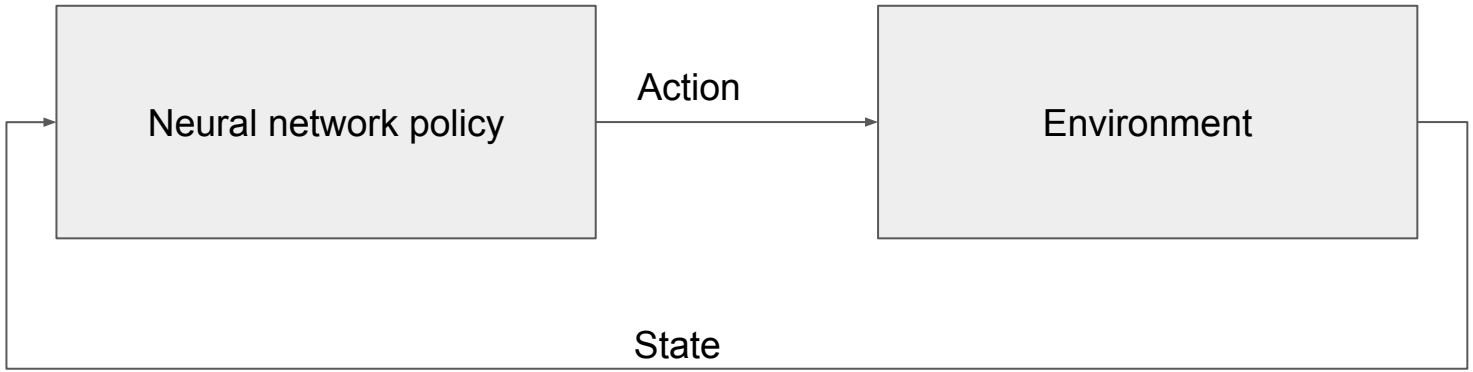
Closed-loop Safety of Bayesian Neural Networks and Stochastic Control Systems

Mathias Lechner

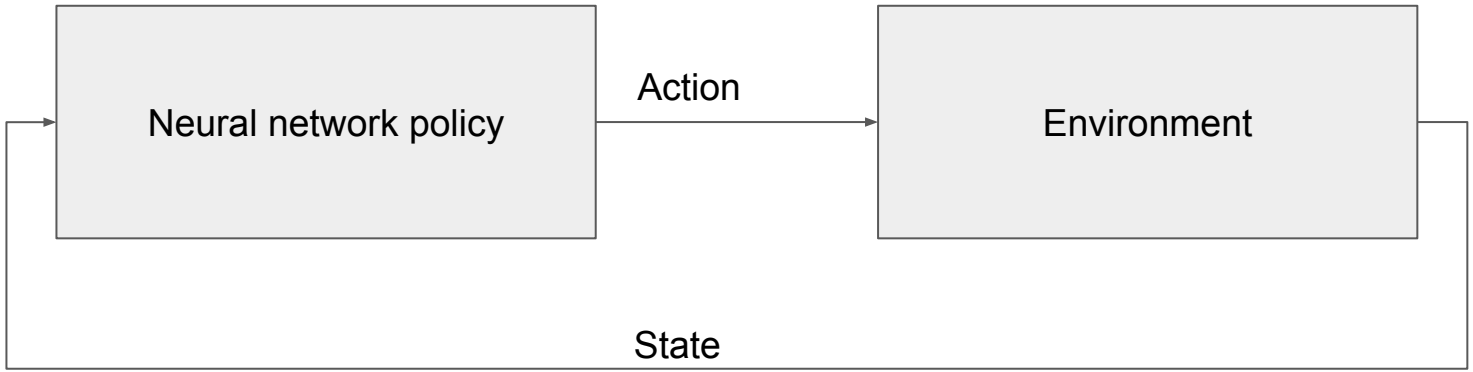
Formal safety verification of neural networks

Safety-critical applications that require formal safety guarantees

Closed-loop systems

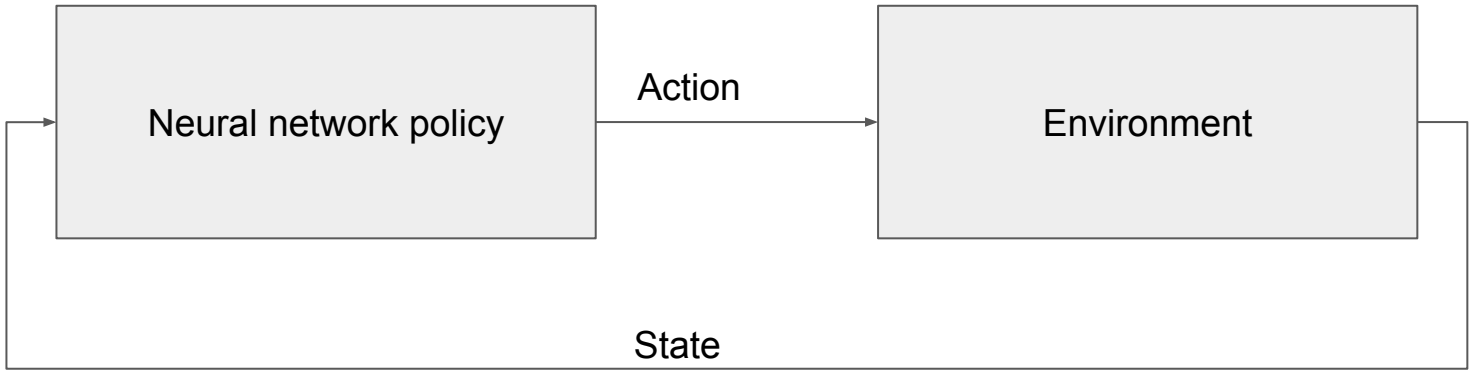


Closed-loop systems



- Safety
  - System never reaches unsafe states
- Stability
  - System always reaches target states

Closed-loop systems



- Safety
  - System never reaches unsafe states
- Stability
  - System always reaches target states

Many existing works assume policy and environment are deterministic

Closed-loop stability and safety

	Deterministic system	**Stochastic system**
Deterministic policy		
Stochastic policy		

Closed-loop stability and safety

	Deterministic system	**Stochastic system**
Deterministic policy	“Solved” via Lyapunov and Barrier functions	
Stochastic policy		

Closed-loop stability and safety

	Deterministic system	**Stochastic system**
Deterministic policy	“Solved” via Lyapunov and Barrier functions	This talk (Lechner et al. 2022 AAAI)
Stochastic policy	This talk (Lechner et al. 2021 NeurIPS)	

Closed-loop stability and safety

	Deterministic system	**Stochastic system**
Deterministic policy	“Solved” via Lyapunov and Barrier functions	This talk (Lechner et al. 2022 AAAI)
Stochastic policy	This talk (Lechner et al. 2021 NeurIPS)	Future work

