Robust MPC Estimation by Deep Learning

Report

CASTETS Edouard

1 Abstract

My objective is to use a deep learning model to predict the control law based on a MPC model, the advantage of this method is that it has a lower computation cost. Yet the issue of this problem is that the deep learning model won't give the ideal control law, to ensure guarantees on constraint and stability we will then use a Robust MPC method and validate it with a statistical method as it is explained in this scheme :

2 Introduction

Improving time and efficiency of optimization algorithm is a constant challenge while optimization problems occupy an ever-increasing place in the world to control systems. On some systems we try to keep the lowest optimization time and computation cost for the lightest and cheapest system. Many solutions can be considered, yet as Deep Learning is a growing center of interest in control in nowadays.
One of the commonly used optimization algorithm is the Method Predictive Control, also known as MPC. A alternative to use this method which has a huge computation cost and time would be to use a deep learning neural network trained on the MPC control law to predict the control for a given point, yet this concept induce stability and constraint issues as the prediction by the model is an approximation of the true solution of the MPC problem. A paper wrote by Hertneck et al. [1] proposes a solution to the stability and constraint issues by learning on a robust MPC model and validating the deep learning model by analyzing its maximal approximation error. If the maximal validation error is below a bound, then the deep learning model will be able to satisfy stability and constraint issues for the problem.

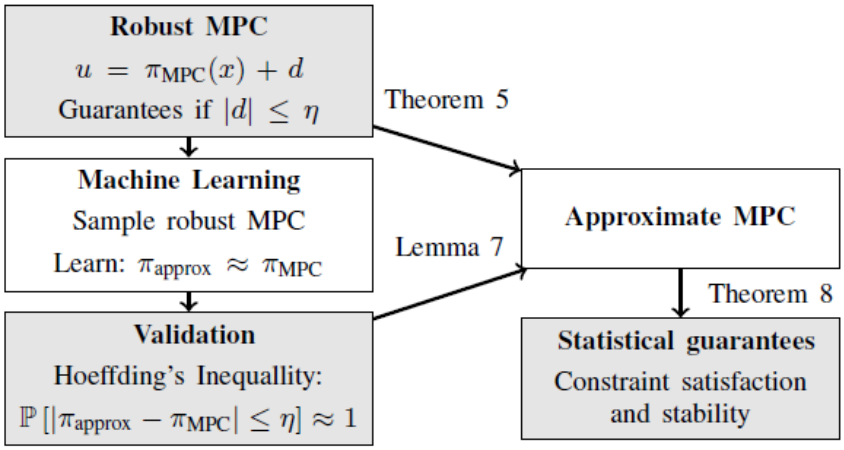

Figure 1: Model

This theory and method for computation has been fully developed in the paper of Hertneck et al. [1] and in a thesis [2]. The objective of my project will be to create my own Deep Learning model based on this paper's methods. In nowadays a lot of machine and deep learning papers are provided without code or precise explanations on how to pre-process the data while they presents very good results. It is sometimes impossible to reproduce their results or create a functional model. Thus the contribution of this project will be to provide a code for pre-processing and a neural network

with my results.

3 Theory and Method

3.1 Robust MPC and Approximation

All the Theory can be found here [1] and here [2]:
As the theory is fully developed in the paper I will only summarized the step that I will follow without proving that these steps ensure guarantees.

The objective is to predict learn the MPC control law for the following system :

$$\begin{aligned} X_{k+1} &= A * X_k + B * u_k \\ A &= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}; B = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ Q &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; R = 0.1 \end{aligned}$$

With Q and R penalization matrix respectively for X_k and u_k.
Here are the constraints :

$$u \in [-2; 2] = U = \{u \in \mathbb{R} | Lu \leq \mathbb{1}_2\}$$

$$X \in [-3; 3] \times [-3; 3] = \chi = \{X \in \mathbb{R}^2 | HX \leq \mathbb{1}_4\}$$

