Lecture 9

EAIS S'26

Andrea Bajcsy

Last Time:

- updating safety online

This time:

- latent-space safety

Why Latent States?

So far, we have assumed a hand-designed representation of state, $x \in \mathcal{X}$, that we can perfectly observe. But this is never true in practice! We need to deal with real obs, $o_t \in \mathcal{O}$, coming from sensor.

Ok, well the traditional thing to do is to design a <u>state estimator</u>: $p(x_t \mid o_{0:t}, a_{0:t})$ which gives us a (distribution) over the current state given a history of the robot's observations and actions.

history of observations (e.g. RGB camera img)

history of robot actions

⚠ BUT what *is* the representation of x? Recall the problem of <u>cutting a rope</u> -- what is x?

 ← <u>cutting an onion</u> -- what is x?

$f(x_t, a_t) = x_{t+1}$

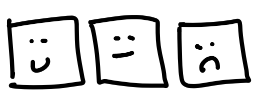
⚠ On top of this, I can't hand-design a <u>dynamics model</u> for such a complicated interaction!

<u>IDEA:</u> Let's <u>learn</u> the state representation and dynamics directly from $\mathcal{D} = \{(o^i_{0:t}, a^i_{0:t})\}_{i=1}^N$ observation-action data:

INPUTS

OUTPUTS

"latent" state

$\mathcal{D} = \{$  , down-down-down $\}$ → $\{z \in \mathcal{Z},\ f(z_t, a_t)\}$

$o_{0:t}$ $a_{0:t}$

"images of smile"   "muscle twitches"   "face state"   "model of how you make expressions"

<u>GOAL:</u> with learned (z, f) maybe we can extend all our safe ctrl math to compute safety $(\pi(z), V(z))$!

ROADMAP

1. "world models" (i.e. how to learn z, f jointly)
   denoted by WMs
2. safe ctrl in WMs
   → specifying safety constraints
   → policy and value learning
   → runtime filtering of visuomotor policies

   + what is uniquely hard in safety!

3. open challenges

PART 1 – WORLD MODELS

world model (intuitive) : given current observation and action(s), predict future outcome.

It can mean many things:

- Markovian state-based model (today!)
- video diffusion / flow matching model

⊛ note: some communities call text-to-img or text-to-video WMs. I usually mean an embodied action-conditioned model!

↳ see: [1] Mei et. al. "Video Gen. Models in Robotics", 2026.
[2] Li et. al. "Survey on WMs in Embodied AI", 2025.

Markovian State-based Models

Assume the future evolution of the agent's environment only depends on the state variable z_t at time t and action a_t → we have been making this assumption too!

this is def$^{\underline{n}}$ of Markovian

$$z_{t+1} \sim p_\eta(z_{t+1} \mid \underline{z_t}, \underline{a_t}) \quad \text{vs.} \quad p(z_{t+1} \mid \underline{z_{0:t}}, \underline{a_{0:t}})$$

learned model parameters

Non-Markovian

⚠ video models do not explicitly model Markovian state – they directly transform patches of pixels into future pixels / video frames.

