Methods for Data-driven Model Predictive Control

Application to intelligent building control

Achin Jain, Truong X. Nghiem, Manfred Morari, and Rahul Mangharam

POC: A. Jain (achinj@seas.upenn.edu)

May 23, 2019

Optimal decision making is necessary and safety critical to the success of a wide range of control applications. Consider, for example, the case of autonomous racing. A common design objective for a controller is to minimize lap times while staying on a known track, avoiding collision with other competing cars, and simultaneously keeping below maximum possible acceleration governed by engine power. The *decision variables* or degrees of freedom for this controller are the force provided by the engine and steering angle. The controller essentially generates a sequence of engine force and steering angle that drive the car along the track in minimum time. Another application where optimal decision making plays an important role is of energy management in buildings. The design objective for the controller in this case is to minimize energy costs while regulating temperatures in all the rooms so as to keep the occupants comfortable. The *decision variables* for this controller include room temperatures, settings of equipments like air handling units and chillers. Now actuating the building with these optimized decisions provides cost savings while keeping occupants happy.

What's common in these control applications? A common theme in designing a controller for autonomous racing and optimal energy management is that the best controller is predictive in nature. Such a controller uses a mathematical model of the car or building to predict their behavior in the future. A mathematical model is a set of differential equations that describe the physics behind operation of a phyical system. In the first case, the model predicts precisely how the position and orientation of the car will change on varying the engine force and steering angle, and, in the second case, how the power consumption and measured room temperatures change on varying the temperature setpoints and equipment settings. Thus, a model of a physical system allows for planning the system's behavior in the future. This technique for control design is called model predictive control (MPC) [1]. An MPC controller solves an optimization problem using a mathematical model to optimize the decision variables.

What differentiates these control applications? Although, the general principle of MPC design remains the same in these two applications, the key difference is that designing a controller for minimal lap times is relatively easier than for optimal energy management in buildings. The difference arises due to the complexity of mathematical models. While in autonomous racing the physics is easier to capture in ordinary differential equations, it is much harder in the case of building operations. The theory of vehicle kinematics and dynamics is well understood in the literature [2]. There are very few tuning parameters in the standard kinematic or dynamical vehicle models like mass, wheelbase, distance between the center of gravity and wheels, and tire slip angles which all can be easily measured for a given configuration of a racing car. Thus, the mathematical model is accurately known except at insanely high speeds when tires are operating at the limits of slipping. On the other hand, identifying the mathematical model of a building is a complex process that requires massive engineering effort, expert domain knowledge and periodic retuning making the system identification cost and time prohibitive. The modeling complexity in buildings arises due to the nonlinear interaction between a large number of subsystems like variable air volume boxes, air handling units, chiller systems/compressors, air ducts and water loops, etc., with heat and mass flows within the building and with the external environment. Thus, the major barrier in developing high fidelity models (white box and grey box) of buildings, is the user expertise, time, and associated high costs required to develop a mathematical model that accurately reflects reality [3]. This includes the installation cost of retrofitting the building with additional sensors, the cost of training the engineering, commissioning and service personnel required to implement model-based control and the cost of the necessary engineering effort required for constructing a model. The modeling difficulty is compounded due to the fact that each building is designed and used in a different way and thus, has to be uniquely modeled.

MPC has been proven to be very powerful for multivariable dynamical systems in the presence of input and output constraints, and forecast of the disturbances. With a reasonable forecast of the external disturbances, the mathematical models predict the state of the system in the future and thus a predictive controller based on MPC can act preemptively to provide a desired behavior. The caveat is that MPC requires a reasonably *accurate* mathematical model of the system. As we have seen, physics-based modeling is not suitable for all applications, especially, building control. We therefore need accurate mathematical models at an economical cost in order to deploy MPC at scale. This article specifically focuses on different data-driven algorithms based on machine learning that reduce the cost of model development so that the deployment of MPC can be scaled up at a much lower cost in the applications like building control. Although models for vehicle dynamics can be improved even further using machine learning [4], but the scope of this article is limited to intelligent building control that requires entirely different set of techniques. In general, the algorithms introduced in this article are applicable to physical systems with slow response time.

