RL (Silver)

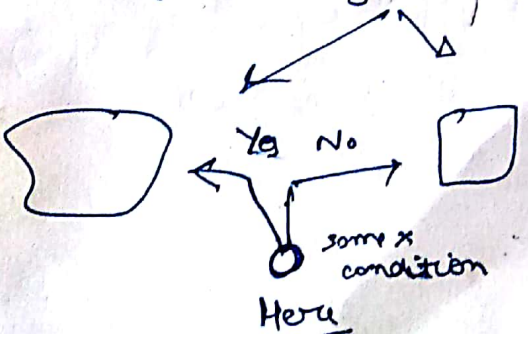
Lecture 1.

- RL crosses across various fields of science like CS, Neuroscience, Engineering, Psychology, etc.
  The reward system in RL algorithms is drawn from the findings in Neuroscience about how the human brain makes decisions, by devoting to the dopaminic system.

Machine Learning subdivision
- Supervised
- Unsupervised
- Reinforcement

Reinforcement Vs supervised

- No best ~~reward~~ decision mentioned in RL. We just say that for some X action, you get Y points. Not that this is the best action.
- Feedback delayed, not instantaneous -> In supervised, we directly get the classification result → good/bad. But in RL, so this decision is delayed, as in, we might get some good points in some of the actions until a catastrophic (large -ve reward) loss in the end.
- non-iid data → What a robot or a sees in one second is very correlated to what it sees in the next second. So, time really matters.
- Agent gets to make decisions / take actions and based upon its action, it gets very different data.

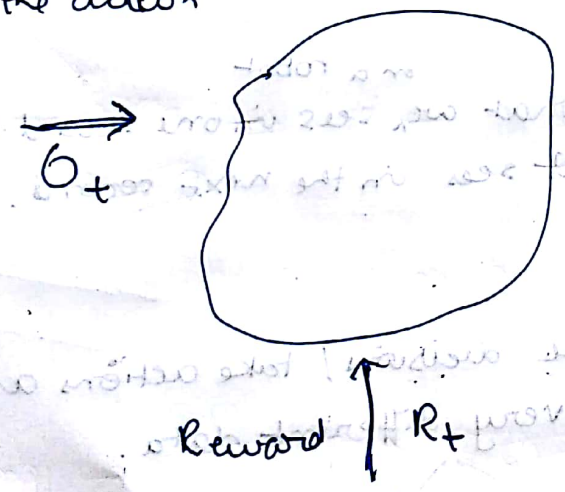


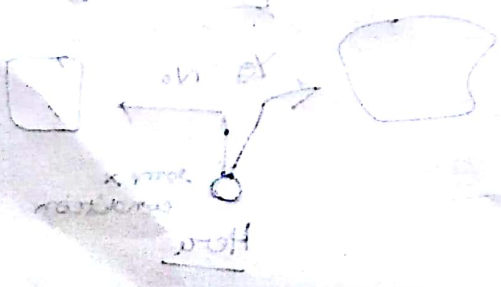
The RL problem

- Rewards
  - → Reward R → scalar feedback and there's this hypothesis that any goal can be achieved by maximizing the cumulative rewards at each time steps.
  - eg. say want to do some task in shortest time then add reward = -1 at each time step, so when we take the cumulative reward, it learns to take the minimum # steps to maximize cumulative reward.
  - eg. Defeat world champion at Backgammon.
    - → no intermediate reward but final +1/-1 for win/lose.

- Sequential decision making
  - Goal → select actions to max. future total ahead, it may mean to plan ahead and do short-term losses for long term gains.

- Agent & Environment
  - Goal → want to build such algo. that sits inside the agent.
  - responsible for taking the action





O_t → observation → sees something of the real world

R_t → Reward → how well it's doing at that step.

