Bayesian Optimization in Model Predictive Control

Frederic Dlugi

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

March 19, 2022

Abstract

When using MPC to control a vehicle it is often necessary to fine tune model parameters and control costs 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 model parameters. We try to find the MPC cost matrices and real vehicle parameters by optimizing controller performance with different parameters using Bayesian Optimization.

I. Introduction

When using Model Predictive Control (MPC) to solve a problem a lot of fine tuning and calibration needs to be done for an algorithm to work on a plant. Assuming the popular quadratic cost function the cost matrices for input and state of the model need to be chosen. Many models also have parameters that represent physical aspects of the plant for best performance the model parameters need to match the plant. In many cases the model is a simplification of the plant, therefore multiple physical features may influence the optimal parameters of the model and can't be easily measured. In the design process of a MPC-controller the selection of optimal parameters and cost matrices needs deep understanding of the plant and time for manual tuning of the variables. In this paper we explore to use of Bayesian Optimization (BO) for the optimal selection of cost matrices and model parameters using a simulated bicycle vehicle.

i. What is Bayesian Optimization?

Bayesian optimization is a strategy for global optimization of black-box 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 acquisition 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.

ii. What is MPC?

MPC 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 (Fig-

Algorithm 1 Bayesian Optimization

f ← black box function
initPos ← initial positions
evaluations ← f(initPos)
for $i = 1 \rightarrow N$ **do**
  $\alpha \leftarrow$ acquisitionFunction(evaluations)
  nextPos ← argmax(α)
  evaluations.$append$(f(nextPos))
$result \leftarrow$ max(evaluations)

Algorithm 2 Expected Improvement

$\mu, \sigma \leftarrow$ gp.predict(x)
$z \leftarrow (\mu - y_{max} - \xi)/\sigma$
$result \leftarrow (\mu - y_{max} - \xi) * \text{cdf(z)} + \sigma * \text{pdf(z)}$

Algorithm 3 Probability of Improvement

$\mu, \sigma \leftarrow$ gp.predict(x)
$z \leftarrow (\mu - y_{max} - \xi)/\sigma$
$result \leftarrow$ cdf(z)

Algorithm 4 Upper Confidence Bound

$\mu, \sigma \leftarrow$ gp.predict(x)
$result \leftarrow \mu + \kappa * \sigma$

ure 2), this is called receding horizon approach.

Figure 1: *Block-diagram of MPC algorithm*

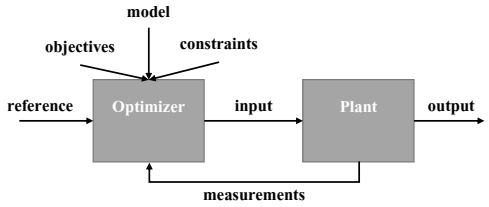
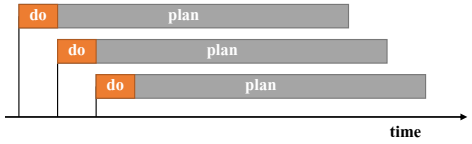


Figure 2: *Plan horizon of MPC algorithm*



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
$L \leftarrow L_r + L_f$

II. Literature 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

Algorithm 6 Update Bicycle Model

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

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 collaborators [3] explored the use of BO to decrease model mismatch. 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 collaborators [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.

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 (acceleration 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

We explore two uses of BO for MPC. Firstly we try to find the optimal ratio of input cost to state cost ratio (See Fig. 3). Secondly we estimate L_r and L_f parameters of our controller (See Fig. 4).

Figure 3: *Ratio experiment*

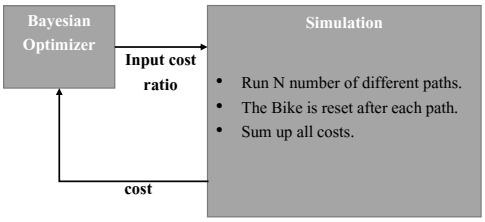
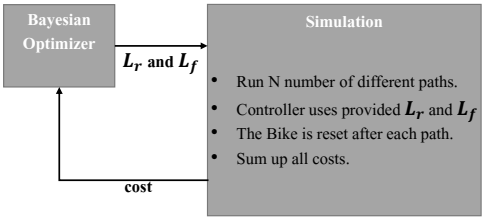


Figure 4: *L-Value experiment*



We used a standard MPC implementation that uses the cost function that is specified here (Alg. 7). We use scipy's SLSQP optimizer to minimize the cost function.

