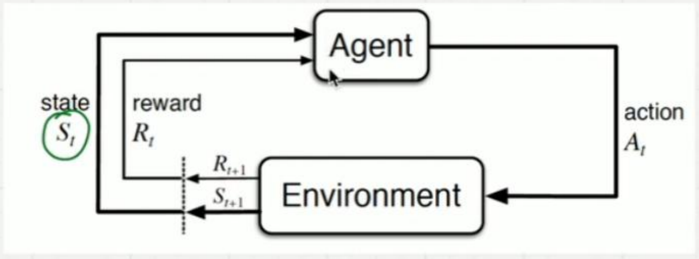
Lecture 6 - Finite Markov Decision Processes

This problem defines the field of reinforcement learning. This lecture will focus on understanding this problem from the very basics.

The Agent-Environment Interface:-



S_t Set A_t

$S_1, S_2 \ldots$ $a_1, a_2 \ldots$

R_t

At each time step, the agent receives information of the state of the environment. The state is denoted by S_t.

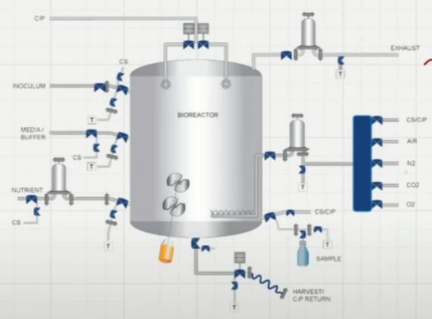
One time step later, the agent receives a reward and goes into a new state. The reward is denoted by R_{t+1} and the new state is denoted by S_{t+1}.

At each time step, the agent implements a mapping from the states to the probability of selecting each possible action. This mapping is called as the agents policy. We denote the policy as π_t, where $\pi_t(a|s)$ is the probability that $A_t = a$ if $S_t = s$.

Examples of States:- Sensor reading; chess game intermediate position, Abstract- Not sure where my carkeys are.

Examples of Actions:- Voltage applied to motors of a robot arm. Whether or not have lunch, Pedaling wheels of a bicycle.

Understanding the Agent-Environment interface with practical examples.



→ **States:** Sensor readings, Thermocouple readings

→ **Actions:**

Target temperature: Activate heating element

Target stirring rate: Activate motors

Rewards:

Moment to moment measures of the rate at which useful chemical is produced.

To summarize, $\text{states} = \begin{bmatrix} \text{sensor reading} \\ \text{thermocouple reading} \end{bmatrix}$ $\text{actions} = \begin{bmatrix} \text{Target temperature} \\ \text{Target stirring rate} \end{bmatrix}$

Example 2: Pick and Place Robot:

→ **States:** Readings of Joint Angle and Velocities.

→ **Actions:** Voltages applied to motors at each joint.

Reward: +1 for each object successfully picked up and placed.
-0.02 (small negative): Jerky motion.

To summarize, states = $\begin{bmatrix} \theta_1, \\ \omega_1, \\ \theta_2, \\ \vdots \end{bmatrix}$, actions = $\begin{bmatrix} V_1, \\ V_2, \\ \vdots \end{bmatrix}$, reward = $\begin{bmatrix} +1, \\ -0.02 \end{bmatrix}$ for $\begin{bmatrix} \text{success} \\ \text{jerky motion} \end{bmatrix}$

Goals and Rewards:

In this section, we will look at rewards more closely.

The use of a reward signal to formalize the idea of a goal is one of the most distinctive features of reinforcement learning.

(1) Game of chess: Rewards are +1 for winning, -1 for losing and 0 for drawing.

+1 taking the queen

(2) Robot escaping a maze: Reward is -1 for every time step that passes before the robot escapes the maze.

10 time steps 2 tim
-10 -2

(3) Robot collecting empty soda cans: +1 for each can collected, else 0.

* Rewards should not be given for achieving sub-goals, but for actually attaining final goal (e.g: Giving reward in chess for taking opponent's queen)

Returns:-

Agent's goal is to maximize the $\underbrace{\text{cumulative reward}}$ it $\underbrace{\text{receives in the long run}}$.

We want to maximize the $\underbrace{\text{expected return}}$.

The expected return at time 't' is denoted as G_t. We can express it as follows:-

$$\boxed{G_t} = \underbrace{R_{t+1} + R_{t+2} + \dots + R_{\boxed{T}}} \rightarrow \text{Rewards after time step t are added up.}$$

$$\boxed{G_t} = \boxed{R_{t+1}} + R_{t+2} + \dots + R_T$$

T: Final Time Step.

