..., St−1, At−1, Rt
In this case, the environment and task as a whole are also said to have the Markov property

For example, consider a weather forecasting model that predicts the weather in a given location based on the weather conditions in the recent past. Suppose that the model satisfies the Markov property. This means that the probability of tomorrow's weather depends only on today's weather, and not on the weather from the previous days. In other words, if we know today's weather, we can predict tomorrow's weather regardless of what the weather was like yesterday or the day before. This simplifies the model significantly and allows for more efficient computation of the probabilities of future weather conditions.

13. Consider the image given below:



Define Markov chain and convert this into transition probability matrix and table.

The Markov Chain consists of a sequence of states that follow the Markov property. This Markov chain actually is the probabilistic model that depends on the current state to predict the next state.

When the current state is cloudy, 70% of the next state will be rainy, or 30% will be windy. If the current state is windy, there is a 100% possibility the next state will be rainy. When the current state is rainy, there is an 80% probability that the next state will keep rainy, and there is 20% will be changed to cloudy.

We can show these states and transition probability into a table or a matrix as below:

Current State	Next State	Transition Probability
Cloudy	Rainy	0.7
Cloudy	Windy	0.3
Rainy	Rainy	0.8
Rainy	Cloudy	0.2
Windy	Rainy	1.0

	Cloudy	Rainy	Windy
Cloudy	0.0	0.7	0.3
Rainy	0.2	0.8	0.0
Windy	0.0	1.0	0.0

Markov chain consists of a set of states along with their transition probabilities.

14. Define MRP and MDP

MARKOV REWARD PROCESS

- The Markov Reward Process (MRP) is an extension of the Markov chain with an additional reward function. So, it consists of states, a transition probability, and a reward function.
- This reward function gives us the reward that we get from each state. This function will tell us the reward we obtain in the state cloudy, the reward we obtain in the state of windy, and the rainy state.
- This reward also can be a positive or negative value. Mathematically can define reward as:

$$R_s = E[R_{t+1} \mid S_t]$$

- A reinforcement learning task that satisfies the Markov property is called a Markov decision process, or MDP.
- If the state and action spaces are finite, then it is called a finite Markov decision process (finite MDP).
- Finite MDPs are particularly important to the theory of reinforcement learning.

We define **MRP** as (S,P, R,ɤ) , where :

- S is a set of states,
- P is the Transition Probability Matrix,
- R is the Reward function,
- ɤ is the discount factor

•We define **MDP** as (S,P, R,A,ɤ) , where :

- S is a set of states,
- P is the Transition Probability Matrix,
- R is the Reward function, we saw earlier,
- A is a set of actions
- ɤ is the discout factor

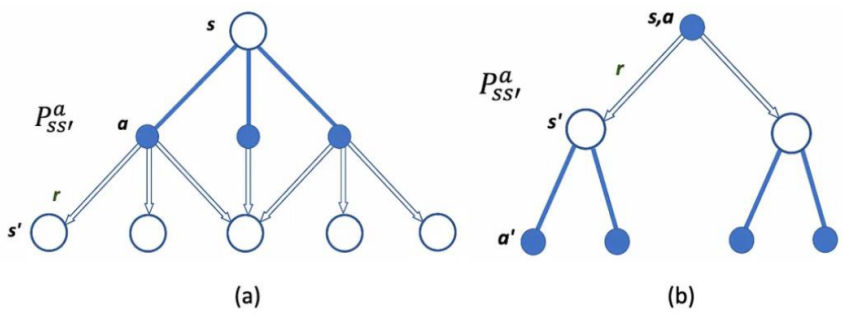
15. Explain Bellman equation for state value and action value function using backup diagrams

BELLMAN EQUATION FOR STATE VALUE FUNCTION

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right]$$

- This equation tells us how to find the value of a state **s** following a policy $\boldsymbol{\pi}$.
- We can intuitively see that it recursively breaks down the value computation into an immediate expected reward from the next state, ***r(s,a)***, plus the value of a successor state, ***Vπ(s')***, with a discount factor $\boldsymbol{\gamma}$.
- The above equation also expresses the stochasticity of the Environment with the sum over the policy probabilities.

BELLMAN EQUATION



Backup diagrams for (a) $V_\pi(s)$ and (b) $Q_\pi(s,a)$.

