Safe control under input limits with neural control barrier functions

Simin Liu

Dolan Lab, Intelligent Control Lab

Outline

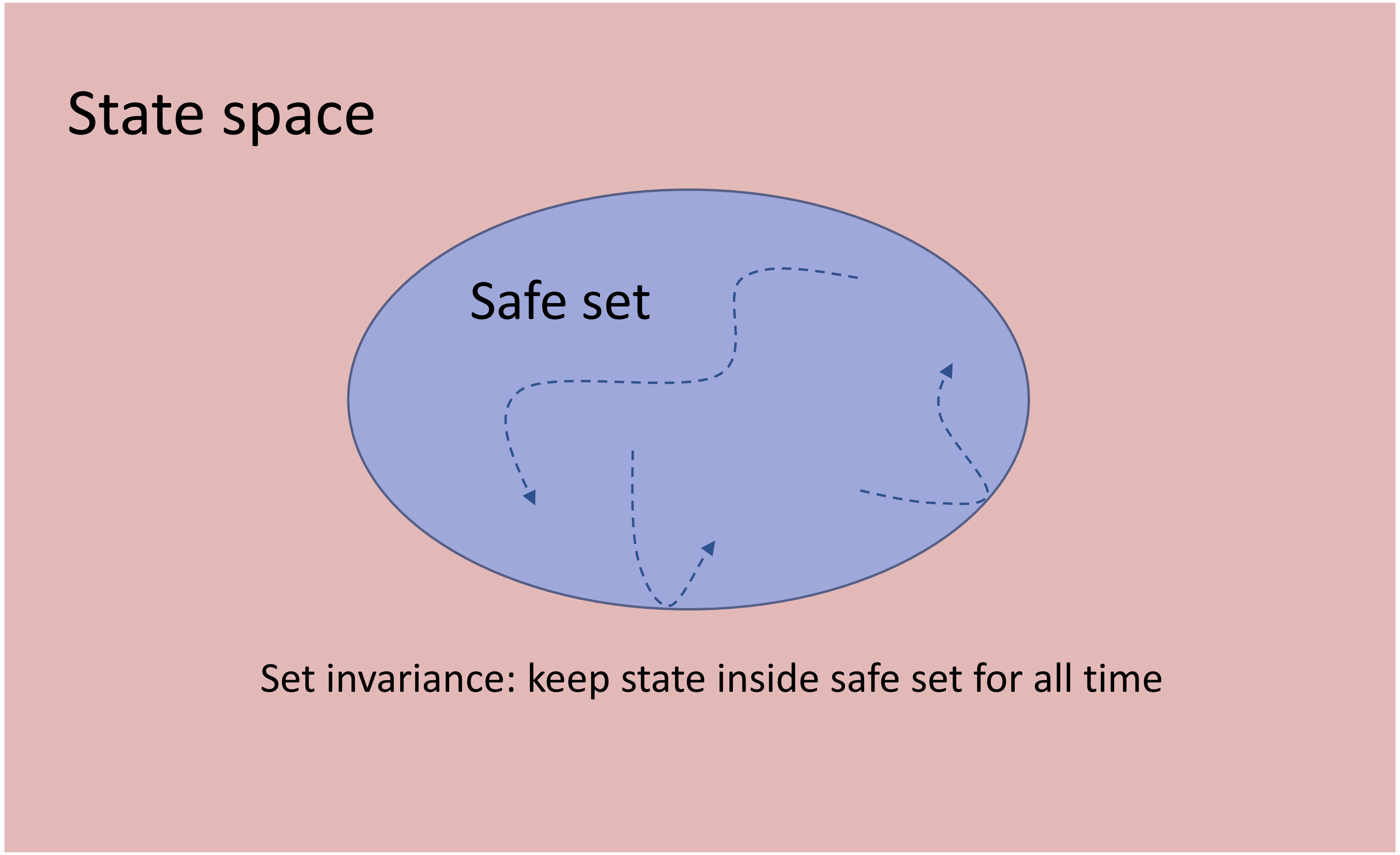
1. Why control barrier functions (CBFs)?

2. How do they work and why do input limits make them hard to construct?

3. How can we leverage tools from machine learning (neural networks, gradient-based training) to tackle this problem?

Why CBFs?

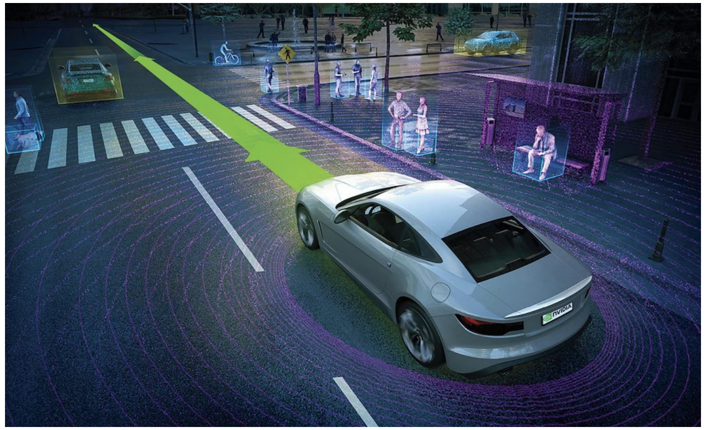
CBFs cover the expansive "set invariance" class of safety problems



CBFs cover the expansive “set invariance” class of safety problems

Safe cobot-human interaction

Bipedal locomotion



Safe trajectory planning for AVs

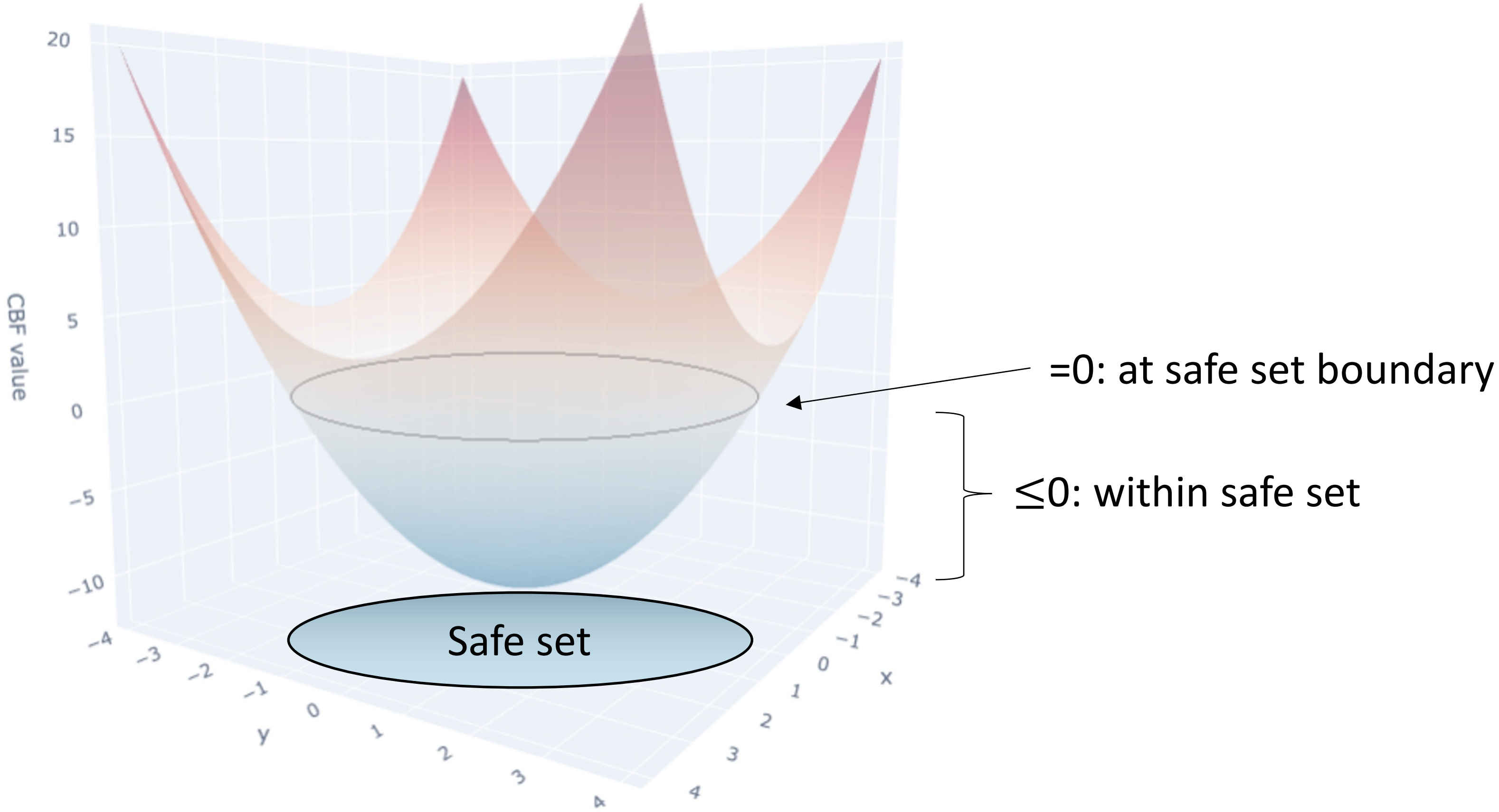
CBFs can offer provable, flexible safe control

- Provable: has mathematical guarantees of safety
- Flexible: acts as “safety layer” on top of any other policy

How do CBFs work?

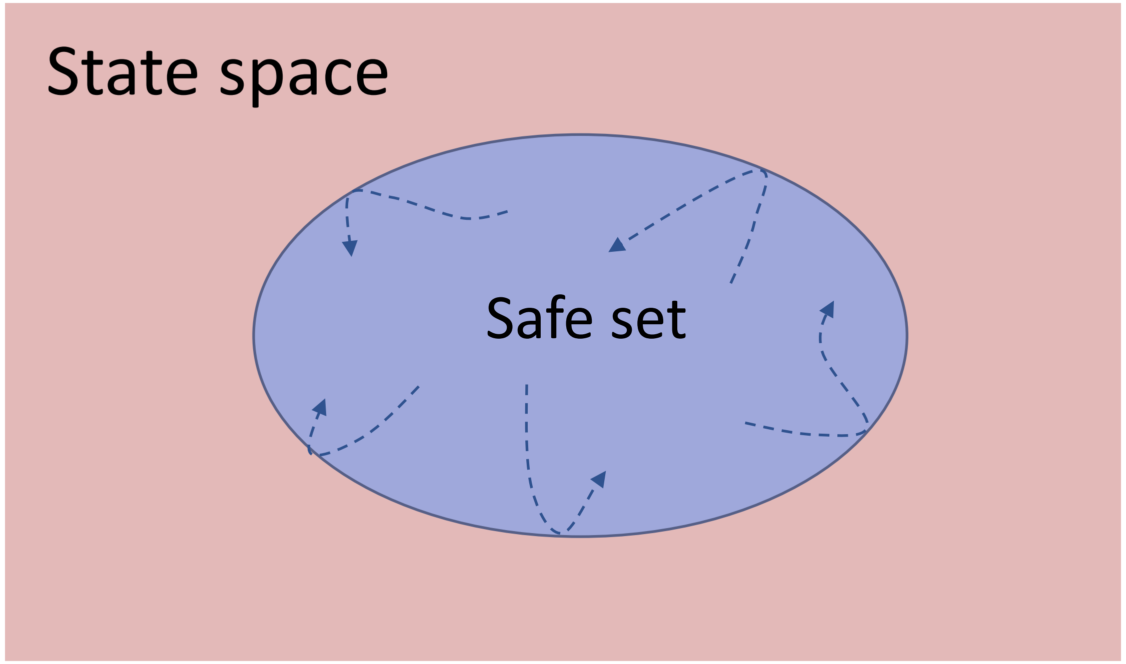
CBFs are a "danger index"

- CBFs map each state to a scalar measure of danger



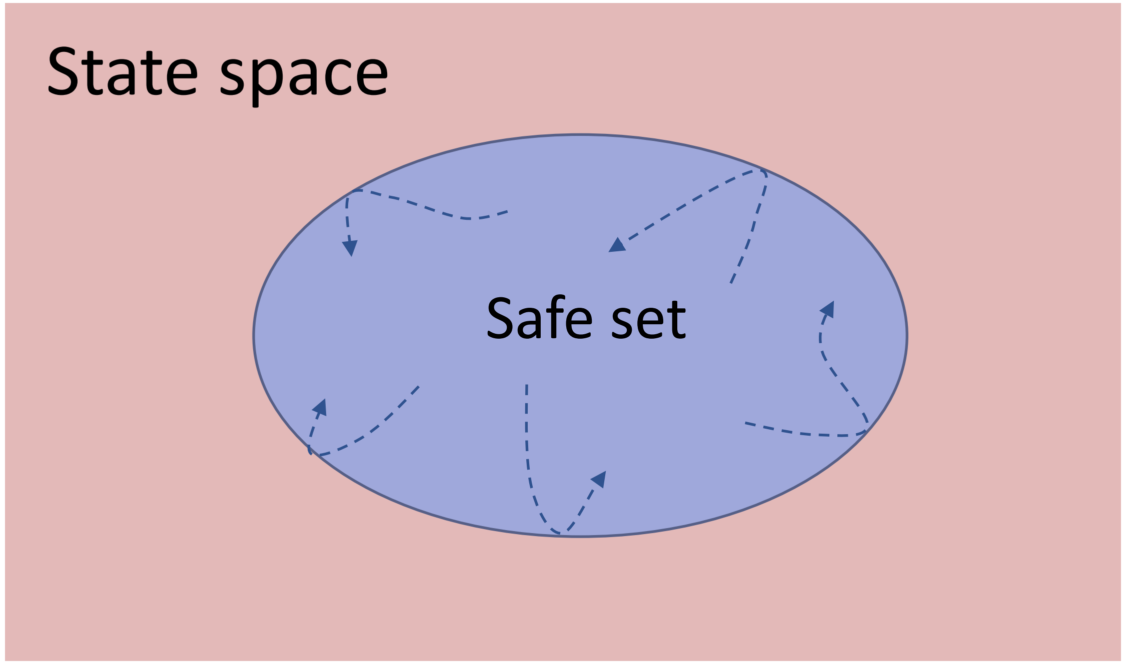
System provably safe if CBF never becomes positive

- A safe controller will decrease the CBF if it ever reaches 0



System provably safe if CBF never becomes positive

- A safe controller will decrease the CBF if it ever reaches 0
- → Constraint on what inputs a safe controller can provide at boundary states



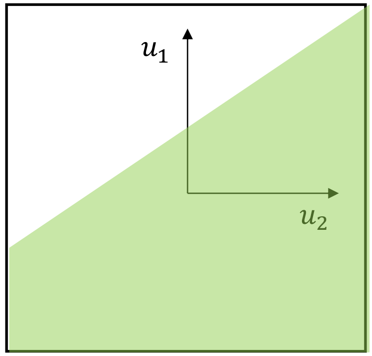
CBF gives affine "input safety constraint"

$\forall x \in \partial\mathcal{S}$, controller $k(x)$ must satisfy $\dot{\phi}(x, k(x)) \leq 0$.