This formulation is suited for applications which have a final step, like a game of chess or trips through a maze.

These tasks are called episodic tasks.

There are cases where the agent-environment interaction goes on continously. These are called as continuing tasks. Example: Rover on a Mars expedition. For these tasks, we cannot use the above formulation.

$$R_{T+1} + R_{T+2} + \dots \quad R_{T+\infty} \quad \text{``}\infty\text{''}$$

We naturally come to the concept of $\underbrace{\text{discounting}}$. Let us understand this concept using an analogy:-

100 Rupees are more valuable now compared to 5 years later. This is due to inflation.

Similary, $\underbrace{\text{immediate rewards}}$ are more valuable compared to rewards $\underbrace{\text{received later}}$. We take this into account by saying that every reward is 'γ' times less valuable than the reward before. Here γ is the discount rate which is less than 1.

So, the expected return can now be written as:-

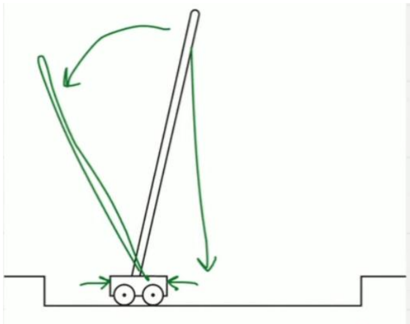
$$G_t = R_{t+1} + 0.5\,R_{t+2} + 0.25\,R_{t+3} + \dots + (0.25)(0.5) = 0.125\,R_{t+4}$$

$$\boxed{G_t = \overset{\downarrow 1}{R_{t+1}} + \boxed{\overset{\downarrow 1}{\gamma} R_{t+2}} + \boxed{\overset{\downarrow 1}{\gamma^2} R_{t+3}} + \dots = \sum_{k=0}^{\infty} \underset{\text{discount rate}}{\gamma^k} R_{t+k}}$$

If $\gamma = 0$, the agent is only concerned about immediate rewards

As γ approaches 1, the agent considers future rewards more strongly.

Example to illustrate the difference between episodic and continuing tasks.


The pole-balancing task

Objective: Apply forces to a cart moving along a track such that the pole does not fall over.

How would you write the rewards and returns for this problem?

Option 1: Treating the problem as an episodic task.

Reward = +1 for all time steps when the pole does not fall over, 0 for timesteps when the pole falls over.

2 timestep :- +2

5 timestep = +5

$$G_t = \underbrace{R_{t+1} + R_{t+2} + R_{t+3} + \ldots}\ \Big\}\ \text{Return}$$

Return will be maximum only if the pole does not fall over for maximum number of time steps. This is what we want. So, our reward formulation is correct.

Option 2: Treating the problem as a continous task.

Reward = -1 for failure (if pole falls), otherwise 0.

$$G_t = \underbrace{R_{t+1}} + \underbrace{\gamma R_{t+2}} + \underbrace{\gamma^2 R_{t+3}} + \ldots$$

2

$0 + 0 + \gamma^2(-1) = -\gamma^2$

$-(0.5)^2$

$= -$ ~~(0.025)~~

$-(0.25)$

If failure occurs after k time steps, $G_t = -\gamma^k$

$0 + 0 + 0 + \gamma^3(-1) =$

$-(0.5)^{\boxed{3}}$ ✓

$= -(0.0125)$

So, return will be maximized if failure occurs very late and K is very large.
This is exactly what we want. So, our reward formulation is correct.

The Markov Property:-

We know that, the agent makes a decision after receiving signal from the environment. This is called as a state signal.

Let us look at state signals which have a specific property:-

Conversations

When we are speaking with someone, the entire history of our past conversations are known to us.

Cannonball

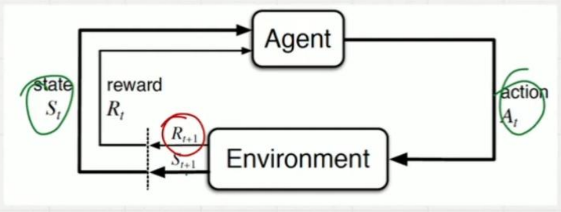
The current position and velocity of the cannonball is all what is needed to predict the next position and velocity. Past history does not matter.

