Acrobot : mechanics + optimization
Anymal : RL-based
manipulation : Imitation-learning

△ underactuated

$\begin{cases} \dot{x} = f(x,u) \\ y = g(x) \end{cases}$ $\begin{cases} \dot{x} = Ax + Bu \\ y = Cx + Du \end{cases}$

△ second-order nonlinear system

$\ddot{q} = f(q, \dot{q}, u)$

or

$\dot{x} = \tilde{f}(x,u) = \begin{bmatrix} \dot{q} \\ f(q,\dot{q},u) \end{bmatrix}$

△ "control affine" nonlinear systems

$\ddot{q} = f_1(q,\dot{q}) + f_2(q,\dot{q})u$ (1)

based on this, define underactuated

def

(1) is fully actuated in $q, \dot{q}$ i.f.f.

$f_2(q,\dot{q})$ is full row rank (matrix)

i.e. $\dim(q) = m$, $\dim(u) = n$

$f_2(q,\dot{q}) \in \mathbb{R}^{m \times n}$, $\text{rank}[f_2(q,\dot{q})] = m$

controllable : can choose some inputs over time to achieve some goal

actuable : can choose some inputs to cause acceleration

(1) is underactuated iff in $q, \dot{q}$ $\text{rank}[f_2(q,\dot{q})] < m$

$\forall q, \dot{q}$, $\text{rank}[f_2(q,\dot{q})] < m$

⇒ "system" is underactuated

△ Feedback equivalence (fully-actuated)

given

$\ddot{q} = f_1(q,\dot{q}) + f_2(q,\dot{q})u$

& $\ddot{q}^d$

⇒ then $u = f_2^{-1}(q,\dot{q})[\ddot{q}^d - f_1(q,\dot{q})]$

⇒ $\ddot{q} = \ddot{q}^d$

give me an "acceleration", I give u an "u"

⇒ feedback equivalent to $\ddot{q} = u$ (double integrator which we hv optimal sol^n for)

△ feedback equivalence be broken;

input saturations
state constraints
model uncertainty

△ Manipulator Eqs

actuator selector Matrix

$\underbrace{M(q)}_{\text{mass matrix}}\ddot{q} + \underbrace{C(q,\dot{q})\dot{q}}_{\text{coriolis terms}} = \underbrace{\tau_g(q)}_{\text{gravity}} + \underbrace{Bu}_{\text{torque input}}$ (+?)

$M \succ 0$

⇒ $\ddot{q} = M^{-1}(q)[-C(q,\dot{q})\dot{q} + \tau_g(q) + Bu]$

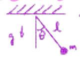
$\ddot{q} = f_1(q,\dot{q}) + f_2(q,\dot{q})u$ is in this form!

Q! whether $f_2(q,\dot{q})$ is full rank relies on "B"

as $\underbrace{M^{-1}(q)}_{\text{full rank}} \underbrace{B}_{\text{not sure!}}$

△ Dynamic Programming

$ml\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u$



△ Control as an optimization

- given trajectory $x(\cdot), u(\cdot)$ $\forall t$ $x(t)$ $t \in [0, T_{max}]$
- Assign a score (scalar, reward, cost) e.g. time, avg distance
- subject to constraints

e.g. minimum time for double integrator

$\ddot{q} = u$

goal : drive to $q = \dot{q} = 0$ in minimum-time from any initial condition

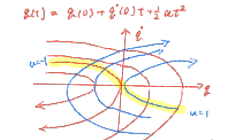
△ intuition : "bang-bang" policy (accelerate as much as possible) (unsmooth)

△ intuition #2 :

$\ddot{q} = u$, $u = -1$ $u = 1$

$\dot{q}(t) = \dot{q}(0) - t$

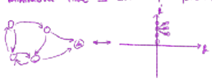
$q(t) = q(0) + \dot{q}(0)t + \frac{1}{2}at^2$



Q: How to generalize ?