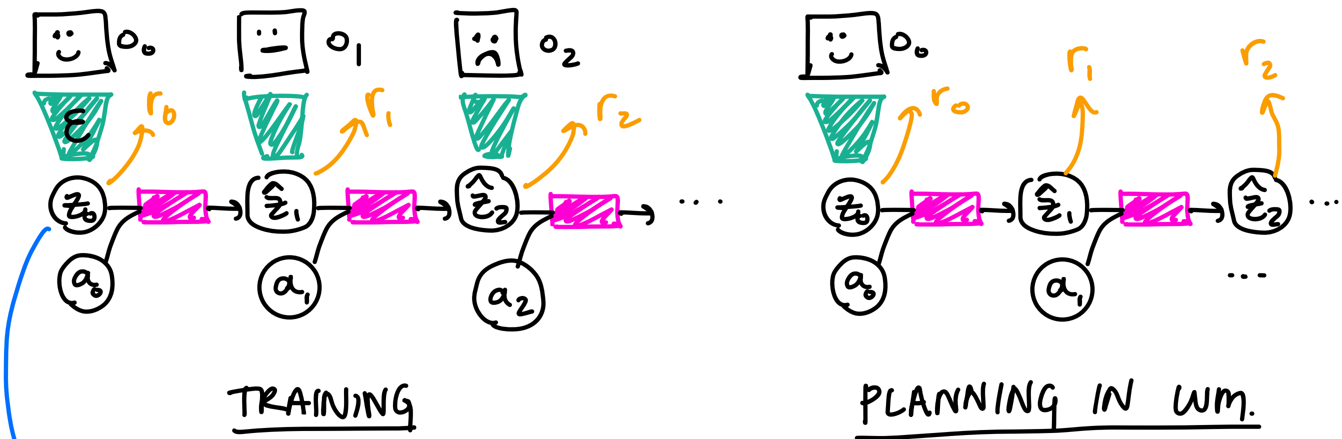
Markovian world models consist of 2-3 ingredients:

uninterp. vector $\in \mathbb{R}^{1000}$ or $\mathbb{R}^{512}$...

Encoder: $z_t \sim \mathcal{E}_{\gamma}(z_t \mid o_t)$ params

Dynamics: $\hat{z}_{t+1} \sim p_{\eta}(\hat{z}_{t+1} \mid z_t, a_t)$ params

Reward: $r_t \sim p_{\varphi}(r_t \mid z_t)$ params → or in safety it's going to be our constraint

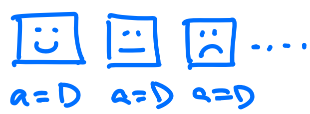


ex. this is an 512-D vector whose values are learned

Three things matter in ML: DATA, MODEL ARCH, LOSS FUNC.!

(A) DATA: The only data we assume is:

(o_t, a_t, o_{t+1}) → ex.



Q what is our dream "composition" of this data? i.e. what data distrib. do we want?

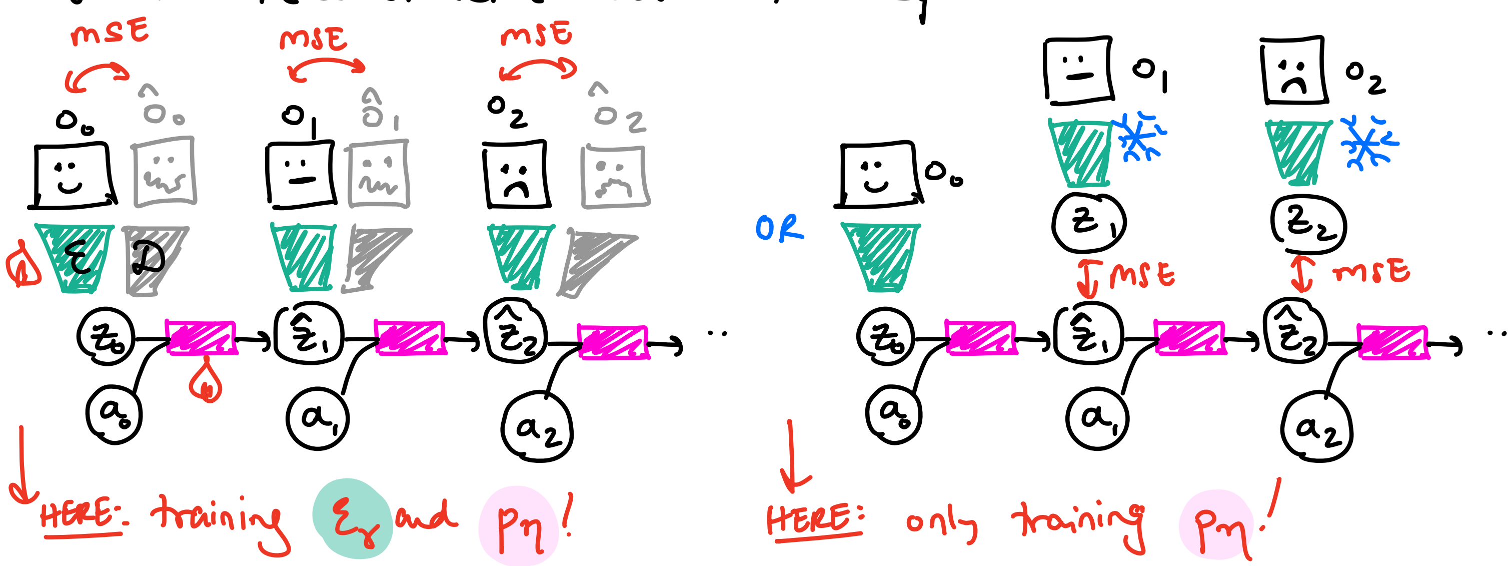
A High coverage! want for every o_t, want to see diverse a_t and o_{t+1}'s so we can learn to predict consequences. It's a system ID problem! Only expert data here is less helpful.