What is common in all of the above examples?

In all of these examples, the state signal (knowledge of people, current configuration of chess pieces, current position and velocity of cannonball) retains all the relevant information from the past.

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

Let us define the Markov property formally:-



For most general cases, the dynamics may depend on everything that has happened earlier.

This can be expressed as follows:-

All past event

$$P(R_{t+1} = r, S_{t+1} = s' \mid \underbrace{S_0, A_0, R_1, \ldots, S_{t-1}, A_{t-1}, R_t, S_t, A_t}_{\text{all past events}})$$

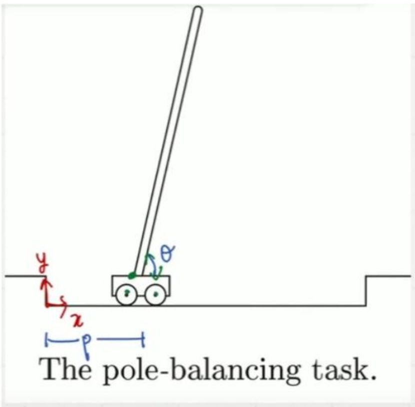
But if the state signal has Markov property, the environment response only depends on the state and action values in the previous time step.

This can be expressed as follows:-

$$\boxed{P(R_{t+1} = r, S_{t+1} = s' \mid \underbrace{S_t, A_t}_{\text{only current event}})}$$

If a state signal has a Markov property, it allows us to predict the next state and the expected next reward given the current state and action.

Practical examples to demonstrate Markov property:-



The pole-balancing task.

The $\underbrace{\text{state signal}}$ which includes the position of the cart, velocity of the cart, angle between the cart and the pole and the rate at which this angle is changing would satisfy the Markov property

But only in an idealized setting.

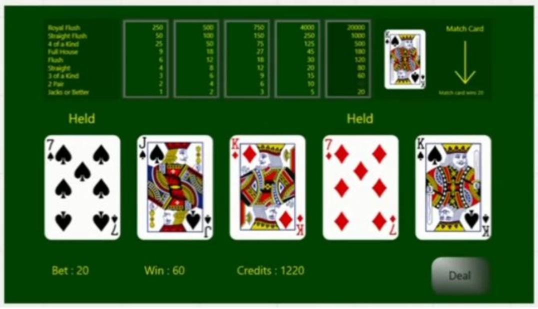
$$\begin{bmatrix} p, \\ \dot{p}, \\ \theta, \\ \dot{\theta} \end{bmatrix}_t \rightarrow \boxed{t+1}$$

→ Markov property.

In a real pole-cart system, the bending of the pole, temperature of the pole and wheel bearings would also affect the behavior of the system.

So, our state signal → $\left\{ \begin{bmatrix} p, \\ \dot{p}, \\ \theta, \\ \dot{\theta} \end{bmatrix} \right\}$ would violate the Markov property in a real setting.

Game of Draw Poker:-



Royal Flush	250	500	750	4000	20000
Straight Flush	50	100	150	250	1000
4 of a Kind	25	50	75	125	500
Full House	9	18	27	45	180
Flush	6	12	18	30	120
Straight	4	8	12	20	80
3 of a Kind	3	6	9	15	60
2 Pair	2	4	6	10	[illegible]
Jacks or Better	1	2	3	5	20

Match Card

Match card wins 20

Held Held

Bet : 20 Win : 60 Credits : 1220

Deal

In this game, for the state to satisfy the Markov property, the player should know:-

1. Knowledge of one's own cards.
2. Bets made by other players.
3. Number of cards drawn by other players
4. Past history with other players.

↳ Does Raj like to bluff? Does his expressions reveal something when he is bluffing?

However, no one remembers this much information while playing poker. Unless you are James Bond :p

Hence, the state representations which people use to make poker decisions are non-Markov.

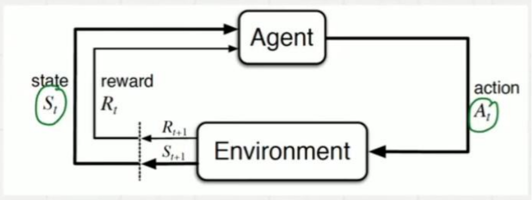
Markov Decision Processes (MDPs)

A reinforcement learning task which satisfies the Markov property is called a Markov Decision Process or MDP.