Assuming a control-affine system, $\dot{x} = f(x) + g(x)u$:

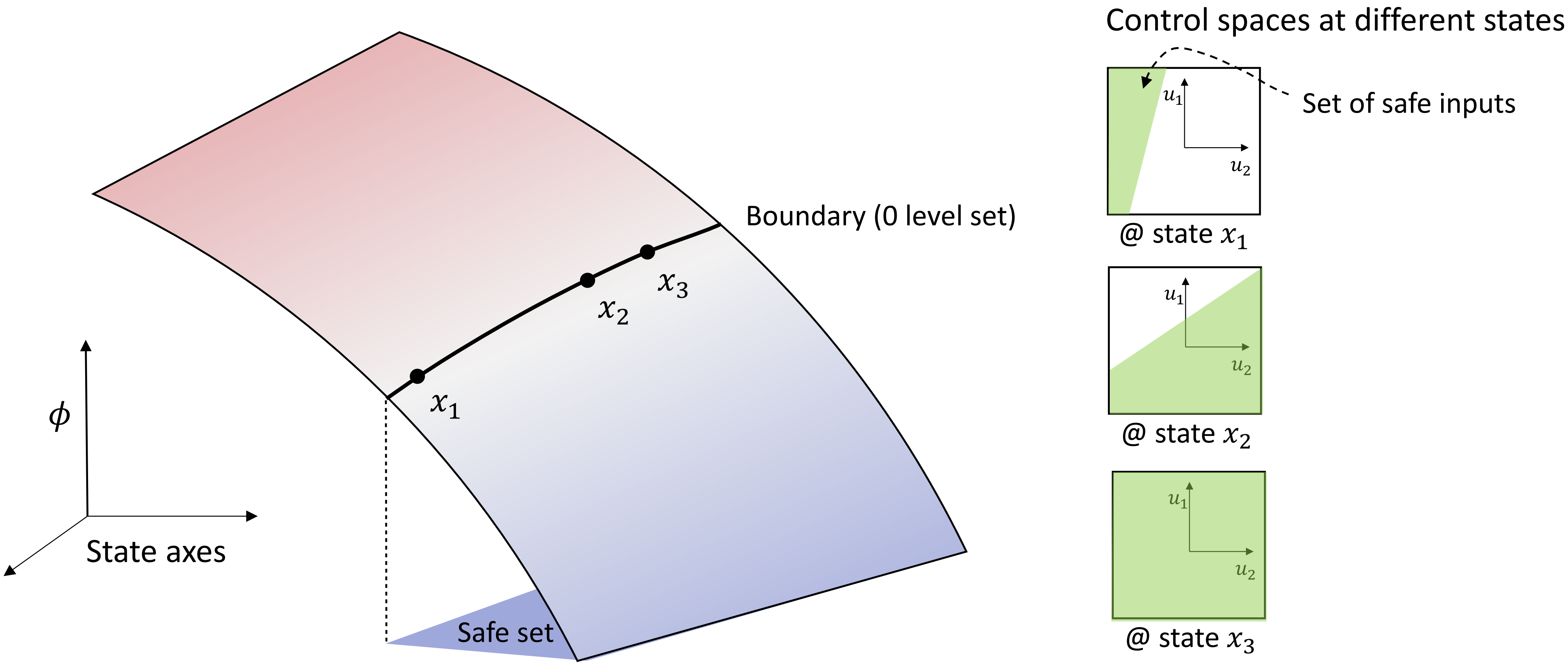
$$\dot{\phi}(x, k(x)) = \nabla_x \phi(x)^\top \dot{x} = \underbrace{\nabla_x \phi(x)^\top f(x)}_{\text{scalar}} + \underbrace{\nabla_x \phi(x)^\top g(x)}_{\text{vector}} k(x) \leq 0$$

Note: constraint also state-dependent



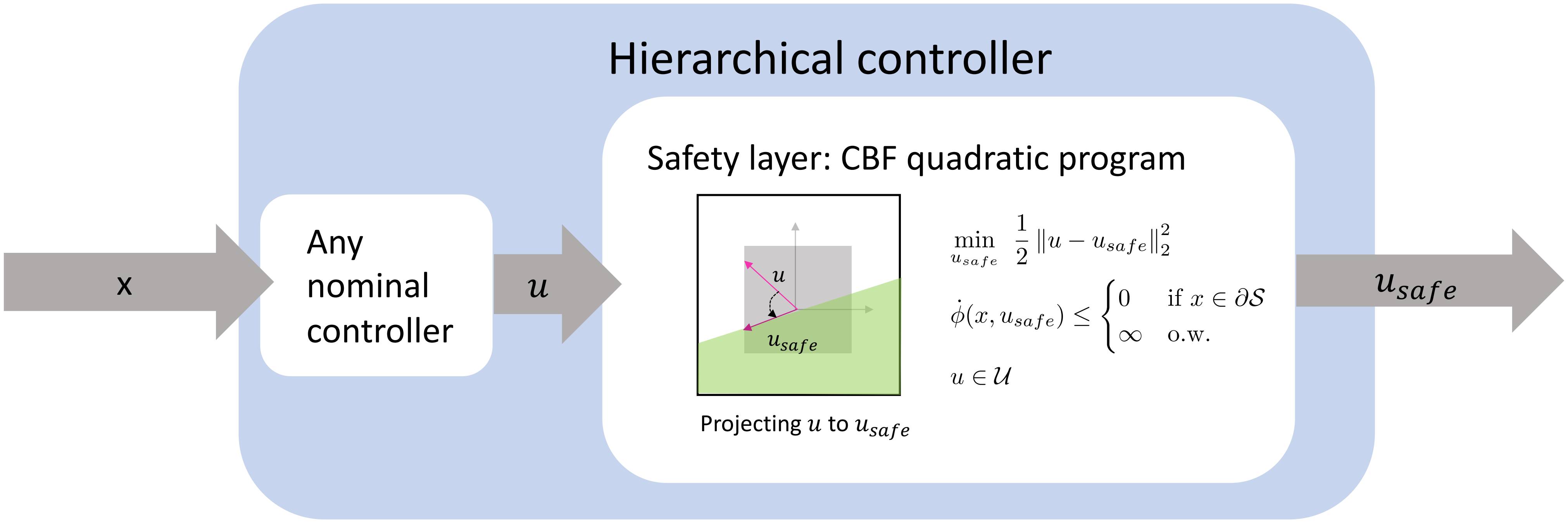
Example of constraint set in control space

CBF gives affine "input safety constraint"



“Input safety constraint” can be implemented as top layer in hierarchical controller

- This safety layer can operate on top of any controller, modifying its inputs minimally to comply with the safety constraint



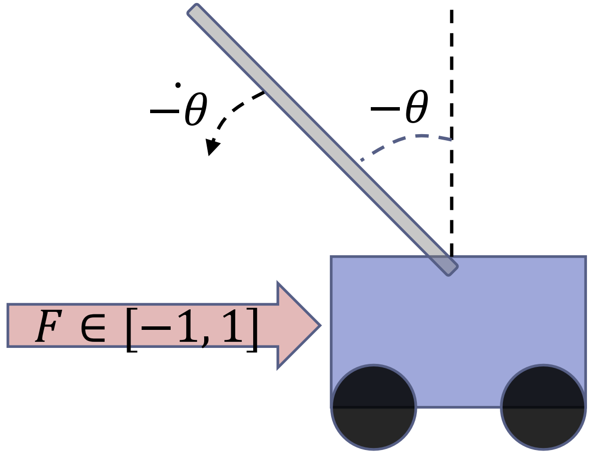
CBFs sound great! So, what’s the catch?

If CBF constructed in limit-blind way, we'll run into issues later

- Where did the CBF even come from? 🤔

- In the absence of limits, CBF constructed from safety spec using known formula (functional)

Example: limit-blind CBF for balancing cartpole



Safety spec: $\rho(x) = \theta^2 - (\pi/4)^2$

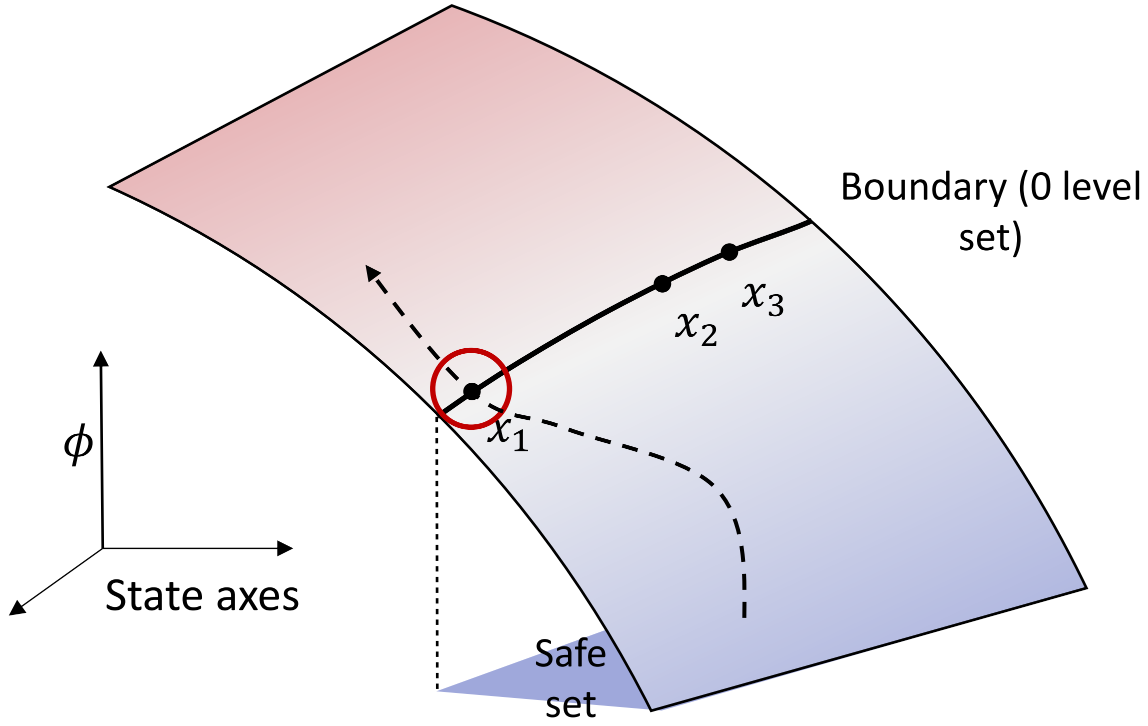
Implicitly defined by a function to keep negative

Limit-blind CBF:

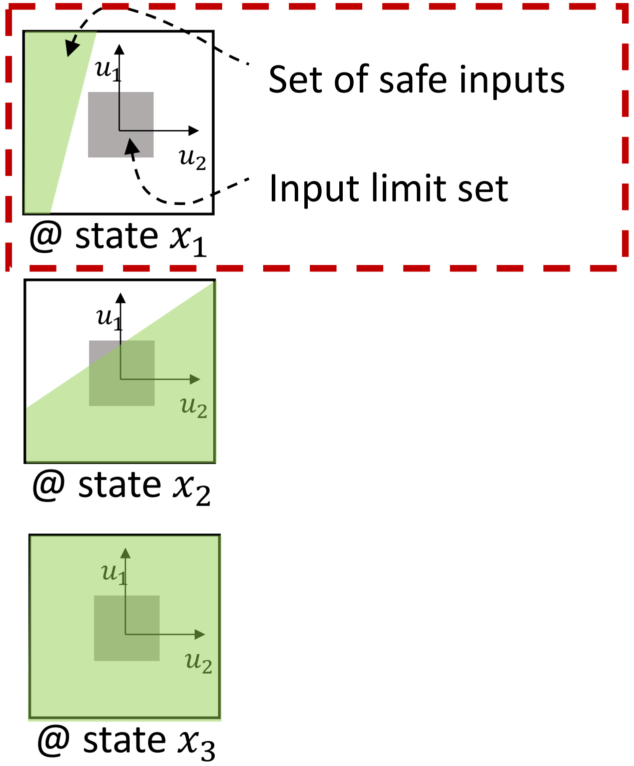
$$\phi = \rho + k\dot{\rho} = \theta^2 - (\pi/4)^2 + k\theta\dot{\theta}$$

for any $k > 0$

For limit-blind CBF, sometimes no feasible input satisfies safety constraint...



Control spaces at different states

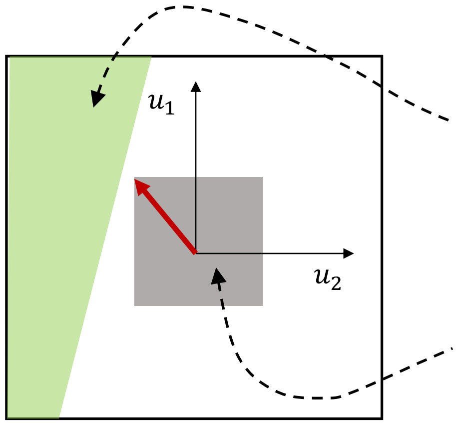


Sets don’t intersect!

In practice, can max out limits, but that doesn't ensure safety

The best we can do is max out the limits trying to minimally violate the safety law.

