Last Time:

- ☐ computation (grid, SSL, RL)

This Time

- ☐ online updates

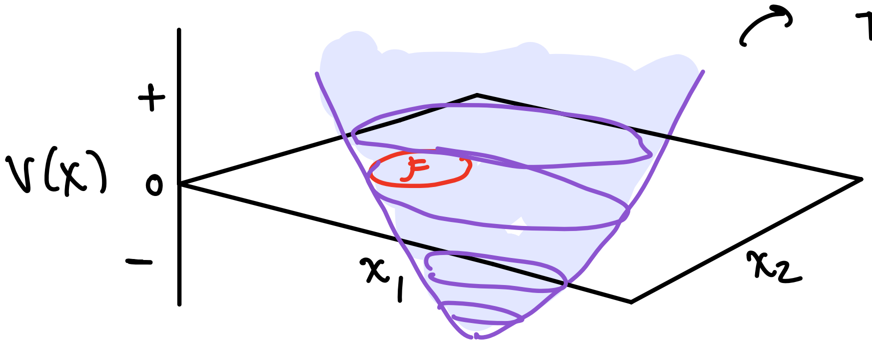
Lecture 8

EAIS S'26

Andrea Bajcsy

Online Updates

So far, we have discussed <u>offline methods</u> for computing (even approximately) the safety value function + optimal policy



$$\pi^*(x) = \arg\max_{u \in \mathcal{U}} V(f(x,u))$$

or

$$\pi^*(x) = \arg\max_{u \in \mathcal{U}} \frac{\partial V}{\partial x} \cdot f(x,u)$$

<u>Specification</u> → <u>Offline Computation</u> ⇒ <u>Online:</u>

$\mathcal{F}$, $f(x,u)$, $\mathcal{D}$ ⟶ Grid-based, RL + NN, SSL + NN (Deepreach) ⇒ $V(x)$, $\pi(x)$

$\mathcal{F}$: failure set; $f(x,u)$: dyns.; $\mathcal{D}$: disturbance bounds

TODAY: But what if I need to update $V(x)$ & $\pi(x)$ <u>online</u>?

Q When would I need online updates?

<u>A</u>: $\mathcal{F}$ is unknown a priori! (e.g. floormap)

$\mathcal{D}$ is unknown a priori! (e.g. wind gust strength)

aspects of $f_\theta(\cdot,\cdot)$ unknown a priori (e.g. mass of obj.)

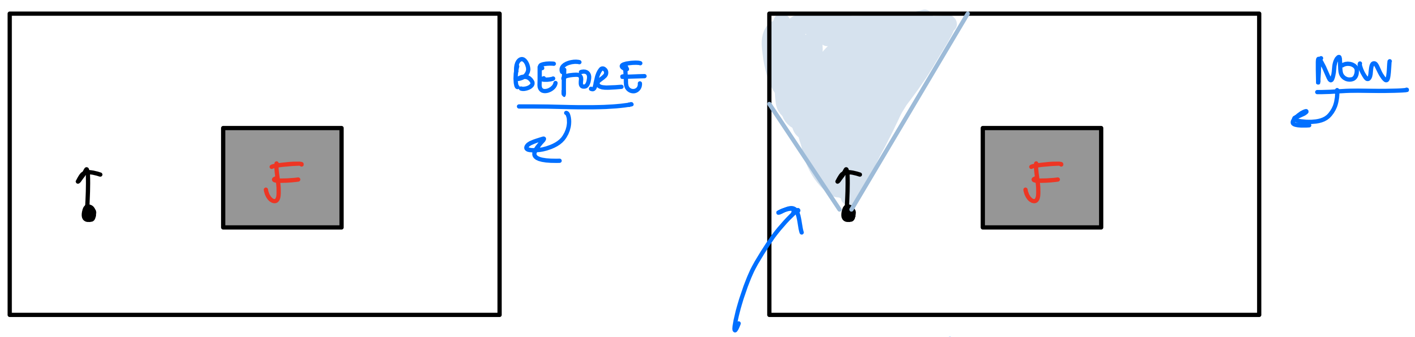
Broadly, there are 3 main methods/algorithms:

