Aquntic AI Course

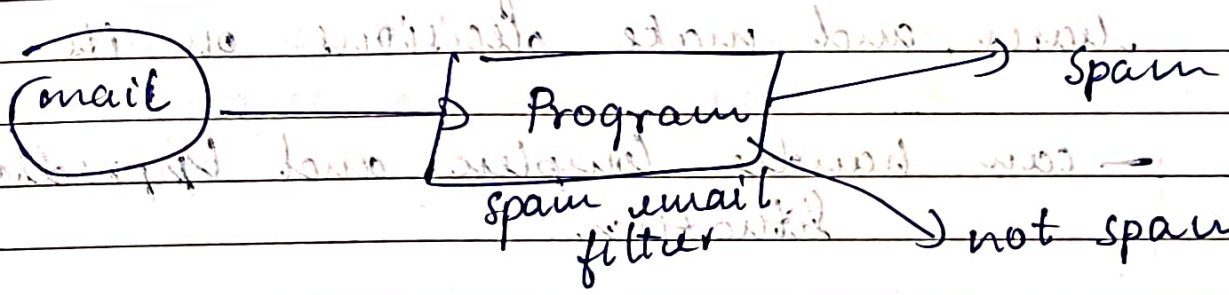
from course youtube - Code with Aaroli

learnt and maintained by Rohit Chigatapu

Limitations of Traditional AI:

helping us every day from chatbots to robots

Supose you have a spam email filter



The problem is a new type of spam mail

the program doesnt know this so a human must train again to spam this new mail type

So there is no ability to adapt and recognise these new unknown scenarios on their own.

They only reply or act on the way you train them they wont adapt

* So they can adapt to new situation or learn from interaction without human input

* Agentic AI

is a new kind of AI that can adapt, learn, and make decisions on its own.

- can handle Complex and Unpredictable situation
- Adapt and learn Over time
- Make Autonomous Decisions

* How can Generative AI and Agentic AI used together ?

While Open AI doesn't adapt or make decisions Agentic AI can make decisions and adapt and we open AI to generate new stuff for us

Ex:- Advertising and Marketing

Agentic AI: Analyzes customer data to decide which ads to show

Open AI: Creates personalized ad content (text, images, videos) based on customer preferences

* So we see Agentic AI Chooses the target audience, while Open AI designs the ad content for them.

* What is an Agent?

perceive → reason → Act

Example:- a robot vacuum

Step-1: Understand environment through its sensors

Step-2: Decides what to do

Step-3: taking action

* What are LLM agents?

Same like agents. perceive, reason, act but use LLM to give and ~~understand~~ reason

Ex:- ① Understand user query

Ex:- "Tell me the weather in Delhi today"

② Decides what needs to be done to answer the user's query

Ex:- agent decides to use API of weather to fetch the details

③ Taking Action

Ex:- agent will get the details in raw format and use LLM to convert it to understandable human text and answer the user

What is an Agent in AI?

Types of Agents in AI

1. Simple Reflex Agent
2. Model - based reflex agent
3. Goal - based Agent
4. Utility - based Agent
5. Learning Agent

Simple Reflex Agent

They are the simplest agents
like Automatic door
opens when senses human near
and closes when not near

Same for sensor light
If human in there lights on

So they only react on current situation
they dont have any memory and dont
plan, just react

Model Based Reflex Agent

as we know Simple Reflex Agent

in this MBR Agent instead of just reacting it also remembers things

this memory is called model of the world

it is useful when to handle situation when current input is not enough to make a good decision

Goal Based Agents

It has a Goal in mind

Plans before acting

Ex:- flight booking assistant

→ book the flight under Rs 5000.

so agent will compare flights from different sites and prices and will give the best option

they think before they act

Utility based Agents

= Goal-Based Agents + Smart Optimization

it is an upgrade to the goal based agent

as it just doesnt focus on reaching the goal but also wants to find the best possible way to do it

before reaching the goal it tries to max the utity score
→ how good each option is

uses utility function to assign scores to each different options and picks the best one

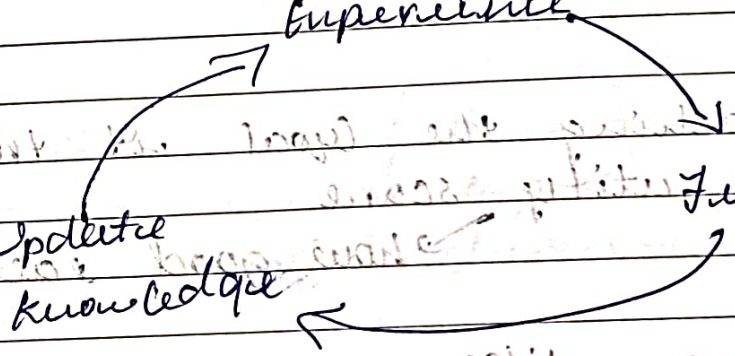
Eg: In Chess game
The agent considers 10 possible moves