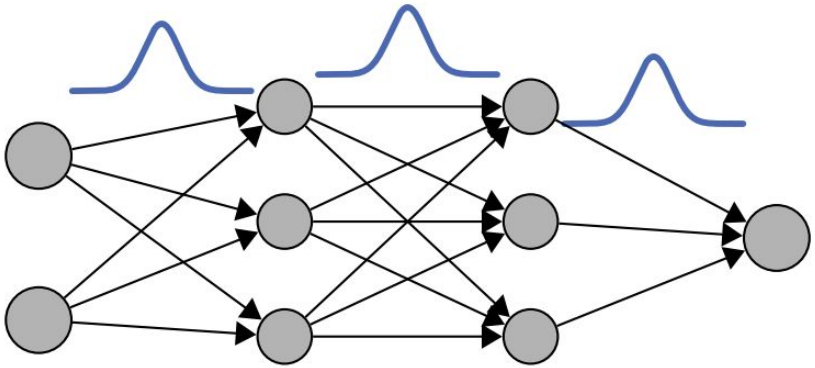
Infinite Time Horizon Safety of Bayesian Neural Networks

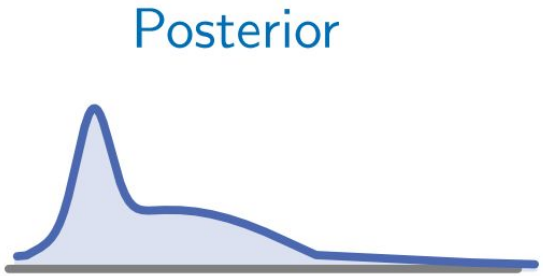
Mathias Lechner*, **Đorđe Žikelić***, Krishnendu Chatterjee, Thomas A. Henzinger
IST Austria

* equal contribution

Bayesian neural networks (BNNs)

- Weights are random variables
- Learns uncertainties as arbitrary posterior distributions
- In this work: ReLU activations and Gaussian priors





BNN verification

Existing verification methods: Sampling-based

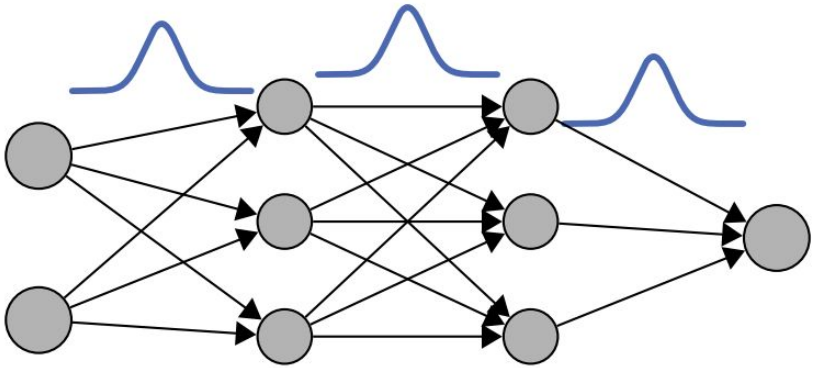
- Statistical guarantees [1] or lower bounds on safety probability [2]
- In closed-loop systems with BNN policies, statistical guarantees on safety over finite and bounded time horizon [3]

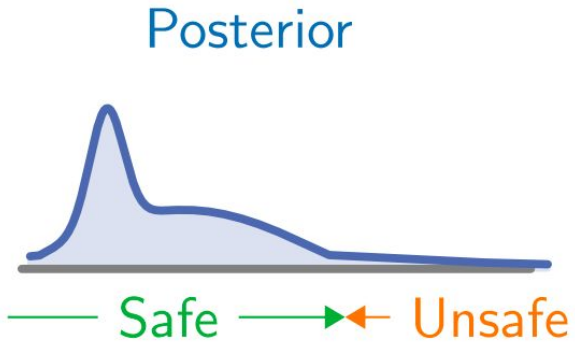
[1] Cardelli et al. *Statistical Guarantees for the Robustness of Bayesian Neural Networks*. IJCAI 2019
[2] Wicker et al. *Probabilistic Safety for Bayesian Neural Networks*. UAI 2020
[3] Michelmore et al. *Uncertainty quantification with statistical guarantees in end-to-end autonomous driving control*. ICRA 2020

Need for sure safety guarantees

- Prior weight distributions have unbounded support
- Posteriors also likely to have unbounded support

  -> BNNs are typically unsafe by default





Verification problem: safe weight set computation

Verification problem: identify a safe weight set for a BNN π, in the form of a product of intervals around means

$$W_\epsilon^\pi = \prod_{i=1}^{p+q} [\mu_i - \epsilon, \mu_i + \epsilon] \subseteq \mathbb{R}^{p+q}.$$

Verification problem: safe weight set computation

Verification problem: identify a safe weight set for a BNN π, in the form of a product of intervals around means

$$W_\epsilon^\pi = \prod_{i=1}^{p+q} [\mu_i - \epsilon, \mu_i + \epsilon] \subseteq \mathbb{R}^{p+q}.$$

Problem 1: feed-forward BNNs

Problem 2: closed-loop system with a BNN policy

Verification problem: safe weight set computation

Verification problem: identify a safe weight set for a BNN π, in the form of a product of intervals around means

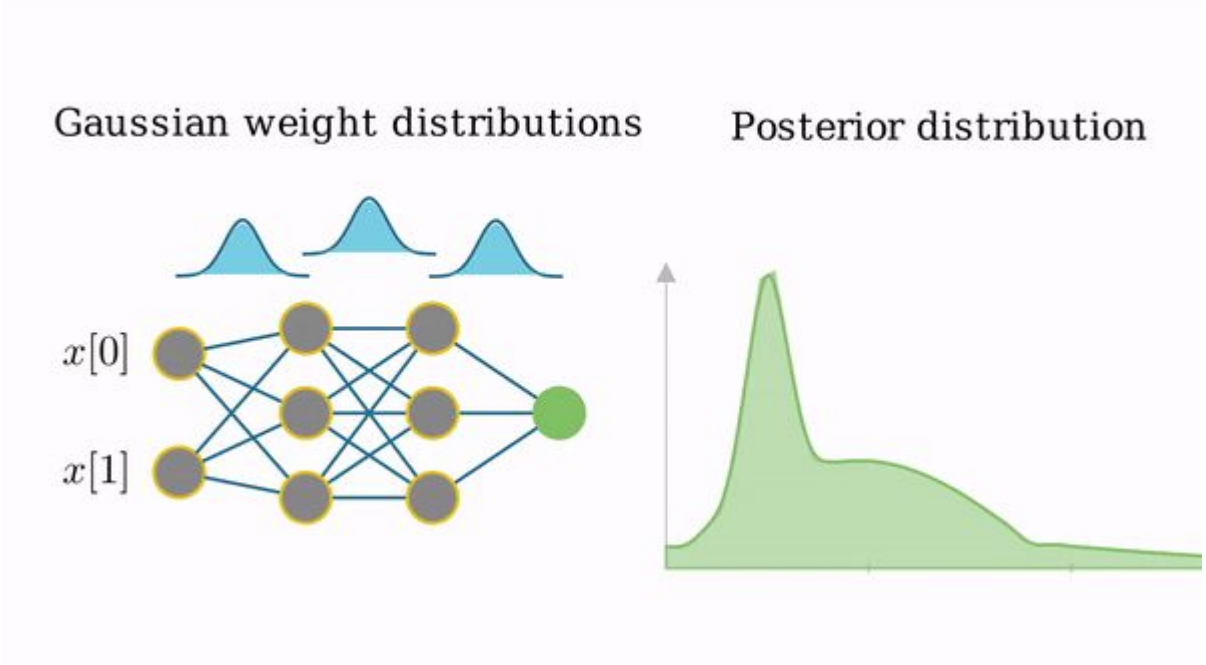
$$W_\epsilon^\pi = \prod_{i=1}^{p+q} [\mu_i - \epsilon, \mu_i + \epsilon] \subseteq \mathbb{R}^{p+q}.$$

