Lecture-19

State-action reward

$R : S \longmapsto R$ (Previously new state maps to reward)

Now:-

$R : S \times A \longmapsto R$ (Now state & action maps to reward)

$$R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2)$$

↳ Different actions might have different rewards

→ Bellman's equation: (Updated)

$$V^*(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^*(s') \right]$$

- $R(s,a)$: Current Reward
- $\sum_{s'} P_{sa}(s') V^*(s')$: Future Reward
- $\max_a$: Max over the whole equation

⊛ Remeber reward depends on current reward position.

⊛ Value iteration can still be used

⊛ $\pi^*(s) = \underset{a}{\text{argmax}} R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^*(s')$

argmax on the whole equation

⊛ Finite horizon MDP:-

→ $(S, A, \{P_{sa}\}, T, R)$

→ We are going to replace γ with T.

→ ~~$s_0 a_0$~~

→ MDP would run for a definate period of time:-

$s_0 a_0 \; s_1 a_1 \cdots s_T a_T$ → World Ends

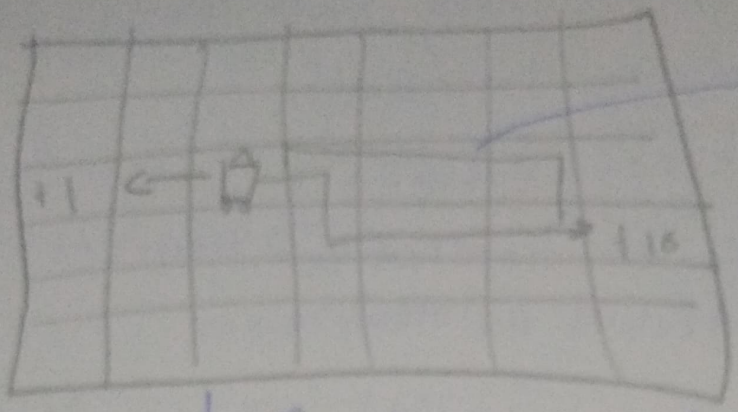
$[R(s_0, a_0) + R(s_1, a_1) + \cdots R(a_T, s_T)]$

↳ Finite Reward

↳ $E[R(s_0, a_0) + R(s_1, a_1) + \cdots R(a_T, s_T)]$

↳ Goal is to maximize this expected payoff.

(x)



↳ Sometime reward depends on time. If you have two to three time steps left then you should rush towards +1 but if you more than that than you should rush towards +10.

→ Stationary Policy does not change over time

↳ $\pi^*(s)$ depends on time.

↳ So $\pi^*(s)$ should be written as $\pi^{*(t)}(s)$ (non-stationary policy/ changes over time).

(*) Non-stationary state transitions:

① $S_{t+1} \sim P^{(t)}_{S_t a_t}$

↳ Non-stationary state-transition Probability

② $R^{(t)}(s, a)$

↳ Reward depends on time.

* When do you need finite horizon or non-stationary transitions:-

→ Changing dynamics.
- Weather forecast (Different environment)
- Industrial automation.
-

→ How you would solve finite horizon MDP:-

$$V_t^*(s) = E[R(s_t, a_t) + R(s_{t+1}, a_{t+1}) + \cdots + R(s_T, a_T) \mid \pi^*, s_0 = s]$$

↑ Expected total payoff starting s at time t, execute π^*.

→ Value iteration: (Dynamic Programming)
in this case evolves:-

$$\rightarrow V_t^*(s) = \max_a R(s,a) + \sum_{s'} P_{sa}(s') V_{t+1}^*(s')$$

$$\rightarrow \pi^*(s) = \arg\max_a (\ldots)$$

→ So let start with $V_T^*(s)$ the T is last timestep:-

$$V_T^*(s) = \max_a R(s,a)$$

↳ No time step after this

→ The essential idea is that we first compute $V_T^*(s)$ than $V_{T-1}^*(s)$ then V_{T-2}^* and so on... till V_0^* and then compute $\Lambda_t^*(s)$ for each step. ~~for each step Ea station~~ for each step.

↳ How Value iteration for finite time horizon changes.

» Linear Quadratic Regression (LQR):-

① $(S, A, \{P_{sa}\}, T, R)$ → finite horizon MDP

② $S \in \mathbb{R}^n$ $A \in \mathbb{R}^d$

③ P_{sa}: $S_{t+1} = AS_t + Ba_t + W_t$ → Next state evolves as function of $P_T \cdot S_t$ and a_t.