It picks the one most likely to lead to victory

Learning Agents

A learning agent is an agent that gets better over time.

it learns from its own experiences just like we do



it just tries to maximize total reward over time

So they try, they observe, they learn

Reinforcement Learning

Getting Reward for Good Actions
Getting penalities for bad Actions

A Learning Agent learns by Reinforcement learning

Agents:
1. Observe its Environment
2. Agent takes action
3. The environment gives feedback
4. agent learns from the feedback and adjusts its future decisions to get more rewards.

Environment responds to agents action by:
1. update state
2. Reward agent

State: is a snapshot of the environment at a specific moment of time

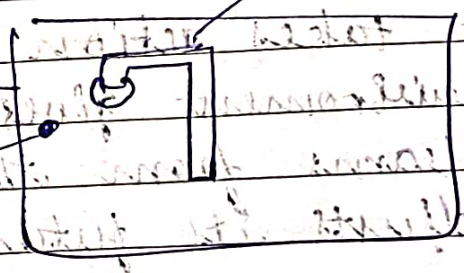
Policy -
Is like a agents brain
Types of policy
- Deterministic
- Stochastic (probabilistic)

Goal : Goal is to pick the best policy.

Value function : V(s).

How good it is for the agent to be in a particular state

Eg:- In a snake eating ball game



State with high V(s)

State with low V(s)

state near to food V(s) = 8

state near to wall V(s) = -80

* How does agent learn these values?

By simply playing the game exploring different situations of the game. Over time it starts to notice that some states often lead to food those states are high value states

How to Calculate V(s)?

Bellman Expectation Equation

$$V(s) = E_\pi [R + \gamma V(s')]$$

- V(s) - Value of current state
- R - Immediate reward for current action
- γ - Discount factor (how much we care about future rewards)
- V(s') - Value of Next State
- π - Policy
- E - Expectation (avg over possible outcomes)

And the eqn only works when you know these values in the beginning

Agent must already know!

The policy ? Agent must learns
next state ? this after many
rewards experience

The agent doesn't know Bellman
can directly

Solution:

We can use learning algorithm
like:

1) Monte Carlo
2) Temporal Difference (TD) Learning
3) Q-Learning

What is Q-learning ?

→ means agent doesn't know anything about the env

is a model-free Reinforcement learning
algorithm

→ agent will not know what is
it ? or ?
→ next state

What is a Q-Value ?

$Q(s, a)$

- Quality
- state
- action

tells the agent the quality of taking an action in state s

What is a Q-Table?

states \ actions	a_1	a_2	a_3
s_1	$Q(s_1, a_1)$	$Q(s_1, a_2)$	$Q(s_1, a_3)$
s_2	$Q(s_2, a_1)$	$Q(s_2, a_2)$	$Q(s_2, a_3)$
s_3	$Q(s_3, a_1)$	$Q(s_3, a_2)$	$Q(s_3, a_3)$
s_4	$Q(s_4, a_1)$	$Q(s_4, a_2)$	$Q(s_4, a_3)$

14	2	7
0.01	-5	-10
0.4	-15	+15
0.5	0.9	100

agent $s_1 \rightarrow a_1$ (as 14 highest reward)

$s_2 \rightarrow a_1$

$s_3 \rightarrow a_3$

$s_4 \rightarrow a_4$

the value of Q-Table is updated after each step

How to count $Q(s,a)$?

→

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a) \right]$$

- $Q(s,a)$: current Q value
- α: learning rate
- r: reward
- γ: discount factor
- $\max_{a'} Q(s',a')$: next state Quality

Example

Current state s: Snake is near food

Action a: Turn left

Reward r: +10 (food eaten!)

next State s' - New snake position.

Max future Q-value $(\max_{a'} Q(s',a')) = 8$

Current $Q(s,a) = 5$

$\gamma = 0.9$ (discount factor)

$\alpha = 0.1$ (learning Rate)

$r + \gamma \cdot \max_{a'} Q(s',a')$

compute target -

$= 10 + 0.9 \times 8 = 17.2$

$r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)$

compute error -

$= (17.2 - 5) = 12.2$

update the Q-value - $Q(s,a) \leftarrow 5 + 0.1 \times 12.2 = 5 + 1.22$

$= 6.22$

This is now each Q value for each action and each state are updated in the Q-table