Deep Learning Tubes for Tube MPC

Johan Gronqvist

2020-11-30

Overview

Contents

- MPC
- Tubes
- Problems
- Deep Learning
- Summary

Reference

- Based on "Deep Learning Tubes for Tube MPC" by David D. Fan, Ali-akbar Agha-mohammadi & Evangelos A. Theodorou. (arXiv:2002.01587).

Why am I interested

Tube MPC, Linear dynamics

- Assuming A, we guarantee B for all disturbances, uncertainties and other issues captured by Ω.

Deep Learning Theory

- Assuming A, we can guarantee B for all disturbances, uncertainties and other issues captured by our dataset.
- Nonconvex global optimization problem with large number of parameters.

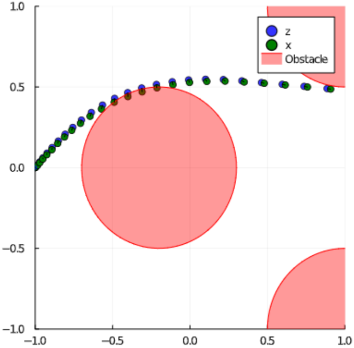
Pragmatic Deep Learning

- We did some sensible stuff, and we saw some sensible results, so we are happy.

MPC

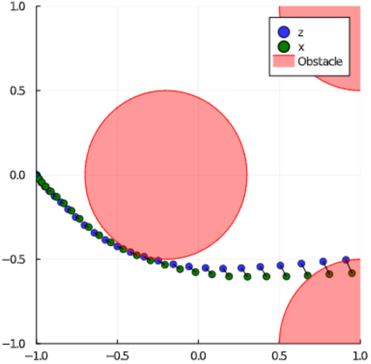
MPC

- Reach a Goal
- Avoid obstacles
- Solve a sequence of open-loop control problems
- At each time-step: Observe, Plan, Execute first step



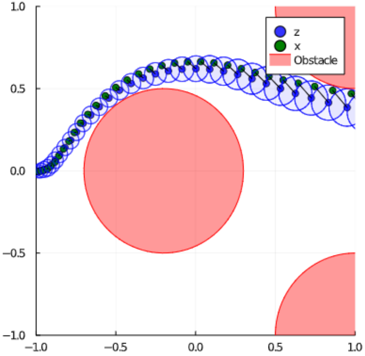
Disturbances

- Consider disturbances or model uncertainties
- Future states cannot be known precisely
- Set of potential futures



Many futures

- Planning a trajectory that avoids obstacles is not enough
- Need to plan with uncertainty
- Tube of futures
- Choose a control that yields a good tube.



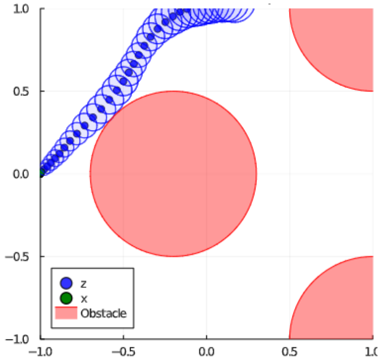
Tubes

First attempt: MPC in open loop

- True state-space x
- A nominal model $z_+ = f_z(z, u)$
- Tube width model $\omega_+ = f_\omega(\omega, z, u)$
- $u_1, \dots, u_N \longrightarrow (z_1, \omega_1), \dots (z_N, \omega_N)$
- Assume $|z_k - x_k| < \omega_k$ with high probability
- Ensure z_k is at least a distance ω_k away from any obstacle.

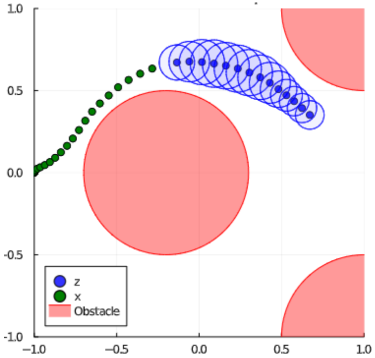
The previous picture is inaccurate

- Typically, uncertainty will grow over time
- Cautious control required, adaptinng to tube-growth
- We pretend that we need to plan an open-loop controller for the remainder of the track



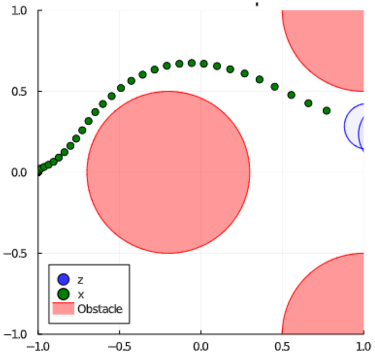
The previous picture is inaccurate

- At each later time-step, we re-plan with perfect state information
- This will prevent unbounded error growth



The previous picture is inaccurate

- Our MPC setup did not know that, and was overly cautious



Reality

MPC at time t should be aware that there will be MPC optimizers at each later stage with full state information.