(B) MODEL ARCH: Foundational works use RNN or RSSMs to model $p_\eta(z_{t+1} | z_t, a_t)$ (PlaNet, Dreamer v1 - v4)

This is so the latent state had a notion of memory

⇒ Recent works freeze $\mathcal{E}(z_t | o_t)$ with pre-trained encoder like DINOv2 / v3 and only train $p(z_{t+1} | z_t, a_t)$ [DINO-WM, Zhou et. al 2025]

(C) LOSS FUNCTION: usually trained to minimize error btwn. predicted and "true" next state, either in latent space or in reconstructed observation space



PART 2 Safe Ctrl. in Latent Spaces

OK, so now we have our ingredients for control!

Encoder: $z_t \sim \mathcal{E}_\gamma(z_t | o_t)$

Dynamics: $\hat{z}_{t+1} \sim p_\eta(\hat{z}_{t+1} | z_t, a_t)$

We now need to tackle 3 things:

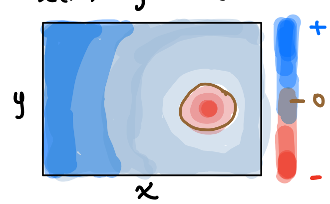
SAFETY SPEC., POLICY & VALUE LEARN, FILTERING

(4) **SAFETY SPEC**: in traditional safety, we defined

$$\mathcal{F} = \{x : \ell(x) < 0\}$$

where $\ell(x)$ was signed distance function.

$\ell(x)$ signed dist.



$$\ell(x) = \sqrt{x^2+y^2} - R$$

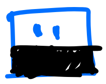
simple idea: let's learn the ℓ function

Constraint: $l_t = \ell(z_t)$

where $l_t < 0 \iff z_t \in \mathcal{F}$ & $l_t > 0 \iff z_t \notin \mathcal{F}$.

[Q] How do we train this? What data + assumptions do we need?

[A] Assume failure is observable in observation o_t.

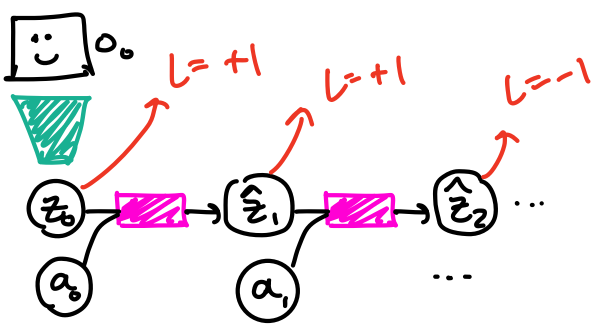
o_t  I can see you are sad! vs. o_t  Your smile is unobservable!