Problem 1: feed-forward BNNs

Problem 2: closed-loop system with a BNN policy

⟹ Use rejection sampling to re-calibrate BNNs and ensure safety

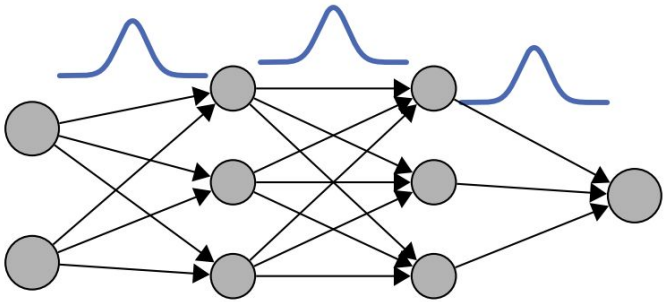
Verification problem: safe weight set computation



⟹ Use rejection sampling to re-calibrate BNNs and ensure safety

Safe weight sets for feed-forward BNNs

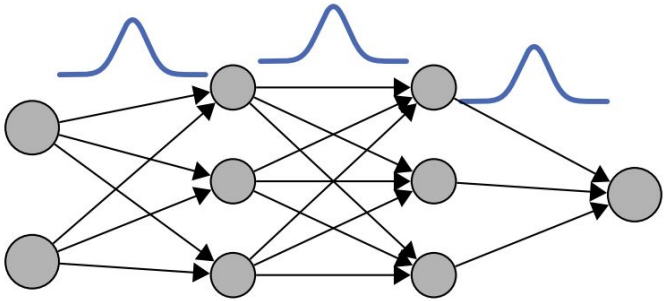
Input set



Unsafe set

Safe weight sets for feed-forward BNNs

Input set



Unsafe set

$$\mathbf{x}_0 \in \mathcal{X}_0, \quad \mathbf{x}_l \in \mathcal{X}_u \qquad \text{(Input-output conditions)}$$

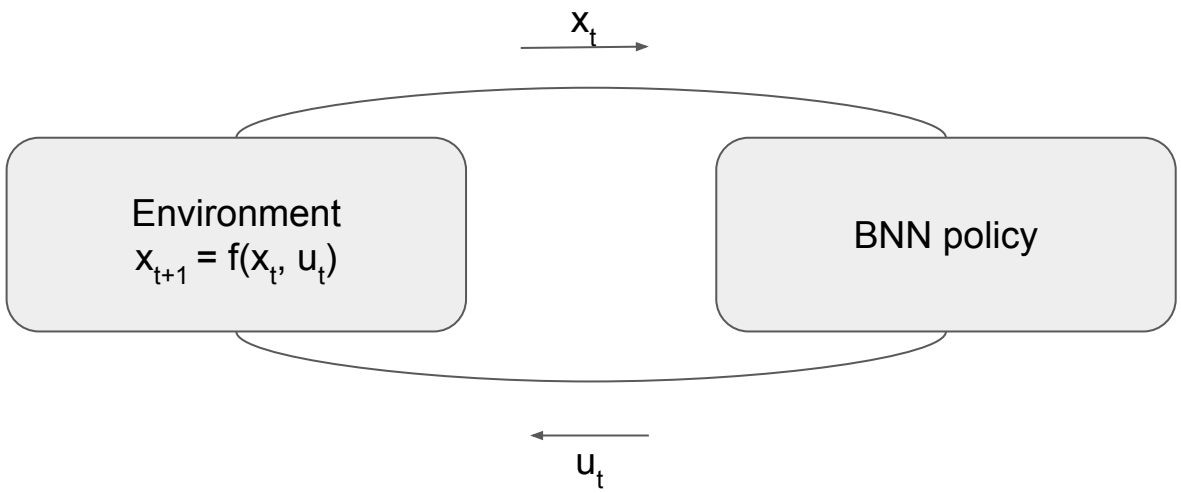
$$\mathbf{x}_i^{\text{out}} = \text{ReLU}(\mathbf{x}_i^{\text{in}}), \text{ for each } 1 \leq i \leq l-1 \qquad \text{(ReLU encoding)}$$

$$(\mathbf{M}_i - \epsilon \cdot \mathbf{1})\mathbf{x}_i^{\text{out}} + (\mathbf{m}_i - \epsilon \cdot \mathbf{1}) \leq \mathbf{x}_{i+1}^{\text{in}}, \text{ for each } 1 \leq i \leq l-1$$
$$\mathbf{x}_{i+1}^{\text{in}} \leq (\mathbf{M}_i + \epsilon \cdot \mathbf{1})\mathbf{x}_i^{\text{out}} + (\mathbf{m}_i + \epsilon \cdot \mathbf{1}), \text{ for each } 1 \leq i \leq l-1 \qquad \text{(BNN hidden layers)}$$

$$\mathbf{x}_{0,\text{pos}} = \text{ReLU}(\mathbf{x}_0), \quad \mathbf{x}_{0,\text{neg}} = -\text{ReLU}(-\mathbf{x}_0)$$
$$(\mathbf{M}_0 - \epsilon \cdot \mathbf{1})\mathbf{x}_{0,\text{pos}} + (\mathbf{M}_0 + \epsilon \cdot \mathbf{1})\mathbf{x}_{0,\text{neg}} + (\mathbf{m}_0 - \epsilon \cdot \mathbf{1}) \leq \mathbf{x}_1^{\text{in}} \qquad \text{(BNN input layer)}$$
$$\mathbf{x}_1^{\text{in}} \leq (\mathbf{M}_0 + \epsilon \cdot \mathbf{1})\mathbf{x}_0^{\text{out}} + (\mathbf{M}_0 - \epsilon \cdot \mathbf{1})\mathbf{x}_{0,\text{neg}} + (\mathbf{m}_0 + \epsilon \cdot \mathbf{1})$$