The classical Robust MPC problem aims to find the $u_{0|t}, ..., uN-1|t$ for a given $X(t)$:

$$\min_{u(.|t)} \sum_{k=0}^{N-1} l(X(k+t|t), u(k+t|t)) + V_f(X(N+t|t))$$

with these constraints :

$$\begin{aligned} x(t|t] &= x(t) \\ x(k+t+1|t) &= f(x(k+t|t), u(k+t|t)) \\ x(k+t|t) &\in \chi_k, k = 0, \ldots, N-1 \\ u(k+t|t) &\in U_k, k = 0, \ldots, N-1 \\ x(N+t|t) &\in \chi_f \end{aligned}$$

Assume that the solution of the MPC problem for a given $X(t)$ is $Pi_{RMPC}(X(t))$ and the solution given by the trained neural network is $Pi_{DNN}(X(t))$.
Control law from RMPC :

$$u(.|t)^* = \Pi_{RMPC}$$

A well designed RMPC problem is a problem that constraints the solution across time such that for

$$u(.|t)^* = \Pi_{RMPC} \pm d, d \in [-\eta; \eta]$$

the control law will satisfies the constraint and stability of the problem.
Then here all the problematic here is to compute χ_k ,U_k for all k in order to get the control law from this RMPC problem because the Neural Network will learn from existing RMPC control laws. and predict the control law with a quantified maximum error η, the RMPC problem must be well constructed such that the error on prediction η does not impact constraints and stability guarantees.

Control law from Deep Neural Network after training and validation :

$$u(.|t)^* = \Pi_{DNN} = \Pi_{RMPC} \pm d, d \in [-\eta; \eta]$$

The constraining of the set will depend on η and the validation process will consist on checking that $d \in [-\eta; \eta]$. In the paper the team is using a probability law to measure the risk and validate the DNN model, yet for our simple problem it is easy to obtain a well fitted model that ensures $d \in [-\eta; \eta]$ so we won't be using the paper's method here.

3.2 Design the Robust MPC

As defined in the paper we define the set :

$$u \in [-(2-\eta); (2-\eta)] = U_t = \{u \in \mathbb{R} | L_t u \leq \mathbb{1}_2\}$$

χ_k ,U_k can be computed as :

$$\chi_k = (1 - \epsilon_k)\chi$$
$$U_k = (1 - \epsilon_k)U_t$$

All the problematic is around the calculus of ϵ_k. Applied to our problem, the method of the paper proposes to solves the following problems to compute ϵ_k :

$$\min_{Y,X} -log(det(Ymax))$$

With constraints :

$$Y \preceq Ymax$$
$$\begin{bmatrix} Y & YA^T + X^T B^T & YQ^{1/2} & X^T R^{1/2} \\ AY + BX & Y & 0 & 0 \\ Q^{1/2}Y & 0 & I & 0 \\ R^{1/2}X & 0 & 0 & I \end{bmatrix} \succeq 0$$

To solves this LMI it I used Matlab using YALMIP package and I used a YALMIP solver SDPT3 found on a GitHub which is made for logdet optimization with matrix. Once it was solved I was able to compute :

$$P = Y^{-1}$$
$$K = XP$$
$$Q = P - (A + BK)^T P (A + BK)$$

(as it is not specified in the paper we take the Q solution that equalizes the Assumption 24 of [2])
We can now compute :

$$\epsilon_l = \min \lambda(Q); c_{\delta,u} = \max \lambda(P); c_{\delta,l} = \min \lambda(P)$$

$$\rho = 1 - \frac{\epsilon_l}{2 * c_{\delta,u}}; k_{max} = ||K||_{L2}$$

$$\tilde{k}_{max} = ||P^{-1/2}K^T||_{L2}$$

We won't need to compute δ_{loc} mentioned in the paper because the T value included in the formula is issued of a limited development of the problem with X and u as variable(see []koler), yet as A and B are constant matrix $T = 0$, thus the value in [] are $\delta_{loc} = \infty$ and $\eta_1 = \infty$. Thus we can choose η as we want as long as we can train a deep learning model to achieve the validation.
Also as A and B are constant matrix the Lipschitz constant is : $\lambda = ||B||_{L2}$.
The paper is introducing for simplicity of computation :

$$\epsilon = \eta * \lambda * \sqrt{c_{\delta,u}} * \max\{\tilde{k}_{max}||L_t||_\infty, \frac{||H||_\infty}{\sqrt{c_{\delta,l}}}\}$$

