RL Lecture 2

Markov Decision Process

→ Markov Process:

Intro to MDPs:

- MDPs formally describe the environment for RL.
- Environment is fully observable. $O_t = S_t^e = S_t^a$.
- Almost all RL problems → can be formalized as MDPs.

Owing to the markov property,
the state captures all relevant information from the history.

State transition matrix:

$$P_{ss'} = P[S_{t+1} = s' \mid S_t = s]$$

P → transition probability →

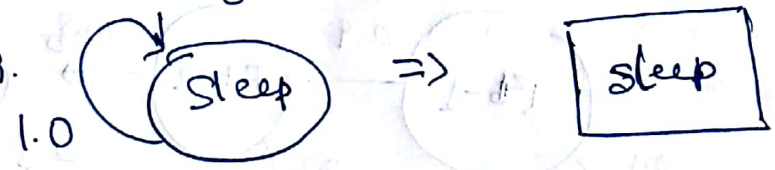
$$\begin{bmatrix} P_{11} & \cdots & P_{1n} \\ \vdots & & \\ P_{n1} & \cdots & P_{nn} \end{bmatrix}$$

markov process:

Random process (sequences of states $S_1, S_2 \ldots$) following the markov property.

Internal state → doesn't need any modelling

→ self-loop → eg.



Modifications:

→ probability is changing overtime.

2 ways of modelling

① model it as a non-stationery process.

[Stationery process:
For an ∞ sequence of states, it is called a stationery process when:

$$P(X_1, X_2, \ldots X_t) = P(X_{1+k}, X_{2+k}, \ldots X_{t+k})$$

basically

$\ldots (X_1, X_2 \ldots X_t) \ldots$

$\ldots (X_{1+k}, \ldots, X_{t+k}) \ldots$

t consecutive at any point of time.

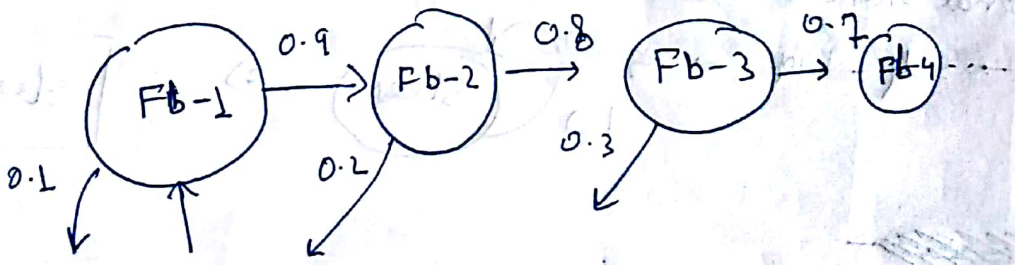
where $P(X_1, X_2 \ldots X_t)$ → joint distribution]

So, the 1st proposed sol² is to use non-stationary MDP and incrementally adjust your sol². (later)

(2) non-stationary dynamics ⇒ just a more complicated Markov process

eg.

Facebook in the simple case Becomes



Fb-i represents that you spent i seconds in that job (here, i seconds on facebook).

Markov Reward Process → represented by tuple (S, P, R, γ)

→ Markov chain with values

We care about the max of this reward ie. maximizing this reward.

S → set of states

P → transition probability

R → Reward funct^n (immediate) $= E(R_{t+1} | S_t = s)$

γ → discount factor

$G_T = R_t + \gamma R_{t+1} + \gamma^2 R_{t+1} \ldots$ (return)

γ^2 to make sure that it is finite.

$= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

Note: No expectat^n here, as G_T → is just a sample, not the overall value. ↘ at a time instant.

$\gamma = 0$ → maximally short / short sighted

$\gamma = 1$

(why)

→ mathematically convenient

→ model is not perfect → uncertainty of the future

G_T → means from that time-step onwards.

Value funct^n → long-term value of being in a state.

$v(s) \doteq E[G_T | S_T = s]$

R_{t+1} → indexing doubt → just a convention

→ stick to the concept that:
- move to state
- get a reward.

