Intelligent Control Project: Neural Lyapunov Control

Department of Computer, Control, and Management Engineering Antonio Ruberti

Sapienza
Università di Roma

Introduction

- Stability analysis for nonlinear systems is done mainly through Lyapunov functions
- Hard to find in general, lots of manual work
- Only sufficient conditions

- Idea: Use NN to learn control and Lyapunov function for the closed loop system

Introduction

- Problem: NN can only approximate functions depending on loss
- The NN might not represent a Lyapunov function
- Use a constraint solver to verify Lyapunov conditions

NN: structure

- Use another neural network to learn a control

  $u = Kx$
- Use a multilayer feedforward neural network to learn a candidate LF, V(x)
- Use tanh activation functions to ensure smoothness of V(x)

Recap: Lyapunov stability conditions

Consider a system $\dot{x} = f(x)$, with f: $R^n \rightarrow R^n$ locally Lipschitz and with an equilibrium at the origin.
Consider a continously differentiable function V: $D \rightarrow R$

If V(x) is positive definite and $\dot{V}(x)$ is negative semidefinite, the origin is stable. If $\dot{V}(x)$ is negative definite the origin is asymptotically stable

NN: loss function

- Use a loss function to minimize the “degree of violation” of the Lyapunov conditions, i.e. *Lyapunov Risk*

$$L_\rho(\theta, u) = \frac{1}{N} \sum_{i=1}^{N} \Big(\max(0, -V_\theta(x_i)) + \max(0, L_{f_u} V_\theta(x_i)) \Big) + V_\theta^2(0)$$

Falsifier

- The falsifier uses a contraint solver to certify the Lyapunov conditions are respected

$$\Phi_\epsilon(x) := \left(\sum_{i=1}^{n} x_i^2 \geq \epsilon\right) \wedge \left(V(x) \leq 0 \vee \nabla_{f_u} V(x) \geq 0\right)$$

- We look for states x that violate Lyapunov conditions
- ϵ upper bounds the tolerable numerical error

Falsifier - δ−completeness

- The solver is δ-complete : it solves a relaxation of the problem to account for numerical precision
- If the relaxation is unsatisfiable, so is the original problem
  - i.e. if no states violating the contraints are found, the function is Lyapunov

Region of attraction tuning

- The framework only takes into consideration controls that are linear in the state
- These limit the region of attraction
- Try to enlarge the region of attraction by adding appropriate terms in the loss function $-\alpha V_\theta(x_i)$
- α is a hyperparameter

Experiments

Four experiments:

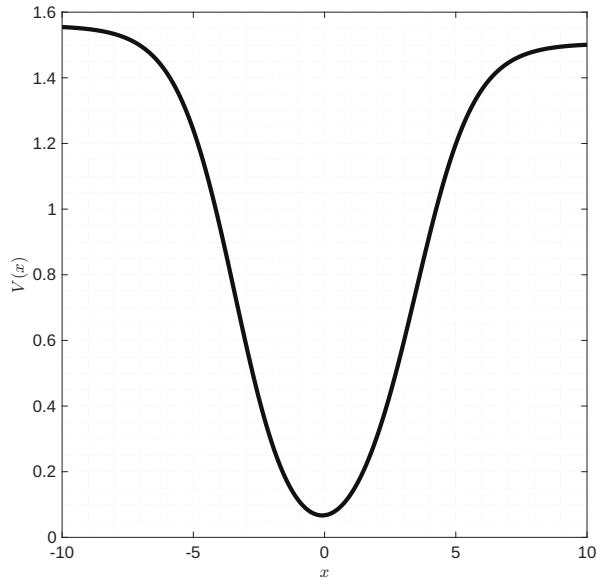
- Integrator (linear)
- Inverted pendulum (nonlinear)
- Cart-Pole (nonlinear, underactuated)
- Pendubot (nonlinear, underactuated)