Reduction to MILP/SMT solving

Closed-loop system with BNN policies

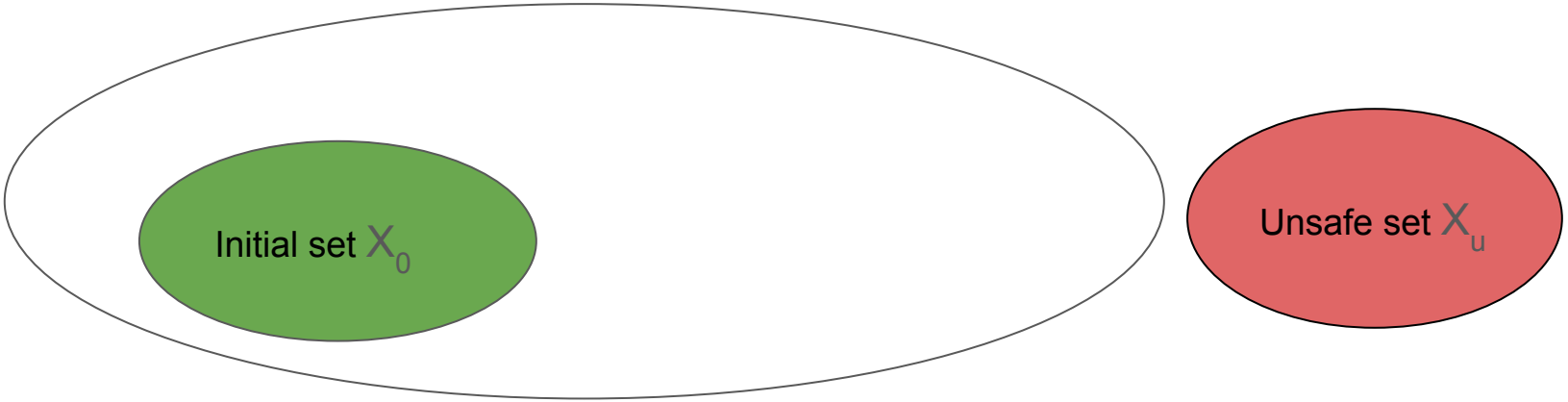


Input set: X_0

Unsafe set: X_u

Infinite time horizon safety verification

Positive invariants as safety certificates

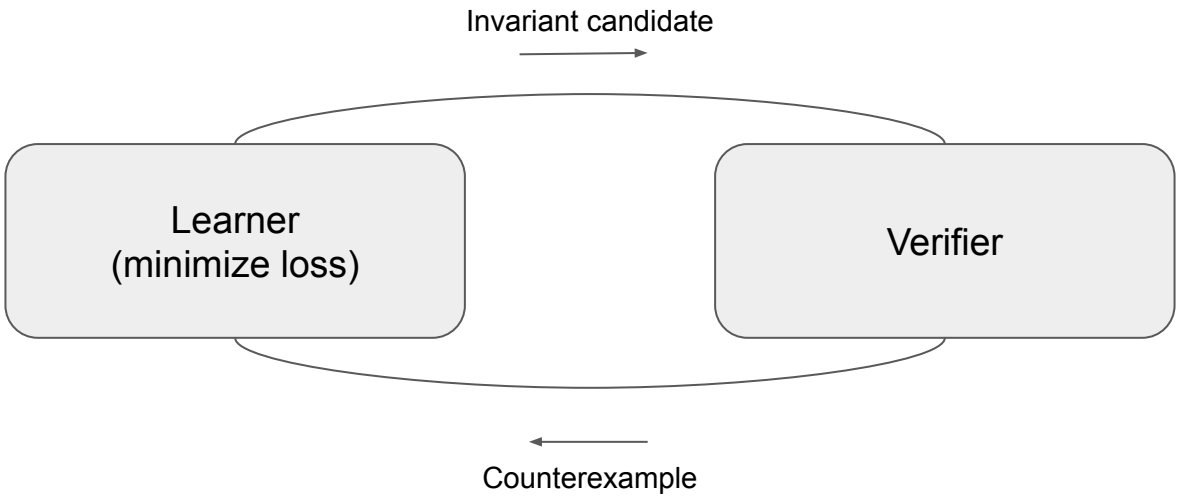


- $X_0 \subseteq Inv$
- $X_u \cap Inv = \varnothing$
- For each $x \in Inv$ and $(w,b) \in W_\epsilon^\pi$, we have $f(x,\pi_{w,b}(x)) \in Inv$

Learner-verifier framework

- Consider invariant membership as a binary classification problem
- Represent invariant as a (deterministic) neural network

$$\mathsf{Inv} = \{\mathbf{x} \in \mathcal{X} \mid g^{\mathsf{Inv}}(\mathbf{x}) \geq \vec{0}\}$$



Learner

$$\mathcal{L}(g^{\mathsf{Inv}}) = \frac{1}{|D_{\text{spec}}|} \sum_{(\mathbf{x},y)\in D_{\text{spec}}} \mathcal{L}_{\text{cls}}\left(g^{\mathsf{Inv}}(\mathbf{x}), y\right) + \lambda \frac{1}{|D_{\text{ce}}|} \sum_{(\mathbf{x},\mathbf{x}')\in D_{\text{ce}}} \mathcal{L}_{\text{ce}}\left(g^{\mathsf{Inv}}(\mathbf{x}), g^{\mathsf{Inv}}(\mathbf{x}')\right),$$

Learner

$$\mathcal{L}(g^{\text{Inv}}) = \frac{1}{|D_{\text{spec}}|} \sum_{(\mathbf{x},y)\in D_{\text{spec}}} \mathcal{L}_{\text{cls}}\left(g^{\text{Inv}}(\mathbf{x}), y\right) + \lambda \frac{1}{|D_{\text{ce}}|} \sum_{(\mathbf{x},\mathbf{x}')\in D_{\text{ce}}} \mathcal{L}_{\text{ce}}\left(g^{\text{Inv}}(\mathbf{x}), g^{\text{Inv}}(\mathbf{x}')\right),$$

Positive in initial states
Negative in unsafe states

Learner

$$\mathcal{L}(g^{\text{Inv}}) = \frac{1}{|D_{\text{spec}}|} \sum_{(\mathbf{x},y)\in D_{\text{spec}}} \mathcal{L}_{\text{cls}}\left(g^{\text{Inv}}(\mathbf{x}), y\right) + \lambda \frac{1}{|D_{\text{ce}}|} \sum_{(\mathbf{x},\mathbf{x}')\in D_{\text{ce}}} \mathcal{L}_{\text{ce}}\left(g^{\text{Inv}}(\mathbf{x}), g^{\text{Inv}}(\mathbf{x}')\right),$$

Positive in initial states
Negative in unsafe states

Closedness under system dynamics w.r.t. the weight set

Verifier

Check that the candidate satisfies 3 conditions (via reduction to MILP/SMT)

- $X_0 \subseteq \text{Inv}$
- $X_u \cap \text{Inv} = \varnothing$
- For each $x \in \text{Inv}$ and $(w,b) \in W_\epsilon^\pi$, we have $f(x, \pi_{w,b}(x)) \in \text{Inv}$