P means the probability of action ***a***, issued in state ***s***, ending up in state ***s'*** (with reward ***r***)

16. Write Bellman optimality equation for V* and Q* and explain them with the help of back up diagrams

$$\begin{aligned}
v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi^*}[G_t \mid S_t = s, A_t = a] \\
&= \max_a \mathbb{E}_{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}_{\pi^*}\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \,\middle|\, S_t = s, A_t = a\right] \\
&= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \\
&= \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_*(s')\right].
\end{aligned}$$

- The Bellman optimality equation for q∗ is:

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right] \\
&= \sum_{s',r} p(s', r|s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right].
\end{aligned}
$$

BELLMAN OPTIMALITY EQUATION

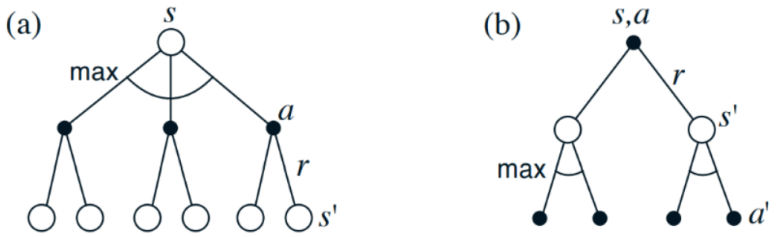


Figure 3.7: Backup diagrams for (a) v_* and (b) q_*

These diagrams spans of future states and actions considered in the Bellman optimality equations for v∗ and q∗. These are the same as the backup diagrams for Vπ and Qπ except that arcs have been added at the agent's choice points to represent that the maximum over that choice is taken rather than the expected value given some policy

17. What is the difference between Bellman Expectation and Bellman Optimality Equation?

BELLMAN OPTIMALITY AND BELLMAN EXPECTATION EQUATION

Bellman expectation equation ⟶

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_\pi(s')\right]$$

Bellman optimality equation ⟶

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_*(s')\right]$$

Bellman Optimality equation is the same as Bellman Expectation Equation, but the only difference is instead of taking the average of the actions our agent can take we take the action with the max value.

18. Explain the solution of Recycling Robot using the Bellman Optimality Equation

EXAMPLE: RECYCLING ROBOT

- At each step, robot has to decide whether it should (1) actively search for a can, (2) wait for someone to bring it a can, or (3) go to home base and recharge.
- Searching is better but runs down the battery; if runs out of power while searching, has to be rescued (which is bad).
- Decisions made on basis of current energy level: high, low.
- Reward = number of cans collected

$S = \{\texttt{high}, \texttt{low}\}$

$A(\texttt{high}) = \{\texttt{search}, \texttt{wait}\}$

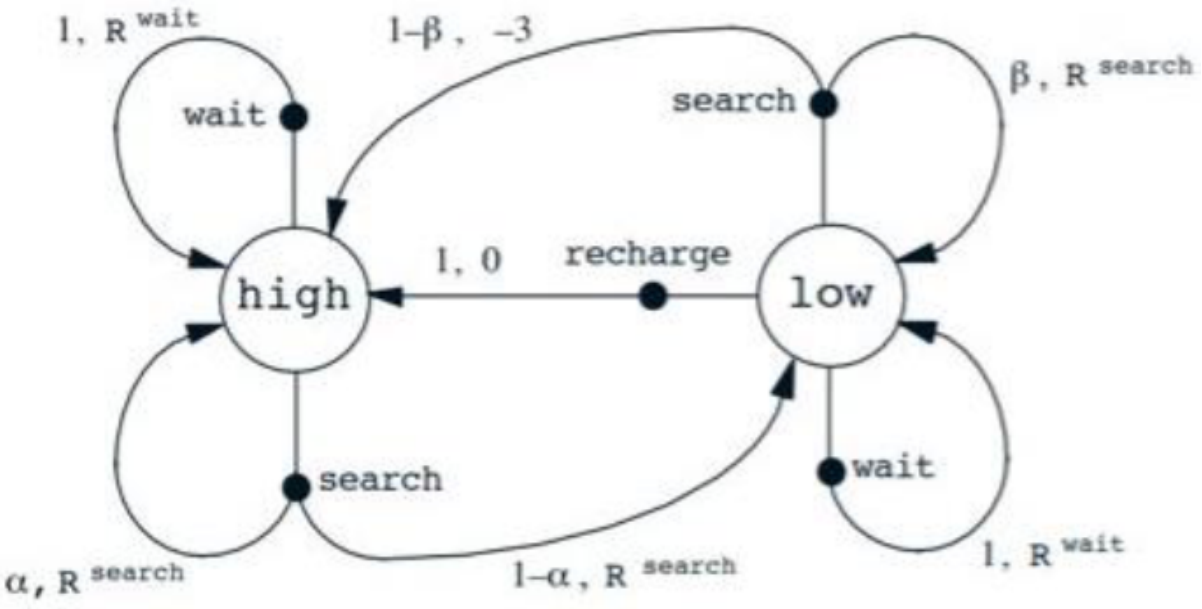
$A(\texttt{low}) = \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}$

