Lecture 7
EAIS S'26
Ken Nakamura

Last Time

- ☐ robust safety
- ☐ zero-sum games

This time

- ☐ Computational tools

note: We will use AI notation this lecture

$s =$ "state" $a =$ "action" / "control"

Recap

failure set $F \Rightarrow$ never enter F $\begin{cases} \text{Collisions} \\ \text{exceed joint limits} \\ \text{spill etc...} \end{cases}$

$$F := \{ s \mid \ell(s) < 0 \}$$

O.C. problem

$$V(s,t) = \max_{a(\cdot) \in A_t^T} \left[\min_{\tau \in [t,T]} \ell(\xi_s^a(\tau)) \right]$$

Hamilton-Jacobi-Bellman Variational-Inequality

$$\min \left\{ \ell(s) - V(s,t), \frac{\partial V}{\partial t} + \max_{a \in A} \frac{\partial V}{\partial s} \cdot f(s,a) \right\} = 0$$

$$V(s,T) = \ell(s)$$

cont time: $\dot{s} = f(s,a)$

Hamilton-Jacobi-Bellman Eqn

$$V_t(s) = \min \left\{ \ell(s), \max_{a \in A} V_{t+1}(f(s,a)) \right\}$$

$$V_T = \ell(s)$$

disc time: $s' = f(s,a)$

Re-organized a bit...

$$V_t(s) = \min\left\{ \ell(s), \max_{a \in A} V_{t+1}(f(s,a)) \right\}$$

$$0 = \min\left\{ \ell(s) - V_t(s), \underbrace{\max_{a \in A} V_{t+1}(f(s,a)) - V_t(s)}_{\text{"difference in value in 1 timestep"}} \right\}$$

$$0 = \min\left\{ \ell(s) - V(s,t), \underbrace{\frac{\partial V}{\partial t} + \max_{u \in \mathcal{U}} \frac{\partial V}{\partial s} \cdot f(s,u)}_{\text{total derivative wrt time}} \right\}$$

$$\frac{dV(s,t)}{dt} = \frac{\partial V}{\partial t} + \frac{\partial V}{\partial s} \cdot \underbrace{\frac{\partial s}{\partial t}}_{\text{dynamics}}$$

Even when $\ell(s)$, $f(s,a)$ are known analytically, really hard to get analytic expression for V

$\Rightarrow$ need computational tools to obtain V

Today:

- ↳ grid-based numerical methods
- ↳ self-supervised learning
- ↳ reinforcement learning

$\rightarrow$ all work for HJI, reach-avoid, etc.

Design decision: How to represent V?

HW1: Represent $V(s,t)$ as a grid

Solve via numerical methods

helper_oc (MATLAB)
BEACLS (C++)
optimized_dp (Python)
hj_reachability (JAX)

$s \in S \subset \mathbb{R}^d$ $\quad a \in A \quad t \in [0, T]$

d-dim state

- discretize each dim m times
- discretize time K times

HW1: $[51 \times 51 \times 51]$

$$0 = \min \left\{ \ell(s) - V(s,t), \frac{\partial V}{\partial t} + \max_{a \in A} \frac{\partial V}{\partial s} \cdot f(s,a) \right\}$$

approximated w/ fancy finite-differences on grid nodes

"essentially non-oscillatory" methods

- at each grid point, need to compute "Hamiltonian"

$$H(s,t) = \max_a \frac{\partial V}{\partial s} \cdot f(s,a)$$

$s = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad \dot{s} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ a \end{bmatrix} \quad \frac{\partial V}{\partial s} = \begin{bmatrix} P_x \\ P_y \\ P_\theta \end{bmatrix} \quad a \in [-1, 1]$

$\Rightarrow H(s,t) = \max_a \left[P_x \cos\theta + P_y \sin\theta + P_\theta \cdot a \right]$

$\Rightarrow a^* = \text{sign}[P_\theta]$

$\Rightarrow$ control affine dyn w/ box ctrl limits is simple sign check

Challenge Q (think at home)

$s = \begin{bmatrix} x \\ y \end{bmatrix}$ $\dot{s} = \begin{bmatrix} \cos a \\ \sin a \end{bmatrix}$ $a \in [0, 2\pi]$

directly ctrl θ

What is θ^*?

$\Rightarrow$ need to store Value @ each grid pt in time

$\Rightarrow$ need to do optimization @ each grid pt

Q: How many grid pts?

$K m^d$ $\Rightarrow$ expensive!

Note: you can do "smarter" gridding. See:
"Scalable learning of ..."
Herbert + Choi ICRA'21

Computation time + memory requirement blows up in state dim d

↳ "Curse of dimensionality"

Need ~1 TB RAM to store $V(s,t)$ on 7-8 dim state spaces

<u>In-practice</u> Some dimensions matter "less"

Assign higher discretization to state dims that "matter more"

Leung et al. "On infusing Reachability-based..." IJRR'19

$\text{State} = [y, \psi_R, V_{xR}, V_{yR}, r_R]$

$\text{Grid} = [21, 9, 9, 9, 9] \rightarrow$ higher fidelity for y

Grid-based numerical methods

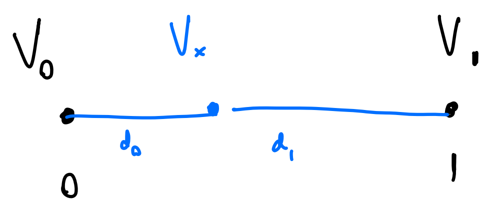
Accuracy depends on grid resolution + derivative approx

↳ can get arbitrarily accurate w/ finer gridres

Computation + memory scale exponentially w/ state dim

Need to interpolate for 'off-grid' states

1-D example



$$V_x = V_0 \cdot \frac{d_1}{d_1 + d_0} + V_1 \cdot \frac{d_0}{d_1 + d_0}$$

CODE DEMO

How to scale up?

