Last Time:

- System-level failures
- risk in robotics

Lecture 15

SP '26

Andrea Bajcsy

This Time:

- Controlling In-Distribution

Recap:

Thus far, we have a set of tools + mathematical frameworks for

↳ quantifying uncertainty: $P(o_t)$, $P(o_t, a_t)$, $P(y_t \mid o_t, a_t)$

($P(o_t)$: (ex. Junwon's lecture) observation density; $P(y_t \mid o_t, a_t)$: (ex. ensembles, CP...) prediction uncertainty; $P(o_t, a_t)$: observation-action density (ex. FAIL-DETECT))

↳ system-level failures: find o_t s.t. $\text{Cost}(\xi^{\pi(o_t)}) > \delta$

But, all these methods were still focused on **detection** of uncertain situations. What about **mitigation**?

Q How can a robot use its control / actuation to avoid "uncertain situations" / "OOD states"?

(OOD: out-of-distribution)

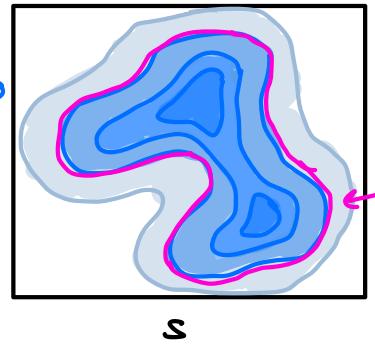
CONTROLLING IN-DISTRIBUTION

For now, let's focus on density models. So models that fit a distribution:

$P(s_t, a_t)$ over some dataset $\mathcal{D} = \{(s_t, a_t)_i\}_{i=1}^N$

(for now, assume perfect state; actions robot took)

We will say that (s_t, a_t) are **in-distribution** if $P(s_t, a_t) \geq c$, for some density level $c > 0$.



Set of in-D states & actions:

$\mathcal{D}_c := \{(s_t, a_t) : P(s_t, a_t) \geq c\}$

We want to only query our learned model on these $\mathcal{D}_c$ indistribution states/actions to mitigate shift @ test-time.

⊛ NOTE: being in-distribution does not necessarily guarantee that the learned models will be performant, but still useful.

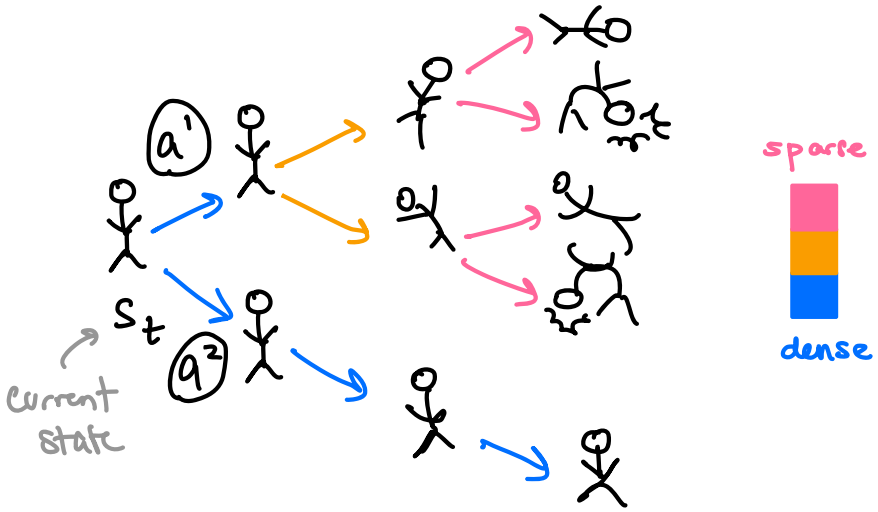
Since we don't have access to the true data generating distribution $P(s_t, a_t)$, we fit it with model $P_\theta(s_t, a_t)$. For the rest of this lecture, assume P_θ is well-fit.

Q How can we design a controller such that $P_\theta(s_t, a_t) \geq c$? (we get to pick this!)

IDEA 1 (GREEDY): At each timestep, just enforce the constraint

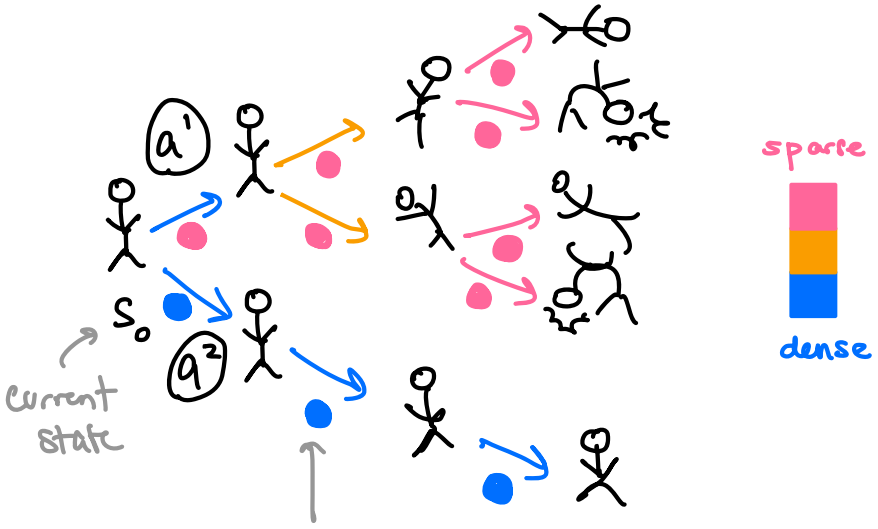
pick $a_t \in \mathcal{A}$ s.t. $P_\theta(s_t, a_t) \geq c$