△ Dynamic Programming

minimum-time ≅ shortest path problems



- weighted shortest path
- DP

discrete states $s_i \in S$
discrete actions $a_i \in A$
discrete time $s[n+1] = f(s[n], a[n])$
"edge cost" $g(s,a)$
total cost $\sum_n g(s,a)$

Key idea : Additive cost

e.g. time to goal $g(s,a) = \begin{cases} 1 & \forall s \neq s_{goal} \\ 0 & \text{o.w.} \end{cases}$

$\int g(s,a)dt$

△ DP

- DP is a recursive algorithm
- solve backwards from the goal
- cost-to-go

$J^*(s_i) = \min_{a[\cdot]} \sum_{n=0}^{\infty} g(s[n], a[n])$ (watch for convergence)

$s[0] = s_i$

hard to search

⟺ $J^*(s_i) = \min_a [\underbrace{g(s,a)}_{\text{single}} + J^*(f(s,a))]$

condition to satisfy optimality

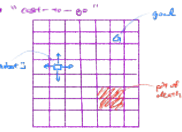
- Algorithm of DP (= value iteration)

$\hat{J}^*$ ← estimate of optimal cost-to-go

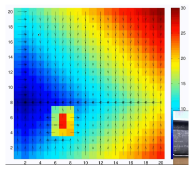
$\forall i$ $\hat{J}^*(s_i) \leftarrow \min_a [g(s,a) + \hat{J}^*(f(s,a))]$

$\hat{J}^* \longrightarrow J^* + c$

- "cost-to-go"



$g(s,a) = \begin{cases} 0 & @ \text{ goal} \\ 10 & @ \text{ pit} \\ 1 & @ \text{ o.w.} \end{cases}$



- Caveats
  - accuracy (discretization errors)
  - scalability (dimensions ~5) curse of the dimensionality
  - model is known
  - cost function
  - assume "full state" feedback

△ Lim DP ($\Delta x \to 0$, $\Delta u \to 0$, $\Delta t \to 0$)

- eg minimum-time double integrator

$\ddot{q} = u$, $|u| \leq 1$

optimal policy is "bang-bang"

- we want "discrete" → "continuous"

△ optimal cost-to-go

- discrete

$s[n+1] = f(s[n], a[n])$

- cost

$\sum \ell(s[n], a[n])$

$\forall s$ $J^*(s) = \min_a [\ell(s,a) + J^*(f(s,a))]$ (cost-to-go function)

∴ $\pi^*(s) = \text{argmin}_a [\ell(s,a) + J^*(f(s,a))]$ (optimal controller/policy)

△ discrete v continuous

states	s	x
actions	a	u
time	n	t

$s[n+1] = f(s[n], a[n])$ $\dot{x} = f_c(x,u)$

$\sum_n \ell_d(s[n], a[n])$ $\int_0^\infty \ell_c(x,u)\,dt$

△ informal derivation

$x[n+1] \approx x[n] + h\, f_c(x[n], u[n])$

euler-integration $t_{n+1} - t_n$

$\ell_d(x,u) = h\,\ell_c(x,u)$

∴ $J^*(x) = \min_u [h\,\ell_c(x,u) + J^*(x + hf_c(x,u))]$

$J^*(x) + \frac{\partial J^*}{\partial x} h f_c(x,u)$

⇒ $0 = \min_u [\ell_c(x,u) + \frac{\partial J^*}{\partial x} f_c(x,u)]$

Hamilton-Jacobi-Bellman Eqn. HJB

e.g. infinite-time Grid world

$J^*(x) = c$

$\frac{dJ}{dt} = \frac{\partial J}{\partial x} f(x,u)$

for u^* (cost-to-go)

$0 = \ell(x,u^*) + \frac{\partial J}{\partial x}\frac{\partial x}{\partial t}$ ⇒ $\frac{\partial J}{\partial t} = -\ell(x,u^*)$ (running cost)