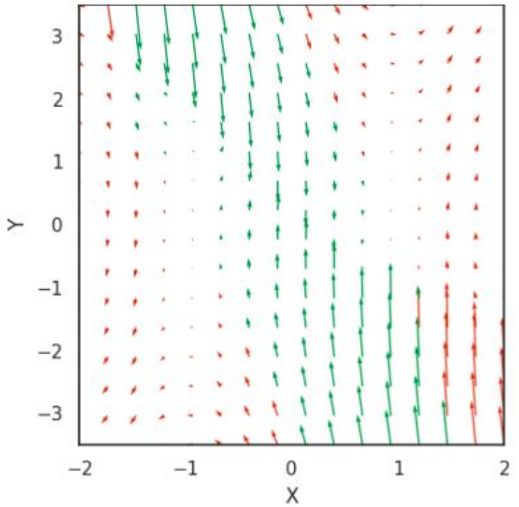
Safe exploration RL

- Exploration in RL requires randomized actions

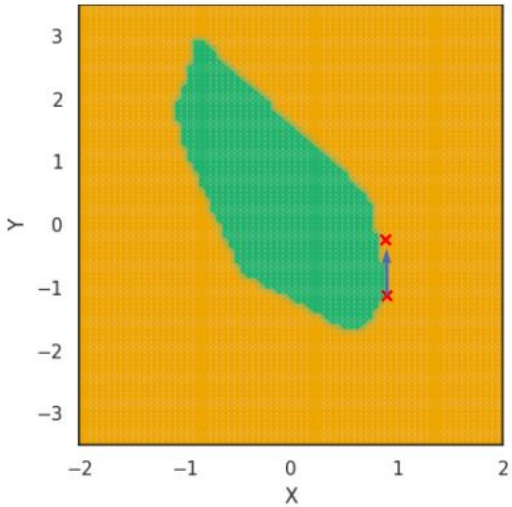
Experiments

Environment	No re-training Verified	No re-training Runtime	Init D_{spec} with $\mathcal{X}_0$ and $\mathcal{X}_u$ Verified	Init D_{spec} with $\mathcal{X}_0$ and $\mathcal{X}_u$ Runtime	Bootstrapping D_{spec} Verified	Bootstrapping D_{spec} Runtime
Unstable LDS	-	3	1.5σ	569	2σ	760
Unstable LDS (all)	0.2σ	3	0.5σ	6	0.5σ	96
Pendulum	-	2	2σ	220	2σ	40
Pendulum (all)	-	2	0.2σ	1729	1.5σ	877
Collision avoid.	-	2	-	-	2σ	154
Collision avoid. (all)	-	2	-	-	1.5σ	225

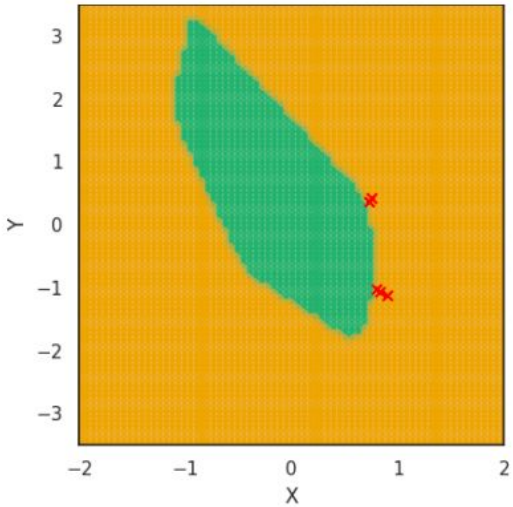
Closed-loop system vector field



Learned invariant (1 iteration)



Learned invariant (5 iterations)



Conclusion

- Novel view of the verification problem for BNNs – need for sure safety and the computation of safe weight sets
- For feedforward BNNs – reduction to constraint solving
- For closed-loop systems with BNN policies – a learner-verifier framework to learn positive invariants
- Experimental results that demonstrate the effectiveness

Code available: https://github.com/mlech26l/bayesian_nn_safety

Stability Verification in Stochastic Control Systems via Neural Network Supermartingales

Mathias Lechner*, **Đorđe Žikelić***, Krishnendu Chatterjee, Thomas A. Henzinger

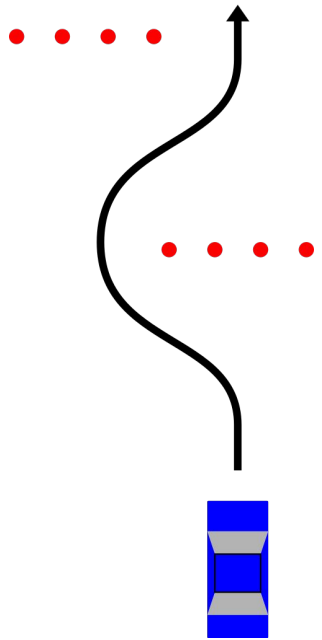
IST Austria

* equal contribution

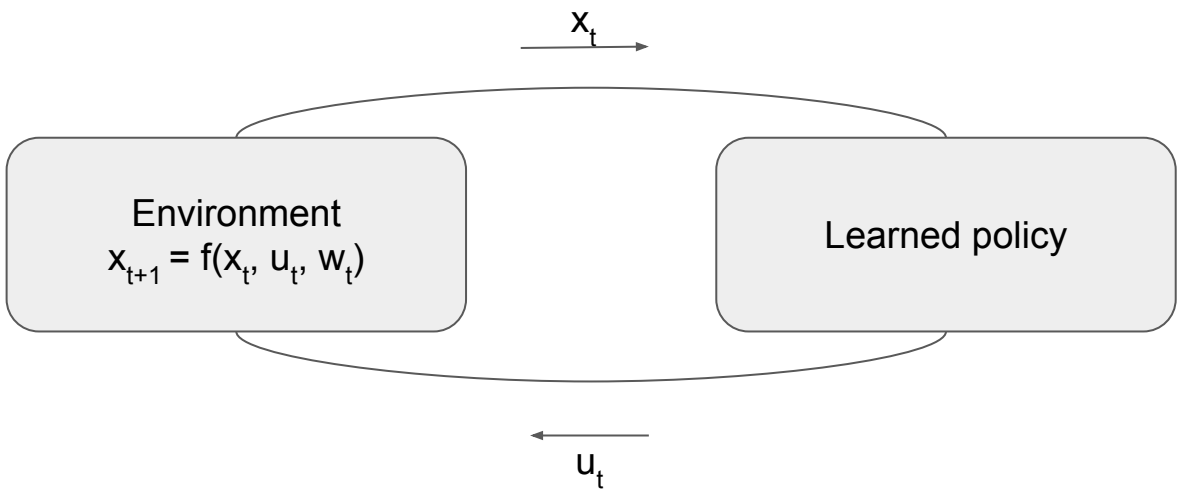
IST AUSTRIA
Institute of Science and Technology