1. <u>warm-starting</u> → use old/previously computed $V(x)$ to inform <u>new</u> $V(x)$!
2. <u>local updates</u> → update only <u>small parts</u> of $V(x)$
3. <u>parameterization</u> → offline, during training, parameterize $V(x;\beta)$ where β are some params you adapt <u>online</u>

① WARM STARTING

Consider the following motivating scenario:

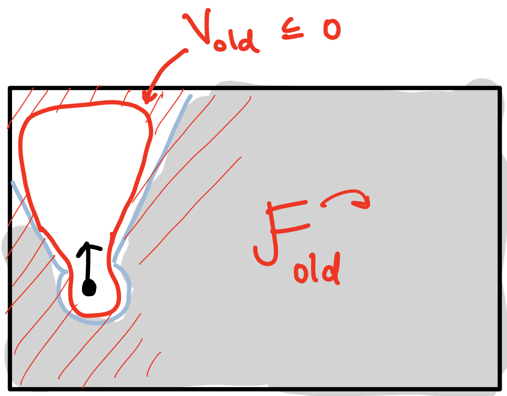
So far, we assumed we knew where $\mathcal{F}$ was apriori — ex. all obstacles in env. But, in reality we don't!



robot has front-facing camera - can't even see $\mathcal{F}$ @ start of deployment

Goal is to compute safety filter that is provably safe in the unknown environment (i.e, we will only sense $\mathcal{F}$ @ deployment time and build it over time)

At $\tau = 0$, robot is at starting state @ real time:



Initial Failure Set

$$\mathcal{F}_{old} = \{x : l_{old}(x) \leq 0\}$$

Initial Safety Comp:

$$\begin{cases} \min\left\{ l_{old}(x) - V(x,t),\ \frac{\partial V}{\partial t} + \boxed{H\left(x, \frac{\partial V}{\partial x}\right)} \right\} = 0 \\ V(x,T) = l_{old}(x) \end{cases}$$

hamiltonian

$$H := \max_{u \in \mathcal{U}} \frac{\partial V}{\partial x} \cdot f(x,u)$$

$\downarrow t \to 0$

$V_{old}(x)$

Robot takes small action + senses new obstacles!

$\tau = 1$

New Failure Set

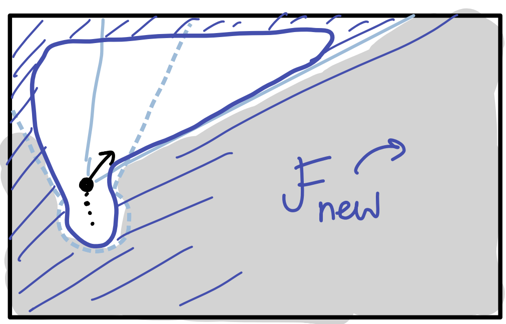
$$\mathcal{F}_{new} = \{x : l_{new}(x) \leq 0\}$$

Warm-Started Safety Comp:

$$\begin{cases} \min\left\{ l_{new}(x) - V(x,t),\ \frac{\partial V}{\partial t} + H\left(x, \frac{\partial V}{\partial x}\right) \right\} = 0 \\ V(x,T) = V_{old}(x) \end{cases}$$

$\downarrow t \to 0$

$V_{new}(x)$



Lemma (Informal; Bajcsy, CDC 2019): The safe set obtained via warm-starting is a guaranteed under-approx. of the true safe set obtained via HJI-VI.

under-approx of safe-set ⇒ more conservative ⇒ ensure safety while faster compute!

For a 4D dynamical system:

$\dot{p}^x = v\cos\theta \,, \quad \dot{p}^y = v\sin\theta \,, \quad \dot{v} = a \,, \quad \dot{\phi} = \omega$

ctrl input $u = (a, \omega)$

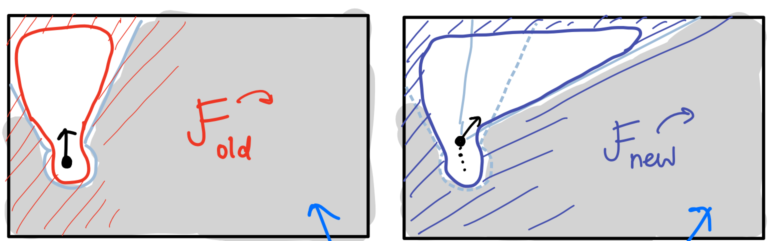
↳ Full Reach: 51.7 s (Grid-Based, MATLAB)

↳ Warm-Started: 12.5 s

② LOCAL UPDATES

OK, but even w/ warm-starting, our computation is "touching" all the states during the update, but most don't change!