Bellman equat^n for MRPs:

Value funct^n → / two parts

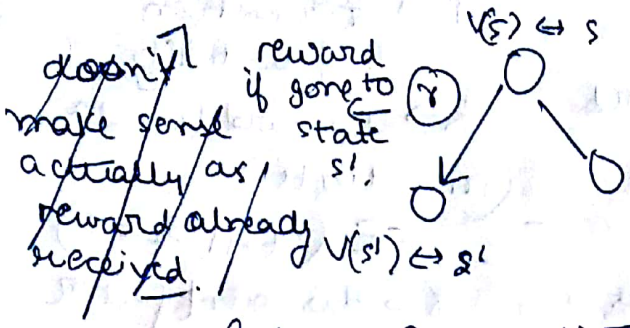
R_{t+1} ↘ discounted value of successor state → $\gamma v(s_{t+1})$

$v(s) = E[G_T | S_t = s]$

$= E[R_{t+1} + \gamma G_{t+1} | S_t = s] = E[R_{t+1} + \gamma v(s_{t+1}) | S_t = s]$

Value funct needs to obey Bellman equations

Backup diagram ⇔ one step look ahead search



$$V(s) = R_s + \gamma \sum_{s' \in S} P_{ss'} V(s')$$

matrix formulation:

$V = R + \gamma P V \rightarrow O(n^3) \rightarrow$ not feasible to compute directly

$V = (1 - \gamma P)^{-1} R$

→ use DP / Monte-Carlo / Temporal-~~Difference~~ ~~Process~~ Difference Learning

Till now, no actions

Enter: MDP → MRP with decisions (or actions).

Environment in which all states are Markov.

add A to tuple ↘ set of actions.

$$P^a_{ss'} = P(S_{t+1} = s' \mid S_t = s, A_t = a)$$

↓ dependent on 'a' now

$$R_s^a = E[R_{t+1} \mid S_t = s, A_t = a]$$

We need to formalize the meaning of making decisions and taking decisions.

↓

Policy → distr over ~~agent~~ actions given states.

$$\Pi(a \mid s) = P(A_t = a \mid S_t = s)$$

→ markov policy → only dependent on current state & not the time step you are in.

policy → fully characterizes the behaviour of an agent.
& same at each time instant → just depends on the state.

[future rewards → fully characterized by state].

We can always recover a MRP from a given MDP.

No matter what policy chosen, it defines some chain of ~~sequences~~ states which ultimately form a markov chain

& sequences of states & rewards ⇒ MRP $\langle S, P^\pi, R^\pi, \gamma \rangle$

→ Just average over the policy.

$$P^{\pi}_{ss'} = \sum_{a \in A} \pi(a|s) P^{a}_{ss'}$$

$$R^{\pi}_{s} = \sum_{a \in A} \pi(a|s) R^{a}_{s}$$

Earlier, value funct^n didn't have any action associated with it. Now:

① state value function: $V_\pi(s)$ → expected return starting from state s and following policy π → sampling actions according to π.

$$V_\pi(s) = E_\pi(G_T | S_T = s)$$

② action-value function → expected return starting from state s, taking action a & following policy π.
[$q_\pi(s,a)$]

$$q_\pi(s,a) = E_\pi[G_T | S_t = s, A_t = a]$$

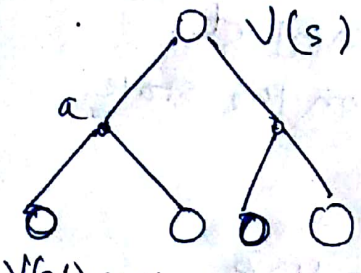
[Difference: here action at this state is chosen as 'a' and then find the expectation].

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s]$$

$$q_\pi(s,a) = E_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1}) | S_t = a, A_t = a].$$

$$V_\pi(s) = \sum_{a \in A} \pi(a|s) q_\pi(s,a)$$

$$q_\pi(s,a) = R^{a}_{s} + \gamma \sum_{s' \in S} P^{a}_{ss'} V_\pi(s')$$



$$V_\pi(s) = \sum_{a} \pi(a/s) \left[R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_\pi(s') \right]$$

$$q_\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a \sum_{a' \in A} \pi(a'/s') \, q_\pi(s',a')$$

↳ Recursive relation.