Environment → the world or the thing that the agent interacts with

↓ responsible for generating O_t & R_t

The only influence the agent can have over the environment is through its action, eg, where it is in the environment by → moving to

This goes on in a trial-and-error loop where the agent performs (tries) different actions and based upon its observations & reward, it learns what the most optimal actions are. This time series of O_t, R_t & A_t → experience of the agent and forms the data for the RL problem.

History → $H_t = A_1, O_1, R_1, \ldots A_t, O_t R_t$

Goal of our algorithm:

Build a mapping from this history to the next action that the agent should take.

Problem: RL agents may be living for a entire lifetime & hence going back to this history everytime is not feasible.

Solution: Define a 'state' → summary of the history to be used to determine next action.

$S_t = f(H_t)$ → function of History

Environment State: Information used within the environment to determine what happens next → i.e. some sort of representation (using numbers, etc.) of the environment to generate the next O_t & R_t.

However, this is not visible to the agent → i.e. agent/algorithm → should not be dependent on the environment state.

Thus, this doesn't influence building our algorithm.

Agent State → the set of numbers used to capture the summary of the information seen by the RL agent, which influences what action it should take next. → used for building our algorithm.

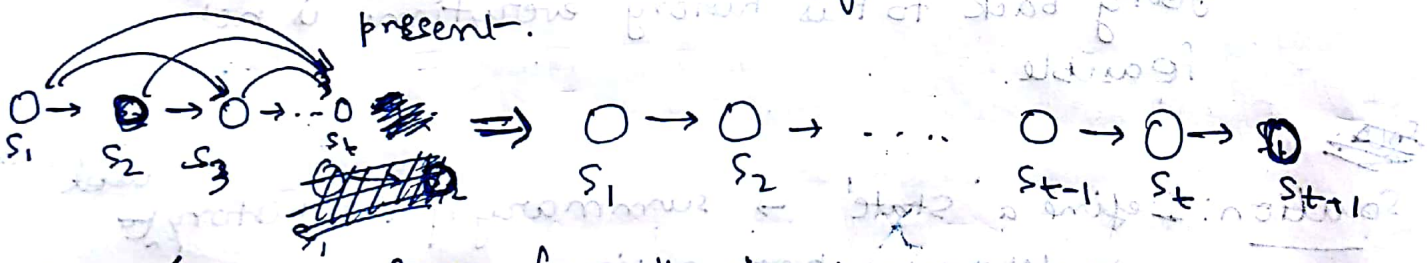
$$S_t^a = f(H_t).$$

Information state / markov state ⇒ contains all useful information from the history

State S_t is 'markov' if:

$$P(S_{t+1} \mid S_t) = P(S_{t+1} \mid S_1, \ldots, S_t)$$

i.e. future is independent of the past, given the present.



comes from the basic of independency, than $P(A \mid B) = P(A)$ if A & B are independent

→ $P(A \mid B, C) = P(A \mid B)$ → A & C are conditionally independent given B.

it is a simplifying assumption / property.

This means the state → S_t ⇐ $H_{1:t}$ — [$H_{1:t} = H_1 H_2 \dots H_t$]

is sufficient to determine the future → $H_{t+1:\infty}$ → determines

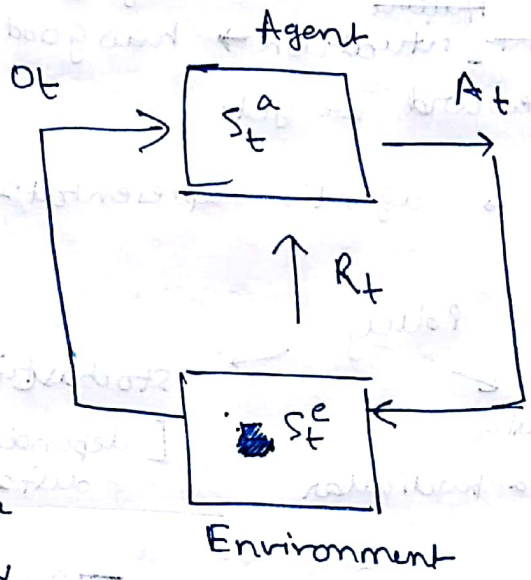
environment state → S_t^e → markov (by definition)

also → can always find markov states, need to see if we can find some that is practical.