⇒ if you are following optimal policy your cost ↓

HJB sufficiency theorem

e.g. double integrator / Quadratic cost

$\ddot{q} = u$ $\ell(x,u) = q^2 + \dot{q}^2 + u^2$ (no constraints)

claim:

$\pi^*(x) = -q - \sqrt{3}\dot{q}$

$J^*(x) = \sqrt{3}q^2 + 2q\dot{q} + \sqrt{3}\dot{q}^2$ $\left(x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}\right)$

proof:

(1) $0 = \min_u [q^2 + \dot{q}^2 + u^2 + \frac{\partial J}{\partial x} f(x,u)]$

$\left(\frac{\partial J}{\partial x} f(x,u) = [\frac{\partial J}{\partial q} \frac{\partial J}{\partial \dot{q}}]\begin{bmatrix} \dot{q} \\ u \end{bmatrix}\right)$

$= \min_u [q^2 + \dot{q}^2 + u^2 + \frac{\partial J}{\partial q}\dot{q} + \frac{\partial J}{\partial \dot{q}}u]$

$= \min_u [0]$

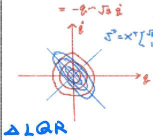
(2) $u = \text{argmin}_u [u^2 + \frac{\partial J}{\partial \dot{q}}u]$

⇒ $\frac{\partial}{\partial u}(\cdot) = 2u + \frac{\partial J}{\partial \dot{q}} = 0$

⇒ $u = -\frac{1}{2}\frac{\partial J}{\partial \dot{q}}$

$= -\frac{1}{2}(2q + 2\sqrt{3}\dot{q})$

$= -q - \sqrt{3}\dot{q}$



$J^* = x^T \begin{bmatrix} \sqrt{3} & 1 \\ 1 & \sqrt{3} \end{bmatrix} x$

△ LQR

- $\dot{x} = Ax + Bu$

$\ell(x,u) = x^TQx + u^TRu$

$Q \succeq 0$ $R \succ 0$

- inf horizon

$J^*(x) = x^TSx$ (cost-to-go) (can capture the dynamics)

$S \succeq 0$

$\frac{\partial J}{\partial x} = 2x^TS$

$0 = \min_u [x^TQx + u^TRu + \frac{\partial J}{\partial x}(Ax + Bu)]$

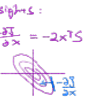
⇒ $\frac{\partial [\,]}{\partial u} = 0$, $u = -R^{-1}B^TSx = -Kx$

Riccati Equation

$0 = Q - SBR^{-1}B^TS + A^TS + SA$

- insights :

$-\frac{\partial J}{\partial x} = -2x^TS$ (where we want our cost-to-go func. to decrease)



$u = -R^{-1}B^TSx$

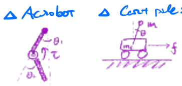
(gradient)
(underactuated B matrix to adjust Sx)
(weighting on B^TSx)

△ DP so far

- Tabular (discrete → graph search)
- Linear dynamics + convex objective (LQR)

↳ here, use acrobot, cartpole, quadrotors to demonstrate DP

△ Acrobot   △ Cart pole:



△ Equations of motion :

$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau_g(q) + Bu$

- Acrobot	- Cart pole
$q = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$	$q = \begin{bmatrix} x_{cart} \\ \theta_{pole} \end{bmatrix}$
$u = [\tau]$	$u = [f_{cart}]$
$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$B = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$
$\lvert u \rvert \leq u_{max}$	$\lvert u \rvert \leq u_{max}$
	$\lvert x \rvert \leq x_{max}$

↳ target : swing up + balance

sol^n 1. tabular value iteration ? yes, not elegant

2. LQR ? Beautiful solution for balancing (need more for swing-up)

△ LQR for nonlinear systems

$\dot{x} = f(x,u) \Longleftrightarrow \dot{x} = Ax + Bu$

⇒ choose nominal x_0, u_0 (want a fixed pt)