Control space @ state x_1

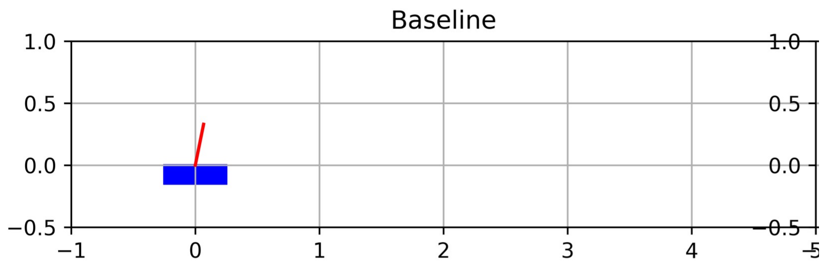


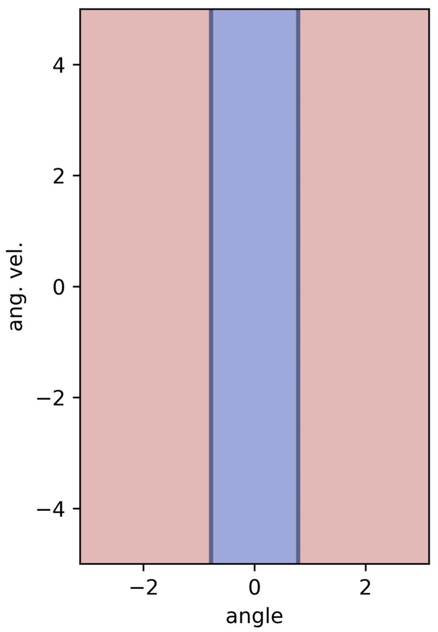
"As safe as possible" input

Set of safe inputs

Input limit set

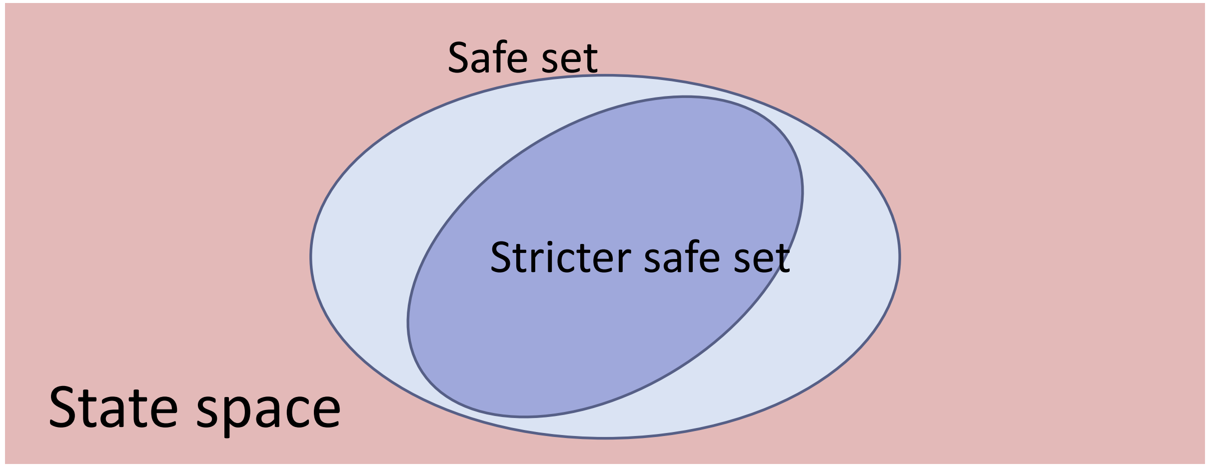
Example: limit-blind CBF for balancing cartpole





Our aim: find limit-friendly CBF

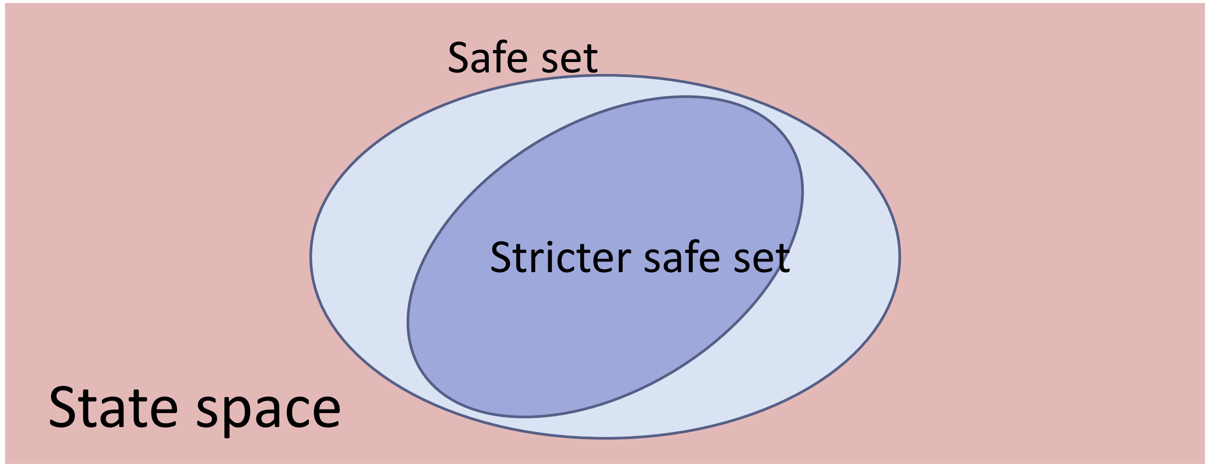
- It is valid to employ any CBF stricter than the original limit-blind CBF



- In most cases, there exists a limit-friendly CBF that is stricter
- Q: Why?

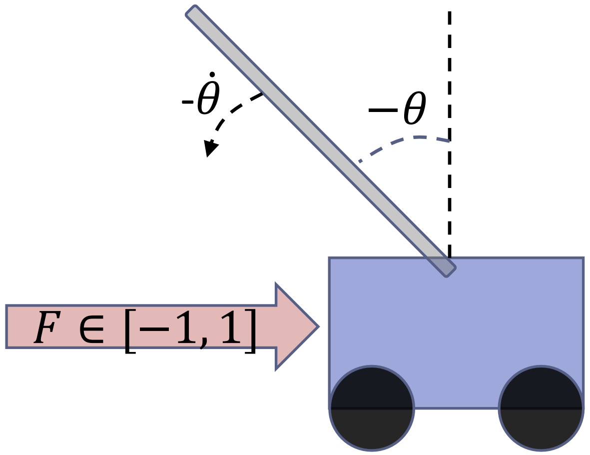
Our aim: find limit-friendly CBF

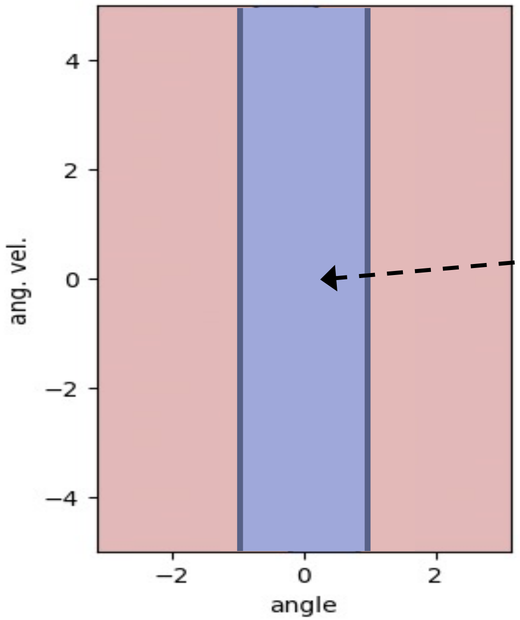
- It is valid to employ any CBF stricter than the original limit-blind CBF



- In most cases, there exists a limit-friendly CBF that is stricter
- Q: Why? Can exclude irrecoverable states

Example: balance cartpole

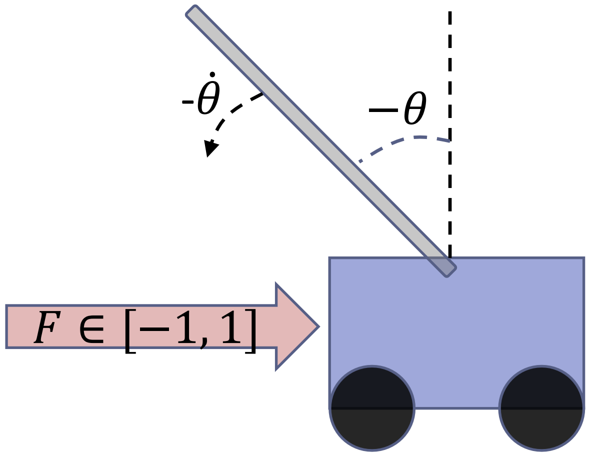


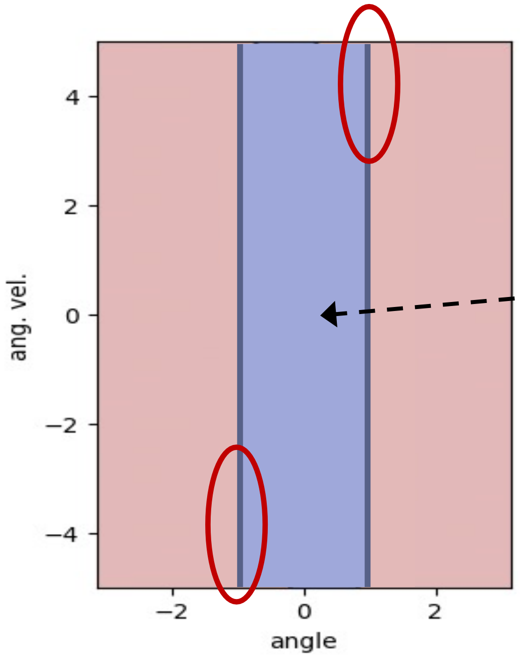


Safe set for limit-blind CBF

Q: Where on the boundary are there irrecoverable states?

Example: balance cartpole

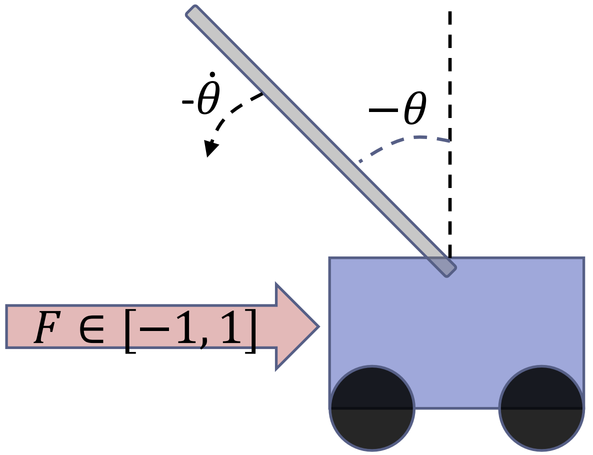


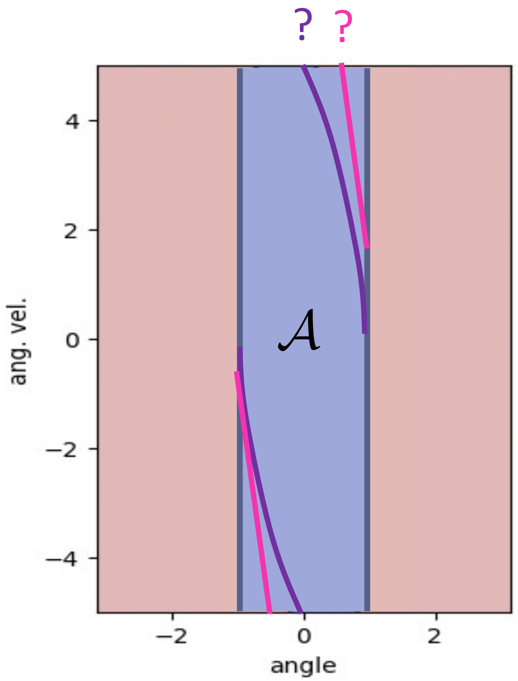


Safe set for limit-blind CBF

Q: Where on the boundary are there irrecoverable states?

Example: balance cartpole



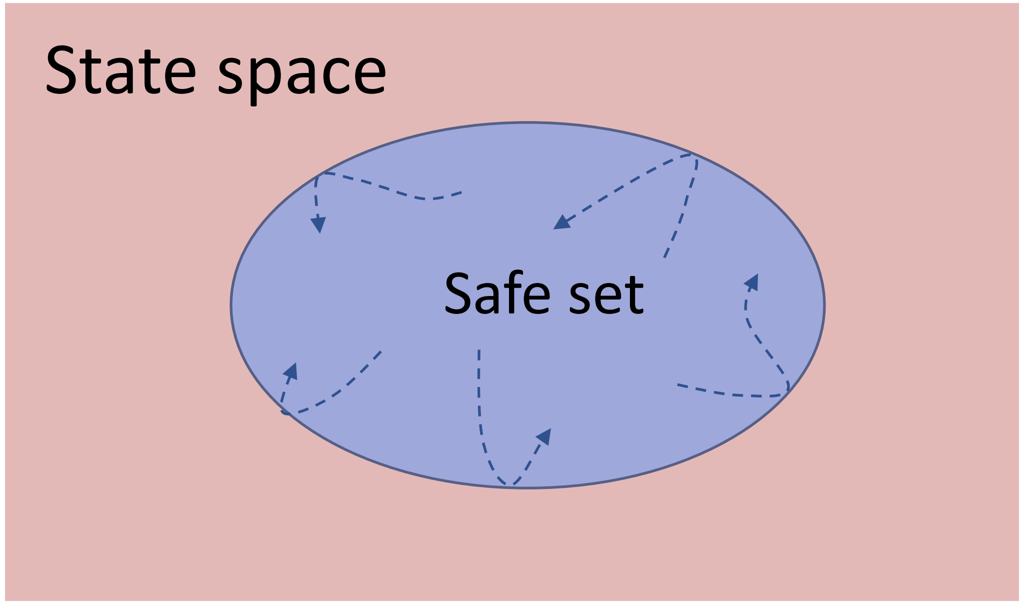


But the exact shape of a non-sat safe set is still hard to guess….