$A \in \mathbb{R}^{n \times n}$ $B \in \mathbb{R}^{n \times d}$ $W \sim (0, \Sigma_w)$

④ $R(s,a) = -(s^T U s + a^T V a)$

↓ Another Assumption

$U \in \mathbb{R}^{n \times n}$, $V \in \mathbb{R}^{d \times d}$

$U, V \geq 0$ $s^T U s \geq 0$, $a^T V a \geq 0$

* Want $s \approx 0$ (We want this to be stable)

Then Choose $U = I$, $V = I$

then $R(s,a) = -(\|s\|^2 + \|a\|^2$

↳ Reward penalizing large movement from stable state.

→ Just an example.

* Where to get A, B? → Method 1 - Learning A, B

Collecting Data ←

$$s_0^{(1)} \xrightarrow{a_0} s_1^{(1)} \longrightarrow \cdots s_T$$

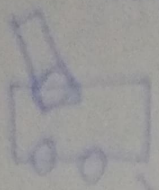
$$s_0^{(m)} \xrightarrow{a_0} s_1^{(m)} \longrightarrow \cdots s_T^{(m)}$$

$$\rightarrow s_{t+1} = As_t + Ba_t$$

$$\rightarrow \min_{A,B} \sum_{i=1}^{m} \sum_{t=0}^{T} \left\| \left(As_t^{(i)} + Ba_t^{(i)}\right)^2 - s_{t+1}^{(i)} \right\|^2$$

↳ You will be able to find A and B.

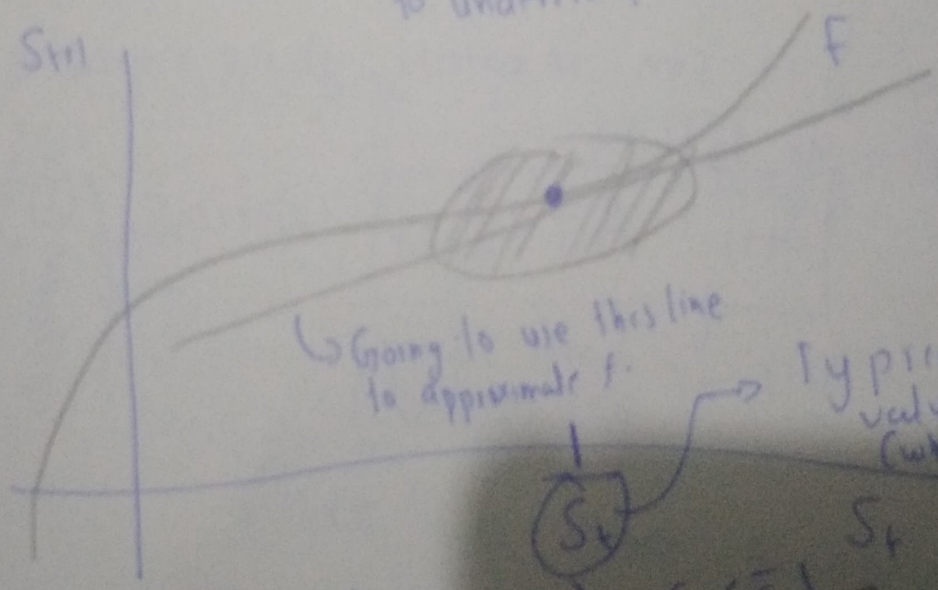
④ → Method 2 (Find A, B) — Linearize a non-linear model



$S_{t+1} = f(S_t, a_t)$ → Physics Law of simulator

$$\begin{pmatrix} x_{t+1} \\ \dot{x}_{t+1} \\ \theta_{t+1} \\ \dot{\theta}_{t+1} \end{pmatrix} = f\left(\begin{pmatrix} x_t \\ \dot{x}_t \\ \theta_t \\ \dot{\theta}_t \end{pmatrix}, a_t \right)$$

→ $S_{t+1} = f(S_t)$ (Ignoring action for now) to understand linearization



⊙ $S_{t+1} \approx f'(\bar{S}_t)(S_t - \bar{S}_t) + f(\bar{S}_t)$ (Approximating f)

$\bar{S}_t$ is a constant

* And if $S_{t+1} = f(S_t, a_t)$ then linearized equation would be: (Linearize f around $\bar{S}_t, \bar{a}_t$)