⇒ $\dot{x} \approx f(x_0,u_0) + \underbrace{\frac{\partial f}{\partial x}\Big|_{x=x_0, u=u_0}}_{A}(x - x_0) + \underbrace{\frac{\partial f}{\partial u}\Big|_{x=x_0, u=u_0}}_{B}(u - u_0)$

let

$\bar{x} = x - x_0$

$\dot{\bar{x}} = \dot{x} - \dot{x}_0$

$\bar{u} = u - u_0$

⇒ $\dot{\bar{x}} = A\bar{x} + B\bar{u}$

e.g. simple pendulum

- $ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = \tau$

$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$ $u = [\tau]$

- linearize about $\theta = \pi$, $\dot{\theta} = 0$, $u = 0$
- $\dot{x} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{ml^2}[\tau - b\dot{\theta} - mgl\sin\theta] \end{bmatrix}$
- check fixed pt ✓

$\dot{x}_0 = \begin{bmatrix} 0 \\ \frac{1}{ml^2}[0 - b\cdot 0 - mgl\sin\pi] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

- linearization

$\frac{\partial f}{\partial x} = \begin{bmatrix} 0 & 1 \\ -\frac{g}{l}\cos\theta & -\frac{b}{ml^2} \end{bmatrix}\Bigg|_{\theta=\pi} = \begin{bmatrix} 0 & 1 \\ \frac{g}{l} & -\frac{b}{ml^2} \end{bmatrix}$

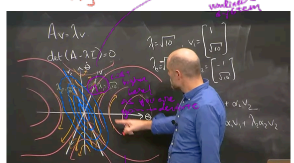
$\frac{\partial f}{\partial u} = \begin{bmatrix} 0 \\ \frac{1}{ml^2} \end{bmatrix}$

let $m = l = 1$ $g = 10$

$A = \begin{bmatrix} 0 & 1 \\ 10 & 0 \end{bmatrix}$

$B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

orange: approximate of a nonlinear system



red: nonlinear system

- LQR Design

$\begin{cases} \dot{\bar{x}} = A\bar{x} + B\bar{u} \\ J = \int_0^\infty \bar{x}^TQ\bar{x} + \bar{u}^TR\bar{u}\, dt \\ [K,S] = lqr(A,B,Q,R) \\ u = -Kx = -R^{-1}B^TSx \\ J^* = \bar{x}^TS\bar{x} \end{cases}$

⇒ Stable

$\dot{x} = \underbrace{(A - BK)}_{\text{Hurwitz}}x$

△ Is (A,B) controllable?

underactuated but controllable

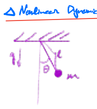
△ controllable : given $x(0)$ can I find $u(t)$ $\forall t \in [t_0, t_f]$ s.t. $x(t_f) = 0$

Is (A,B) stablizable

△ stablizable : given $x(0)$ can I find $u(t)$ $\forall t \in [t_0, \infty)$ s.t. $\lim_{t\to\infty} x(t) = 0$

(no control input)

△ Nonlinear Dynamics



(kinetic energy)

$T = \frac{1}{2}ml^2\dot{\theta}^2$ $v_{tan} = l\dot{\theta}$

(potential energy)

$U = -mgl\cos\theta$

Lagrangian mechanics

$L = K - U$

$K = \frac{1}{2}mv^2$
$= \frac{1}{2}m\dot{s}^2$
$= \frac{1}{2}ml^2\dot{\theta}^2$ $\theta = \frac{s}{l}$

$U = mgy$
$= -mgl\cos\theta$

∴ $L = \frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos\theta$

Euler-Lagrange equation

$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta}$

$L = \frac{1}{2}ml^2\dot{\theta}^2 + mgl\cos\theta$

$\frac{\partial L}{\partial \theta}$: $ml^2\ddot{\theta} + mgl\sin\theta = Q$ (generalized force)

$Q = \underbrace{-b\dot{\theta}}_{\text{friction}} + \underbrace{u}_{\text{torque}}$

∴ $ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u$

⇕

$M(q)\ddot{q} + C(q,\dot{q})\dot{q} = \tau_g(q) + Bu$

⇕

$\ddot{q} = f(\theta, \dot{\theta}, u)$

⇕

$\theta(t) = ?$

△ Nonlinear Dynamics Questions

- What is $\lim_{t\to\infty} \theta(t)$?
- Will my robot fall down ?

$ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u$ (N·m)

$kg\frac{m^2}{s^2}\frac{1}{s}$ $kg\,m^2\frac{1}{s^2}$ $kg\frac{m^2}{s^2}$

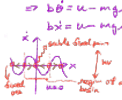
case : $b\dot{\theta} >> ml^2$

$b\sqrt{\frac{l}{g}} >> ml^2$ (heavily damped region)

⇒ linear approximation!

⇒ $b\dot{\theta} = u - mgl\sin\theta$

$b\dot{x} = u - mgl\sin x$ $x \in R$



△ Stability

- local stability (do not leave a ball)
  - in the sense of Lyapunov (i.s.L.)

$\begin{cases} \epsilon, \delta, \text{ small positive constants} \\ \forall \epsilon, \exists \delta, \text{ s.t. } \lVert x(0) - x^* \rVert < \delta \\ \Rightarrow \forall t\ \lVert x(t) - x^* \rVert < \epsilon \end{cases}$

  - locally attractive — get to the region

$\{\lim_{t\to\infty} x(t) \to x^*$

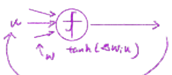
  - asymptotically stable — i.s.L + attractive

{ i.s.L + attractive

  - exponentially stable — get to the region w/ rate

$\begin{cases} \forall t \\ \lVert x(t) - x^* \rVert < Ce^{-\alpha t} \\ C, \alpha > 0 \end{cases}$

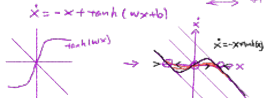
△ Simple RNN

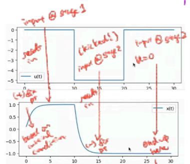


Autapse

±b offset

$\dot{x} = -x + \tanh(wx + b)$





△ 2nd-order system

$ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = u$

$\dot{x} = f(x,u)$

$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}$

$\dot{x} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{ml^2}[u - b\dot{\theta} - mgl\sin\theta] \end{bmatrix}$

★ Essence of controls for underactuated robotics :

we hv all these dynamics

→ plug w/ u, s.t. phase portraits behave as we want!!!

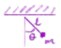
△ Lyapunov Analysis

recall DP:
- Tabular — easy to explode
- LQR — only for linear case
- Approximate DP (NN) — works quite well (but solve a different DP, e.g. discounting)

↳ all are trying to get "cost-to-go" function $J^*(x)$

△ now Lyapunov ⟷ optimal value
good enough — very good
might replace the original optimal val.

△ **Example** : stability analysis of simple pendulum



$ml\ddot\theta + mgl\sin\theta = -b\dot\theta$

△ $\theta(t) \longrightarrow$ hard to solve, cannot be analysis
△ Lyapunov instead !!!

△ $E = K + U$

$= \frac{1}{2}ml^2\dot\theta^2 - mgl\cos\theta$

△ $\frac{d}{dt}E(x)$, $x = \begin{bmatrix}\theta \\ \dot\theta\end{bmatrix}$

$= \frac{\partial E}{\partial x}\frac{\partial x}{\partial t}$

$= \frac{\partial E}{\partial t}$

△ DP sum

$\frac{d}{dt}x = f(x(t), u(t), x)dt$

- $J(x(t), u(t), t_0, t_f)$

$= Q(x(t_f), t_f) + \int_0^{t_f} \ell(x(\tau), u(\tau))\, d\tau$

- $V(x(t_0), t_0, t_f)$ (cost-to-go)