We will now look at some notations:-

We will now look at some notations:-

$p(s'|s,a)$:- State Transition Probability



$$p(s'|s,a) = Pr(S_{t+1} = s' \mid S_t = s, A_t = a)$$

$r(s,a)$:- Expected reward for the state action pair

$r(s,a,s')$:- If I started in 's', took action 'a' and landed in 's'', what is the expected reward.

A practical example to understand a Markov Decision Process:-

Recyling Robot

At each time, the robot should decide one of the following:-

(1) Whether to actively search for a can

(2) Whether to remain stationery and wait for someone to bring the can

(3) Go back to home base and recharge battery

The 'State Set' of the robot has 2 possible states - High, Low

The 'Action Set' of the robot has 3 possible actions:- Wait, Search & Recharge

How do we formulate the rewards?

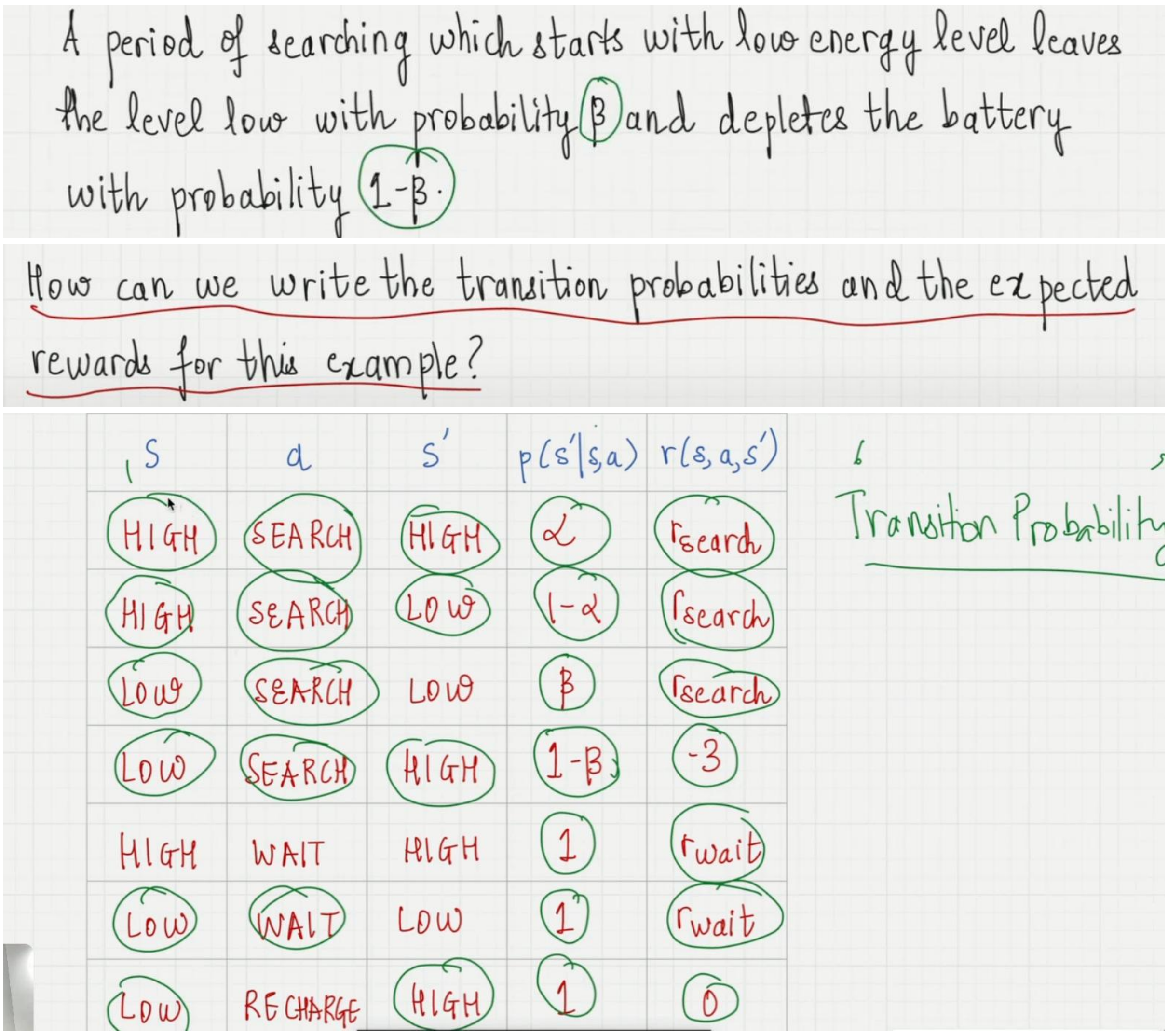
Let r_{search} denote the expected number of cans the robot collects while searching $\rightarrow \boxed{3}$

Let r_{wait} denote the expected number of cans the robot receives while waiting. $\rightarrow \boxed{1}$

So, while searching, the reward value would be r_{search} and while waiting the reward value would be r_{wait}.