Finding limit-friendly CBF = finding CBF that obeys complex design constraint

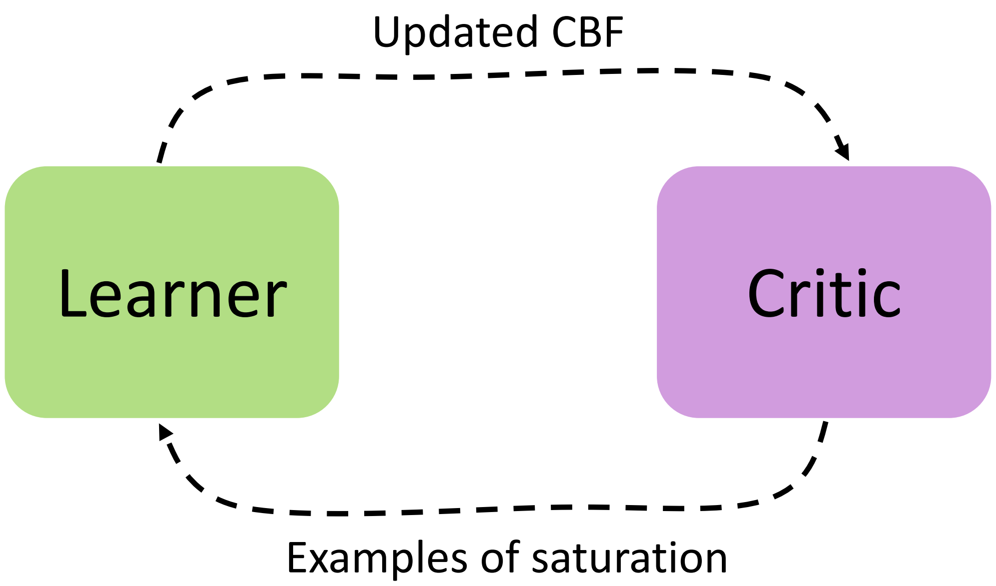
A limit-friendly CBF has parameters θ that satisfy:

$$\inf_{u \in \mathcal{U}} \dot{\phi}_\theta(x, u) \leq 0 \ , \ \ \forall x \in \partial \mathcal{S}$$



Our key ideas:

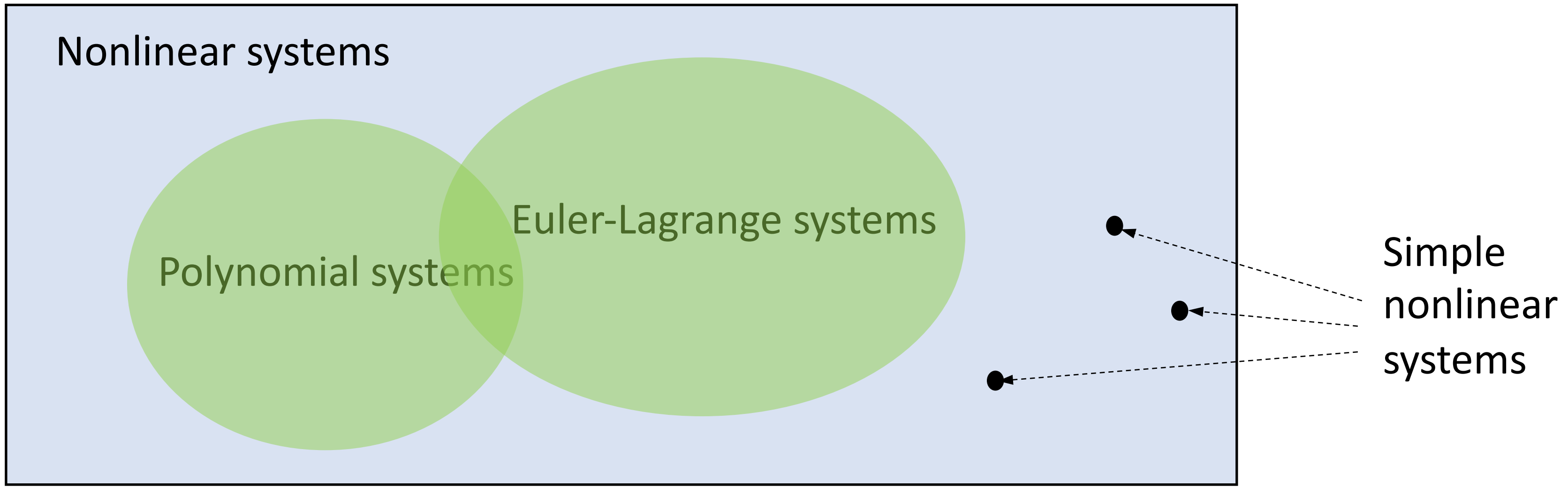
- Train generic *neural* CBF to satisfy design constraint!
- Pose as min-max optimization
- Optimize using efficient learner-critic algorithm



Unlike previous synthesis methods, ours scales to nonlinear, high-dimensional systems

- Synthesizing limit-friendly CBF is a hard problem, and the more general the system, the harder it is

- Previous works consider subclasses of nonlinear systems*



*See references 1-9.

Recap

- CBF's promise provable safety, but they're hard to construct given input limits
- Input limits pose a tough constraint on CBF
- Our idea: train neural CBF to satisfy constraint, using learner-critic algorithm
- Our synthesis method is generic, scalable, automatic

Roadmap

- Posing synthesis as min-max optimization
  - Our choice of loss function
  - Design of parametric (neural) CBF

- Using learner-critic optimization algorithm

Posing the min-max optimization

Loss function measures "how unsafe" at state x

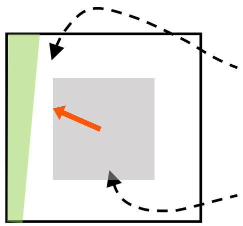
$$\inf_{u\in\mathcal{U}} \dot{\phi}_\theta(x,u) = \underbrace{\dot{\phi}_\theta(x,u^*(x))}_{\mathcal{L}(x,\theta)} \leq 0 \quad , \quad \forall x \in \partial\mathcal{S}$$

Design constraint → loss function

Interpretation:

$\mathcal{L}(x,\theta) \leq 0$: $\exists$ feasible safe input x

$\mathcal{L}(x,\theta) > 0$: measures "how unsafe" the "most safe" input is



Set of safe inputs

is the most safe input

Input limit set

Satisfying design constraint is equivalent to min-max over loss

$$\text{Want } \theta \text{ such that } \inf_{u \in \mathcal{U}} \dot{\phi}(x, u) = \mathcal{L}(x, \theta) \leq 0 \;\;, \quad \forall x \in \partial \mathcal{S}$$

$$\rightarrow \;\; \theta \text{ such that } \max_{x \in \partial \mathcal{S}} \mathcal{L}(x, \theta) \leq 0$$

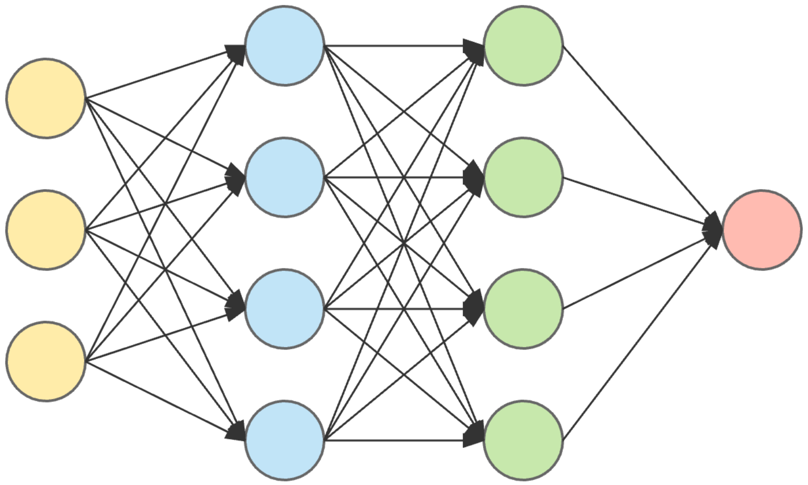
$$\rightarrow \;\; \min_{\theta} \max_{x \in \partial \mathcal{S}} \mathcal{L}(x, \theta)$$

Q: How to interpret this?

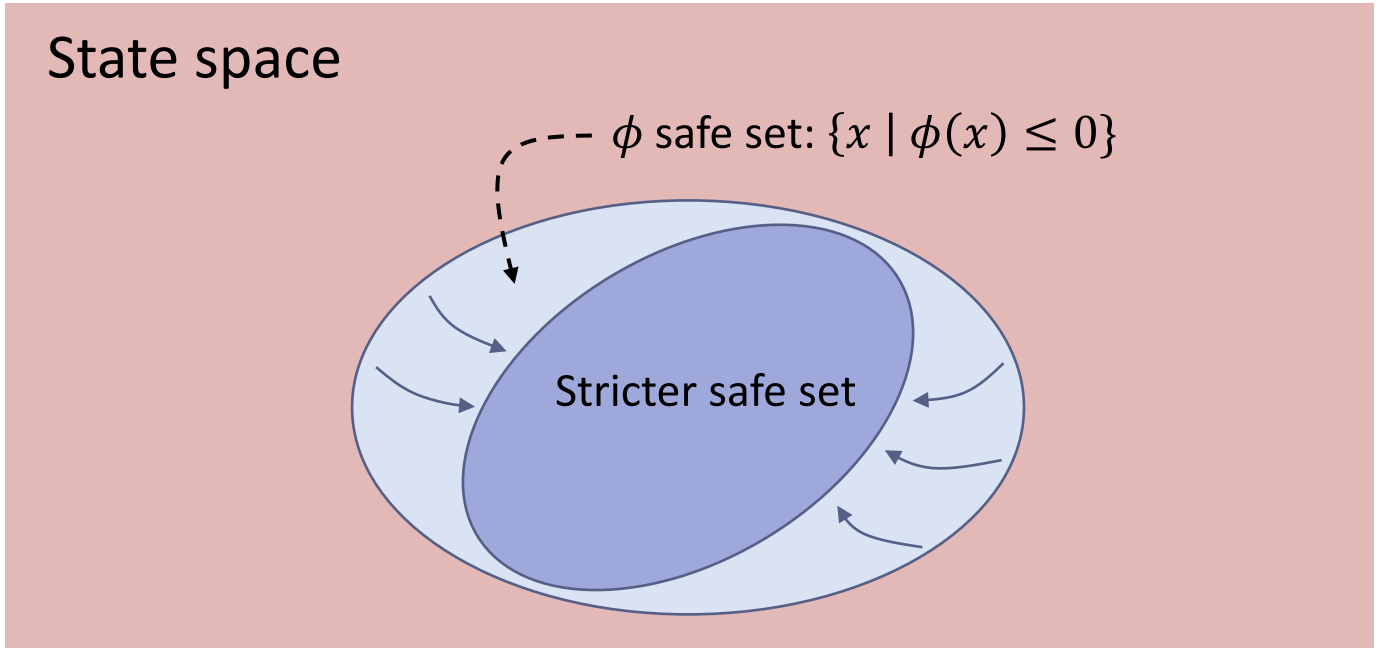
Parametrizing the CBF to optimize over

We choose a *neural* CBF, enabling generic, scalable synthesis

- Generic:
  - Can express wide range of nonlinear functions
- Scalable:
  - Can be efficiently trained on large inputs (high-dimensional systems)

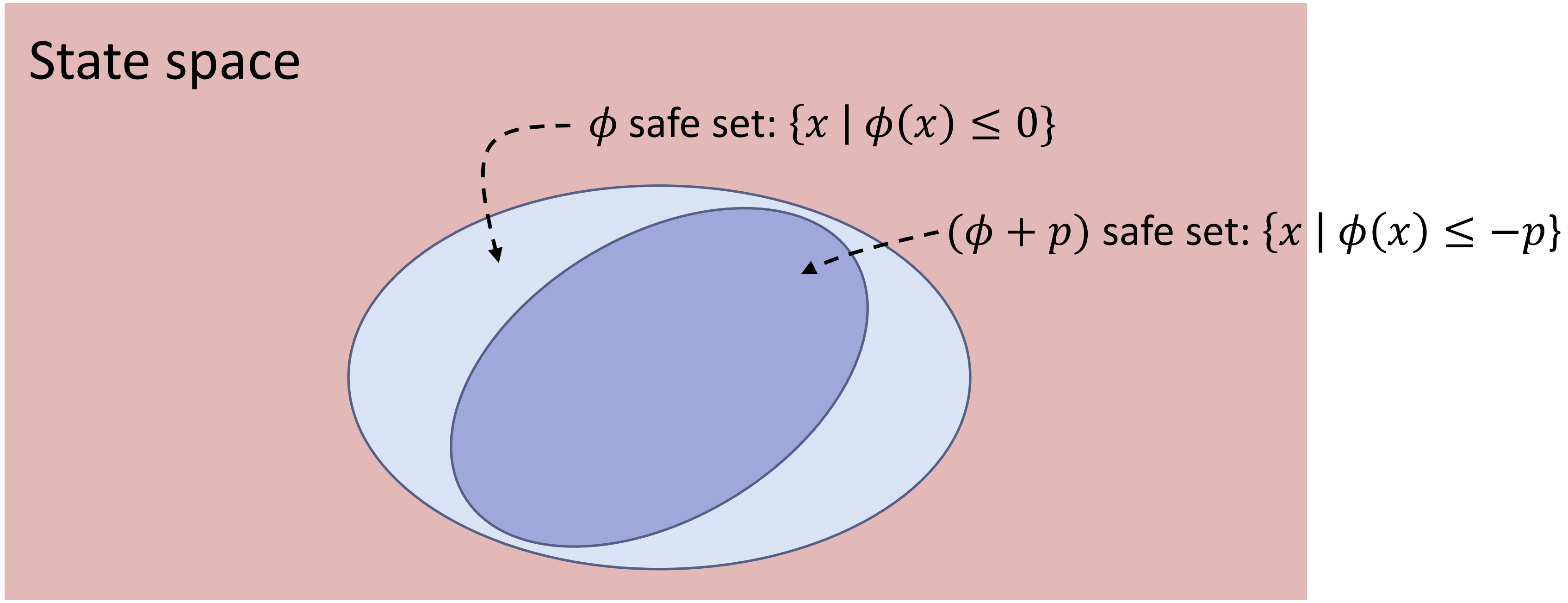


We design a neural CBF that is stricter than the limit-blind CBF



Q: how would we modify ϕ to shrink its safe set?

