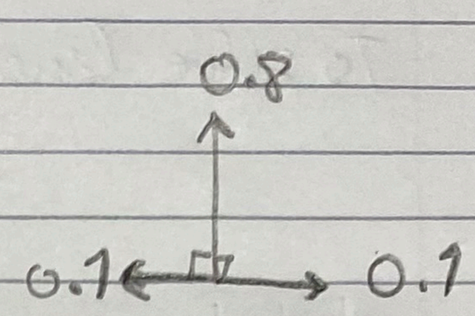
Lecture 17

Reinforcement Learning.

- MDPs (recap).
- Value function.
- Value iteration/Policy iteration.
- Learning s.t.p./putting it all together.



MDP: $(S, A, \{P_{st}\}, \gamma, R)$

state, actions, transition probabilities, discount factor, Reward

3	→	→	→	+1
2	↑		↑	−1
1	↑	←	←	←
	1	2	3	4

s_0, a_0
$s_1 \sim P_{s_0 a_0}, a_1$
$s_2 \sim P_{s_1 a_1}, a_2$
$\vdots$

$\Rightarrow R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$

Policy: $\pi : S \mapsto A$. Eg $\pi((3,1)) = \text{"}\leftarrow\text{"}$

Define: V^π, V^*, π^*

V^π: For a policy π, $V^\pi : S \to \mathbb{R}$ is st $V^\pi(s)$ is the expected total payoff for starting in state s and executing π.

$V^\pi(s) = E[R(s_0) + \gamma R(s_1) + \dots \mid \pi, s_0 = s]$

Eg $\hat{\pi}$:

→	→	→	+1
↓		→	−1
→	→	↑	↑

is a pretty bad policy, and

$V^{\hat{\pi}} =$

.52	.73	.77	+1
−.1		−.82	−1
−.88	−.87	−.85	−1

Bellman's equations:

$$V^\pi(s) = R(s_0) + \gamma \overbrace{\sum_{s'} P_{s\pi(s)}(s') V^\pi(s')}^{\text{future reward}} \quad (I)$$

$$V^\pi(s) = E[\underbrace{R(s_0)}_{\text{immediate reward}} + \gamma \underbrace{(R(s_1) + \gamma R(s_2) + \dots)}_{\sim V^\pi(s_1)} \mid \pi, s_0 = s] \quad (II)$$

Mapping between Eqs (I) & (II): $s \leftrightarrow s_0$; $s' \leftrightarrow s_1$

$\Rightarrow V^\pi(s) = E[R(s) + \gamma V^\pi(s')]$, $s' \sim P_{s\pi(s)}$.

Given π, get a linear system of eqs in terms of $V^\pi(s)$.

$\square$ = unknown variables of linear system.

$$\Rightarrow \boxed{V^{\pi}((3,1))} = R((3,1)) + \gamma\left(0.8\boxed{V^{\pi}((3,2))} + 0.1\boxed{V^{\pi}((2,1))} + 0.1\boxed{V^{\pi}((4,1))}\right)$$

To solve linear system: $\begin{bmatrix} V^{\pi}((1,1)) \\ V^{\pi}((1,2)) \\ \vdots \\ V^{\pi}((4,3)) \end{bmatrix} = x$ must be solved for a system of 11 eqs.

$\underline{\underline{V^*}}$: V^* = optimal value function $\Rightarrow V^*(s) = \max_{\pi} V^{\pi}(s)$

New v. Bellman's equations:

$$V^*(s) = R(s) + \max_a \underbrace{\left(\gamma \sum_{s'} P_{sa}(s') V^*(s')\right)}_{\text{expected future reward}}$$

$$\pi^*(s) = \arg\max_a \gamma \sum_{s'} P_{sa}(s') V^*(s')$$

Practise (w/ confusing notation):

$$V^*(s) = \underbrace{V^{\pi^*}(s)}_{\text{value function of } \pi^*} \geq V^{\pi}(s) \quad \text{for every } \pi, s$$

Strategy:

1) Find V^*; 2) Use argmax equation to find π^*.

Value iteration:

- Initialise $V(s) := 0$ for every s.
- For every s, update
  $V(s) := R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V(s')$

$\begin{bmatrix} V((1,1)) \\ \vdots \\ V((4,3)) \end{bmatrix} \in \mathbb{R}^{11}$. "Synchronous update": update all 11 values simultaneously.