Stability

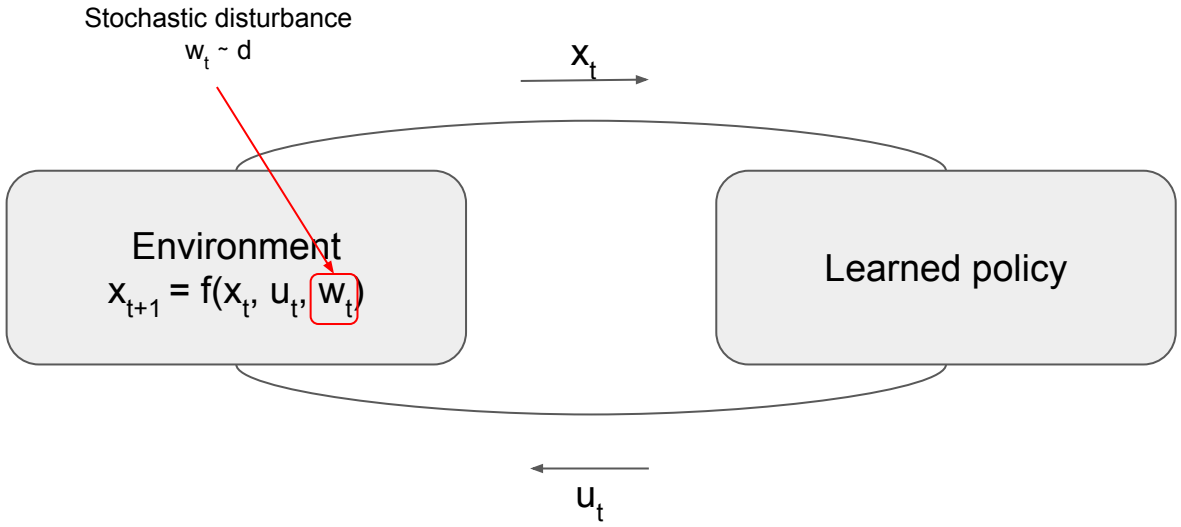
System is stable ⇔ System can recover to safe region from any system state



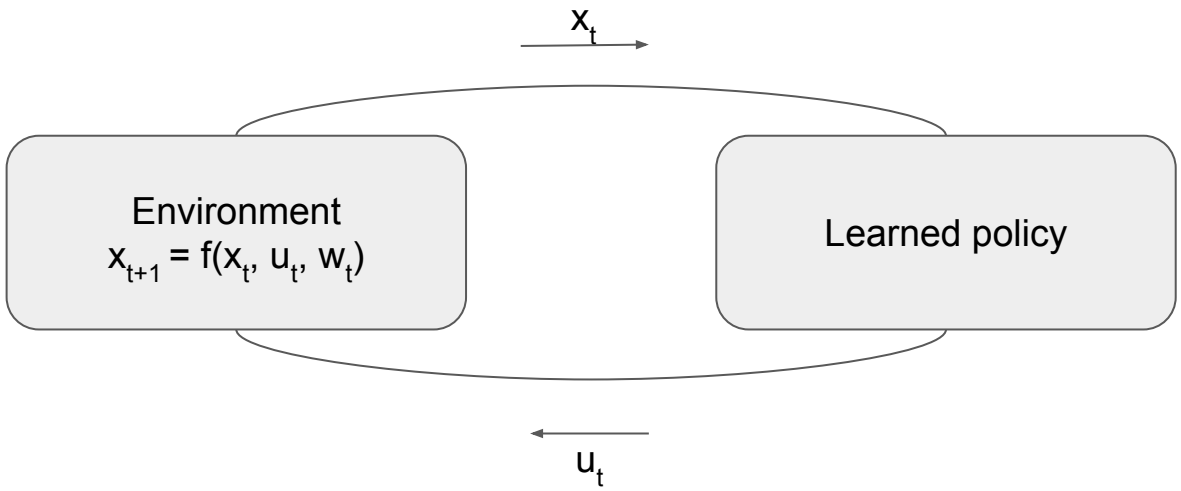
Stochastic feedback loop systems



Stochastic feedback loop systems



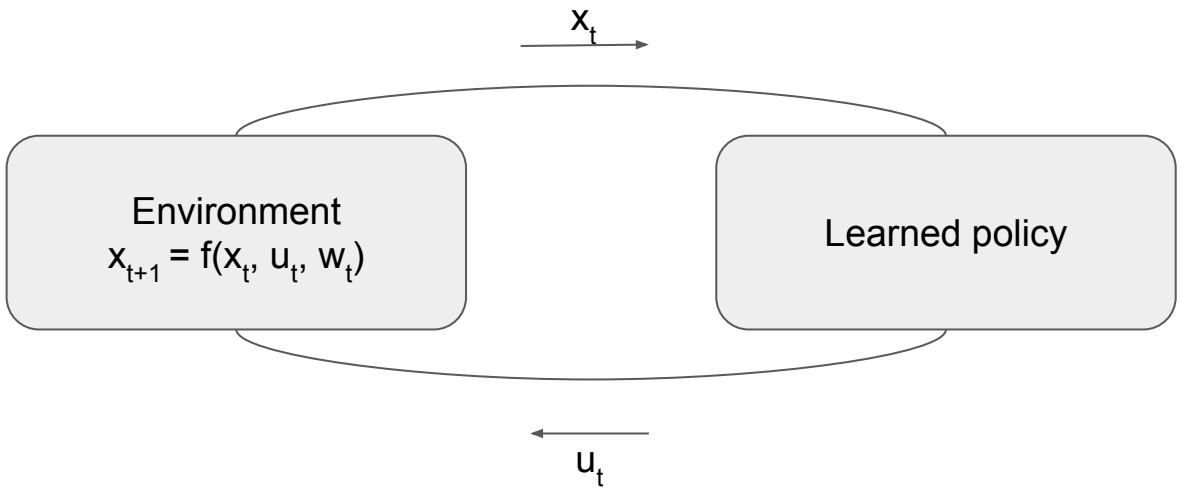
Stability verification for stochastic feedback loop systems



Stabilization set X_s, policy π

Assumptions: X_s is closed under system dynamics
X is compact
f, π are Lipschitz continuous

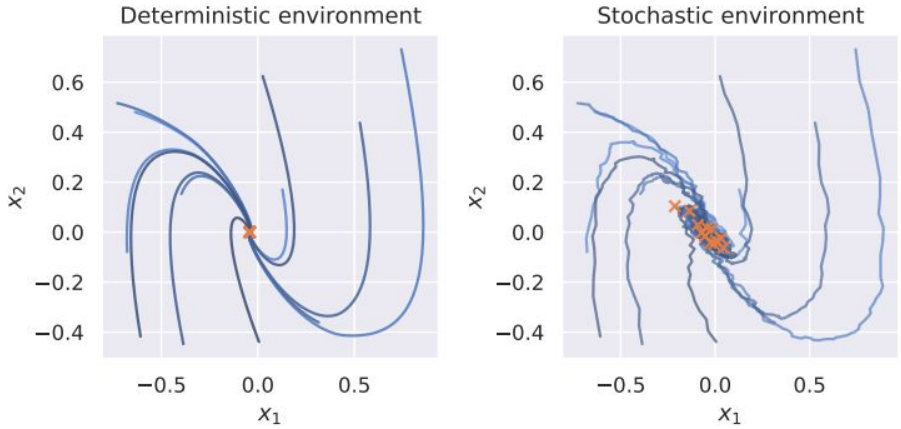
Stability verification for stochastic feedback loop systems



Almost-sure asymptotic stability verification: Verify that Prob[reach X_s] = 1 for each initial state x_0
(each x_0 induces a probability space over trajectories that start in x_0)

Related work

- Stability verification for deterministic systems via Lyapunov functions (LFs)
  - Convex optimization for polynomial systems
  - More recently: Learning neural network LFs
- Stability verification for stochastic control systems
  - Mostly theoretical works on stochastic extensions of LFs
  - Abstraction based methods, verify weaker notion of stability or over finite time horizon