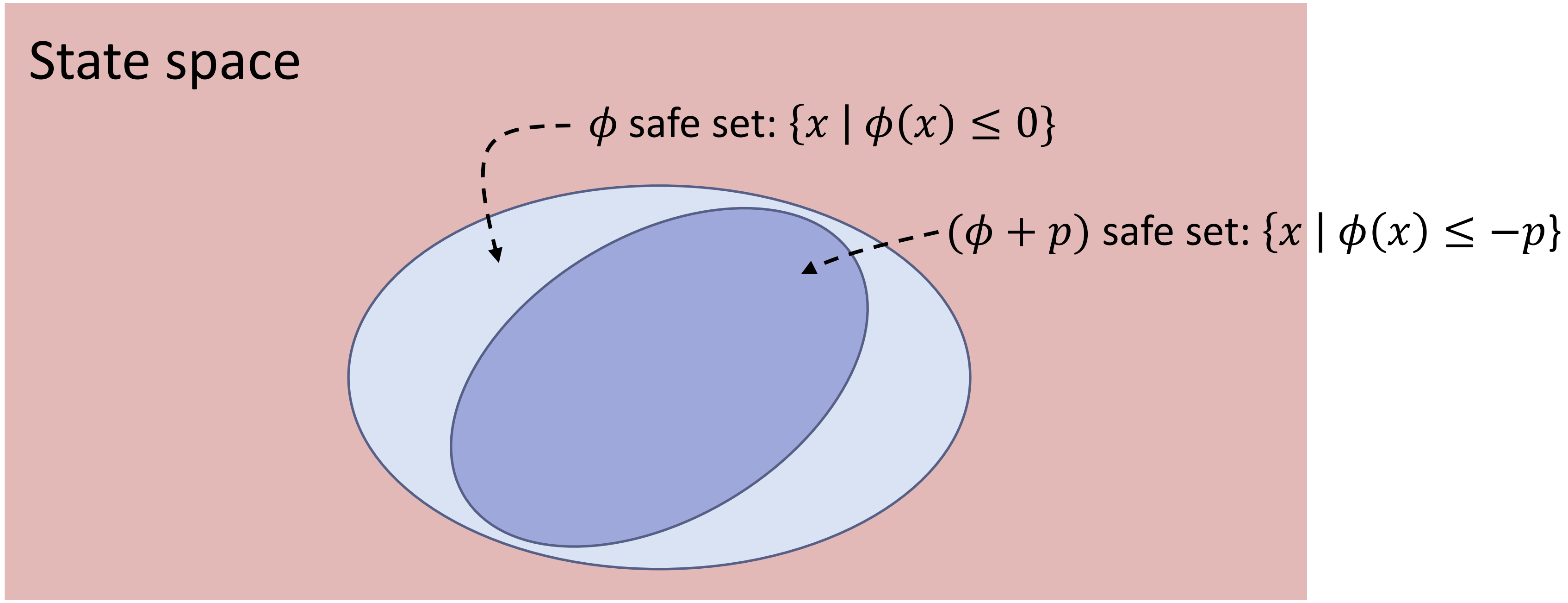
We design a neural CBF that is stricter than the limit-blind CBF



Q: how would we modify ϕ to shrink its safe set?

Add a positive function to ϕ.

We design a neural CBF that is stricter than the limit-blind CBF



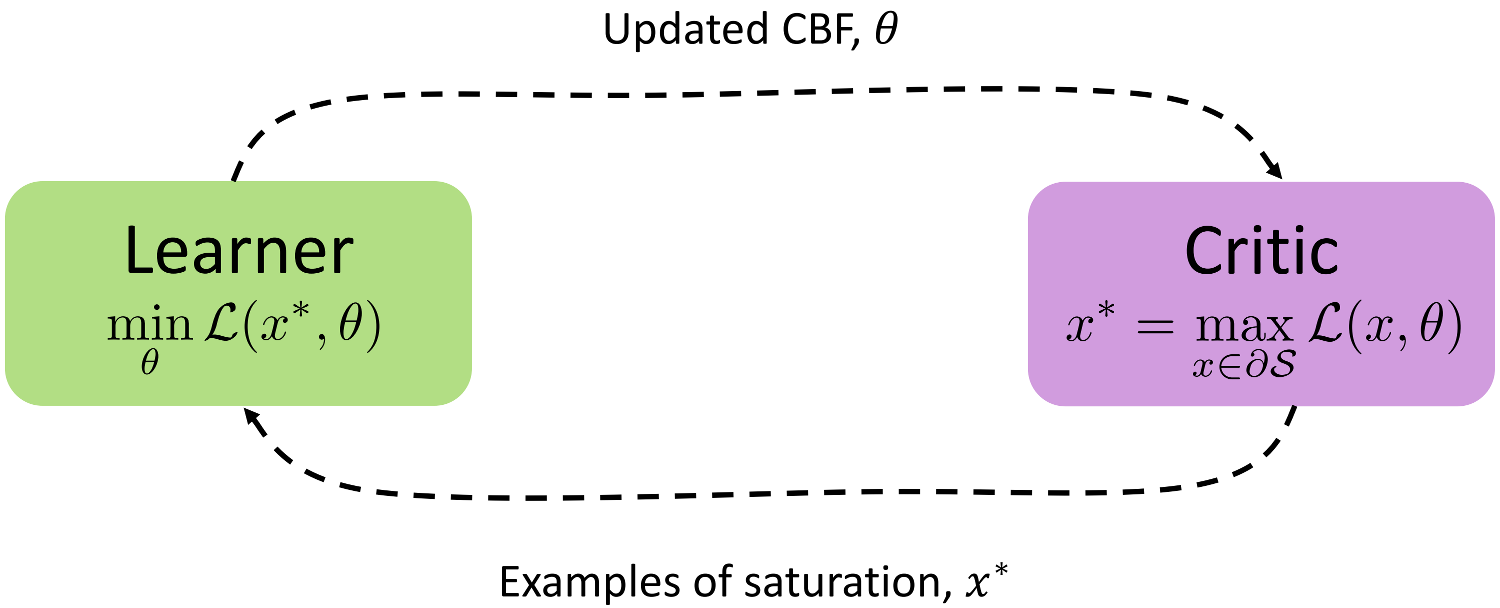
Let $\phi^* = \phi + p(nn(x))$ with $p(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^+$ and $nn(\cdot)$ a feedforward NN with $tanh$ activations.

p chosen depending on the type of safety specification, etc. For example, $p = softmax$.

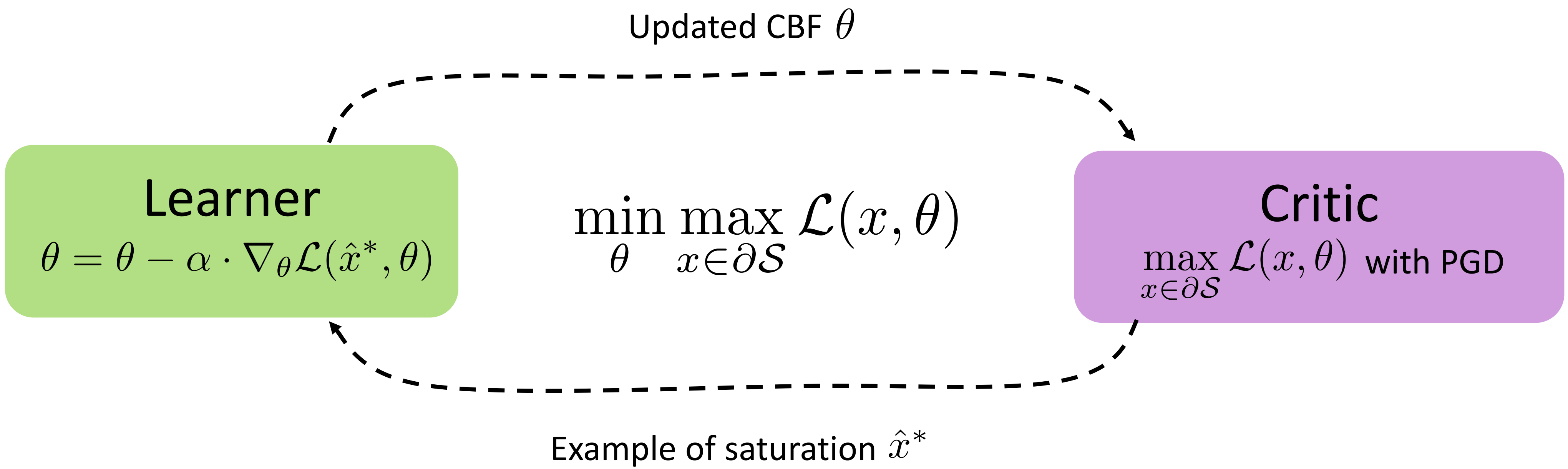
Designing an optimization algorithm

Optimize min-max using learner-critic framework

$$\min_{\theta} \max_{x \in \partial \mathcal{S}} \mathcal{L}(x, \theta)$$



Learner and critic both use gradient descent



Techniques not covered:

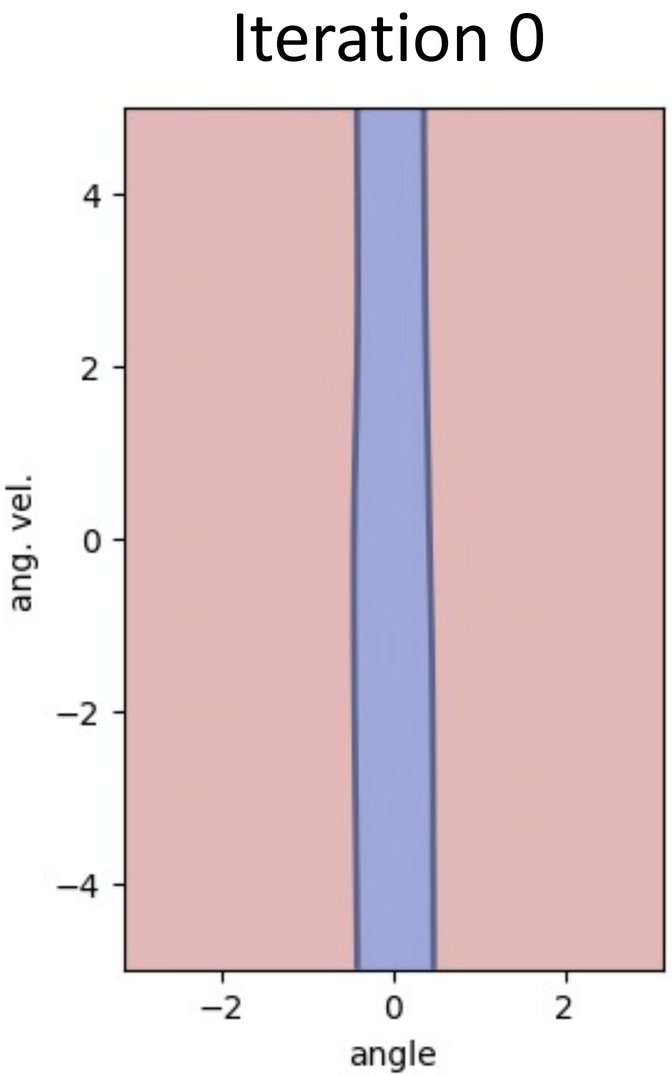
- Simple trick to get a differentiable objective
- Details of critic’s batch optimization and learner’s batch update
- “Warm-start” technique that boosts critic efficiency
- Regularization term that encourages a larger safe set

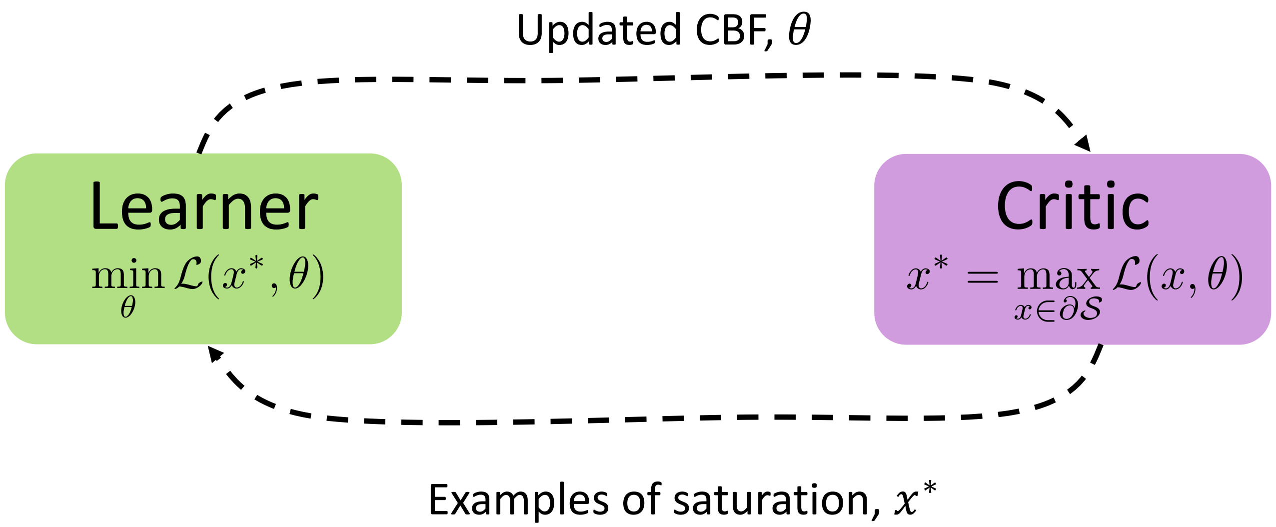
Improves efficiency

But feel free to ask afterwards!

Let’s see some examples.