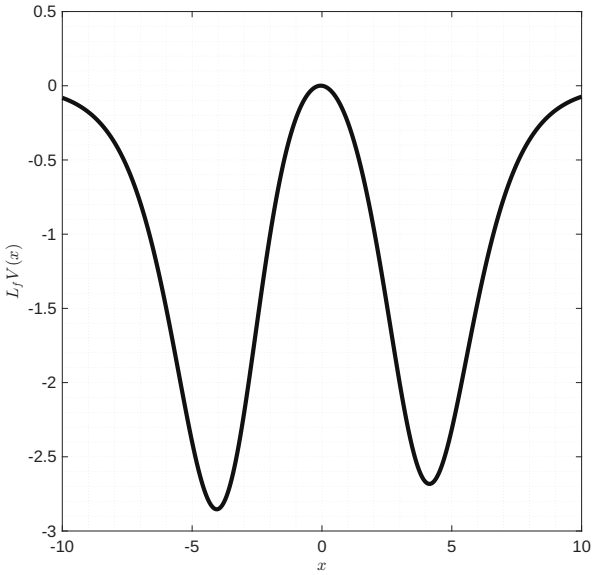
Experiments - setup

- NN training always on GPU
- Falsification always on CPU (12/16 threads)
- One falsification every 20 iteration of NN training
- Counterexamples are added to training dataset
- Control weights init. To LQR solution, LF weights random
- Random re-initialization after 10000 iterations of NN training, with loss roughly zero
- Training failed if no solution after 5 re-initializations

Experiments (1): Integrator

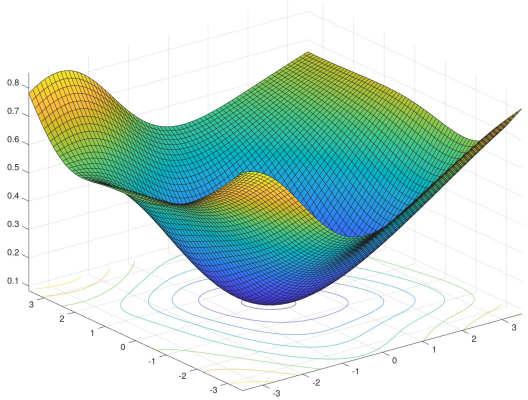
- A solution is found quickly, with a large valid region

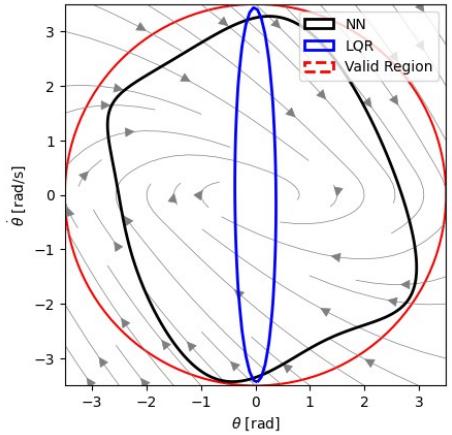


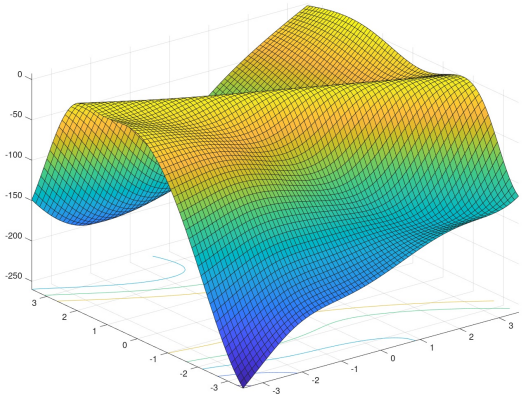


Experiments (2): Inverted pendulum

- A solution is found
- Origin is globally asymptotically stable with both LQR and NN controllers
- Level sets of NN Lyapunov function larger that LQR inside valid region