Idea: Represent $V(s,t)$ as a NN, solve w/ some algorithm

Simplest Idea: supervised learning

$$\min_{\theta} \mathbb{E}_{s} \Big\| \underbrace{V_{gt}(s,t)}_{\text{true val fn}} - \underbrace{V_{\theta}(s,t)}_{\text{NN approx}} \Big\|$$

What's wrong w/ this?

↳ we do not have $V_{gt}(s,t)$!

What we do know

$V_\theta(s,T) = \ell(s)$

$$0 = \min\left\{ \ell(s) - V_\theta(s,t),\ \frac{\partial}{\partial t} V_\theta(s,t) + \max_a \left\langle \frac{\partial}{\partial s} V_\theta(s,t), f(s,a) \right\rangle \right\}$$

Idea: Just enforce these conditions!

Deepreach (Bansal + Tomlin ICRA '20)

$$L_1(s,t;\theta) = \| V_\theta(s,T) - \ell(s) \|$$

↳ terminal time constraint

$$L_2(s,t;\theta) = \| \min \{ \ell(s) - V_\theta(s,t), \frac{\partial}{\partial t} V_\theta(s,t) + \underbrace{\max_{a \in A} \frac{\partial}{\partial s} V_\theta(s,t) \cdot f(s,a)}_{H(s,t)} \} \|$$

↳ Implicit supervision of HJ-VI

Goal: $\min_\theta \mathbb{E}_{(s,t)} \left[L_1(s,t;\theta) + \lambda L_2(s,t;\theta) \right]$

Q: How to sample (s,t)?

Sampling t: "backwards in time"

→ Pretrain w/ $t=T$ → fit $V(s,T) = \ell(s)$

→ Uniformly sample $[t_{start}, T]$ where $t_{start} = T \rightarrow 0$ over training

→ Conceptually same as grid-based methods

Sampling s: Uniform

→ assumes all states are equally important

CODE Demo

In practice

↳ Scales to high-D (100+ dim) sys

↳ requires special architectures

↳ sinusoidal activations

↳ why not ReLU?

↳ still need to do inner optimization

↳ Weak, implicit source of supervision

↳ struggle w/ large T

↳ May need to modify sampling procedure over states for higher quality

So far

always need to compute $\max_a \frac{\partial V}{\partial s} \cdot f(s,a)$

→ may not have access to $\dot{s} = f(s,a)$

→ Maximization may not be easy to do

Idea 2: Learn $\arg\max_a \frac{\partial V}{\partial s} \cdot f(s,a)$?

⇒ Off-policy Actor-Critic Reinforcement learning

↳ SAC, DDPG, TD3, DrQ etc...

⇒ discrete time: $s' = f(s,a)$ $\quad V(s) = \min\{\ell(s), \max_a V(f(s,a))\}$

Reachability RL

(Akametalu et al 2018)

(Fisac et al. ICRA 2019)

Jointly learn π and Q s.t.

$$Q(s,a) = \min\{\ell(s), \max_{a'} Q(s',a')\}$$

$$\pi(s) = \arg\max_a Q(s,a)$$

$$V(s) = \max_a Q(s,a) \qquad Q(s,a) = \min\{\ell(s), V(s')\}$$

But first...

Discounted Safety Bellman eq

$$Q(s,a) = (1-\gamma)\,\ell(s) + \gamma\left[\min\left\{\ell(s),\ \max_{a'} Q(s',a')\right\}\right]$$

$$\underbrace{\gamma \in [0,1)}_{\text{discount factor necessary for convergence of RL algs}}$$

discount factor necessary for convergence of RL algs

1. Collect and store (s, a, ℓ, s') tuples
   ↳ usually using π_φ + noise

2. fit Q: $L(\theta) = \mathbb{E}_{(s,a,\ell,s')} \| Q_\theta(s,a) - y_{target} \|^2$

   $$y_{target} = (1-\gamma)\,\ell(s) + \gamma\left[\min\left\{\ell(s),\ Q_\theta(s', \pi_\varphi(s'))\right\}\right]$$

3. Update π_φ : $L(\varphi) = \mathbb{E}_s\left[-Q_\theta(s, \pi_\varphi(s))\right]$

4. Repeat til done

Note: at beg of training, y_{target} is inaccurate

Fisac et al. : "anneal" $\gamma : 0 \rightarrow 1$ over training

↳ Familiar?

CODE Demo

RL in practice

↳ Can often use fixed $\gamma = 0.95$ or 0.99

↳ $Q(s,a) \approx Q_\theta(s,a)$ can be unreliable, try using

$s' = f(s,a)$ ⇒ not always possible / desirable

$Q(s,a) \approx \min\{l(s), Q_\theta(s', \pi(s'))\}$

Q: What distribution is $\mathbb{E}_{(s,a,s')}$ over?

A: distribution induced by π_ψ
(and reset dist)

Follow-up Q: Where will Q_θ be good?

A: States where the safety policy visits

Takeaway Learning is not a silver bullet, each method has trade-offs

	Scales?	Supervision Strength	dynamics	Accuracy
Grid	✗	✓	analytic + cont time	Grid resolution over states
Deepreach	✓	✗	analytic + cont time	Chosen Sampling dist. over states and time
RL	✓	✓	Simulator access + discrete time	Induced Sampling dist over states and actions