~~$S_{t+1} \approx f'(\bar{S}_t, \bar{a}_t) \cdot (S_t - \bar{S}_t) + f(\bar{S}_t, \bar{a}_t) \cdot (a_t - \bar{a}_t)$~~

$$S_{t+1} = f(\bar{S}_t, \bar{a}_t) + \left(\nabla_{S_t} f(\bar{S}_t, \bar{a}_t)\right)^T \cdot (S_t - \bar{S}_t) + \left(\nabla_{a_t} f(\bar{S}_t, \bar{a}_t)\right)^T \cdot (a_t - \bar{a}_t)$$

↳ Approximation of orginal function (Expression S_{t+1} as linear function S_t & a_t)

→ You can rexpress above equation as follow

$$\boxed{S_{t+1} = AS_t + Ba_t}$$

↳ Same answer as method 1.

where S_t ① → Need to add a intercept term.

→ the above methods show you model finite horizon MDP as linear ~~dynamical~~ dynamical model

→ Main Problem

① $S_{t+1} = AS_t + Ba_t + W_t$

and if

② $R(s,a) = -(s^T U s + a^T V a)$

Total Pay off:-

③ $R(s_0, a_0) + R(s_1, a_1) + \cdots R(s_T, a_1)$

→ Dynamic Programming algorithm to solve for V^*:-

(Final time step) $V_T^*(s) = \max_{a_T} R(s_T, a_1)$

$= \max_{a_T} -s_T^T U s_T - a_T^T V a_T$

← Choose action to maximize $V_T^*(s)$ – Value of last time

$= -s_T^T U s_T$

$\pi_T^*(s_T) = 0$ (choose action 0).

→ Suppose

$$V^*_{t+1}(s_{t+1}) = s_{t+1}^T \Phi_{t+1} s_{t+1} + \psi_{t+1}$$

where

$$\Phi_{t+1} \in \mathbb{R}^{n \times n}, \quad \psi_{t+1} \in \mathbb{R}$$

Given MDP is linear System and reward function has the form we mentioned $V^*_{t+1}(s_{t+1})$ would be Quadratic.

Assuming continous state.

$$V^*_t(s_t) = \max_{a_t} R(s_t, a_t) + E_{s_{t+1} \sim P_{s_t a_t}} \left[V^*_{t+1}(s_{t+1}) \right]$$

→ Now Putting in equations:-

$$V^*_t(s_t) = \max_{a_t} -s_t U s_t - a_t V a_t + E_{s_{t+1} \sim P_{s_t a_t}} [C]$$

Continue on next page for clarity

(x)

$$V_t^*(S_t) = \max_{a_t} -S_t U S_t - a_t V a_t + E_{S_{t+1} \sim N(AS_t + Ba_t, \Sigma_w)} [\ldots$$

↓ Remember from past linear model

$$\ldots S_{t+1}^T \Phi_{t+1} S_{t+1} + \Psi_{t+1}]$$

$V_{t+1}^*(S_{t+1})$

→ This simplifies to a big quadratic function of a_t

↳ Take derivate wrt a_t
Set to zero and solve for a_t
then you end with following a_t formula

$$a_t = \underbrace{(B^T \Phi_{t+1} B - V)^{-1} B^T \Phi_{t+1} A}_{L_t} \cdot S_t$$

therefore

$$\Pi_t^*(S_t) = L_t \cdot S_t$$

→ Optimal action is a linear function of state S_t if you have a linear dynamical MDP.

→ If put the optimal action into $V_t^*(S_t)$. then:

$$V_t^*(S_t) = S_t^T \Phi_t S_t + \Psi_t$$

where

$$\Phi_t = A\left(\Phi_{t+1} - \Phi_{t+1} B\left(B^T \Phi_{t+1} B - \nu\right)^{-1} B \ldots\ldots B\Phi_{t+1}\right) A_t - U$$

$$\Psi_t = -tr \sum \omega_t \Phi_{t+1} + \Psi_{t+1}$$

ance
noise system.

→ Summarizing the whole algorithm:

Next page

→ Intialize $\underline{\Phi}_T = -U$, $\gamma_T = 0$

Recursively calculate

$\underline{\Phi}_t$, γ_t using $\underline{\Phi}_{t+1}$, γ_{t+1} (for $t = T, T$

, $t-2 \cdots 0$) (Equations defined above

Calculate a_t

$\Lambda^*(S_t) = L_t S_t$

→ Curx to solve a finite horizon MDP (But MDP has to be approximated to linear dynamical system).