Why + how would this fail? Consider this simple example from Kang et. al, ICML 2022 with discrete, limited s and as:



Suppose we want $P_\theta(s_t, a_t) \geq c_{orange}$. This constraint imposed @ the initial time would allow the agent to take a^1 and a^2. After a^1, the agent is on boundary of constraint, but is doomed to leave the high density region & enter pink states!

Instead, we want to constrain the agent's actions with the lowest density it will encounter in the future. If we did this restriction, we would see



IDEA 2 (CONTROL PROBLEM IN DENSITY SPACE):

Ok, so mathematically what do we need to satisfy? We need:

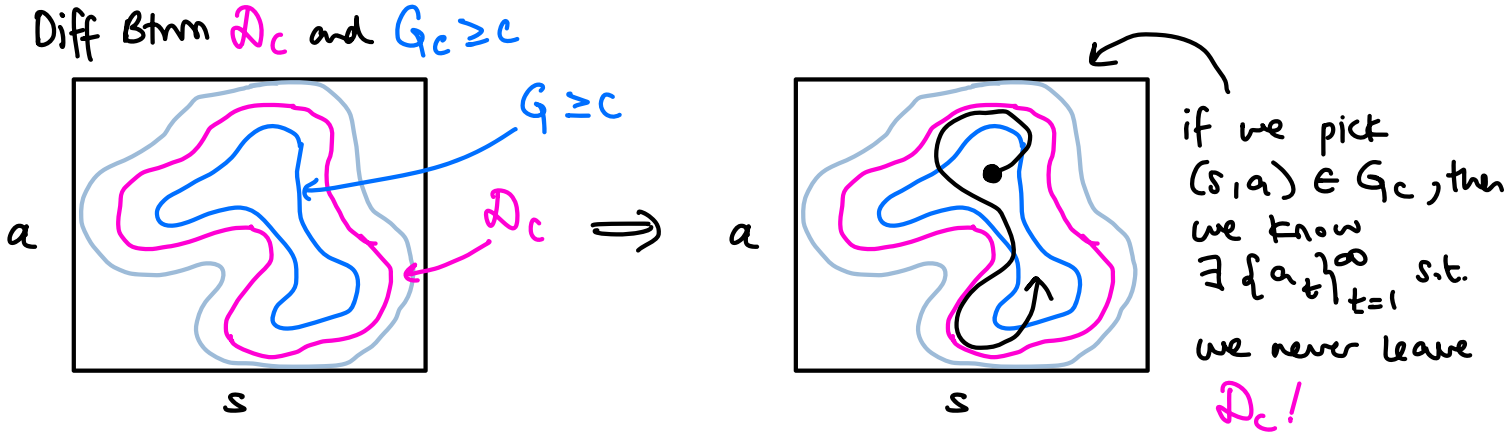
$$G(s_0, a_0) := \max_{\{a_t\}_{t=1}^{\infty}} \min_{t \geq 0} P_\theta(s_t, a_t)$$

$P_\theta(s_t, a_t)$: tells us the density @ any (s_t, a_t)

$\max_{\{a_t\}_{t=1}^{\infty}}$: robot is trying to find action sequence that stays in maximally dense region.

$\min_{t \geq 0}$: keeps track of lowest density over traj

The function $G(s_0, a_0)$'s level sets correspond to the c threshold we got to pick @ start of lecture! Bc P_θ's range is bounded by $[0, 1]$ then so will G's range. Thus, G represents the lowest density we ever experience under our ctrl. seq.



We can also use the function $G(s,a)$ within another control problem, ex. MPC:

$$a^*_{1:H} = \underset{a_{1:H}}{\text{argmax}} \sum_{t=1}^{H} r(s_t, a_t)$$

$$\text{s.t. } s_{t+1} = f(s_t, a_t), \quad \forall\, 1 \leq t \leq H-1$$

$$G(s_t, a_t) \geq c, \quad \forall\, 1 \leq t \leq H \quad \circledast$$

As with standard MPC, the controller executes the first planned action + then replans. The constraint $\circledast$ ensures that as long as the system is above the level set c, it is able to satisfy $P(s_t, a_t) \geq c$ throughout traj. This is also <u>least restrictive constraint</u> b/c $G(s,a)$ was designed to be maximal (via $\max_{\{a_t\}_{t=1}^{\infty}}$) so it gives the mpc problem max flexibility within the support.

<u>NOTE:</u> You may wonder why imposing:

$$P_\theta(s_t, a_t) \geq c \quad \forall\, 1 \leq t \leq H$$

instead of $\circledast$ wouldn't be enough? No! Not enough in gen.