"Asynchronous update": update one at a time.

- Bellman backup operator $V := B(V)$

Value iteration convergence: V to converge to V^*.

Running VI on the MDP will give

$V^* =$

.86	.90	.93	+1
.81		.69	−1
.78	.75	.71	.49

& $\pi^* =$

→	→	→	+1
↑		↑	−1
↑	←	←	←

$s = (3,1)$ & $a = \leftarrow$: $\sum_{s'} P_{sa}(s') = .8 \times .75 + .1 \times .69 + .1 \times .71 = \underline{.740}$;
while $a =$ "↑" gives $\underline{.676}$

Policy iteration:
Initialize π randomly.
Repeat:
Set $V := V^{\pi}$ (ie, solve Bellman's eqs to get V^{π})
Set $\pi(s) := \arg\max_a \sum_{s'} P_{sa}(s') V(s')$

Since V^{π} is a lin. sys. of eqs., only need to do some inversion.

VI vs PI: PI better for smaller systems, due to inversion then being cheap & quick convergence. VI better for very large systems.

What if you don't know P_{sa}?

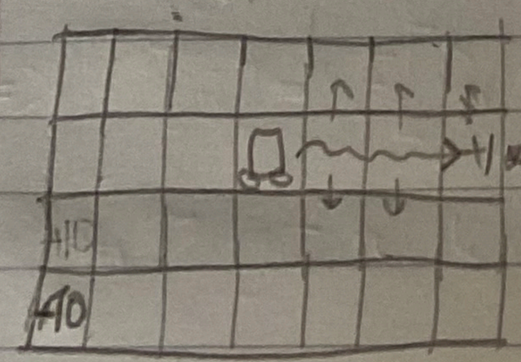
Need to estimate from data. ⇒ $P_{sa}(s') = \dfrac{\#\text{ times took action "a" in state s and got to s'}}{\#\text{ times took action "a" in state s}}$, for PI/VI

or $\frac{1}{|S|}$ if above is "$\frac{0}{0}$" (Laplace smoothing would also work, but unnecessary).

Putting it together:
Repeat: {
Take actions wrt π to get experience in MDP.
Update estimates of P_{sa} (and possibly R)
Solve Bellman eqs using VI to get V
Update $\pi(s) = \arg\max_a \sum_{s'} P_{sa}(s') V(s')$
}

Exploration vs Exploitation

Exploration policy: ε-greedy ≈ 0.1 (strange name)
0.9 chance wrt π
0.1 chance randomly



−.02 at every step
"Greedy" & exploitative

CS229 Lecture 18

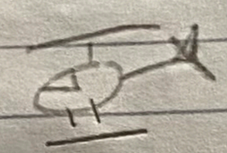
- Continuous state MDPs
- Models/simulation
- Fitted value iteration

(simple) state-space of car

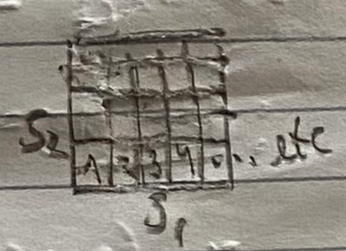
$(x, y, \theta; \dot{x}, \dot{y}, \dot{\theta})$

simple state-space of Helicopter

$(x, y, z, \phi, \theta, \psi; \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi})$

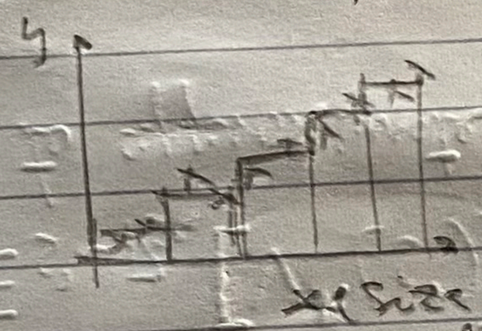


Discretization:



Problems:

- Naive representation for V^* & π^*



- Curse of dimensionality: $\mathbb{R}^n$, and discretize each dim into k values, we get k^n discrete states.