1 way to solve → use the matrix formulation:

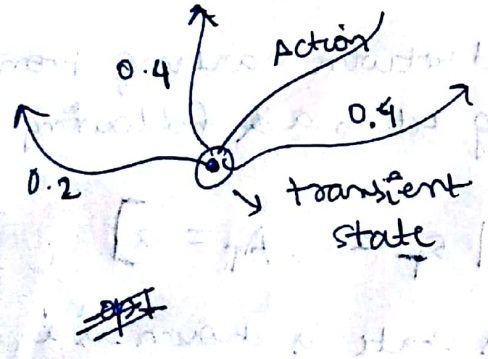
$V_\pi = R^\pi + \gamma P^\pi V_\pi$ (framing MDP as a MRP)

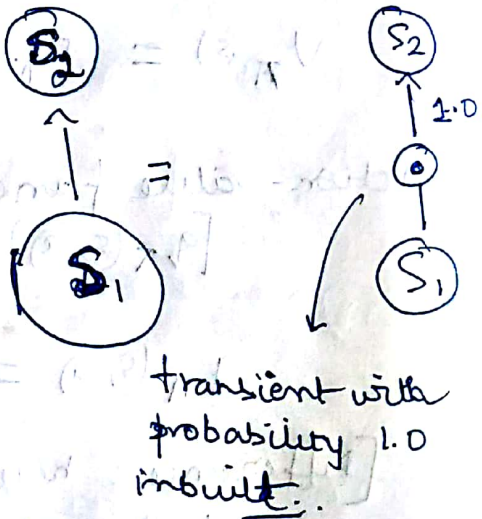
$V_\pi = (1 - \gamma P^\pi) + R^\pi$ → more efficient methods later.

But how to get the optimal?

Reward → dependent on action → R_s^a

i.e actions can decide what happens with the immediate rewards as well.





Optimal value function : $V_*(s) = \max_\pi V_\pi(s)$

$q_*(s,a) = \max_\pi q_\pi(s,a)$ → max, following all the different kinds of policy possible

Knowing the optimal value function

⇒ MDP solved!!

Optimal policy

↳ best possible way to behave

mapping from state to action

Which is best? Partial ordering over policies:

What is optimal? $\pi \geq \pi'$ if $V_\pi(s) \geq V_{\pi'}(s) \; \forall s$,

Imp theorem:

→ ∃ atleast 1 optimal policy $\pi_* \geq \pi \; \forall \pi$

→ can be more than 1 optimal policies → all with same value funct^n: $V_{\pi_*}(s) = V_*(s)$

→ they also get the optimal action-value funct^n:

$q_{\pi_*}(s,a) = q_*(s,a)$

Finding → $\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \max_{a \in A} q_*(s,a) \\ 0 & \text{else.} \end{cases}$

↓

max over $q_*(s,a)$ → Solve for q_* → done ✓

→ ~~also~~ always a deterministic optimal policy

How to get $q_*(s,a)$? Before, we looked at Bellman expectation equation.

Bellman optimal equat^n

→ 1 step look ahead

optimal value funct^{n}s → recursively related.

$$V_*(s) = \max_{a \in A} q_*(s,a)$$

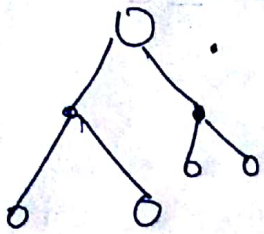
$$q_*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a V_*(s')$$

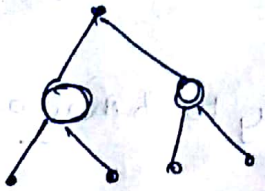
$$V_*(s) = \max_a \left(R_s^a + \gamma \sum_{s'} P_{ss'}^a V_*(s') \right)$$

$$q_*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} q_*(s',a')$$

makes eqn non-linear. Hence, use iterative methods

value iteration, policy iteration, Q-learning, etc...





(Risk sensitive MDPs?)