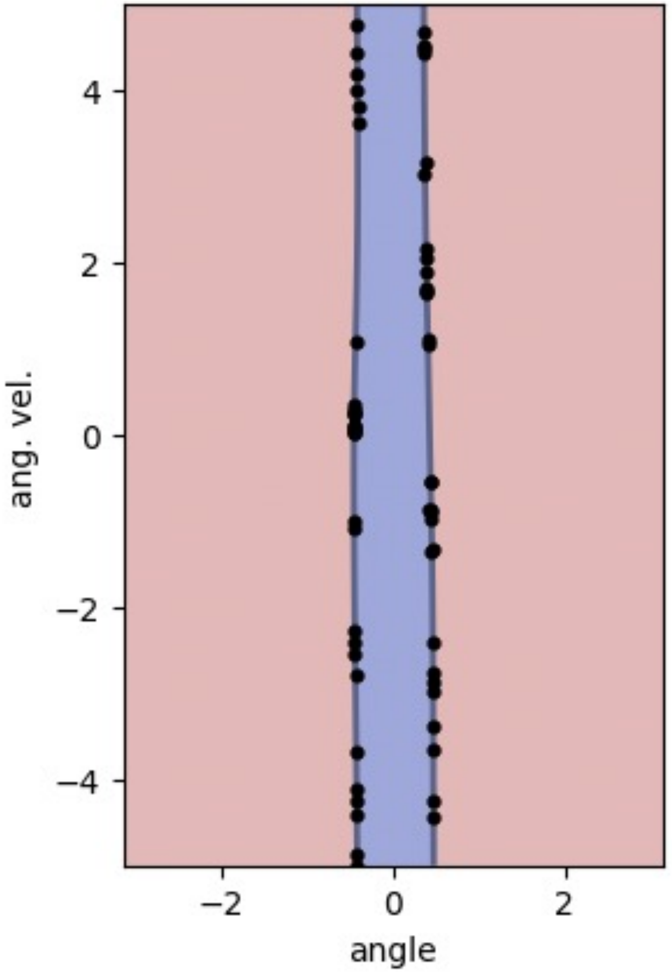
Learner-critic walkthrough for cartpole

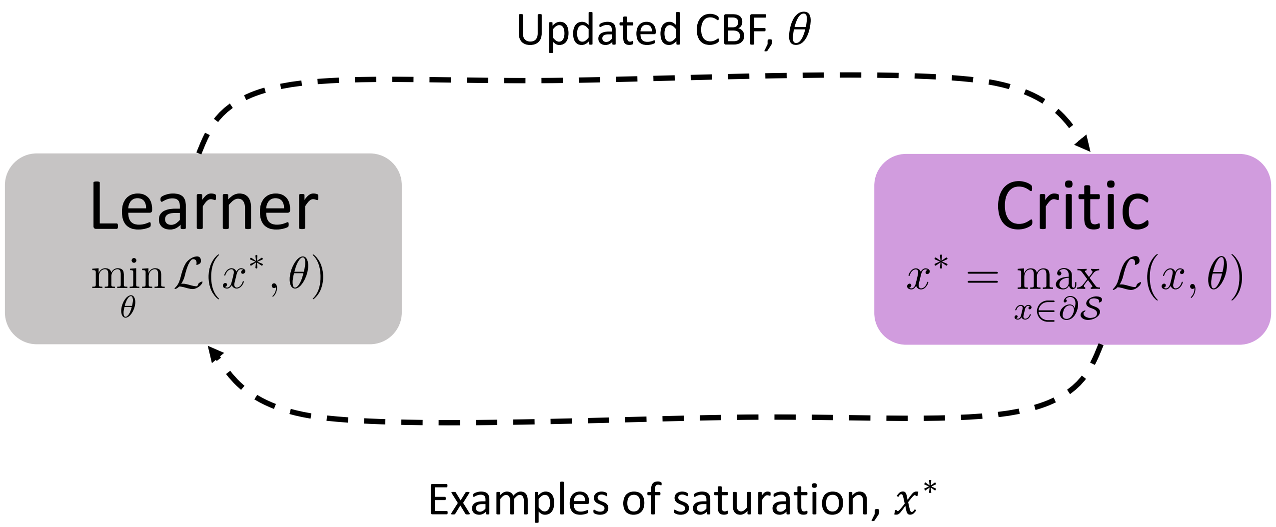




Learner-critic walkthrough for cartpole

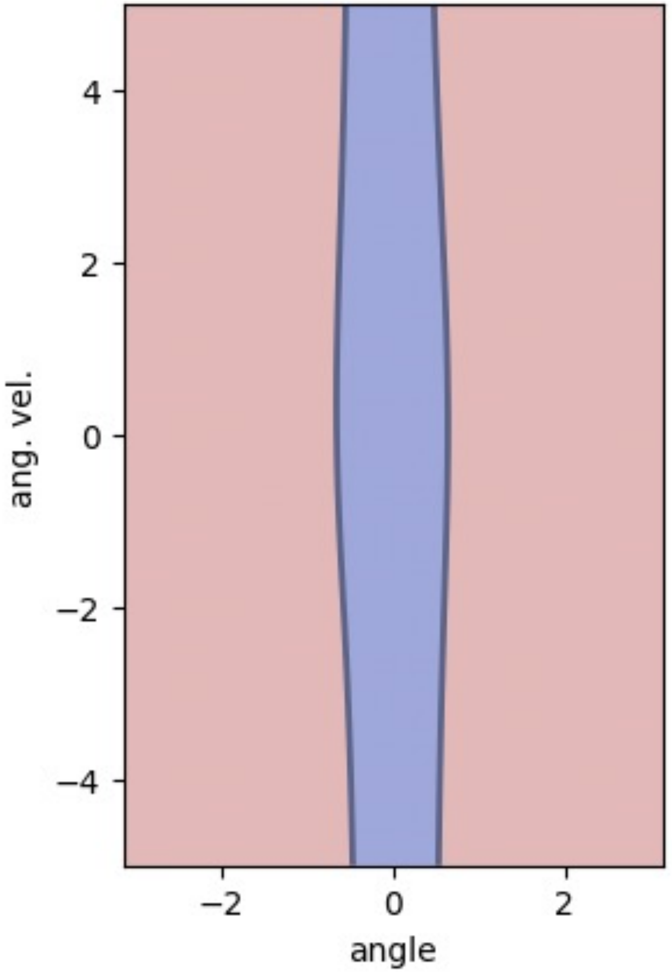
Iteration 0, critic's turn

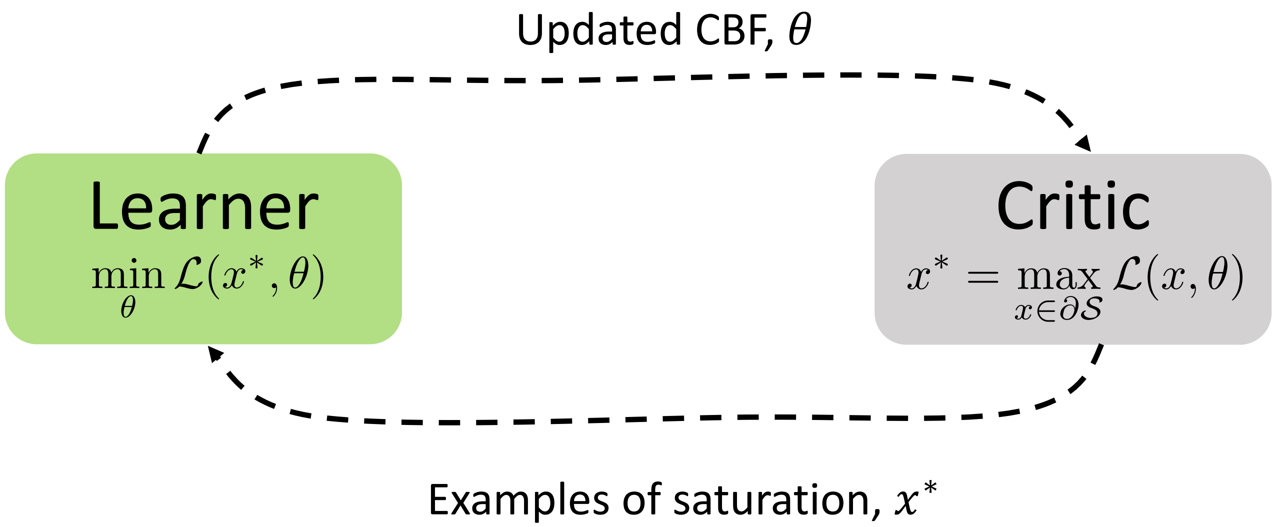




Learner-critic walkthrough for cartpole

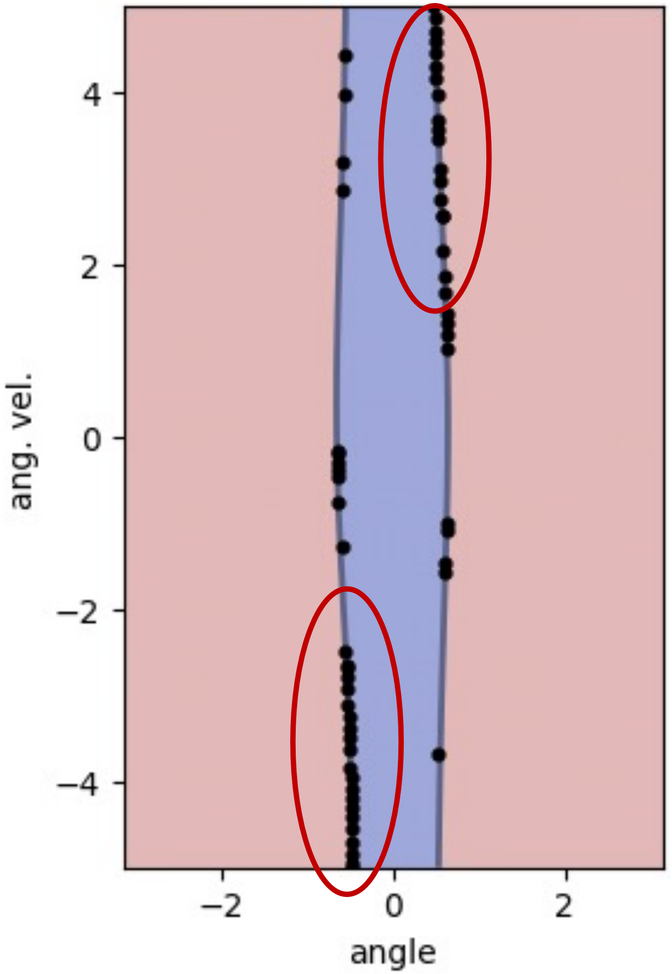
Iteration 0, learner's turn

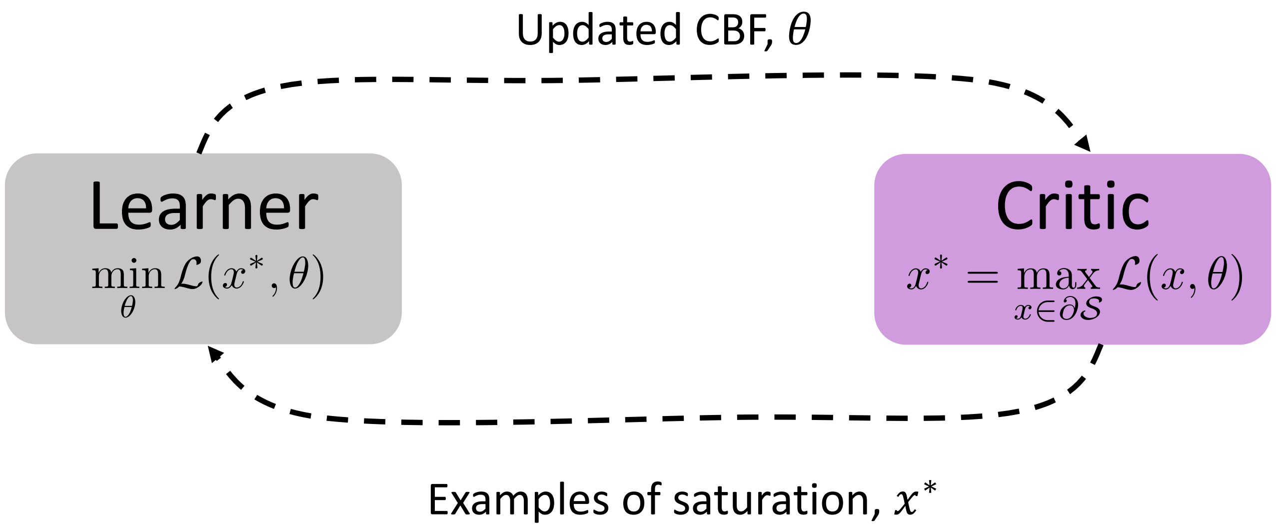




Learner-critic walkthrough for cartpole

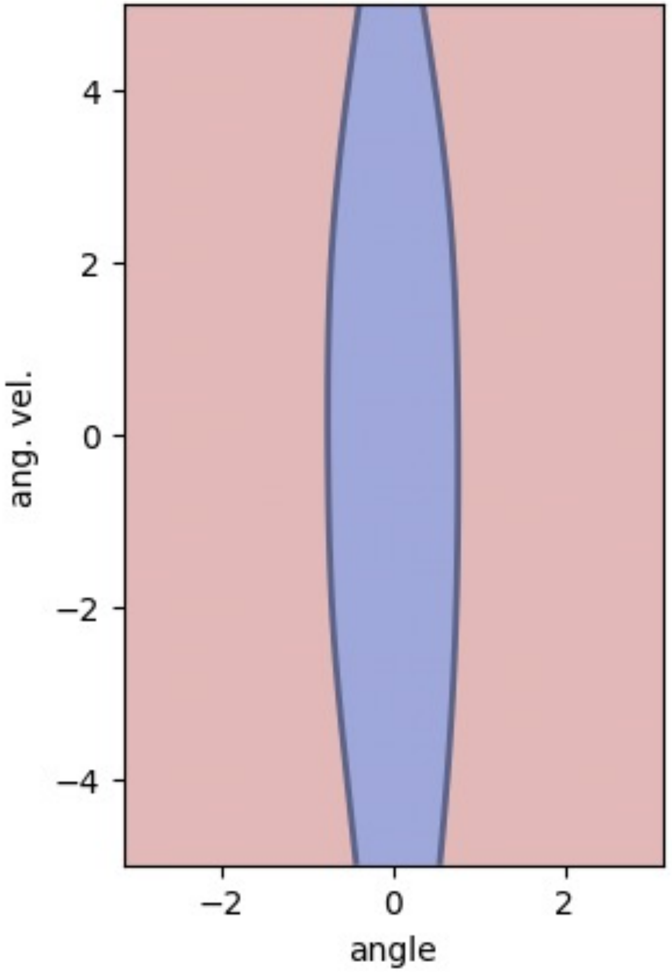
Iteration 1, critic's turn

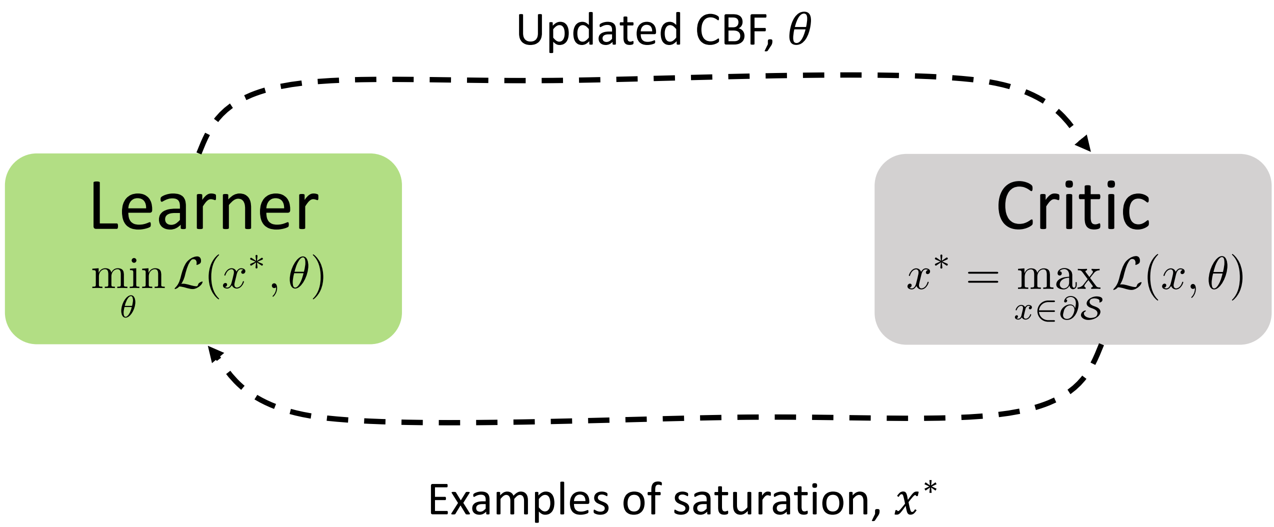




Learner-critic walkthrough for cartpole

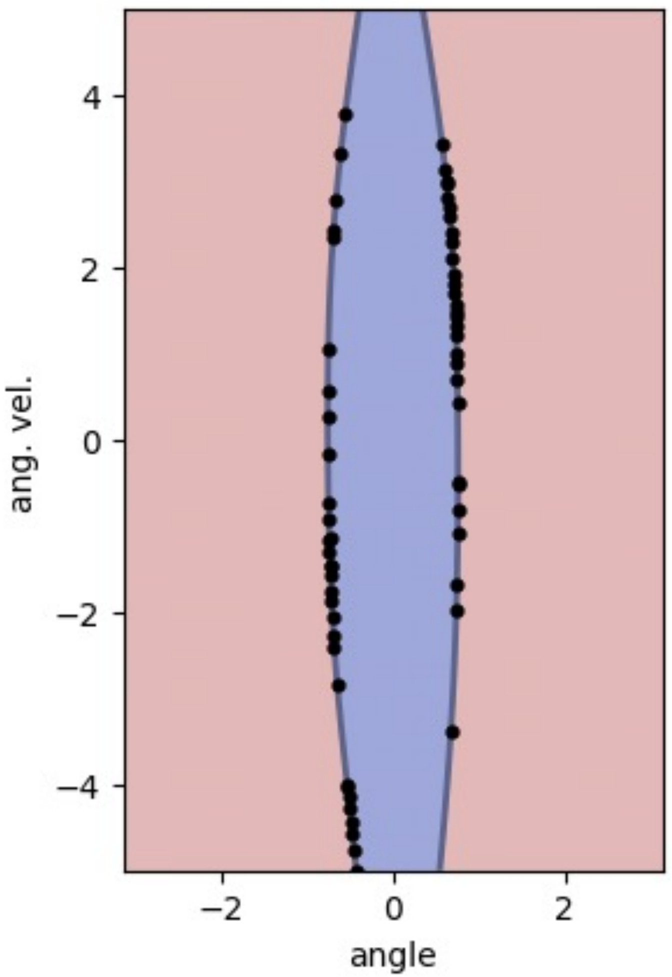