$R^{\texttt{search}}$ = expected no. of cans while searching

$R^{\texttt{wait}}$ = expected no. of cans while waiting

$R^{\texttt{search}} > R^{\texttt{wait}}$

s	s'	a	$p(s'\|s,a)$	$r(s,a,s')$
high	high	search	α	$r_{\texttt{search}}$
high	low	search	$1-\alpha$	$r_{\texttt{search}}$
low	high	search	$1-\beta$	-3
low	low	search	β	$r_{\texttt{search}}$
high	high	wait	1	$r_{\texttt{wait}}$
high	low	wait	0	$r_{\texttt{wait}}$
low	high	wait	0	$r_{\texttt{wait}}$
low	low	wait	1	$r_{\texttt{wait}}$
low	high	recharge	1	0
low	low	recharge	0	0.



$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_*(s')\right]$$

The equation for v∗(h) can be written as follows:

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \sum_{s',r} p(s', r|s, a)\left[r + \gamma v_*(s')\right]$$

$$\begin{aligned} v_*(\mathtt{h}) &= \max \left\{ \begin{array}{l} p(\mathtt{h}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{s})[r(\mathtt{h},\mathtt{s},\mathtt{l}) + \gamma v_*(\mathtt{l})], \\ p(\mathtt{h}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{h}) + \gamma v_*(\mathtt{h})] + p(\mathtt{l}|\mathtt{h},\mathtt{w})[r(\mathtt{h},\mathtt{w},\mathtt{l}) + \gamma v_*(\mathtt{l})] \end{array} \right\} \\ &= \max \left\{ \begin{array}{l} \alpha[r_\mathtt{s} + \gamma v_*(\mathtt{h})] + (1-\alpha)[r_\mathtt{s} + \gamma v_*(\mathtt{l})], \\ 1[r_\mathtt{w} + \gamma v_*(\mathtt{h})] + 0[r_\mathtt{w} + \gamma v_*(\mathtt{l})] \end{array} \right\} \\ &= \max \left\{ \begin{array}{l} r_\mathtt{s} + \gamma[\alpha v_*(\mathtt{h}) + (1-\alpha) v_*(\mathtt{l})], \\ r_\mathtt{w} + \gamma v_*(\mathtt{h}) \end{array} \right\}. \end{aligned}$$

19. Discuss Recycling Robot problem and identify state space, action space and rewards associated with actions for the same. Draw a State Transition Table and State transition diagram for the same.

EXAMPLE: RECYCLING ROBOT

- At each step, robot has to decide whether it should (1) actively search for a can, (2) wait for someone to bring it a can, or (3) go to home base and recharge.
- Searching is better but runs down the battery; if runs out of power while searching, has to be rescued (which is bad).
- Decisions made on basis of current energy level: high, low.
- Reward = number of cans collected

$S = \{\texttt{high}, \texttt{low}\}$

$A(\texttt{high}) = \{\texttt{search}, \texttt{wait}\}$

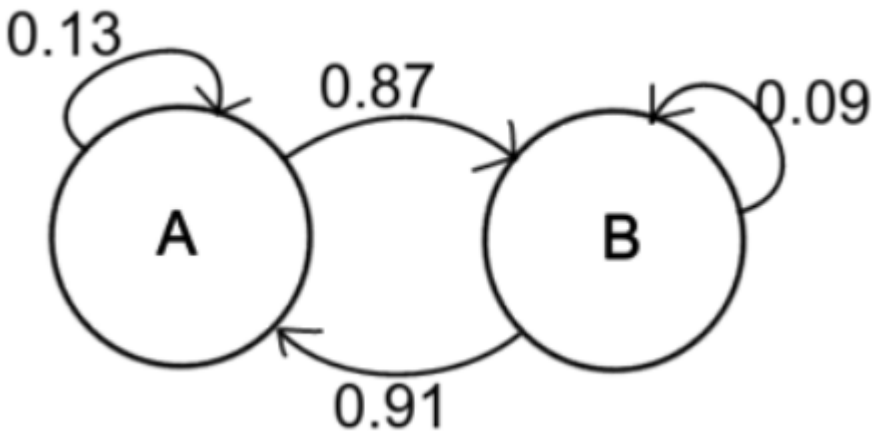
$A(\texttt{low}) = \{\texttt{search}, \texttt{wait}, \texttt{recharge}\}$

$R^{\texttt{search}}$ = expected no. of cans while searching

$R^{\texttt{wait}}$ = expected no. of cans while waiting

$R^{\texttt{search}} > R^{\texttt{wait}}$

20. Draw the transition matrix representing the Markov chain with state diagram shown below. Assume the states are ordered with A before B.



	A	B
A	0.13	0.87
B	0.91	0.09