Machine learning for prediction v/s predictive control

In supervised learning, prediction problems can be classified into either regression or classification. The goal is to identify the mapping (henceforth referred to as model) $f : x \mapsto y$ from features (also called regressors) x to output y, where the output is *continous* for regression and *discrete* for classification. Since, for controller design, we are interested only in regression, hence forth, output y will be assumed to be continuous. The features x are typically multi-dimensional. We denote different dimensions of x by $x_1, x_2, \ldots, x_m$ where each x_i is a scaler. The goal in standard machine learning regression is to identify model f in

$$y = f\left(x_1, x_2, \ldots, x_m\right) \tag{1}$$

given n samples $\{x_1^j, x_2^j, \ldots, x_m^j, y^j\}_{j=1}^{j=n}$. Under this setting, for a new observation defined by $\{x_1^*, x_2^*, \ldots, x_m^*\}$, we can predict $y^* = f\left(x_1^*, x_2^*, \ldots, x_m^*\right)$. Note that it is assumed that all $x_1^*, x_2^*, \ldots, x_m^*$ are known.

It is important to note that this standard machine learning procedure for regression is fundamentally different from using machine learning for control synthesis. In the former, all the features of the model are known, while in the latter some of the features that are the decision (or control) variables must be optimized in real-time for desired performance. To illustrate this, we make two important distinctions for controller design using MPC.

First, model f is dynamical since f captures the dynamical behavior of a physical system. This means that output y_{t+1} at any given time t depends on the output at previous time steps $y_t, y_{t-1}, \ldots, y_{t-\delta}$, where δ is the order of autoregression. To learn a dynamical model, if not all, at least some of the previous outputs $y_{t-1}, y_{t-2}, \ldots, y_{t-\delta}$ are often included in the features. Thus, the regression model is modified to

$$y_{t+1} = f\left(x_1, x_2, \ldots, x_m, y_t, y_{t-1}, \ldots, y_{t-\delta}\right). \tag{2}$$

This goes along the idea of recurrent neural networks [5] and its derivatives like long short-term memory networks [6] which have been widely used for timeseries prediction, the difference being we are specifying the memory dependence explicitly. The order of autoregression δ is a hyperparameter chosen during cross validation.

Second, some of the features in $x_1, x_2, \ldots, x_m$ must be optimized. To understand this, we rewrite (2) in control theory terminology by replacing features with disturbances d, inputs u and their corresponding lags

$$y_{t+1} = f\left(y_t, y_{t-1}, \ldots, y_{t-\delta_y}, d_t, d_{t-1}, \ldots, d_{t-\delta_d}, u_t, u_{t-1}, \ldots, u_{t-\delta_u}\right). \tag{3}$$

In the learning step, we restructure the time series of y, d and u obtained from raw sensor data to create data samples at each time instance in the above format. For optimal decision making, we use model (3) in MPC problem as follows

$$\begin{aligned} \underset{u_t,\ldots,u_{t+N-1}}{\text{minimize}} & \sum_{\tau=0}^{N-1} (y_{t+\tau+1} - y_{\text{ref}})^2 + {u_{t+\tau}}^T R u_{t+\tau} \\ \text{subject to} \;\; & y_{t+\tau+1} = f(y_{t+\tau}, y_{t+\tau-1}, \ldots, y_{t+\tau-\delta_y}, d_{t+\tau}, d_{t+\tau-1}, \ldots, d_{t+\tau-\delta_d}, \\ & \qquad\qquad u_{t+\tau}, u_{t+\tau-1}, \ldots, u_{t+\tau-\delta_u}), \\ & u_{t+\tau} \in \mathcal{U}, \end{aligned} \tag{4}$$

where the constraints hold for all $\tau \in \{0, \ldots, N-1\}$. While (4) is an example of a tracking controller, cost function can be changed depending upon the application. Now the goal has shifted from using (3) for only prediction to optimizing some of the features of the model given the output. Thus, the control problem is essentially an inversion of machine learning regression once model f is learned through standard training procedures. Note that we have not posed any restrictions on f so it can be any nonlinear function in general. In the following sections, we will sepcifically discusss metrits and demerits when f is a random forest, Gaussian process or neural network with the help of case studies on building control. We elaborate on this notion of inversion of machine learning models in the next section listing practical challenges in bridging machine learning and controls.