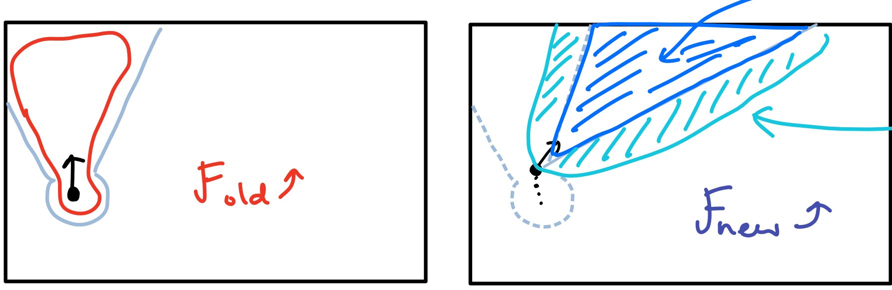
KEY IDEA: prioritize updating states where $\mathcal{F}_{old} \neq \mathcal{F}_{new}$!



not changing!

⊛ see videos from paper!

local update of the BRT (Bajcsy, CDC 2019)



new states we discovered to be free!

also include all neighbors b/c value @ one x is influenced by nearby values (ex. $\frac{\partial V}{\partial x}$)

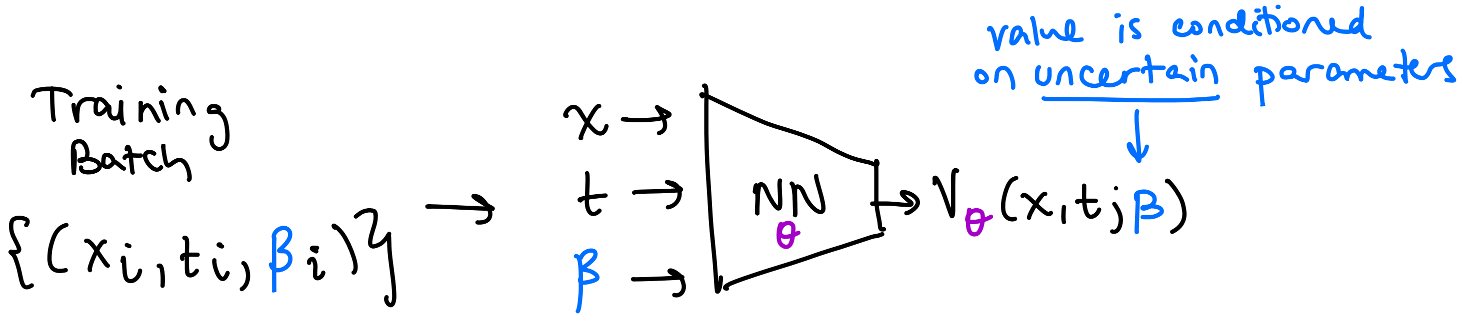
↳ Full Reach: 51.7 s (Grid-Based, MATLAB)

↳ Warm-Started: 12.5 s

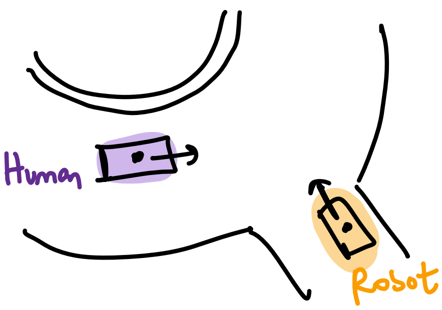
↳ Local-Update: 0.9 s

③ Parameter-Conditioning

A final useful strategy is to condition the safety value function during <u>offline computation</u> in a way that lets you adapt at runtime to new conditions:



<u>ex.</u> Multi-agent Autonomous Driving (Tian et. al, ICRA 2022)