Compromise

- Fix a tracking controller
- Use MPC to plan waypoints for the tracking controller
- Tube width bounded, due to tracking controller
- Good longer term planning with MPC

MPC in closed loop

- Assume that a tracking controller is given
- At every timestep t
  1. Perform full state measurement
  2. Plan set of future waypoints for tracking controller
  3. Take tubes into account

Three Problems

Tube dynamics

- How quickly do tubes grow
- We previously assumed a model for this.
- Where does it come from?
- Why would we trust it?
- The linear case is simpler than the general case.
- Can we do better by learning from data?

Error dynamics

- Tubes were centered around nominal model's trajectories.
- Nominal model could have systematic errors.
- Can we learn those from data?
- Train a network for $e_+ = f_e(e, z_+, z)$
- Use tube of radius ω around $z + e$

Nominal Model

- Where did we get our nominal model?
- Why do we think it is a good one?
- Can we learn a better one from data?

Deep Learning

Nonlinear function approximation

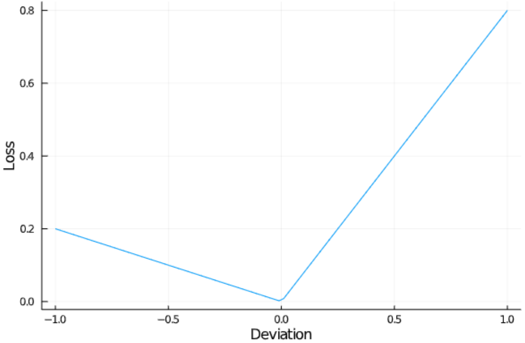
- Deep Learning provides
  - Parametrizations of nonlinear functions
  - Optimization procedures for fitting parameters to data

Tube width dynamics

- We want to parametrize a function f_ω such that

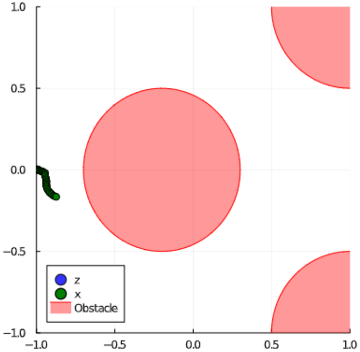
$$\omega_{t+1} = f_\omega(\omega_t, z_t, v_t)$$

- Quantile loss
  - $\mathrm{Prob}[\omega_+ > f_\omega(\omega, z, v)] = \alpha$
  - (Assuming that our network was perfectly trained on our perfect dataset.)



Epistemic uncertainty

- Our dataset is not perfect, unfortunately
- Need to quantify to what extent new data is "new"
- Train a Projector P to project away dataset.
- Anything that remains is new data



Monotonicity

- Additional loss term
- Ensure $f_\omega(\omega + \epsilon, z, u) \geq f_\omega(\omega, z, u)$

Error dynamics

- System $x_{t+1} = f_x(x_t, u_t)$
- Tracking Controller $u_t(z_{t+1}, x_t)$
- Nominal Model $z_{t+1} = f_z(z_t, v_t)$
- Learn systematic errors, $e_t = z_t - x_t$
- Ansats $e_{t+1} = f_e(e_t, z_t, v_t)$
- Train a neural network

Nominal model

- Unknown system $x_{t+1} = f_x(x_t, u_t)$
- Tracking controller $u_t(z_{t+1}, x_t)$
- Assume we do not have a good approximate model for z.
- Learn a model of f_x and use that as the approximate model.

It works, sometimes, maybe

- In a simple world there are guarantees
- In a complex world, Deep Learning can be useful
- Taming Deep Learning is about taking a well understood method, and adding a little deep learning at a time.
- In a controlled way.

The End

Thanks