$= \min_{u(t)} J(x(t), u(t), t_0, t_f)$

- $-\frac{\partial V}{\partial t} = \min_{u(t)} \left(\left(\frac{\partial V}{\partial x}\right)^T f(x(t), u(t)) + \ell(x(t), u(t)) \right)$

$\Leftrightarrow \min_{u(t)} \left(\left(\frac{\partial J^*}{\partial x}\right)^T f(x(t), u(t)) + \ell(x(t), u(t)) \right)$

Hamilton-Jacobi-Bellman (HJB)

- $V(x(t_0), t_0, t_f)$

$= V(x(t_0), t_0, t) + V(x(t), t, t_f)$

$x(t_0)$ $x(t)$ $x(t_f)$

Bellman Optimality

$\Leftrightarrow$ $J^*(s_i) = \min_a [\ell(s,a) + J^*(f(s,a))]$

single

proof — total derivative

$\frac{d}{dt}V(x(t), t, t_f) = \frac{\partial V}{\partial t} + \left(\frac{\partial V}{\partial x}\right)^T \frac{\partial x}{\partial t}$

$= \frac{d}{dt} \min_{u(t)} J(x(t), u(t), t, t_f)$

$= \frac{d}{dt} \min_{u(t)} Q(x(t_f), t_f) + \int_t^{t_f} \ell(x(\tau), u(\tau))\, d\tau$

$= \min_{u(t)} \frac{d}{dt} \left[Q(x(t_f), t_f) + \int_t^{t_f} \ell(x(\tau), u(\tau))\, d\tau \right]$

$= \min_{u(t)} \; 0 + \frac{d}{dt}\left(\left[\mathcal{L}(x(\tau), u(\tau)) \right]_t^{t_f} \right)$

$= \min_{u(t)} \frac{d}{dt}\left(\mathcal{L}(x(t_f), u(t_f)) - \mathcal{L}(x(t), u(t)) \right)$

$= \min_{u(t)} -\ell(x(t), u(t))$

⇕

$\frac{\partial V}{\partial t} + \left(\frac{\partial V}{\partial x}\right)^T \frac{\partial x}{\partial t} = \min_{u(t)} -\ell(x(t), u(t))$

⇕

$-\frac{\partial V}{\partial t} = \min_{u(t)} \left(\left(\frac{\partial V}{\partial x}\right)^T \frac{\partial x}{\partial t} + \ell(x, u) \right)$

$= \min_{u(t)} \left(\frac{\partial V}{\partial x}^T f(x,u) + \ell(x,u) \right)$

★ Get V ⟶ Get u
- two point bounded value problem
- shooting method

△ Discrete-Time HJB

- $x_{k+1} = F(x_k, u_k)$
- $J(x_0, \{u_k\}_{k=0}^{n}, n)$

$= Q(x_n, t_n) + \sum_{k=0}^{n} \ell(x_k, u_k)$

- $V(x_0, n)$

$= \min_{\{u_k\}_{k=0}^{n}} J(x_0, \{u_k\}_{k=0}^{n}, n)$

- $V(x_0, n)$

$= V(x_0, k) + V(x_k, n-k) \; \forall k \in (0, n$

- $V(x_k, n)$

$= \min_{u_k} \left[\ell(x_k, u_k) + \underset{x_{k+1} = F(x_k, u_k)}{V(x_{k+1}, n-1)} \right]$

$\Leftrightarrow$ vs $J(s) = \min_a [\ell(s,a) + J^*(f(s,a))]$

Discrete HJB

⇕

$V(x_k, n)$

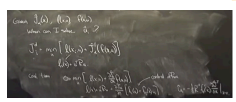
$= \min_{u_k} \left(\ell(x_k, u_k) + V(F(x_k, u_k), n-1) \right)$

