Bayesian Optimization in Model Predictive Control

Frederic Dlugi

Universität zu Lübeck
frederic.dlugi@student.uni-luebeck.de

February 4, 2022

Abstract

When using MPC to control a vehicle it is often neccesary to fine tune control and vehicle parameters to get good performance in reality. This is a time consuming process that often relies on trial and error or grid search of the parameter space. In this paper we evaluate the use of Bayesian Optimization to tackle this problem. We simulate vehicle dynamics of a simple bicycle model with two parameters. We try to find the MPC cost ratios and real vehicle parameters by optimizing controller performance with different parameters using Bayesian Optimization.

I. Introduction

We start by introducing bayesian optimization and MPC on a high level. This is done to let you know what I understand when talking about these algorithms.

i. What is Bayesian Optimization?

Bayesian optimization is a strategy for global optimization of backbox functions. (Alg. 1) It therefore does not require the computation of gradients, but works best on continuous functions. It is best suited for optimizing functions, where each evaluation takes a long time. The number of input dimensions for bayesian optimization is typically less then 20 [1]. Bayesian Optimization uses an aquisition function that operates on a gaussian process of the evaluations. In this paper we evaluate three different acquisition functions: Expected Improvement (Alg. 2), Probability of Improvement (Alg. 3) and Upper Confidence Bound (Alg. 4). We use the Matern ($\nu = 2.5$) kernel for all experiments. We also vary the α Parameter of the Gaussian Process to smooth out the cost landscape.

Algorithm 1 Bayesian Optimization

```
f ← black box function
initPos ← initial positions
evaluations ← f(initPos)
for i = 1 → N do
    α ← acquisitionFunction(evaluations)
    nextPos ← argmax(α)
    evaluations.append(f(nextPos))
result ← max(evaluations)
```

Algorithm 2 Expected Improvement

```
μ, σ ← gp.predict(x)
z ← (μ − y_max − xi)/σ
result ← (μ − y_max − xi) ∗ cdf(z) + σ ∗ pdf(z)
```

Algorithm 3 Probability of Improvement

```
μ, σ ← gp.predict(x)
z ← (μ − y_max − xi)/σ
result ← cdf(z)
```

Algorithm 4 Upper Confidence Bound

```
μ, σ ← gp.predict(x)
result ← μ + κ ∗ σ
```

ii. What is MPC?

MPC is an acronym for model predicive control. It replaces classical control algorithms that usually work in an offline manner with an optimizer, that solves the optimal control problem for a receding horizon (Figure 1). Each action taken is the first action of the optimal control plan. The plan gets updated continuously (Figure 2), this is called receding horizon approach.

Figure 1: *Blockdiagram of MPC algorithm*

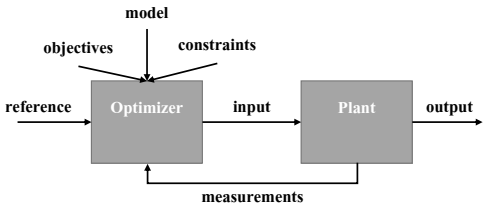
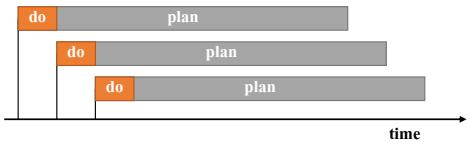


Figure 2: *Plan horizon of MPC algorithm*



iii. Kinematic Bicycle Model

We use a kinematic bicycle model defined by algorithm 5 and 6 [4]. We try to estimate the L_r and L_f in this paper.

Algorithm 5 Kinematic Bicycle Model

$a \leftarrow$ accelaration (input)
$\omega \leftarrow$ steering angle rate (input)
$x_c, y_c \leftarrow$ center position
$v \leftarrow$ speed
$\theta \leftarrow$ absolute angle
$\delta \leftarrow$ steering angle
$\beta \leftarrow$ side slip factor
$L_r \leftarrow$ Distance center to rear wheel
$L_f \leftarrow$ Distance center to font wheel
$L \leftarrow L_r + L_f$