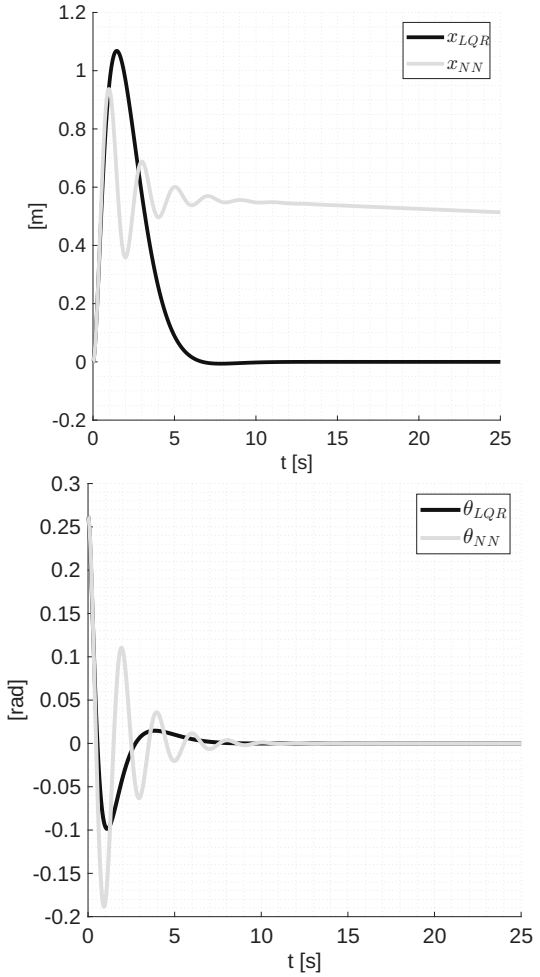






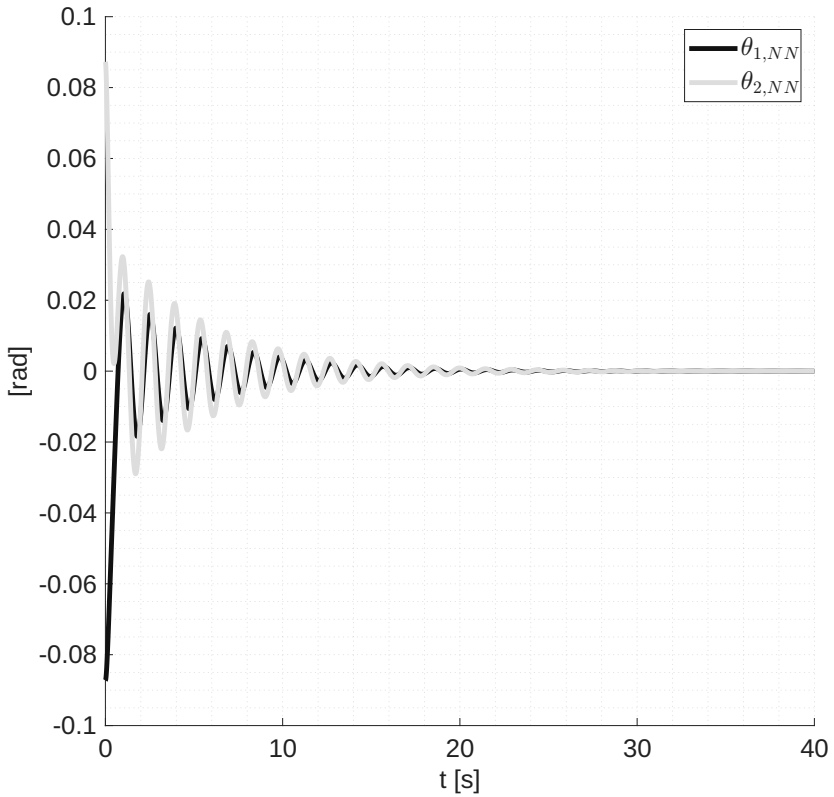
Experiments (3): Cart-pole

- No solution is found within imposed time limits
- Controller computed at the end of training stabilizes the system
  - But no LF to certify it



Experiments (4): Pendubot

- No solution is found within imposed time limits
- Controller computed at the end of training stabilizes the system
  - But no LF to certify it



Experiments: Comparison

Test	ϵ	δ	Valid region UB	Training iterations	Training time [s]	Verification time [s]	Valid?
Integrator	0.1	0.01	10	220	15.25	0.02	Yes
Inverted pendulum	0.5	0.01	3.5	2560	18.48	8.08	Yes
Cart-pole	0.02	0.001	0.1	50000	900	350	No
Pendubot	0.02	0.001	0.05	50000	790	95	No

Conclusions

- Approach works for simple systems and can find valid Lyapunov functions with large ROA
  - Cannot find solutions for slightly more complex systems
- Falsification is NP-hard, takes very long time for slightly more complex systems
  - Difficulty also grows with the dimension of the valid region
  - With a too large valid region falsification might take days for a single step
- Controls only linear in the state might be a limitation
  - Non linear controls make falsification even harder
- Larger networks might help
  - Again, falsification becomes even harder

Thank you for your attention!