[A] Label the WM dataset $\mathcal{D} = \{(o_t, a_t, o_{t+1})\}$ with

$(o_t, l_t = 1)$ if looks safe & $(o_t, l_t = -1)$ if failure

Then, train $\ell(z_t)$ as binary classifier on z:

$$(o_t, l_t) \sim \mathcal{D}_{train} \Rightarrow \mathcal{E}(o_t) = z_t \rightarrow \ell(z_t) = l_t$$



PLANNING IN WM.

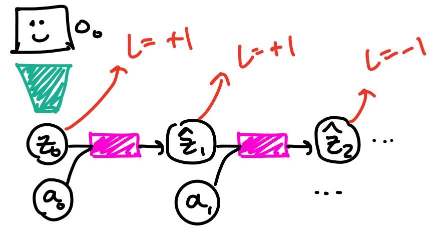
[Q] why don't we train $\ell(z)$ like a proper signed-dist. in the latent space?

[A] hard to assign per-frame real-valued labels indicating how close the robot is to failure!

(B) **POLICY & VALUE LEARN:** In theory, we can run our favorite reachability RL solver, BUT now entirely in the WM's imagination! No need for expensive real-world rollouts / interactions!

Latent HJB Equation:

$$V(z) = \min \left\{ \ell(z), \max_{a \in A} \mathop{\mathbb{E}}_{\hat{z}' \sim p_\eta(\cdot \mid z_t, a_t)} \left[V(\hat{z}') \right] \right\}$$



(!) NOTE: latent dyns. $p_\eta(\cdot \mid z_t, a_t)$ are usually stochastic — need to handle this expectation carefully (open research Q! :))

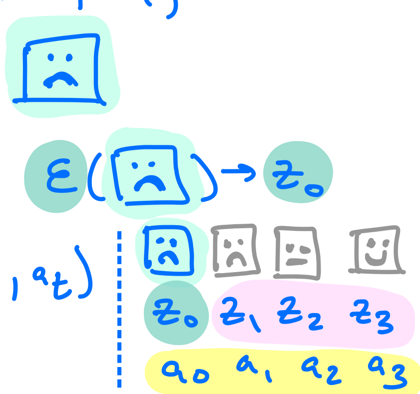
(!!) Need to take care w/ how we do "resets" in the WM when doing RL.
If we just randomly sample an initial z_0 to start a rollout from (i.e. to get $z_{0:T}$ via simulating outcomes of $a_{0:T}$ starting from z_0 using $p_\eta(z_{t+1} \mid z_t, a_t)$) we get $\boxed{\text{noise}}$! To stay on data manifold, do:

(1) sample image: $o_0 \sim \mathcal{D}^{\text{Train}}_{\text{WM}}$

(2) encode: $z_0 \sim \mathcal{E}_\gamma(z_0 \mid o_0)$

(3) simulate $a_{0:T}$ in WM: $z_{t+1} \sim p_\eta(\cdot \mid z_t, a_t)$

ex:



(C) **DEPLOYMENT-TIME FILTERING**

Great! So as we know, we eventually compute a safety policy & safety value function, but now it operates on the latent encodings of high-D obs:

$\mathcal{E}(o)$

$$\pi^{safe}(z),\ V^{safe}(z) = \arg\max_a Q^{safe}(z, a)$$

where we get a "fresh" z_t @ each real timestep encoding the current observation: $\mathcal{E}(o_t) \to z_t$

Let's look @ demo of this in action!

PROBLEM: least-restrictive safety filter can sometimes compromise task performance!

IDEA: what if we did optimization-based filtering (like CBF's do)?

ex. Diffusion Policy predicts a^{nom}

$$a^* = \arg\min_{a \in A} \| a - \pi^{nom}(o) \|_2^2$$

$\mathcal{E}(o)$

$$\text{s.t. } Q^{safe}(z, a) \geq \alpha\left(Q^{safe}(z, \pi^{safe}(z))\right)$$

Recall how Q^{safe} was trained with signal from the BINARY CLASSIFIER $\ell(z)$. What could go wrong with Q^{safe} when I try to use it in ?