Algorithm 7 Cost function

$U \leftarrow$ Array of inputs $(N \times 2)$
$X \leftarrow$ Array of postions $(N \times 2)$
$X_{ref} \leftarrow$ Array of reference postions $(N \times 2)$
for $i = 0 \rightarrow (N - 1)$ **do**
  $cost + (X[i] - X_{ref}[i])^T Q (X[i] - X_{ref}[i])$
  $cost + U[i]^T R U[i]$
return cost

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 (See Alg. 8). We use 200 steps for all paths and all experiments.

ii. Exploring MPC cost ratio using BO

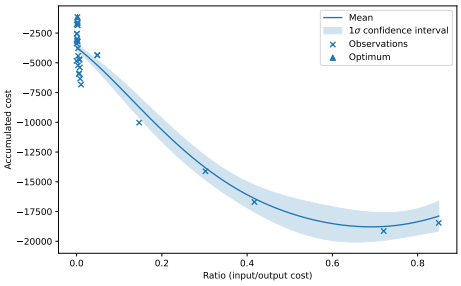
In this experiment we aim to find the optimal ratio between the Q and R matrix in the cost function (Alg 7). We try this for different vehicles that are differentiated in their L-parameter.

Algorithm 8 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

Since a full search of the ratios shows that Ratios between 0.001 and 0.003 result in better performance (See Figure 5). Therefore we explore only this range in all further experiments. We use 40 tracks per step and do 30 steps for both experiments. Increasing the regularization α parameter of the internal GP for BO to 0.01 improved performance.

Figure 5: *GP of full cost ratio search (Higher is better)*

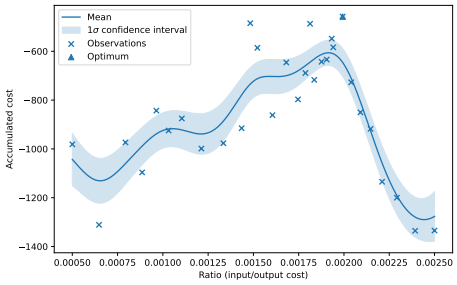


We found that the optimum ratio for all tested vehicles all with a length of $2m$ is close to 0.002 (See Fig 6). This is highly depended on the MPC implementation and vehicle dynamics.

iii. Finding Vehicle parameters using BO

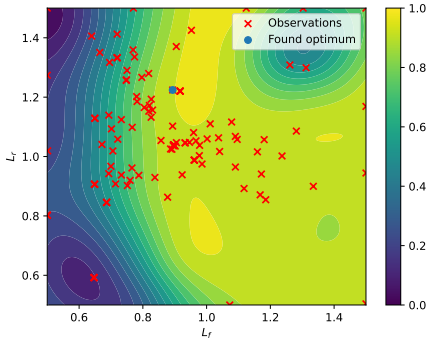
We aim to find the vehicle specific parameters L_f and L_r using BO. We try multiple parameters and different acquisition functions. Since evaluating 40 tracks still lead to noisy

Figure 6: *GP of detailed cost ratio search (Higher is better)*



cost we need to choose a suitable exploration-exploitation parameter in our acquisition function. For Expected Improvement 2 and Probability of Improvement 3 a $\xi = 0.5$ worked the best. For Upper Confidence Bound a $\kappa = 2.576$ worked the best. Increasing the regularization α parameter of the internal GP for BO to 0.1 improved performance.

Figure 7: *Mean of GP from L-parameter search (Target $L_f = 0.8$ $L_r = 1.2$)*



IV. Conclusion and further research

There is no general ratio between input and reference cost. It depends on vehicle dynamics vehicle parameters and the specific MPC implementation. The optimal ratio are still similar and close to 0.002 with our experiment. Therefore fine tuning is still required and us-

ing many different reference tracks, leads to better results.

Finding the vehicle parameters works well using BO if you tweak the BO parameter for exploration and exploitation tradeoff and set the regularization parameter of the internal GP for the BO to a suitable value. Since we only estimated two parameters in our experiment using BO in this case was not helpful.

In further research we would use a more sophisticated MPC since using the SLSQP optimizer from python does not work well for some states. It would also be interesting to try to find more vehicle parameters. Also we would like to try if the BO approach works on real hardware or using one simple and one complex vehicle model this could also be done in simulation by using different vehicle models for the controller and the simulation. In this paper we always measure the current vehicle position and update this inside the controller therefore measuring more or less vehicle states and the effects on the optimization performance would be interesting. To translate this approach to a real vehicle you would need to add uncertainly to measurements and to the 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.