Adaptive MPC Model Selection via Machine Learning

Joshua Pulsipher
Department of Chemical Engineering
University of Wisconsin-Madison
pulsipher@wisc.edu

Viswesh Periyasamy
Department of Computer Science
University of Wisconsin-Madison
vperiyasamy@wisc.edu

Abstract

This report details the development of a novel approach to improve the robustness of model predicative control, MPC, schemes using machine learning. Several MPC systems are simulated with different underlying process models to generate training data that is used to train a support vector machine, a support vector machine regressor, and 2 multilayer perceptrons. These trained machine learning techniques are used to rectify simulated MPC model mismatch. Results provide evidence that these machine learning methods, particularly support vector machines and support vector regressors, can be used to dynamically select an appropriate MPC model given a system's response. Also a brief overview of model predictive control is provided for readers without a control background.

1 Introduction

This study applies machine learning techniques to model predictive control, MPC, schemes in order to make them more resilient to disturbances. In particular, this study attempts to evaluate which machine learning algorithms might serve as a potential alternative to established MPC model mismatch resolution algorithms such as moving horizon estimation, MHE. Current algorithms such as MHE have proven to be effective at adjusting MPC models in response to variant process conditions, but are computationally expensive and are difficult to fine tune [1]. A trained machine learning derived classifier such as a support vector machine has the potential to dynamically vary the MPC model at a lower computational cost. This machine learning based MPC approach would permit MPC control schemes to operate at faster frequencies on inherently variant processes and thus enhance the controllability of a given process [2]. Little research has been conducted to evaluate the feasibility of such a method. Lenz et. al. successfully applied deep learning techniques to an MPC cutting robot to adapt its cutting control to different types of vegetables [3]. Zhejing et. al. studied the use of support vector regression in support of an MPC to control a chemical reactor and showed algorithm to be effective for certain systems [4]. Miao et. al. conducted a similar study in a more broad context and showed that support vector regression was comparable to neural network based regression for MPC control [5]. Such studies show that the traditional machine learning algorithms can be integrated with MPC schemes. However, little to no research has been done to compare machine learning techniques in the context of model mismatch. This study will juxtapose a few different machine learning methods to evaluate their respective effectiveness in responding to model mismatch in a generalized MPC simulation environment.

2 A Brief Overview of MPC

Model predictive control is an advanced control technique that is gaining popularity in the chemical industry to control processes that are inherently unstable/difficult to control with traditional PID

controllers, while only employing the current optimized input; the process is then repeated at the controller frequency (typically on the order of 0.5-30 seconds) [6]. This allows MPC schemes to anticipate future conditions and adjust their performance accordingly, something that simple PID controllers can't do [6]. Many MPC applications are computationally expensive to model because of their complexity and nonlinearity, thus linear models are often used to approximate a system's behavior. However, these linear models are commonly limited to narrow operating regimes and quickly induce substantial error if the system deviates from normal operation due to disturbances and/or if the operational conditions are varied [7]. Currently, common practice is to manually retune these linear models when system conditions change. The current state of the art pertains to MHE state estimation schemes. Moving horizon estimation is useful for nonlinear or constrained systems as it presents a stable online optimization scheme, however the online optimization is computationally expensive and only is viable for systems with low control frequencies [7]. An ongoing field of research is designing MPC schemes that can automatically adapt linear model in accordance to varied system conditions.

3 MPC Simulation Framework

This project only considers simulated processes and MPC controllers. To provide a general set of conditions a generic FOPDT model common to the control community was implemented for both the system model and MPC model. A FOPDT model is given by

$$\tau_p \frac{y(t)}{dt} = -y(t) + K_p u(t - \theta_p) \tag{1}$$

where y is the controlled process attribute, u is the controller input, and τ_p, K_p, and θ_p are adjustable process parameters. Equation 1 is used for these tests as the process model and the MPC model. This relatively simplistic approach is taken as the focus of this work is on the novel application of machine learning techniques and not on advanced nonlinear modeling techniques. This simplistic approach also facilitates the analysis of such machine machine learning techniques in a simulation framework that is analytically well-understood. The FOPDT parameters are set to different values to represent different models. In particular, the value of K_p is varied to distinguish different possible models. This assumption reflects an important class of control systems that exhibit transient process gains and usually are controlled with the aid of MHE methods.