Guidelines: 2,3 is ok. 4-6 is maybe. 7-8: Nah.
Refine most important dims.

- Approximate V^* directly, without resorting to discretisation.

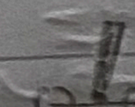
Eg $y \approx \theta^T x$, or $\theta^T \underbrace{\phi(x)}_{\text{features of } x}$

$\Rightarrow V^*(s) = \theta^T \phi(s)$

Model (simulator) MDP:

$s \longrightarrow \boxed{\text{model}} \longrightarrow s' \sim P_{sa}$, $\uparrow a$

How to get model?



$s = \begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}$, $\dot{s} = \begin{pmatrix} \dot{x} \\ \alpha - L\beta\cos\theta/m \\ \dot{\theta} \\ \beta \end{pmatrix}$

physical parameters

$s' = s + \Delta t \cdot \dot{s}$ w/ $\Delta t =$ eg 0.1 seconds

- Can also get model by learning from data.

$$s_0^{(1)} \xrightarrow{a_0^1} s_1^1 \xrightarrow{a_1^1} s_2^1 \xrightarrow{a_2^1} s_3^1 \to \dots \to s_T^1$$
$$s_0^2 \xrightarrow{a_0^2} s_1^2 \xrightarrow{a_1^2} s_2^2 \xrightarrow{a_2^2} s_3^2 \xrightarrow{a_3^2} \dots \to s_T^2$$
$$\vdots$$
$$s_0^m \xrightarrow{a_0^m} \dots$$

Now, apply supervised learning to estimate $\overset{(s')}{s_{t+1}}$ as suggestion @ $\overset{(s,a)}{s_t, a_t}$

Eg linear regression model: (actually ok at low speeds / to 1st order)

$$s_{t+1} = \overset{\in \mathbb{R}^{n \times n}}{A} s_t + B a_t$$

$$\min_{A,B} \sum_{i=1}^{m} \sum_{t=0}^{T} \| s_{t+1}^i - (A s_t^i + B a_t^i) \|^2$$

Model:

$$s_{t+1} = A s_t + B a_t \quad \text{or} \quad s_{t+1} = A s_t + B a_t + \varepsilon_t, \ \varepsilon_t \sim \mathcal{N}(0, \sigma^2 I)$$

deterministic — stochastic — eg windgusts

$$s_t \to \boxed{\text{model}} \to s_{t+1} \sim P_{sa}$$

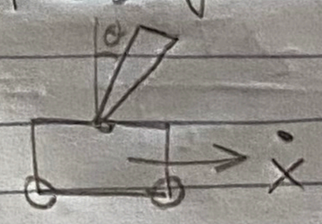
a_t

This is model-based RL

- Fitted value iteration:

Choose feature $\phi(s)$ of states: $V(s) = \theta^T \phi(s)$

$$\phi(s) = \begin{bmatrix} x \\ x \\ x \\ x \\ \vdots \end{bmatrix}$$



Value iteration: $V(s) := R(s) + \gamma \max_a \sum_{s'} P_{sa}(s') V(s') = R(s) + \gamma \max_a \underbrace{E_{s' \sim P_{sa}}[V(s')]}$

Fitted VI: Sample $\{s^1, s^2, \dots, s^m\} \subseteq S$ randomly, & initialise $\theta := 0$

Repeat {

For $i = 1, \ldots, m$ {

For each action $a \in A$, {

Sample $s'_1, s'_2, \ldots, s'_k \sim P_{s_i a}$ using model

Set $q(a) = \frac{1}{k} \sum_{j=1}^{k} [R(s_i) + \gamma V(s'_j)]$

(= Estimate of $\max_a E_{s' \sim P_{sa}} [R(s) + \gamma V(s')]$)

(indent) Set $y^i = \max_a q(a)$

}

(In original VI: $V(s^i) := y^i$ / In fitted VI: Want $V(s^i) \approx y^i$

Ie $\theta^T \phi(s^i) \approx y^i$)

(indent) $\theta := \arg\min_\theta \frac{1}{2} \sum_{i=1}^{m} (\theta^T \phi(s^i) - y^i)^2$

}

Fitted VI gives approximation to V^*.