Full observability →

agent directly observes environment state → $O_t = S_t^e = S_t^a$.

= information state.



This is the main formula behind RL → markov decision process (MDP).

Partial observability → agent indirectly observes the environment state.

agent state

$S_t^a \neq S_t^e$.

eg. Robot with camera → doesn't know precisely where it is → needs to figure out by moving & observing

Multiple ways to define state:

① Naive → entire history → $S_t^a = H_t$

② Probabilistic Bayesian approach → probability distr over which state it can be in.

③ RNNs. (comes in cs224n).

Inside an RL agent

Components (may / may not have)

- Policy → behaviour functn → decide from the current state, which action to take next.
- value functn → estimating how well we are doing at the future ~~current~~ situation → how good is each state/action/ what reward we get
- model → agent's representation the environment

Policy

← Deterministic
[fixed for a particular state]

$a = \pi(s)$

→ stochastic
[depending upon probability distn (distribution).

$\pi(a|s) = P(A=a \mid S=s)$

Value funcn → depends on what policy you choose
eg. robot falling down → less reward than robot which keeps moving

$$V_\pi(s) = E_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} \ldots \mid S=s]$$

[$E_X[x]$ ⇒ expected value of X ⇒ $\sum_{i=1}^{n} p_i x_i$

x → can take values $(x_1, \ldots x_n)$

and p_i → probability of $X = x_i$

$p_i = P(X = x_i)$]

γ → discounting factor

$\gamma < 1$ ∴ $\gamma, \gamma^2, \gamma^3$ → makes the contribution of future rewards subsequently less important than the current reward.

eg. $\gamma = 0.9$

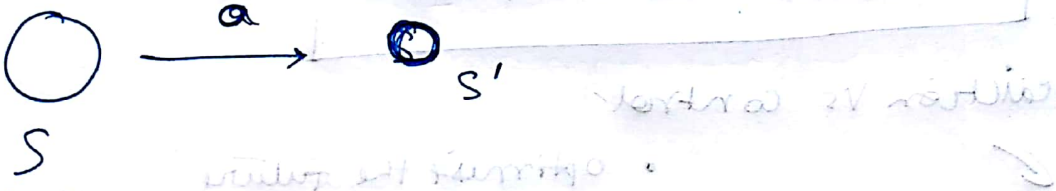
$R_t + 0.9\, R_{t+1} + 0.81\, R_{t+2} \cdots$

Model → Since s_t^a is not visible to the agent, it needs to imagine / predict what environment will do next.

- **Transitions** → Given current state, and action chosen, predict what the next state could be.

$$P_{ss'}^{a} = P[S' = s' \mid S = s, A = a]$$

State transition probability



- **Rewards** → predict next immediate reward for action

$$R_s^a = E(R \mid S = s, A = a)$$

RL agents taxonomy:

- → value based
  as value function depends on policy, policy is implicit
- → Policy Based
  - → no value function
- → Actor Critic
  - → separate policy
  - → separate value function.

- → model free
- → model Based.

2 fundamental problems in RL:

→ General RL
- No environment info known

→ Planning
- model of the environment is given and the agent performs its computation by interacting with this model (not with external environment).

eg. rules of Atari already given and we know what would happen (immediately) if we take a certain step. → Then, we could search over all possible soln's and choose the best ones.

[For those who know about AlphaGo, it uses Tree search to find the best possible move]

RL → exploration Vs exploitation

Prediction Vs control

- policy given
- predict what will happen if I follow this policy

- optimise the future
- choosing the policy (or finding)
- need to solve prediction problem to solve the control problem.