We can finally compute:

$$\epsilon_k = \epsilon\frac{1 - \sqrt{\rho}^k}{1 - \sqrt{\rho}}, k \in [0; N - 1]$$

Now we are able to compute for all k :

$$\chi_k = (1 - \epsilon_k)\chi$$

$$U_k = (1 - \epsilon_k)U_t$$

3.3 Compute RMPC and Build the neural Network

To compute the RMPC I used MPT3 toolbox to constraint the set for X and u across the time during the MPC sequence. I have chosen $\eta = 0.005$ which represent 0.16% of 3, the bound of X.Then the MPC was able to compute a control law u for a give X(t). I computed this u for every X of a meshed grid $[-3; 3] \times [-3; 3]$ with a step of 0.05. I built my dataset : the feature is the point X and the ground truth linked to this feature is $u = \Pi_{RMPC}(X)$. The data used to train the DNN were point where the RMPC was feasible χ_{feas}.
I used Pytorch framework to build and train my neural Network which is a classic full connected neural network with a Leaky ReLU as activation function. Here we clearly want the model to over-fit so there is no worries about regularization methods. I trained with for 50 epochs with a L1 Loss function, a batch size of 32. The optimizer used is an Adam Optimizer with a scheduled learning rate:

Epoch range	Learning rate	Beta 1	Beta 2
0 to 9	0.001	0.9	0.999
10 to 24	0.0001	0.9	0.999
25 to 50	0.00005	0.9	0.999

Table 1: Adam Optimizer

This is the architecture of the neural network :

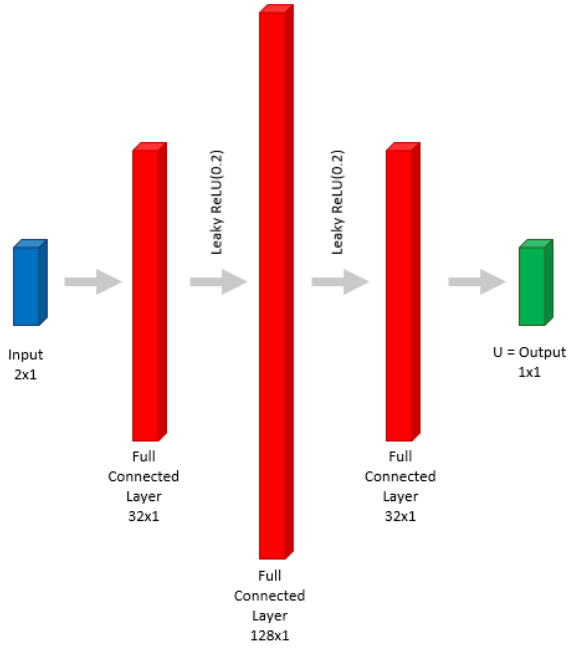

Figure 2: Model

4 Results and analysis

This is a simulation of the path computed in closed loop by the RMPC in Matlab. The white parts represents part of the domain $[-3;3] \times [-3;3]$ where the RMPC had no valid solution. The colored parts are χ_{feas}

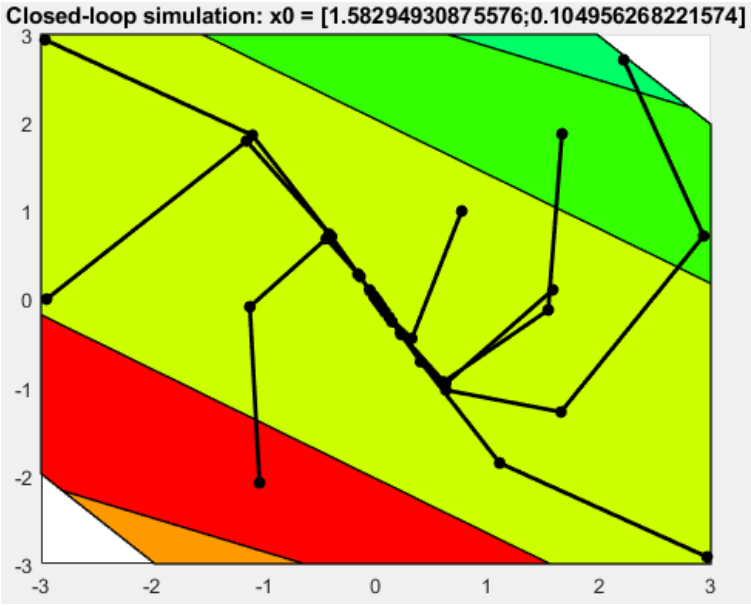

Figure 3: Matlab Path

After a training of the Deep Neural Network, I validated the model by checking that for all point X of the grid, thus all point Xof the data set :