Implicitly defines π^*:

$$\pi^*(s) = \arg\max_a E_{s' \sim P_{sa}} [V^*(s')]$$

Used $s'_1, \ldots, s'_k \sim P_{sa}$ to approximate expectation

Say model is:

$$s_{t+1} = f(s_t, a_t) + \varepsilon_t \quad (\text{eg } As_t + Ba_t + \varepsilon_t)$$

For deployment/run-time,

set $\varepsilon_t = 0$, and $k = 1$

When in state s,

Pick action

$$a = \arg\max_a V(\underbrace{f(s,a)}_{s' \sim P_{sa}, \text{ but deterministic simulator}})$$

Simulator without noise

Lecture 19

Outline:

- State-action rewards
- Finite horizon MDP
- Linear dynamical systems
  - Model
  - LQR (linear quadratic regulation)

MDP: $(S, A, \{P_{sa}\}, \gamma, R)$

Value iteration:

$$V(s) := R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V(s')$$

$$\pi^*(s) = \arg\max_a \sum_{s'} P_{sa}(s') V(s')$$

State-action rewards

$R: S \times A \mapsto \mathbb{R}$

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

Bellman's eqs:

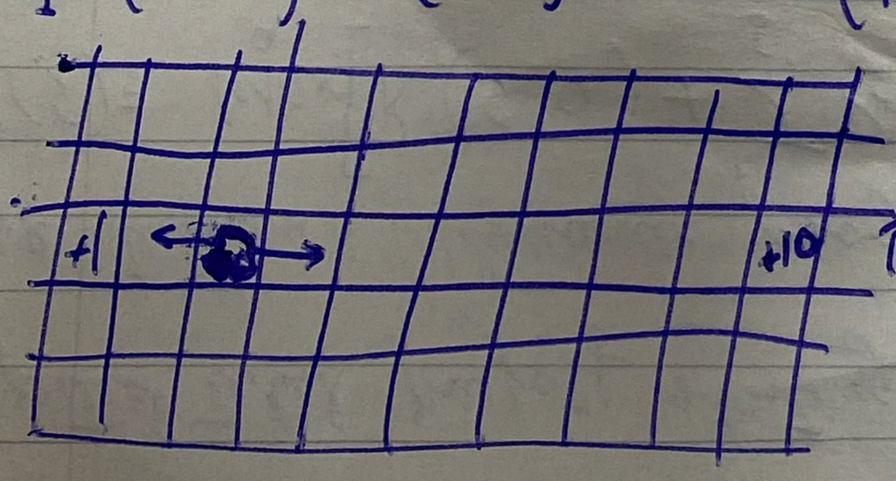
$$V^*(s) = \max_a \Big[\underbrace{R(s,a)}_{\text{immediate reward}} + \underbrace{\gamma \sum_{s'} P_{sa}(s') V^*(s')}_{\text{future rewards}} \Big]$$

Value iteration: $V(s) := RHS \Rightarrow \pi^*(s) = \arg\max_a \Big[R(s,a) + \gamma \sum_{s'} P_{sa}(s') V^*(s') \Big]$

Finite horizon MDP: $(S, A, \{P_{sa}\}, T, R)$ Horizon time T

(no γ):

$$\text{Payoff} = E\big[R(s_0, a_0) + R(s_1, a_1) + \dots R(s_T, a_T)\big].$$



$\pi_t^*(s)$ — stationary policy

$\pi_t^*(s)$ — Non-stationary policy

→ changes over time

Nonstationary state transitions: $s_{t+1} \sim P^{(t)}_{s_t, a_t}$ $\quad R^t(s,a)$

Reasons for change:
- changing dynamics
- industrial automation
- weather forecasts

Lec 19 Cont

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

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

Value iteration: (Dynamic programming)

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

$$\pi^*(s) = \arg\max_a \left[\quad \right]$$

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

By induction, V_T^* gives $V_{T-1}^*, \dots, V_0^*$.