The validity of this alternative constraint above depends on the horizon of the planning problem (H). If $H \to \infty$ then yes, its valid, but in practice we choose $H = 10$ or 20 which is often $150\times$ smaller than full task length for computational efficiency. Thus, we can still get myopic behavior.

⊛ See Kong et. al, ICML 2022 for theoretical proofs of:

↳ Why perfect G function vs. perfect P density differ in their errors when used for MPC.

↳ How learning G_φ and learned P_θ influence error bounds.

COMPUTATION + FRONTIERS

As you may have predicted, the optimization problem:

$$G(s_0, a_0) := \max_{\{a_t\}_{t=1}^{\infty}} \min_{t \geq 0} P_\theta(s_t, a_t)$$

looks very familiar on purpose! It's another version of a reachability problem, and as such we can leverage those tools to obtain an (approx.) $G_\varphi(s_0, a_0)$ in practice.

For example, our "safety Bellman Equation" is now:

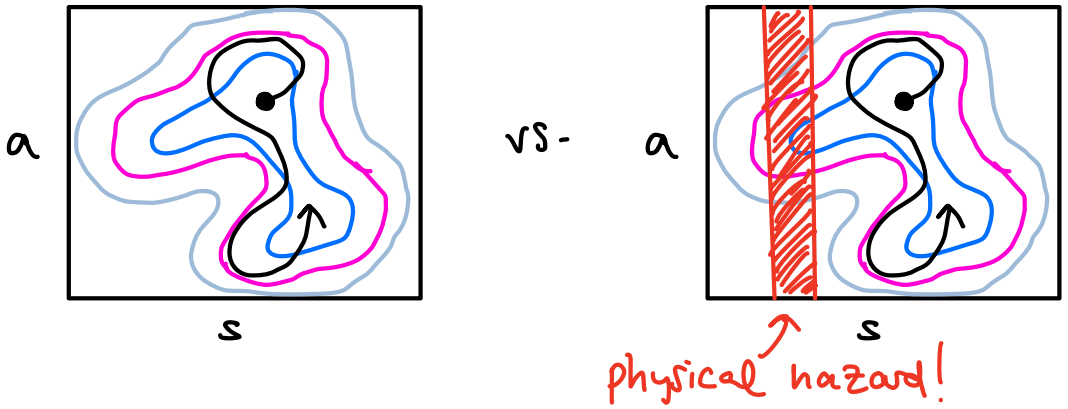
$$G_\varphi(s,a) \leftarrow \max\left\{ P_\theta(s,a), \min_{a' \in A} \gamma\, G_\varphi(f(s,a), a') \right\}$$

(φ: params; γ: discount factor)

$$G_\varphi^{\text{init}}(s,a) \leftarrow P_\theta(s,a)$$

Then we can run our far solvers $\lim_{k \to \infty} G_\varphi^k$, ex. via RL.

Another variant of the problem we may care about is to BOTH avoid OOD situations AND physical hazards.



Using notation from Seo et. al, CoRL 2025, let augmented state:

$$z = \begin{bmatrix} s \\ u \end{bmatrix} \quad \leftarrow s \text{ is physical state (ex. pos, vel, latent state)} \quad \leftarrow u \in \mathbb{R} \text{ is (epistemic) uncertainty of model}$$

Here, we specifically focused on the uncertainty of the dynamics model itself: $P_\phi(s' | s, a)$ rather than the density of (s, a) [although in the paper we do compare to density estimator].

Specifically, we train ensemble of $P_\phi^k(s' | s, a)$, $k \in \{1, \dots K\}$.

and then measure:

$$u_{t+1} = \underbrace{D(s_t, a_t)}_{\text{Epistemic}} = \underbrace{H\left(\sum_{k=1}^{K} \frac{1}{K} P_\phi^k\right)}_{\text{Total Uncertainty}} - \underbrace{\sum_{k=1}^{K} \left(\frac{1}{K} H[P_\phi^k]\right)}_{\text{Aleatoric Uncertainty}}$$

(entropy)

⇒ From: [Shyam et. al, Model-Based Active Explore..." ICML 2019.]

Then we can define two types of constraints:

$$\mathcal{F}_{\text{physical}} := \{ z : l(s) < 0 \}$$

(l: typical signed dist fncl, ex. signedDist(s, obstacle))

$$\mathcal{F}_{\text{OOD}} := \{ z : u > c \} \Rightarrow \underline{\mathcal{F} = \mathcal{F}_{\text{physical}} \cup \mathcal{F}_{\text{OOD}}}$$

We have augmented dynamics:

$$z_{t+1} = f(z_t, a_t) = \begin{bmatrix} f(s_t, a_t) \\ D(s_t, a_t) \end{bmatrix} \leftarrow := s_{t+1}$$

$:= u_{t+1}$ which is measured

via uncertainty of $\mathbb{P}(s_{t+1} | s_t, a_t)$!

ex. divergence of ensemble of predictions!

$\Rightarrow$ from here, standard computations can follow!