$$|\Pi_{DNN}(X) - \Pi_{RMPC}(X)| \leq \eta = 0.005$$

Here is a graph of the maximum error and mean error found after predicting on the whole dataset for each epoch. As we can see the criteria is respected (maximum error is below 0.003). Thus the Deep Learning model is predicting a control law that will satisfies constraint and stability on all the points of χ_{feas}.

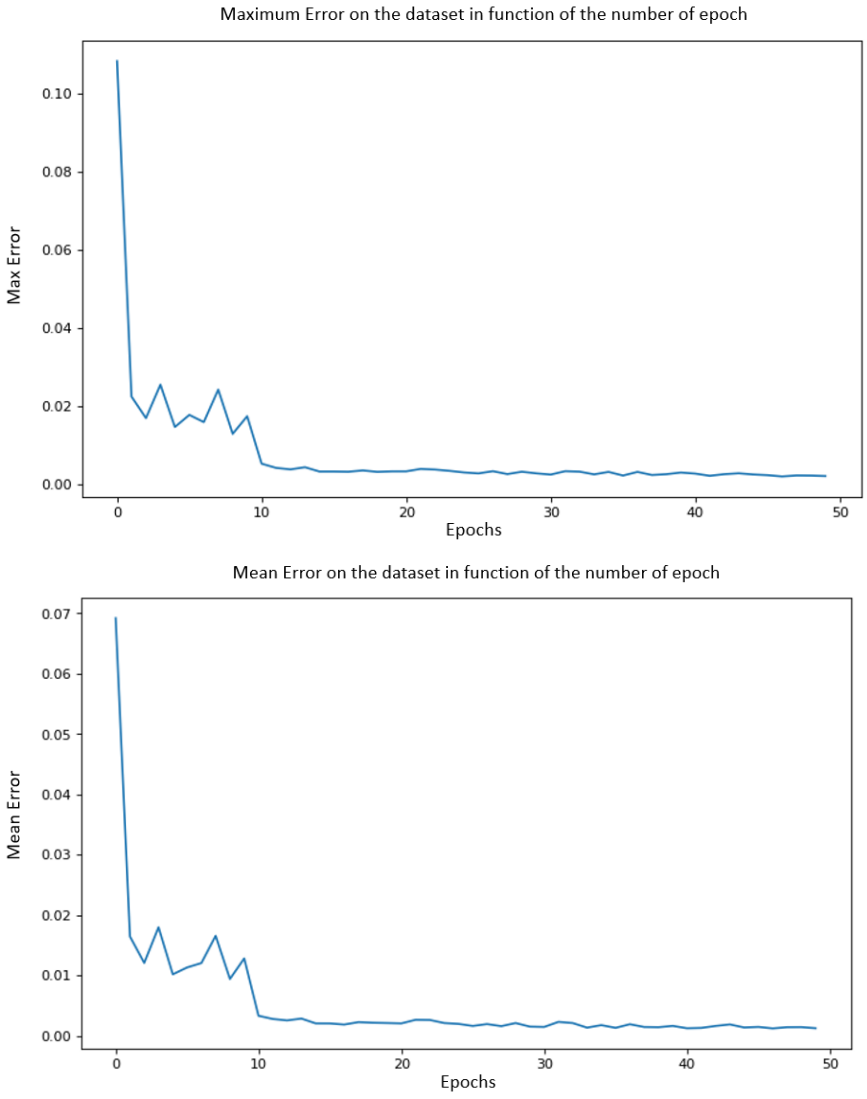

Figure 4: Error Curves

Here is a simulation of closed loop control using the Deep Neural Network to predict the control law. The DNN is working even for points close to the bounds of χ_{feas}.