All of the simulations are carried out via APMonitor which is an open source optimization software (that interfaces with Python) specifically designed for MPC implementation in simulated and physical systems. A Python script is used to link an APMonitor process simulation and an APMonitor MPC controller in order to simulate a process and an MPC controller simultaneously. The script has APMonitor simulate the process for 1 time step then adds some random noise (generated from a uniform distribution) to the measured process variable value to simulate measurement error inherent with a real process. This measurement is then fed into the APMonitor MPC controller which optimizes the control input value needed to drive the system toward a certain setpoint which is specified in the Python script. This optimized control variable value is returned to Python and fed into the process model. This simulation algorithm is then looped over the total number of time steps.

This modular simulation framework easily facilitates the addition of machine learning based classification and/or regression schemes within the main Python loop, such that the MPC FOPDT model can be dynamically selected.

4 Classification and Regression Training

4.1 Training Data Acquisition

A set of 3 model configurations are considered in this study so that a reasonable amount of training data can be collected. Each model specifies the values of τ_p and θ_p to be 10 and 0 respectively. The distinguishing characteristic parameter between the models is the process gain, K_p, which typically is the parameter most associated with variant process model behavior in nonlinear chemical processes as discussed in Section 3. The process gain is set to be 1, 5, and 10 respectively for each model.

Training data is collected from running the MPC simulation scheme described in section 3 over 160 time steps during which the controller setpoint is varied from 10 to 15, from 15 to 5, and then from 5 to 20 to promote a diversity of dynamic data. The measured process variable, the controller input, and the controller's prediction of the process's response in 5 time steps are recorded for each time step. This simulation is carried out for every possible combination of MPC models and process models, accounting for 9 simulations in total. All of the recorded data was saved to be configured as a training data set. This training data does have random elements to it since the process variable is randomly varied with noise. However, it is not necessarily independent and individually distributed since the same control simulation is considered for each different model combination. This is done to limit the computation time as this simulation is very computationally intensive. More $i.i.d$ data could be obtained if more powerful computational resources were available. Admittedly this places a limitation to the scope of this study and future work will include gathering a more randomized set of training data from a wider range of conditions. The methodology used to train various machine algorithms is described in detail below.

4.2 Support Vector Machine Training

There are 3 possible models that the process simulator is allotted to use and thus only 3 classes of model the MPC controller should choose from. A total of 3 linear support vector machines are trained to classify which of the 3 possible models the MPC should select. Meaning that 1 linear SVM is trained for each possible MPC model state such that it can determine if the current model is appropriate or specify which of the other 2 models should be used. The SVMs are designed to accept a feature vector with 3 measurements obtained from the training data. The first feature is the value:

$$x_1^{(t)} = \left| y_p^{(t-5)} - y_m^{(t)} \right| \tag{2}$$

where $y_p^{(t-5)}$ is the MPC's predicted value of y at time step $t-5$, and $y_m^{(t)}$ is the actual measured value of y at time t. This feature gauges how well the MPC model predicts system's response. This feature exacts a delay of 5 time steps for prediction because the incremental prediction at time $t-1$ is less indicative of model mismatch since the respective differences will always tend to be small. Thus this delay necessitates the removal of the initial 5 data points of each training data set. The second feature is defined as:

$$x_2^{(t)} = \left| y_m^{(t)} - y_m^{(t-1)} \right| \tag{3}$$

which provides a value that has a moderate correlation to the gain of the model, K_p, and therefore helps to differentiate between the classes/models. The third feature is that of the process input variable:

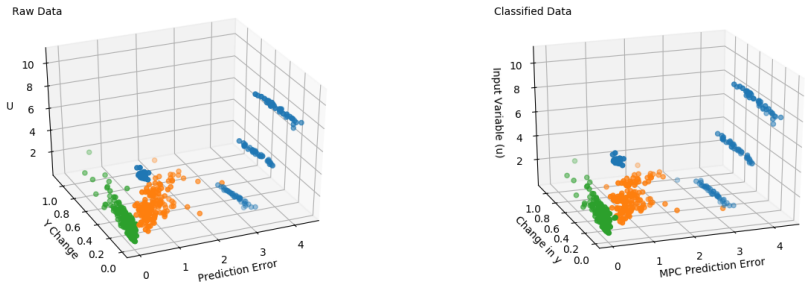
$$x_3^{(t)} = u^{(t)} \tag{4}$$

which complements the second feature since a certain change in y will require a different amount of input, u, depending on the underlying process gain.