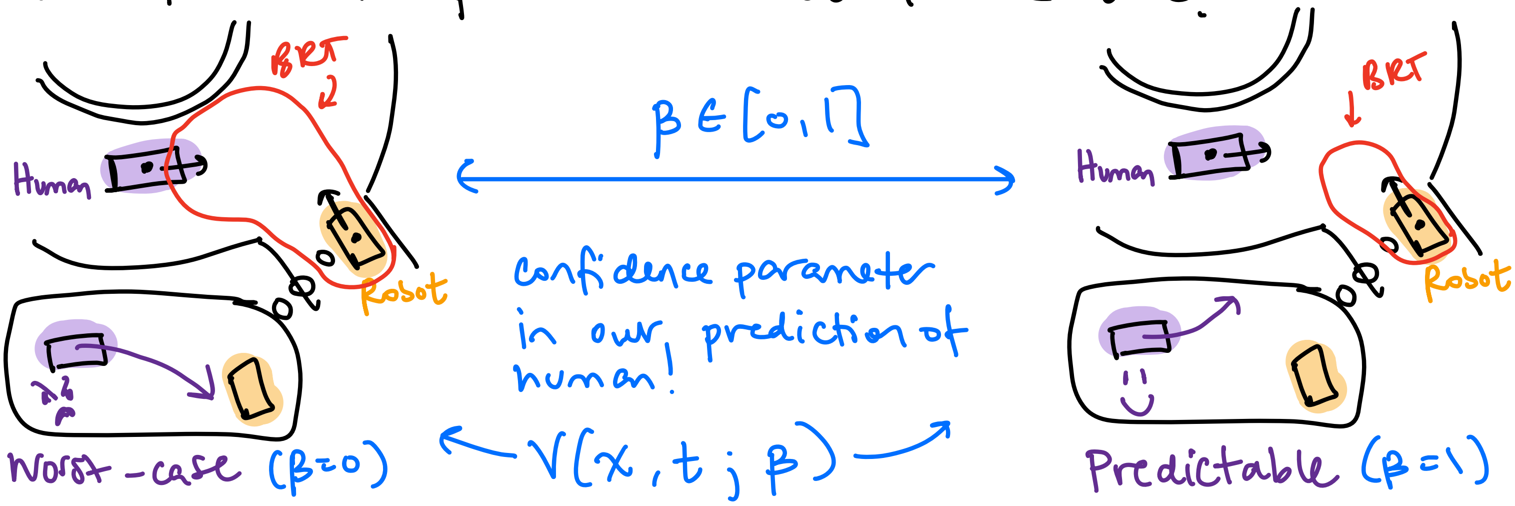
$$\dot{x} = f(x, u_R, u_H)$$

u_R: robot action (steering, accel)

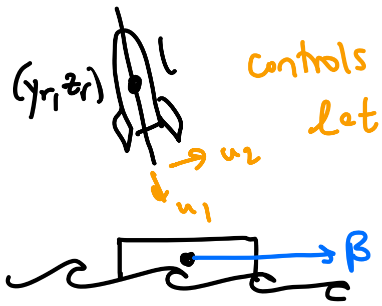
u_H: human action (e.g. steering, accel)

x: joint state (ex. relative pos., orient., velocity of Human w.r.t robot)

Here, we could model the Human as an adversary, which will take <u>any</u> feasible action $u_H \in \mathcal{U}_H$ to collide w/ car but this is too pessimistic most of the time!



ex: Rocket Landing on Floating Pad (Borquez, ICRA 2023)



controls long. & lat. forces

(how much the center of pad is displaced)

6D Dyn. System:

disturbance

$$\ddot{y} = \cos\theta\, u_1 - \sin\theta\, u_2 + d_y$$

$$\ddot{z} = \sin\theta\, u_1 - \cos\theta\, u_2 - g$$

$$\ddot{\theta} = \alpha\, u_1 + d_\theta$$

Parameterized Target Set:

landing pad can change pos. horiz. by $\pm 2l$ b/c ocean

$$\mathcal{L}(\beta) = \{ (y,z) : |y - \beta| \leq 2l,\ 0 \leq z \leq 2l \}$$

$\Rightarrow$ 7D system in total: 6D physical state + 1D β-param.