Linear quadratic regulation (LQR)

$(S, A, \{P_{sa}\}, T, R)$

$S = \mathbb{R}^n$, $A = \mathbb{R}^d$

$\omega_t \sim \mathcal{N}(0, \Sigma_w)$

P_{sa}: $s_{t+1} = \underbrace{A}_{\mathbb{R}^{1\times n}} s_t + \underbrace{B}_{\mathbb{R}^{1\times d}} a_t + \omega_t$ (matrices)

$R(s,a) = -(s^T U s + a^T V a)$

where $U \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{d\times d}$ & $U, V \succeq 0$ (psd)

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

Want $s \approx 0$

Choose $U = I$, $V = I$.

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

Where to get A, B?

$s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \to \dots \to s_T^{(1)}$

$\vdots$

$s_0^{m} \xrightarrow{a_0^{m}} s_1^{m} \to \dots$

Lec 19 Cont 2

$$\min_{A,B} \sum_{i=1}^{m} \sum_{t=0}^{T-1} \| s_{t+1} - (A s_t + B a_t) \|^2 , \quad s_{t+1} \approx A s_t + B a_t$$

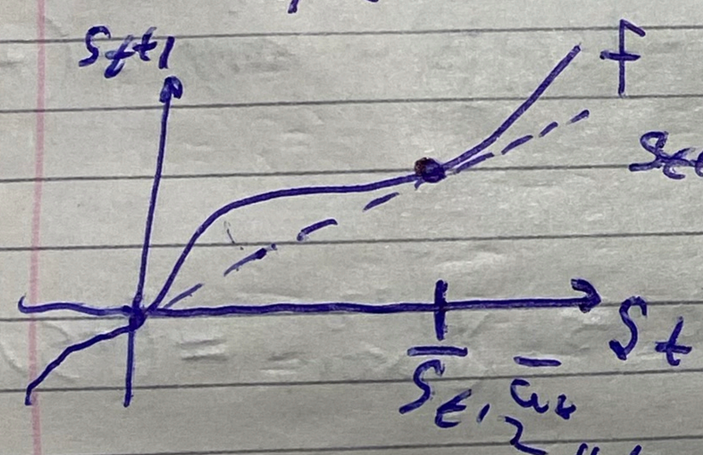
- Learn from data

- Linearize a non-linear model

$$s_{t+1} = f(s_t, a_t)$$

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

$$s_{t+1} = f(s_t, \cancel{a_t})$$



$$s_{t+1} \approx f'(\bar{s}_t) \cdot (s_t - \bar{s}_t) + f(\bar{s}_t)$$

"typical value for s"

Now including a_t: linearize f around $(\bar{s}_t, \bar{a}_t)$: $s_{t+1} \approx f(\bar{s}_t, \bar{a}_t)$

$$+ (\nabla_s f(\bar{s}_t, \bar{a}_t))^T (s_t - \bar{s}_t) \rightsquigarrow s_{t+1} = A s_t + B a_t$$

$$+ (\nabla_a f(\bar{s}_t, \bar{a}_t))^T (a_t - \bar{a}_t)$$

$$s_t = \begin{pmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{pmatrix}$$

Problem:

$$s_{t+1} = A s_t + B a_t + \overset{\text{noise}}{\omega_t}$$

$$R(s,a) = -(s^T U s^T + a^T V a)$$

$$\text{Total payoff} = \sum_{k=0}^{T} R(s_k, a_k)$$

Lec 19 Cont 3

Dynamic Programming:

$$V_T^*(s_T) = \max_{a_T} R(s_T, a_T)$$

$$= \max_{a_T} -s_T^T U s_T - \underbrace{a_T^T V a_T}_{\geq 0}$$

$$= -s_T^T U s_T$$

$$\Rightarrow \pi_T^*(s_T) = \vec{0}$$

Suppose

$$V_{t+1}^*(s_{t+1}) = s_{t+1}^T \Phi_{t+1} s_{t+1} + \psi_{t+1}, \text{ where } \Phi_{t+1} \in \mathbb{R}^{n \times n}, \psi_{t+1} \in \mathbb{R}$$

Want $V_{t+1}^* \to V_t^*$

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

$$V_t^*(s_t) = \max_{a_t} -s_t^T U s_t - a_t^T V a_t + E_{s_{t+1} \sim N(\mathbb{A} s_t + \mathbb{B} a_t, \Sigma_\omega)} \underbrace{\left[s_{t+1}^T \Phi_{t+1} s_{t+1} + \psi_{t+1} \right]}_{V_{t+1}^*(s_{t+1})}$$

$s_{t+1} = \mathbb{A} s_t + \mathbb{B} a_t + \omega_t$, $\omega_t \sim N(0, \Omega_t)$

. Turns out this expectation value simplifies into a big quadratic function of a_t.