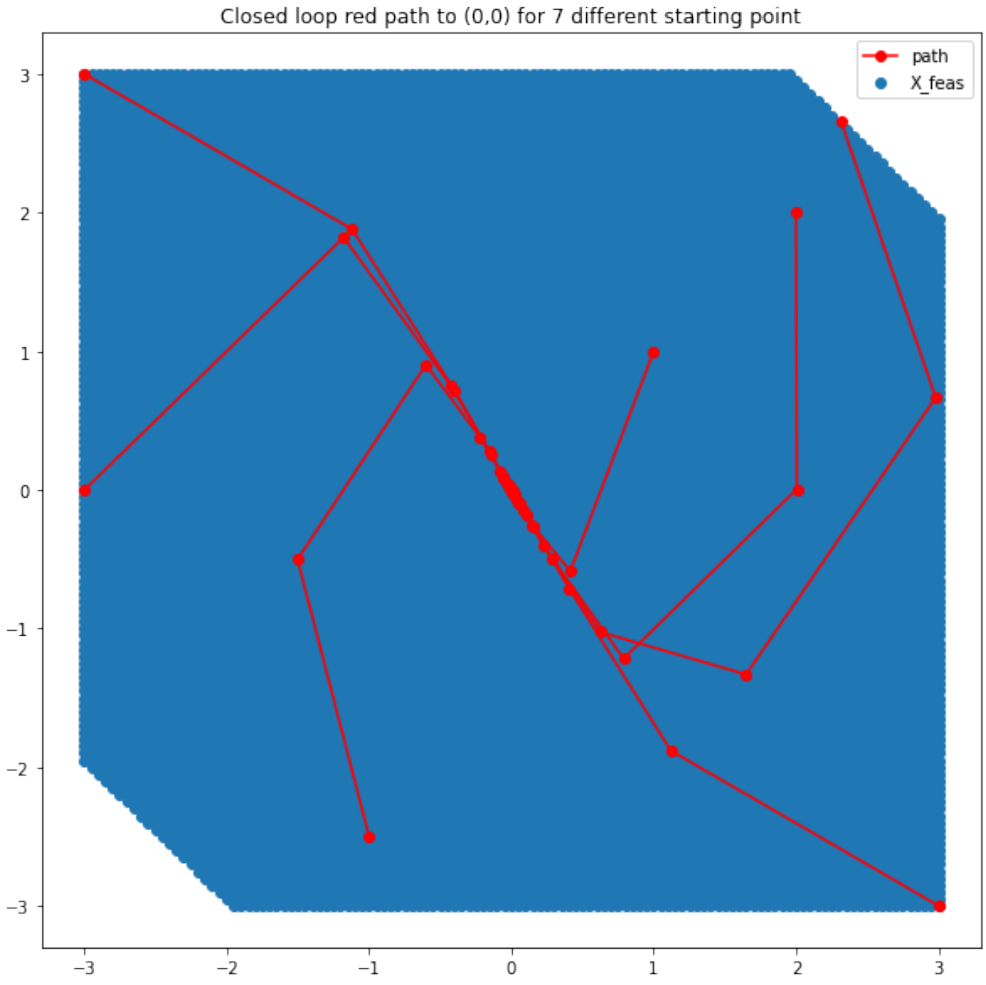

Figure 5: Closed Loop Paths

5 Conclusion

I have succeeded to create a Deep Learning model to predict a robust input in closed loop by applying the method of the paper and transposing it to a linear simple case, I provide the code to compute an RMPC on this case and a code to train a simple model which produces a robust prediction. In the future it could be interesting to see how to apply it to non linear model as they do in the paper.

.

References

[1] Michael Hertneck et al. "Learning an Approximate Model Predictive Controller with Guarantees". In: *IEEE Control Systems Letters* 2 (3 July 2018), pp. 543–548. ISSN: 24751456. DOI: 10.1109/LCSYS.2018.2843682.

[2] Michael Hertneck et al. *Learning an Approximate Model Predictive Controller with Guarantees Student Thesis*. 2018.