⇕

$V(x) = \min_u \left(\ell(x,u) + V(F(x,u)) \right)$

↓

$\pi(x) = \arg\min_u \left(\ell(x,u) + V(F(x,u)) \right)$



- use optimal u : recall — CT LQR: $u^* = -K^* x = -R^{-1}B^T S^* x$; DT LQR: $u^* = -K^* x = -(R + B^T S^* B)^{-1} B^T S^* A x$
- instead of sampling u_i, only sample $x_i \in X_s \subset X$

$\Rightarrow J_i^d = x_i^T Q x_i + \hat{u}_i^T R \hat{u}_i + \hat{J}_\alpha(Ax_i + Bu_i)$

w/ optimal u w/ current $\hat{S}$

△ DP (cont'd)

key idea
1) LQR (linear optimal control
2) local linearization
3) $Q = \begin{bmatrix} 10 & & & \\ & 10 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$ $R = [1]$

@ balancing

Q: How do we swing up?
"value iteration"

⇒ Approximate DP

△ Value iteration on a mesh for Pendulum Swing-up

△ mesh q 51 bins — odd for "0"
$\dot q$ 51 bins
u 9 bins

△ Acrobot
51^4 bins
9 bins
↗ explode!!!

△ NN! Approximating Value Functions

$\hat{J}_\alpha(x)$
└ parameters + bias

- value iteration update
- recall:

$\forall s_i \in S$

$\hat{J}(s_i) = \min_a \left[\ell(s_i, a) + \hat{J}(f(s_i, a)) \right]$

$s[n+1] = f(s[n], a[n])$

$\sum_{n=0}^{\infty} \ell(s[n], a[n])$

- Draw samples

$x_i \in X_s \subset X$

$u_j \in U_s \subset U$

$\forall ij$ $\dot{x}_{ij} = f(x_i, u_j)$ next state samples

$\ell_{ij} = \ell(x_i, u_j)$

desired J @ i — $J_i^d = \min_j [\ell_{ij} + \hat{J}_\alpha(x'_{ij})]$

$\Rightarrow \min_\alpha \text{loss} \sum_i \left[\hat{J}_\alpha(x_i) - J_i^d \right]^2$

⇕

$\min_\alpha \text{loss} \sum_i \left[\hat{J}_\alpha(x_i) - \min_j [\ell_{ij} + \hat{J}_\beta(x'_{ij})] \right]^2$

"target network"

"here we do the training based on the samples. To hv a better sampling, RL kicks in, & guide the state space, to a subset, that is closer to the optimal."

Q: when does it work?
does it converge
↓ △ linear approximator
△ works @ special case : linear NN

- $\hat{J}_\alpha(x) = \alpha^T \phi(x)$

nonlinear basis "input features" (fixed)

$= \sum_j \alpha_j \phi_j(x)$

$\Rightarrow \min_\alpha \sum_i [\alpha^T \phi(x) - J_i^d]^2$

linear least squares
has closed-form soln
⇒ SGD is guaranteed to converge to global minima

△ Back to LQR ; connection to approximator ; w/ discrete LQR

$J^*(x) = x^T S^* x$

NN ⟶ linear approximator

loss $\sum_i [x_i^T \hat{S} x_i - J_i^d]^2$

$\hat{S} \leftarrow \hat{S} - \eta \frac{\partial \text{loss}}{\partial S}$

Based on this, loss explodes ↑

why? LQR tries to solve inf horizon, cost cannot go to inf.
How does normal LQR avoid inf?
$x \longrightarrow 0$, $u \longrightarrow 0$ (exact 0)
How about here, when we do sampling?

popularly seen in RL

- discounting $\sum_{n=0}^{\infty} \gamma^n \ell(x[n], u[n])$ $0 \le \gamma \le 1$

optimal : LQR($\sqrt{\gamma}A$, B, Q, $\frac{1}{\gamma}R$)

pretend that is more stable

$x[n+1] = Ax[n] + Bu[n]$

$|\text{eig}(A)| \le 1$.

$|NFA| \leq |A|$