Near discrete jumps @ boundary $\Rightarrow$ can't evaluate how current actions change long-term safety

WANT This!

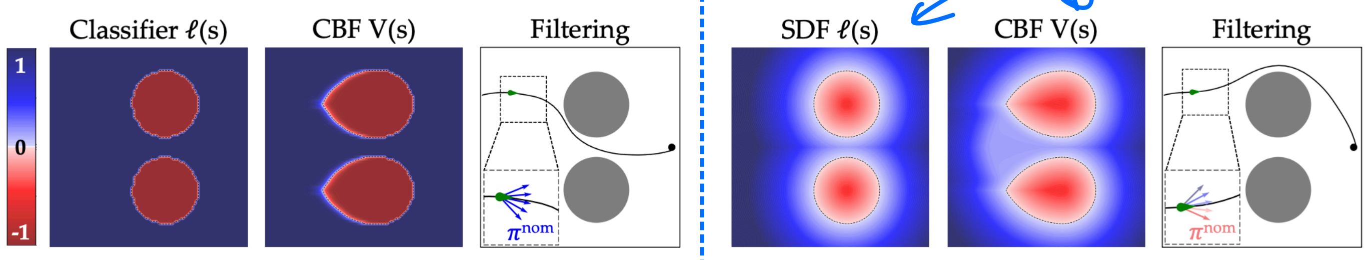


Figure 2: **CBFs as a Function of the Margin Function.** Even with a perfect model, a classifier-based $\ell(s)$ yields a CBF with poor signal during action filtering (left). A smooth margin function provides a rich signal for the CBF to evaluate alternative actions (right).

Initial remedy: gradient penalty when training $\ell(z)$ regularizes its Lipschitz constant (Nakamura et al, 2025)
(i.e. penalize norm of the gradient of $\ell(z)$: $\|\nabla_z \ell(z)\|$)

Q Any other problems? This one is subtle & related to the computation lecture...

A Distribution mismatch btwn π^{safe} (train) & π^{nom} (deploy)! Since $Q^{safe}(z,a)$ is trained only w/ data from π^{safe}, then it is only reliable when evaluating $a \sim \pi^{safe}$ but NOT any other actions (like $a \sim \pi^{nom}$)!

Initial remedy [Nakamura, L4DC 2026]: when training $Q^{safe}(z,a)$, mix rollouts from π^{safe} and π^{nom} that you will shield @ deploym.

video demo of filter working better!

SUMMARY

State → Fully observed (x)

State → Partially observed (o), $o_t = \text{Sensor}(x_t)$
- → Hand-designed → $P(x_t \mid o_{0:t}, a_{0:t})$
- → Learned state → $\boxed{\mathcal{E}(z \mid o_{0:t})}$ TODAY!

Dynamics
- → Hand-designed → $x_{t+1} = f(x_t, a_t)$
- → Simulator → $o_t = \text{Sensor}_{sim}(x_t)$, $x_{t+1} = f_{sim}(x_t, a_t)$
- → Learned → $\boxed{z_{t+1} \sim f(z_t, a_t)}$ TODAY! OR $o_{t+1} \sim f(o_t, a_t)$ for video generation

OPEN CHALLENGES

(A bit in) Reading Day

(0) What "structure" do we need in latent-space?

(1) How to train effectively in high-D z-space

Directly in o-space.

(2) How to generalize framework to video generation.

(3) How to adapt safety behavior to context

(4) How to deal w/ finite data coverage (hallucinations)

(5) How to predict unsafe events w/ minimal unsafe data

(A bit in) Lecture: Semantic Safety

Lecture: controlling "in-distribution"