Can take derivative wrt a_t, set to 0, solve for a_t. This algebra will eventually result in:

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

$$\Rightarrow \pi_t^*(s_t) = L_t s_t.$$

Optimal action is a linear function of the state s_t

$V_t^*(s_t) = s_t^T \Phi_t s_t + \psi_t$, where $\Phi_t = \mathbb{A}(\Phi_{t+1} - \Phi_{t+1} \mathbb{B}(\mathbb{B}^T \Phi_{t+1} \mathbb{B} - U)^{-1} \mathbb{B} \Phi_{t+1}) \mathbb{A}^T$ & $\psi_t = -tr\{\Sigma_\omega \Phi_{t+1} + \psi_{t+1}\}$

Lec Cont 4

To summarise:

Initialize $\Phi_T = -U$, $\Psi_T = 0$

Recursively calculate

Φ_t, Ψ_t using Φ_{t+1}, Ψ_{t+1} (for $t = T-1, T-2, \ldots, 0$)

Calculate L_t

$\pi^*(s_t) = L_t s_t$

<u>Interesting to note:</u>

1. V^* depends on Σ_w
2. π^*, L_t don't depend on Σ_w $\Rightarrow$ estimating noise isn't that important in L

Lecture 20

Outline

- RL debugging / diagnostics
- Policy search over POMDPs
- Conclusion

Helicopter ~~simulator~~ RFL algo of heli.stanford.edu

Assume real-life helicopter flies poorly. Then the diagnostic ~~they used~~ is:

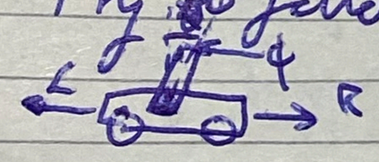
1. Check simulator accuracy.
2. Is the reward function $E[R_0 + R_1 + \dots R_T]$ correct?
3. Is the optimisation algorithm for π^* good?

Policy search

(direct policy search)

$V^* \rightarrow \pi^*$

Try to find a good π directly



$$S = \begin{pmatrix} 1 \\ x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{pmatrix}$$

$$y \approx h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

$$\text{"}a \approx \pi_\theta(s) = (1+e^{-\theta^T s})^{-1}\text{"}$$

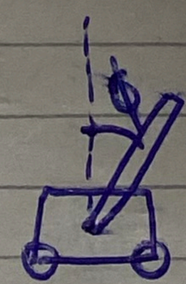
New definition: A stochastic policy is a function $\pi : S \times A \rightarrow \mathbb{R}$ where $\pi(s,a)$ is the prob. of taking action a in state s.

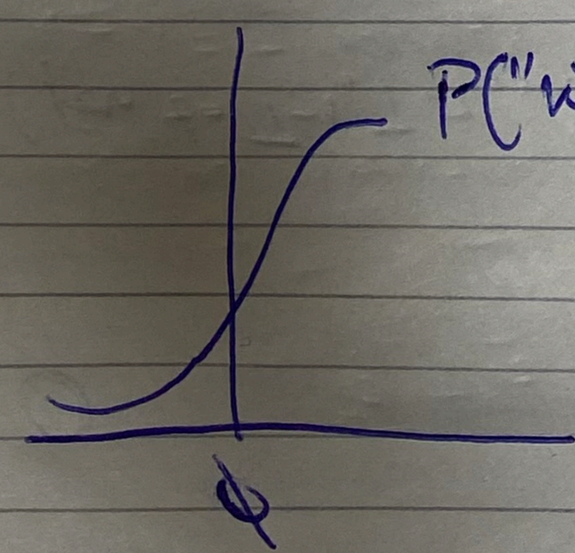
$$\left(\sum_a \pi(s,a) = 1\right)$$

$$\pi_\theta(s, \text{"R"}) = (1+e^{-\theta^T s})^{-1}$$

$$\pi_\theta(s, \text{"L"}) = 1 - (1+e^{-\theta^T s})^{-1}$$

So if eg $S = \begin{pmatrix} 1 \\ x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{pmatrix}$, $\theta = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \Leftrightarrow$





Lec20 Cont

But if $\theta = \begin{bmatrix} 0 \\ -.5 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, then the policy takes into account how far to the right you are.