Algorithm 6 Update Bicycle Model

$\dot{v} \leftarrow a$
$\dot{x_c} \leftarrow v * \cos(\theta + \beta)$
$\dot{y_c} \leftarrow v * \sin(\theta + \beta)$
$\dot{\theta} \leftarrow v * \cos(\beta) * \tan(\delta) / L$
$\dot{\delta} \leftarrow \omega$
$\beta \leftarrow \text{atan2}(L_r * \tan(\delta), L)$

II. Literatur review

In Multi-Objective Optimization of a Path-following MPC for Vehicle Guidance [2] BO is used to weight error terms to find the ratio results in the best reference tracking performance of a complex vehicle model in a simulation. Using BO with expected improvement as acquisition function. They found the BO algorithm used to be more efficient then random search when comparing the number of evaluations needed to find satisfactory results.

Lukas Hewing and colaborators [3] explored the use of BO to decrease model missmatch. They learned a dynamics model of a miniature race car using a gaussian process from real data, while the vehicle is controlled through MPC. They experimented with sparse approximations to gaussian processes to use them in real-time applications.

Lu Qiugang and colaborators [5] explored MPC parameter tuning in HVAC plants. They had good results and with limited evaluations. They used upper confidence bound as acquisition function and also had two tuning parameters. The experiment setup is similar to the one used in this paper.

What papers did similar things? What where their problems? What did they achieve? What do we do differently?

III. Experiments

In this paper we explore two uses of BO in the MPC context. First we try to find the optimal

ratio between input (accelaration and steering angle rate) cost and reference tracking ($x - x_{ref}$ and $y - y_{ref}$). Secondly we try to find vehicle specific parameters using BO.

i. Experiment setup

Show block diagram for both experiments. (Show setup with graph) Show specific MPC implementation. Optimizer used is scipys SLSQP. Show cost function Show generate function Name number of evaluations per BO step and number of BO steps.

To generate the path, we first generate random normal distributed inputs. Then we use the Bicycle model with real L_r and L_f values with these inputs to generate N sequential positions.

Algorithm 7 Generate Path

$L_r \leftarrow$ Distance center to rear wheel
$L_f \leftarrow$ Distance center to font wheel
initPos $\leftarrow (0, 0)$
N $\leftarrow$ Number of path points
bike $\leftarrow$ Bike(L_r, L_f, initPos)
inputs $\leftarrow normalDistibutedArray(N - 1, 2)$
path $\leftarrow zeroArray(N, 2)$
for $i = 1 \rightarrow (N - 1)$ **do**
    bike.$step$(inputs[i - 1])
    path[i] $\leftarrow$ bike.$getPosition()$
return path

ii. Exploring MPC cost ratio using BO

Compare different acquisition functions. Cost ratio was explored with different vehicle parameters (show different gaussian processes)

iii. Finding Vehicle parameters using BO

Using different target parameters Using different acquisition functions Show main target (1.2, 0.8) -> accumulate all evaluations in one gaussian process

IV. Conclusion and further research

There is no general ratio between input and reference cost. It depends on vehicle parameters. The optimal ratio is still similar. Finetuning still required. Using many different reference tracks, leads to better results.

Use more suffisticated MPC. Try more hyper parameters. Try if this works on real hardware or using one simple and one complex vehicle model. Try measuring more or less vehicle states. Add uncertency to measurents and model.

References

[1] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[2] Ali Gharib, David Stenger, Robert Ritschel, and Rick Voßwinkel. Multi-objective optimization of a path-following mpc for vehicle guidance: A bayesian optimization approach. *arXiv preprint arXiv:2104.03773*, 2021.

[3] Lukas Hewing, Alexander Liniger, and Melanie N. Zeilinger. Cautious NMPC with gaussian process dynamics for miniature race cars. *CoRR*, abs/1711.06586, 2017.

[4] Daniel S. Ingram. Self driving cars specialization coursera: Bicycle model. `https://github.com/daniel-s-ingram/self_driving_cars_specialization/blob/master/1_introduction_to_self_driving_cars/Kinematic_Bicycle_Model.ipynb`. Accessed: 2022-01-23.

[5] Qiugang Lu, Ranjeet Kumar, and Victor M Zavala. Mpc controller tuning using bayesian optimization techniques. *arXiv preprint arXiv:2009.14175*, 2020.