Related work

- Stability verification for deterministic systems via Lyapunov functions (LFs)
  - Convex optimization for polynomial systems
  - More recently: Learning neural network LFs
- Stability verification for stochastic control systems
  - Mostly theoretical works on stochastic extensions of LFs
  - Abstraction based methods, verify weaker notion of stability or over finite time horizon
- Termination analysis in probabilistic programs
  - Ranking supermartingales (RSMs) are another stochastic extension of LFs
  - Computation via convex optimization or learning

Related work

- Stability verification for deterministic systems via Lyapunov functions (LFs)
  - Convex optimization for polynomial systems
  - More recently: Learning neural network LFs
- Stability verification for stochastic control systems
  - Mostly theoretical works on stochastic extensions of LFs
  - Abstraction based methods, verify weaker notion of stability or over finite time horizon
- Termination analysis in probabilistic programs
  - Ranking supermartingales (RSMs) are another stochastic extension of LFs
  - Computation via convex optimization or learning

In this work: Learn RSMs for stability verification

Stability verification via RSMs

An RSM for $\mathcal{X}_s$ is a continuous function $V : \mathcal{X} \to \mathbb{R}$ which is nonnegative and for which there exists $\epsilon > 0$ such that

$$\mathbb{E}_{\omega \sim d}\left[V\left(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\right)\right] \leq V(\mathbf{x}) - \epsilon$$

holds for each $x \in \mathcal{X} \backslash \mathcal{X}_s$

Stability verification via RSMs

An RSM for $\mathcal{X}_s$ is a continuous function $V : \mathcal{X} \to \mathbb{R}$ which is nonnegative and for which there exists $\epsilon > 0$ such that

$$\mathbb{E}_{\omega \sim d}\left[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\right] \leq V(\mathbf{x}) - \epsilon$$

holds for each $x \in \mathcal{X} \backslash \mathcal{X}_s$

Theorem 1 (Stability). If there exists an RSM for $\mathcal{X}_s$ then $\mathcal{X}_s$ is almost-surely asymptotically stable for the system.

Stability verification via RSMs

An RSM for $\mathcal{X}_s$ is a continuous function $V : \mathcal{X} \to \mathbb{R}$ which is nonnegative and for which there exists $\epsilon > 0$ such that

$$\mathbb{E}_{\omega \sim d}\left[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\right] \leq V(\mathbf{x}) - \epsilon$$

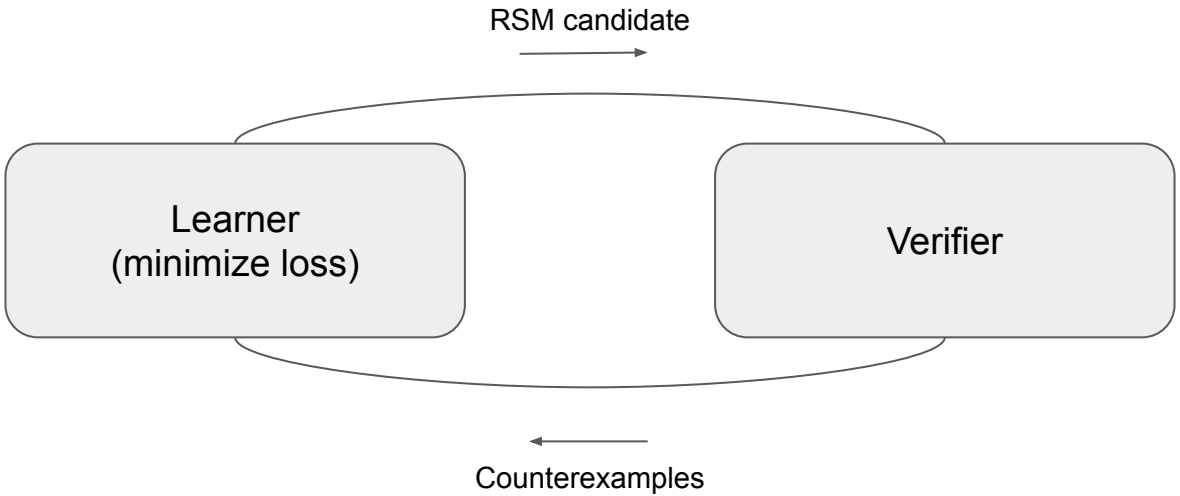
holds for each $x \in \mathcal{X} \backslash \mathcal{X}_s$

Theorem 2 (Bounds on stabilization time). If there exists an RSM for $\mathcal{X}_s$ then:

- $\mathbb{E}_{\mathbf{x}_0}[T_{\mathcal{X}_s}] \leq \frac{V(\mathbf{x}_0)}{\epsilon}$
- $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_s} \geq t] \leq \frac{V(\mathbf{x}_0)}{\epsilon \cdot t}$
- If the system has c-bounded differences, then $\mathbb{P}_{\mathbf{x}_0}[T_{\mathcal{X}_s} \geq t] \leq A \cdot e^{-t \cdot \epsilon^2 / (2 \cdot (c+\epsilon)^2)}$

Learner-verifier framework

- Represent RSM candidate as a neural network $V_\theta(x)$



Verifier

Due to compactness and continuity, need only to check the condition

$$\mathbb{E}_{\omega \sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \leq V(\mathbf{x}) - \epsilon$$

for every state $x \in \mathcal{X} \backslash \mathcal{X}_s$

Verifier

Due to compactness and continuity, need only to check the condition

$$\mathbb{E}_{\omega \sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] \leq V(\mathbf{x}) - \epsilon$$

for every state $x \in \mathcal{X} \backslash \mathcal{X}_s$

Idea: Discretize states space and check

$$\mathbb{E}_{\omega \sim d}\Big[V\Big(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\Big)\Big] < V(\mathbf{x}) - \tau \cdot K$$

at each state in the grid, where K is a Lipschitz bound of the system

Learner

Training objective: Empirical estimate of the expected value

$$\mathcal{L}_{\mathrm{RSM}}(\theta) = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\mathbf{x} \in \tilde{\mathcal{X}}} \left(\max \left\{ \sum_{\mathbf{x}' \in \mathcal{D}_{\mathbf{x}}} \frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_{\mathbf{x}}|} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} \right).$$

Learner

Training objective: Empirical estimate of the expected value

$$\mathcal{L}_{\text{RSM}}(\theta) = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\mathbf{x} \in \tilde{\mathcal{X}}} \left(\max \left\{ \sum_{\mathbf{x}' \in \mathcal{D}_{\mathbf{x}}} \frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_{\mathbf{x}}|} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} \right).$$

For faster verifier runtime -> add regularization to keep Lipschitz constant of the RSM network reasonable

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{RSM}}(\theta) + \lambda \cdot \mathcal{L}_{\text{Lipschitz}}(\theta)$$

Learner

Training objective: Empirical estimate of the expected value

$$\mathcal{L}_{\text{RSM}}(\theta) = \frac{1}{|\tilde{\mathcal{X}}|} \sum_{\mathbf{x} \in \tilde{\mathcal{X}}} \left(\max \left\{ \sum_{\mathbf{x}' \in \mathcal{D}_{\mathbf{x}}} \frac{V_\theta(\mathbf{x}')}{|\mathcal{D}_{\mathbf{x}}|} - V_\theta(\mathbf{x}) + \tau \cdot K, 0 \right\} \right).$$

For faster verifier runtime -> add regularization to keep Lipschitz constant of the RSM network reasonable

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{RSM}}(\theta) + \lambda \cdot \mathcal{L}_{\text{Lipschitz}}(\theta)$$

Theorem. Loss is minimized when $V_\theta(x)$ is an RSM.