Goal: Find θ so that when we execute $\pi_\theta(s,a)$, we maximize

$$\max_\theta E[R(s_0,a_0) + \dots + R(s_T,a_T) \mid \pi_\theta]$$

Note: Here, s_0 is a fixed initial state.

$$\max_\theta E[R(s_0,a_0) + \dots + R(s_T,a_T) \mid \pi_\theta]$$

For simplicity, set $T=1$ (is easily generalizable).

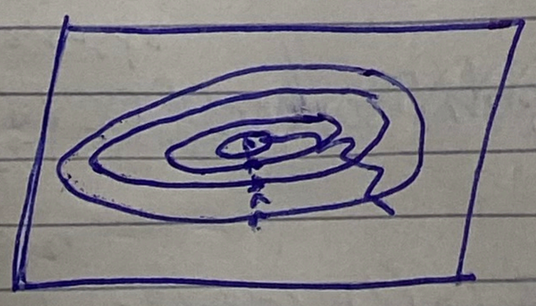
$$= \sum_{s_0 a_0 s_1 a_1} P(s_0, a_0, s_1, a_1) \cdot [\underbrace{R(s_0,a_0) + R(s_1,a_1)}_{\equiv \text{Payoff}}]$$

$$= \sum_{s_0 a_0 s_1 a_1} P(s_0)\, \pi_\theta(s_0,a_0)\, P_{s_0 a_0}(s_1)\, \pi_\theta(s_1,a_1) \cdot [\text{Payoff}]$$

Reinforce Algorithm:

Loop: {
Sample $s_0, a_0, s_1, a_1, \dots (s_T, a_T)$.
Compute payoff $= R(s_0) + R(s_1) + \dots + R(s_T)$.
Update:

$$\theta := \theta + \alpha \left[\frac{\nabla_\theta \pi_\theta(s_0,a_0)}{\pi_\theta(s_0,a_0)} + \frac{\nabla_\theta \pi_\theta(s_1,a_1)}{\pi_\theta(s_1,a_1)} + \dots \right] \times \text{Payoff}$$



$$\left(\frac{d}{db}(fgh) = f'gh + fg'h + fgh' \right)$$

$$\nabla_\theta \sum_{s_0 a_0 s_1 a_1} P(s_0)\, \pi_\theta(s_0,a_0)\, P_{s_0 a_0}(s_1)\, \pi_\theta(s_1,a_1) \cdot [\text{payoff}]$$

$$= \sum P(s_0) \frac{\nabla_\theta \pi_\theta(s_0,a_0)}{\pi_\theta(s_0,a_0)} P_{s_0 a_0} \pi_\theta(s_1,a_1) + \frac{P(s_0)\, \pi_\theta(s_0,a_0)\, P_{s_0 a_0}(s_1)}{\pi_\theta(s_1,a_1)} \cdot \pi_\theta(s_1,a_1)$$

Lec 20 Cont 2

$$\sum P(s_0, a_0, s_1, a_1) \cdot \left[\frac{\nabla_\theta \pi_\theta(s_0, a_0)}{\pi_\theta(s_0, a_0)} + \frac{\nabla_\theta \pi_\theta(s_1, a_1)}{\pi_\theta(s_1, a_1)} \right] [\text{payoff}]$$

$$= \sum_{s_0 a_0 s_1 a_1} P(s_0, a_0, s_1, a_1) \cdot [\text{gradient update}]$$

$a = \theta^T s$ + gaussian noise

partially observable

POMDP

- At each step, get a partial (and potentially noisy) measurement of the state. Have to choose action "a" using that.

Eg: observation is $g = \binom{x}{\dot{x}}$ + noise. Then $\pi(g, \text{"R"}) = \left(1 + e^{-\theta^T \bar{g}}\right)^{-1}$, $\bar{g} = \binom{1}{g}$

can be any set of features

Is π^* simpler, or is V^* simpler?

Low level control tasks, former, otherwise, latter — I think.