$r_{search} > r_{wait}$ } α

A period of searching which starts with high energy level leaves the level high with probability $\boxed{\alpha}$ and reduces it to low with probability $\boxed{1-\alpha}$.



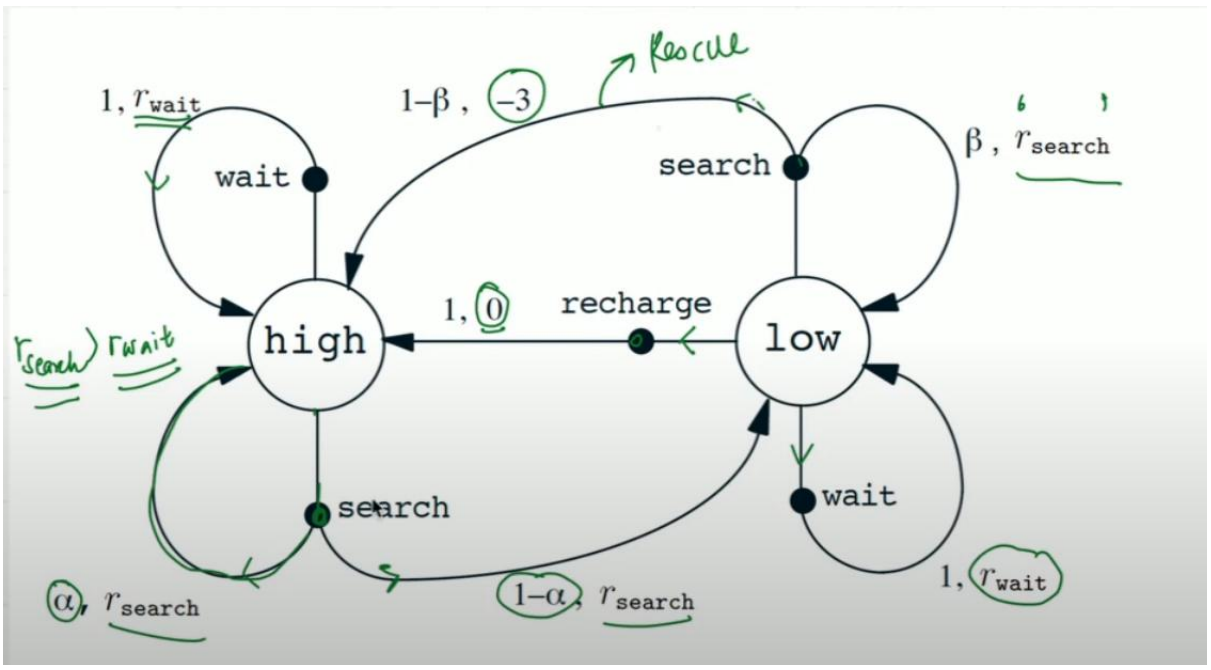
A period of searching which starts with low energy level leaves the level low with probability β and depletes the battery with probability $1-\beta$.

How can we write the transition probabilities and the expected rewards for this example?

s	a	s'	$p(s'\|s,a)$	$r(s,a,s')$
HIGH	SEARCH	HIGH	α	r_{search}
HIGH	SEARCH	LOW	$1-\alpha$	r_{search}
LOW	SEARCH	LOW	β	r_{search}
LOW	SEARCH	HIGH	$1-\beta$	-3
HIGH	WAIT	HIGH	1	r_{wait}
LOW	WAIT	LOW	1	r_{wait}
LOW	RECHARGE	HIGH	1	0

Transition Probabilit

Transition Graph for Recyling Robot :-



1, r_{wait}
wait
1−β , −3
Rescue
search
β , r_{search}
1, 0
recharge
high
low
r_{search} > r_{wait}
search
wait
α, r_{search}
1−α, r_{search}
1, r_{wait}