Expected value computation

Compute $\mathbb{E}_{\omega \sim d}\left[V\left(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\right)\right]$

Expected value computation

Compute $\mathbb{E}_{\omega \sim d}\left[V\left(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\right)\right]$

Problem: V is a neural network -> No simple closed form solution

Expected value computation

Compute $\mathbb{E}_{\omega \sim d}\left[V\left(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\right)\right]$

Problem: V is a neural network -> No simple closed form solution

Solution: Decompose integral to sum and bound sum terms via abstract interpretation

$$\mathbb{E}_{\omega \sim d}\left[V\left(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\right)\right] \leq \sum_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \text{maxvol} \cdot \sup_{\omega \in \mathcal{N}_i} F(\omega)$$

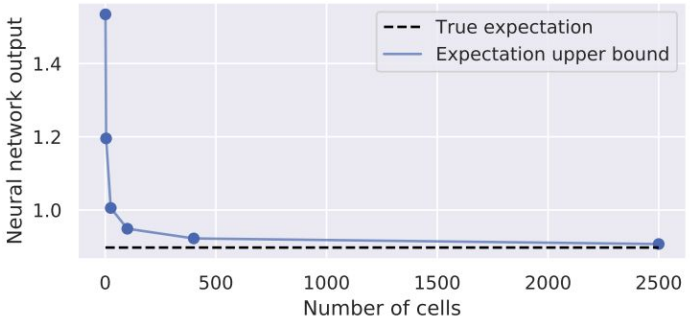
Expected value computation

Compute $\mathbb{E}_{\omega \sim d}\left[V\left(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\right)\right]$

Problem: V is a neural network -> No simple closed form solution

Solution: Decompose integral to sum and bound sum terms via abstract interpretation

$$\mathbb{E}_{\omega \sim d}\left[V\left(f(\mathbf{x}, \pi(\mathbf{x}), \omega)\right)\right] \leq \sum_{\mathcal{N}_i \in \text{cell}(\mathcal{N})} \text{maxvol} \cdot \sup_{\omega \in \mathcal{N}_i} F(\omega)$$



Experiments

- Two environments
  - 2D system
  - Inverted pendulum

Experiments

- Two environments
  - 2D system
  - Inverted pendulum
- Train neural network policy using PPO
  - Policy network: [128,128] hidden dimension with ReLU activation

Experiments

- Two environments
  - 2D system
  - Inverted pendulum
- Train neural network policy using PPO
  - Policy network: [128,128] hidden dimension with ReLU activation
- Run prototype implementation of our algorithm
  - RSM network: [128] hidden dimension with ReLU activation

Experiments

- Two environments
  - 2D system
  - Inverted pendulum
- Train neural network policy using PPO
  - Policy network: [128,128] hidden dimension with ReLU activation
- Run prototype implementation of our algorithm
  - RSM network: [128] hidden dimension with ReLU activation

Environment	Iters.	Mesh (τ)	Runtime
2D system	4	0.002	559
Inverted pendulum	2	0.01	176

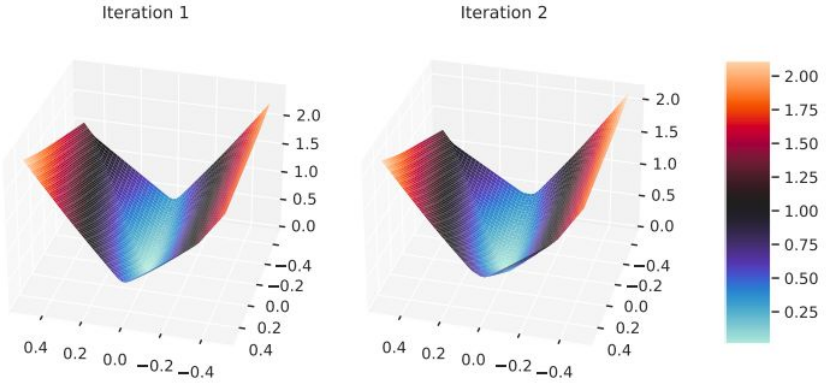
Table 1: Number of learner-verifier loop iterations, mesh of the discretization used by the verifier, and the total algorithm runtime (in seconds).

Experiments

- Two environments
  - 2D system
  - Inverted pendulum
- Train neural network policy using PPO
  - Policy network: [128,128] hidden dimension with ReLU activation
- Run prototype implementation of our algorithm
  - RSM network: [128] hidden dimension with ReLU activation

Environment	Iters.	Mesh (τ)	Runtime
2D system	4	0.002	559
Inverted pendulum	2	0.01	176

Table 1: Number of learner-verifier loop iterations, mesh of the discretization used by the verifier, and the total algorithm runtime (in seconds).



Experiments - Stabilization time

RSM implies stabilization time

$$\mathbb{E}_{\mathbf{x}_0}[T_{\mathcal{X}_s}] \leq \frac{V(\mathbf{x}_0)}{\epsilon}$$

Need to compute eps and min[V(x)]

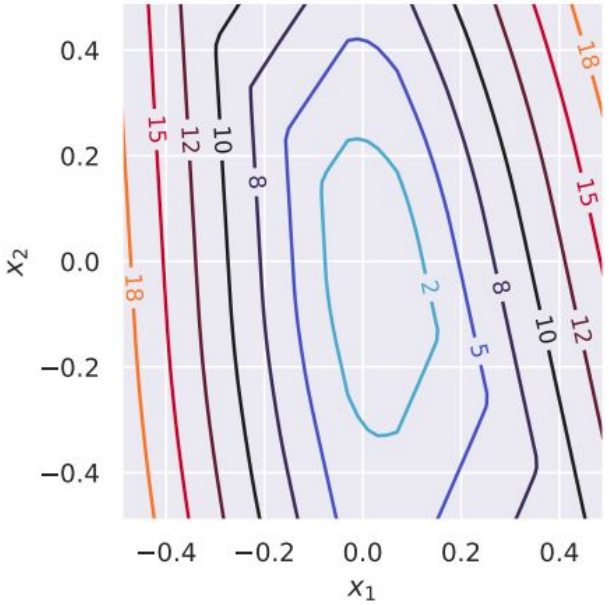
Experiments - Stabilization time

RSM implies stabilization time

$$\mathbb{E}_{\mathbf{x}_0}[T_{\mathcal{X}_s}] \leq \frac{V(\mathbf{x}_0)}{\epsilon}$$

Need to compute eps and min[V(x)]

Stabilization time for the inverted pendulum system (contour lines)



Conclusion

1. RSMs prove almost-sure stability in stochastic feedback loop systems, and provide bounds on stabilization time.

2. A framework for learning neural network RSMs.

3. Method for computing the expected value of a neural network function over a probability distribution.

4. Empirical validation of our approach on two RL benchmarks.