Iteration 1, learner's turn





Learner-critic walkthrough for cartpole

Iteration 2, critic's turn



Updated CBF, θ

Learner

$$\min_{\theta} \mathcal{L}(x^*, \theta)$$

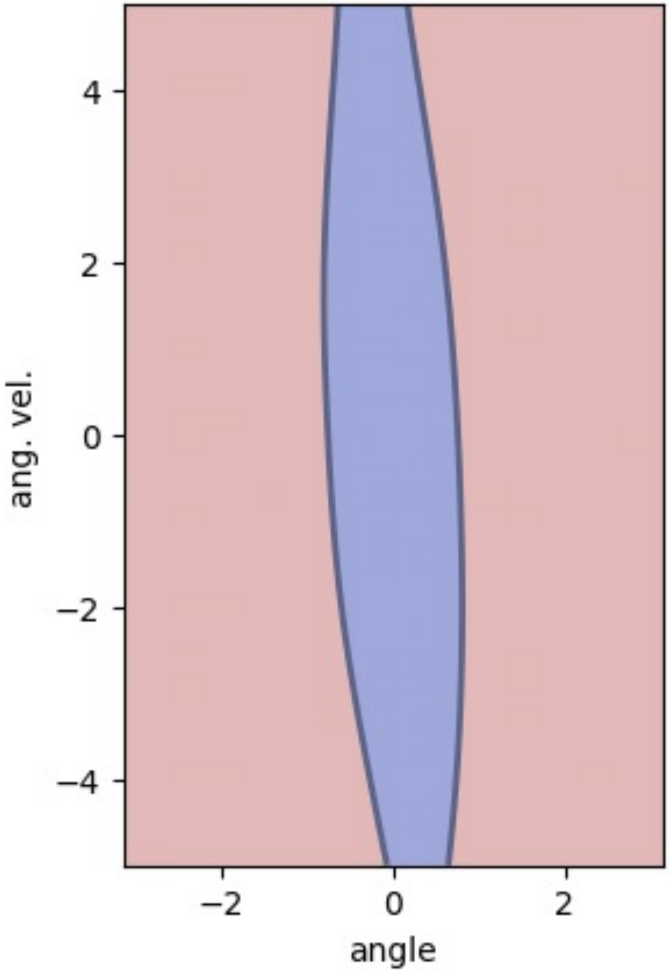
Critic

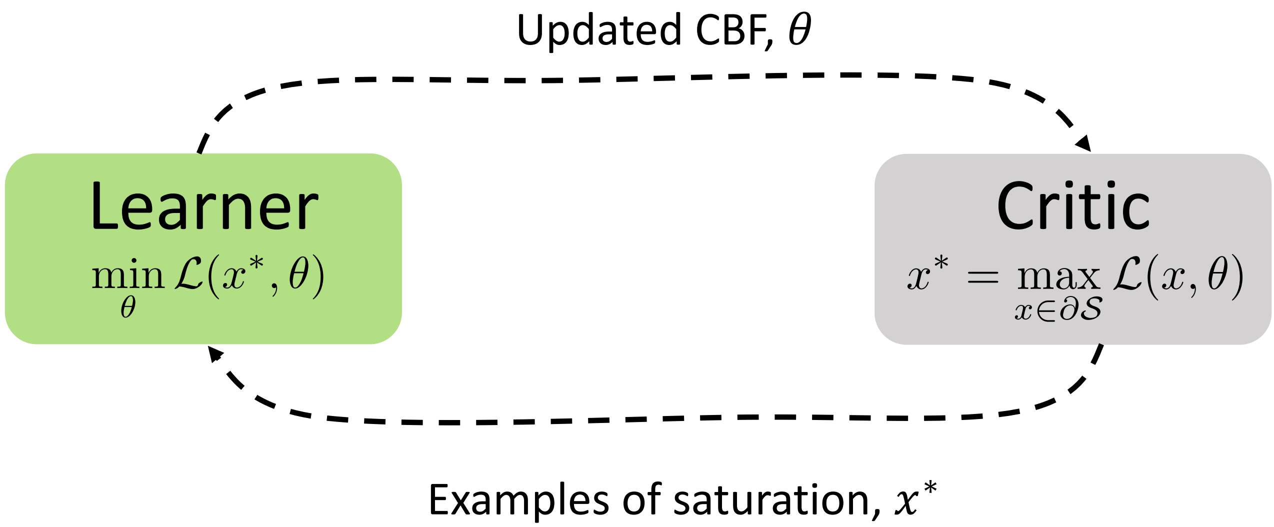
$$x^* = \max_{x \in \partial \mathcal{S}} \mathcal{L}(x, \theta)$$

Examples of saturation, x^*

Learner-critic walkthrough for cartpole

Iteration 2, learner's turn

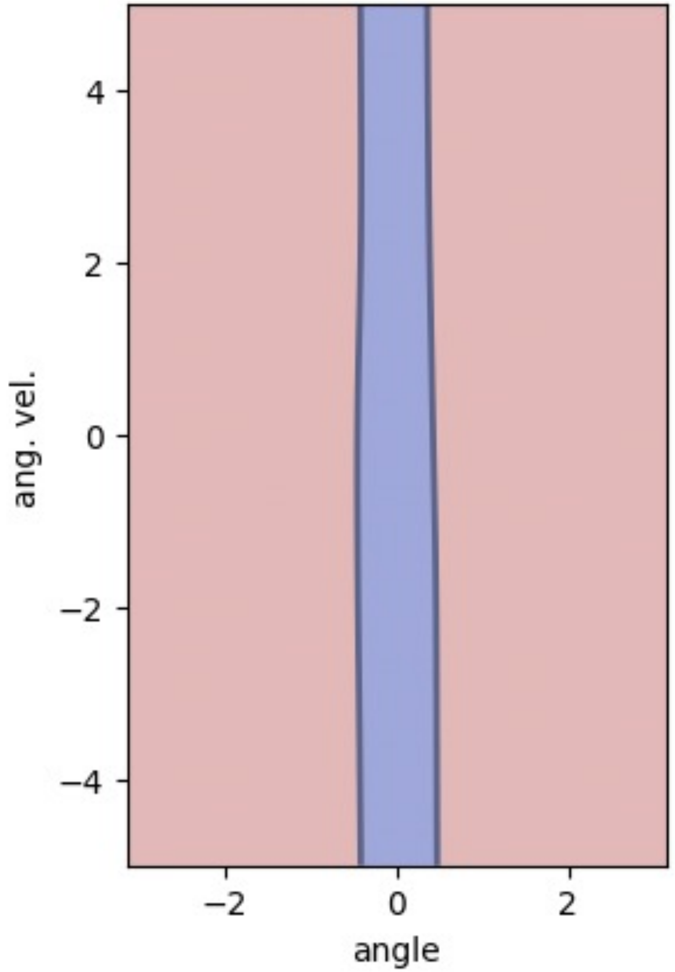




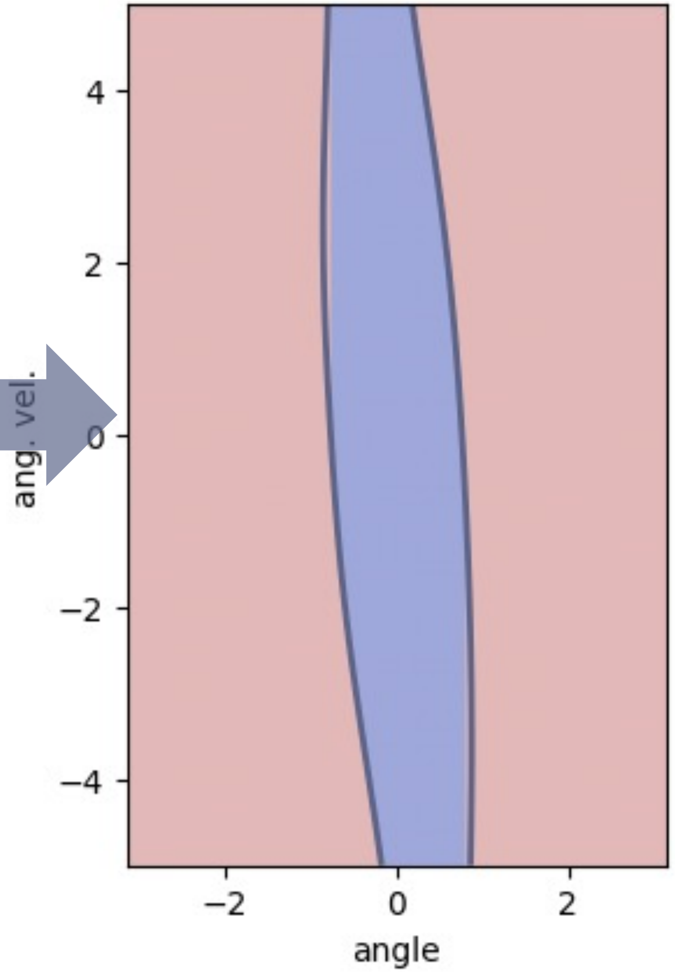
After a while…

Learner-critic walkthrough for cartpole

Iteration 0



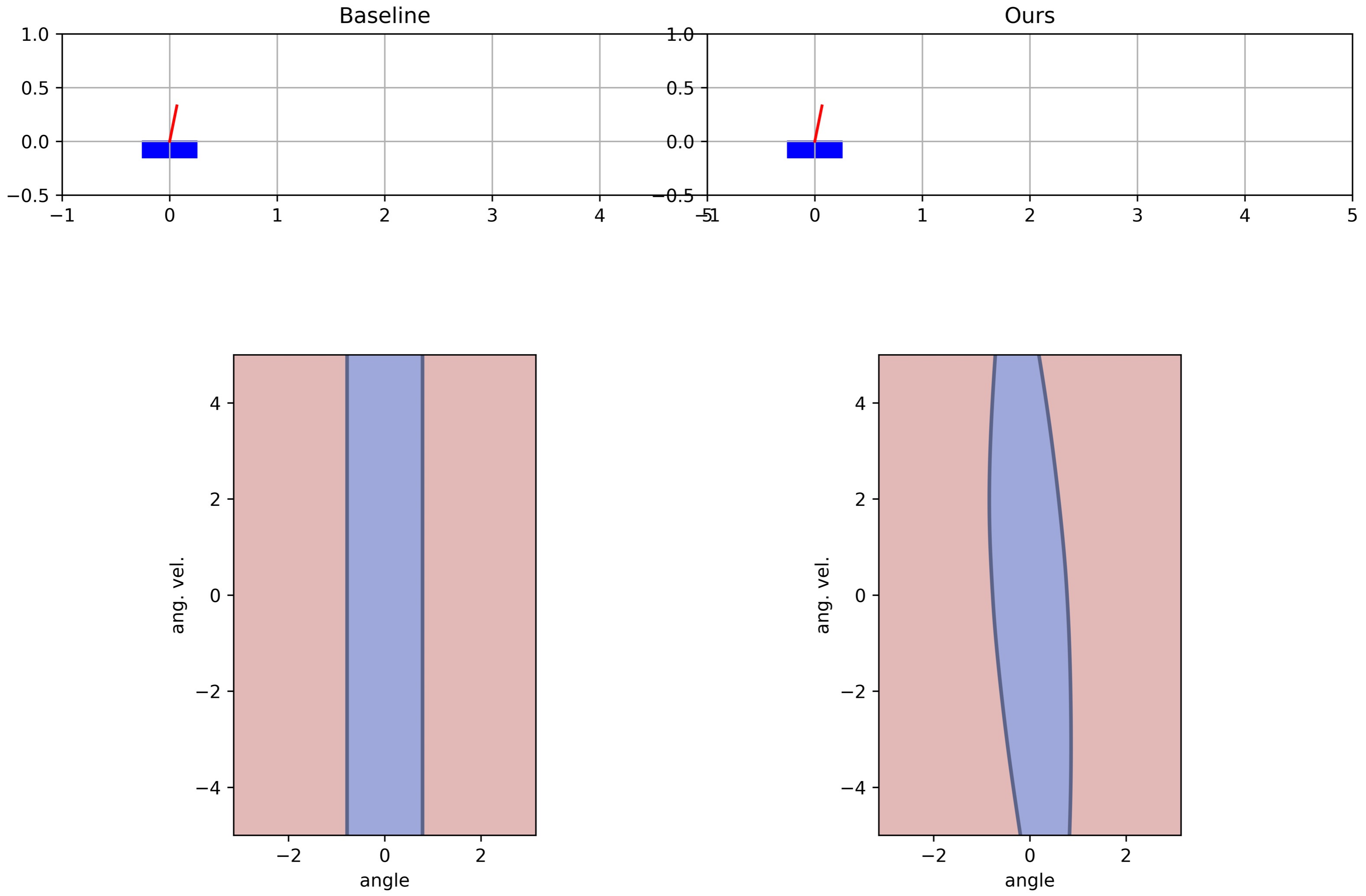
Final iteration



After:

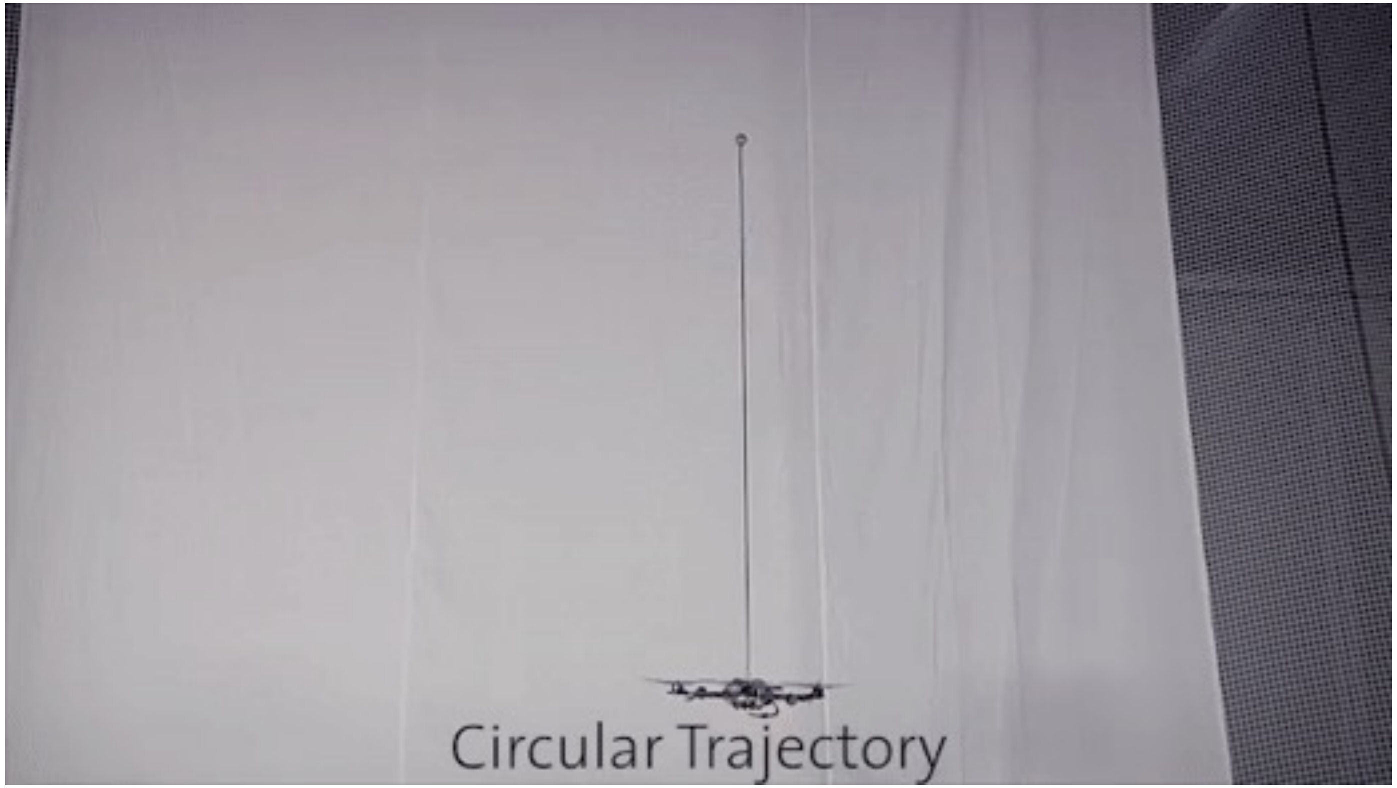
- ~200 iterations
- 15 minutes of training
- 10k cumulative counterexamples

Observing our learned safe controller in action



A harder example now.

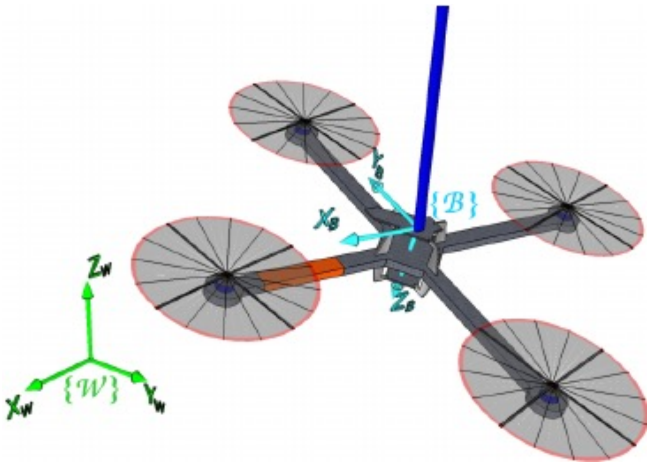
Balance pendulum on a quadcoptor



*See reference 11 for video credits.

Balance pendulum on a quadcoptor

Given: nonlinear, 10D state,
4D limited input system



Safety spec: keep quadcoptor roll, pitch, and pendulum angle to vertical below $\pi/4$

Q: which states do you expect the worst saturation at?

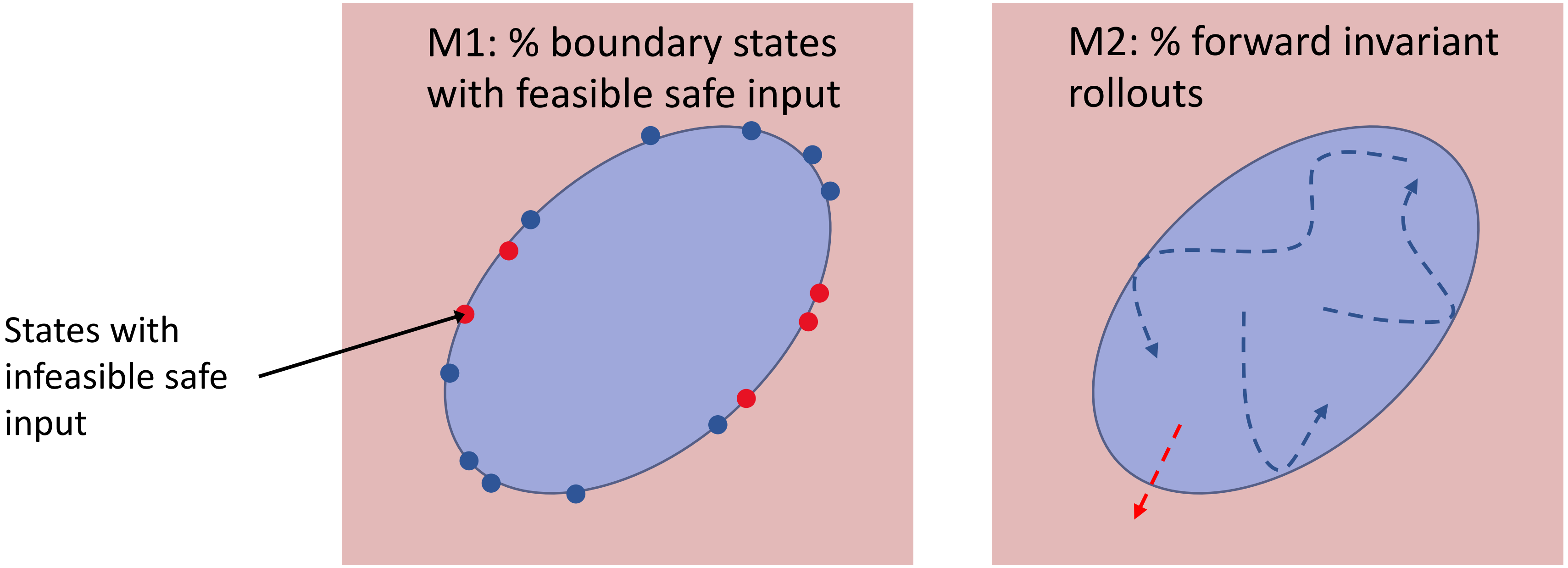
*See reference 12 for figure credit.

It’s much harder to reason about how to shrink this safe set!

Good news – we don’t have to. Just learn it.

Our method far outperforms a non-neural baseline

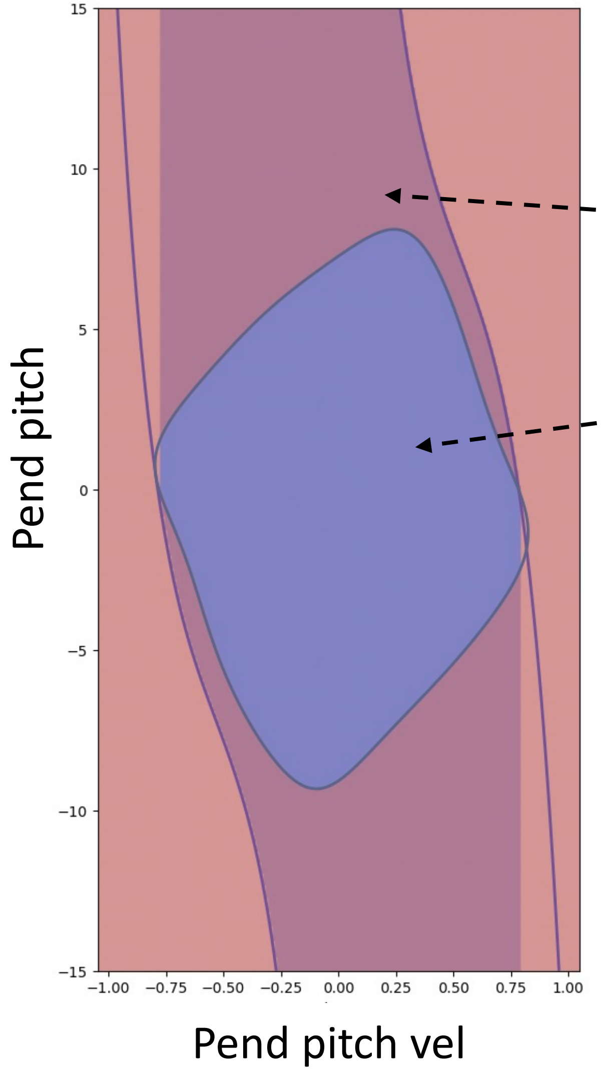
	M1	M2
Baseline (non-neural CBF)*	78.7	49.5-79.5
Ours	99.0	97.0-98.9



*See references 6, 10.

Why does our method outperform?

Safe set diagram



Baseline safe set

Our learned safe set

- Our CBF learned that pendulum pitch and pitch velocity must be bounded
- That requires terms of the form β^2 and $\dot{\beta}^2$ in the CBF (safe set then requires $\beta^2 < 0$ and $\dot{\beta}^2 < 0$)

$\phi_{baseline} = (\beta)^{2 \cdot a_1} - (\pi/4)^{2 \cdot a_1} + a_2 + a_3 \beta \dot{\beta}$
where a_1, a_2, a_3 are tuned parameters

No $\dot{\beta}^2$ term: wrong function form!

$\phi_{ours} = \beta^2 - (\pi/4)^2 + p(nn(\beta)) + k\beta\dot{\beta}$
where $nn(\cdot)$ and k have been learned using learner-critic

NN can learn a $\dot{\beta}^2$ term

Limitations + future work

- Learning required a state transformation first
  - Maybe unnecessary with sinusoidal NN?
- Assumed known, deterministic dynamics
  - Extend to learning *robust* non-saturating CBF?

Final recap

- CBF are hard to synthesize under input limits
- Neural CBF representation + efficient training algorithm = generic, scalable, automatic synthesis
- Addressing this problem makes CBFs more practically useful!

Acknowledgments

I'm grateful to my advisors, Prof. Dolan and Prof. Liu, as well as the other members of my committee, Prof. Held and Jaskaran Grover.

Also thanks to the members of the Dolan Lab and ICL, especially Qin Lin, Tianhao Wei, Ravi Pandya, and also Prof. Andrea Bajcsy, Kate Shih, Ashwin Khadke, Arpit Agarwal.

Questions?

References

1. C. Liu and M. Tomizuka. Control in a safe set: Addressing safety in human-robot interactions. In Dynamic Systems and Control Conference, volume 46209, page V003T42A003. American Society of Mechanical Engineers, 2014.
2. W. Zhao, T. He, and C. Liu. Model-free safe control for zero-violation reinforcement learning. In 5th Annual Conference on Robot Learning, 2021.
3. A. D. Ames, J. W. Grizzle, and P. Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In 53rd IEEE Conference on Decision and Control, pages 6271–6278. IEEE, 2014.
4. Y. Lyu, W. Luo, and J. M. Dolan. Probabilistic safety-assured adaptive merging control for autonomous vehicles. In 2021 IEEE International Conference on Robotics and Automation (ICRA), pages 10764–10770. IEEE, 2021.
5. W. S. Cortez and D. V. Dimarogonas. Safe-by-design control for euler-lagrange systems. arXiv preprint arXiv:2009.03767, 2020.
6. T. Wei and C. Liu. Safe control with neural network dynamic models. arXiv preprint arXiv:2110.01110, 2021.
7. A. Clark. Verification and synthesis of control barrier functions. arXiv preprint arXiv:2104.14001, 2021.
8. S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In 2017 IEEE 56th Annual Conference on Decision and Control (CDC), pages 2242–2253. IEEE, 2017.
9. M. Chen, S. Herbert, and C. J. Tomlin. Fast reachable set approximations via state decoupling disturbances. arXiv preprint arXiv:1603.05205, 2016.
10. W. Zhao, T. He, and C. Liu. Model-free safe control for zero-violation reinforcement learning. In 5th Annual Conference on Robot Learning, 2021.
11. M. Hehn and R. D’Andrea. A flying inverted pendulum. In 2011 IEEE International Confer- ence on Robotics and Automation, pages 763–770. IEEE, 2011
12. R. Figueroa, A. Faust, P. Cruz, L. Tapia, and R. Fierro. Reinforcement learning for balancing a flying inverted pendulum. In Proceeding of the 11th World Congress on Intelligent Control and Automation, pages 1787–1793. IEEE, 2014.