The above description of control synthesis is a two step process. First a dynamical model is identified using historical data and then the learned model is used in optimization to compute optimal decisions. Another approach to designing a controller is to learn control policies directly using data using reinforcement learning (RL), that is the features are directly mapped into control actions. In this setting, we learn the functional representation of the form

$$u_t = f_r\left(y_t, y_{t-1}, \ldots, y_{t-\delta_y}, d_t, d_{t-1}, \ldots, d_{t-\delta_d}, u_{t-1}, \ldots, u_{t-\delta_u}\right). \tag{5}$$

Model f_r can be any nonlinear function, for example, a neural network trained using deep reinforcement learning. While we have seen many sucessful demonstrations of reinforcement learning, for example, in autonomous driving where convolutional neural network (CNN) are used to map raw pixels from a single front-facing camera directly into steering commands [7], in robotics where control policy for a robotic manipulator is learned directly from camera images [8], reinforcement learning often requires massive amount of data. Further, learning by interaction and real-time experiments when a building is occupied limits the use of reinforcement learning. This two factors combined pose a serious practical challenge in the application of reinforcement learning for building control.

-

- need for cost reduction in order to deploy MPC at scale
- this work explores the use of machine learning based models
- how is machine learning used in a traditional sense?
- what are the requirements for predictive control?
  - predictive dynamical model
  - performance guarantees for control
  - distinction with reinforcement learning

Outline of this article

- economic MPC
- what we don't do

Inversion of machine learning models for control

- challenges with constrained optimization using ML models
- hard because of non-convexities, non-differentiabilities, non-closed form solution, give examples with trees, forests, Gaussian processes, neural networks
- cannot use RL, we want model-based
- traditionally optimization in ML unconstrained
- for control we need constrained optimization
- examples of different applications building will be main focus

Application – building control and demand response

- intro to building control and demand response
- need for model predictive control
  - energy efficiency, energy flexibility $->$ energy savings, cost savings
- so we need models, why is traditional way of modeling hard
  - model capture using historical data
  - change in material properties
  - model heterogeneity
- Practical challenges
  - quality of historical data, need for new experiments, sensor failure
  - computational complexity of control/optimization algorithms, real-time control
  - performance guarantees and robustness
  - model adaptability
  - indicator for deterioration, when to update, use statistics of error in prediction

- A concrete example that describes the modeling and control problem
  - description of building: different types of buildings like RTU, central heating/cooling, impact of thermal inertia
  - goals for modeling types of models to be identified
  - goals for control cost minimization, energy minimization, thermal comfort bounds

Conclusion

References

[1] Francesco Borrelli, Alberto Bemporad, and Manfred Morari. *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[2] Rajesh Rajamani. *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[3] David Sturzenegger, Dimitrios Gyalistras, Manfred Morari, and Roy S Smith. Model predictive climate control of a swiss office building: Implementation, results, and cost–benefit analysis. *IEEE Transactions on Control Systems Technology*, 24(1):1–12, 2016.

[4] Lukas Hewing, Alexander Liniger, and Melanie N Zeilinger. Cautious nmpc with gaussian process dynamics for autonomous miniature race cars. In *2018 European Control Conference (ECC)*, pages 1341–1348. IEEE, 2018.

[5] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[8] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

Example of Sidebar

References

[S1] Random reference number one generated for a sidebar, 2016
[S2] Random reference number two generated for a sidebar, 2016

Author Biography

Insert the author bios here.