Figure 1 shows the classification of the training data pertaining to the second linear SVM classifier which takes the MPC model to have a K_p value of 5. The data appears to be easily separable and the linear classifier achieves a stable prediction rule with minimal error on the training set. An error of 1.40% is reported. A Gaussian kernel was also used for comparison, but it yielded less generalizable decision boundaries and produced a comparable error over the training data of 1.14%.

4.3 Support Vector Regression Training

Training the SVM for regression is handled almost identically to the classification approach. The same feature vector $\boldsymbol{x^{(t)}}$ is fed into the SVM but instead of labels, the model's gain value K is used as output. In the case of the 3 possible models in this case study, the possible outputs are $K \in \{1, 5, 10\}$. This contrasts the support vector classifier's (SVC) output of the predicted model index corresponding to the set gain values. An epsilon value of $\epsilon = 0.1$ is used to train the support vector regressor (SVR)



(a) Unclassified raw process data.

(b) Data classified with a linear SVM.

Figure 1: The linear SVM classification of different process models with an MPC model gain of 10.

which controls the epsilon-tube that bounds penalties for training loss. All other parameters (e.g. the linear kernel and and penalty parameter $C = 1.0$) remain unchanged from the SVC.

In the setting of regression, all of the training data is concatenated into 1 set and only 1 SVR is trained (contrasting the 3 classifiers that are trained with separate MPC base models). The separation of classifiers by model index is not necessary for regression because the non-discrete output can be directly used to adjust the MPC model.

4.4 Multilayer Perceptron Training

The deep learning approach is built using a multilayer perceptron (MLP). Both the classifier and regressor use the same configuration with a rectified linear unit (ReLU) activation function and the Adam method of stochastic gradient descent. Additionally, an initial learning rate of $\gamma = 0.001$ is used along with an adaptive procedure which reduces the current learning rate by 80% for every 2 consecutive epochs with no decrease in training loss. Finally, a penalty parameter of $\alpha = 0.0001$ is used for the Euclidean norm regularization term. The MLP classifier and regressor are trained using the same methodologies described above. Thus a total of 3 MLP classifiers are trained where 1 trained classifier is used for each possible MPC model; and 1 MLP regressor is trained from the concatenated set of training data.

5 Adaptive MPC Framework

5.1 Classification Framework

The supervised learning models are used to construct an adaptive MPC framework which adjusts the process gain based on real-time feedback. In the classification domain, the SVC outputs a suggested model at each time step. However, to combat sudden perturbations and make the SVC more robust to noise, a sliding window approach is used to enforce a majority consensus. Specifically, the model is initially set to use the median process gain $K = 5$ and then begins collecting a majority vote of the last 10 classifications for $t \geq 5$. For every setpoint (i.e. desired output y) change in the simulation, the voting data is ignored for the first 5 time steps and resumes after. The entire framework operates in an identical fashion with the MLP classifier in place of the SVC as the supervised learning model.

A model mismatch threshold of $x_1^{(t)} > 0.2$ is introduced to impose a stability constraint on the overall model - this mismatch correlates to the process gain value K_p not agreeing between the MPC model and the process model. The SVC's prediction value is used and stored only if $x_1^{(t)}$ exceeds the mismatch threshold; if the predicted y from 5 previous time steps is not largely different than the current measured value of y, then the previous iteration's majority vote is used again. This protects the framework from over-responding to small differences between predicted and measured output.

Additionally, a classifier threshold is enforced to prevent model oscillation. As the SVC changes its predicted model to reduce error between the predicted and measured output, the drastic reduction in

error will cause the SVC to incorrectly re-choose the original model that caused the large error in the first place. This will induce a cycle of oscillation between the 2 models. To rectify this problem, 3 classifiers are trained on different sets of data and a threshold of $x_1^{(t)} < 0.2$ is used to determine whether it is necessary to swap classifiers.

5.2 Regression Framework

The same adaptive MPC Framework is constructed using regression with some minute changes to achieve similar results. The sliding window approach used in classification is carried over for regression in the form of a moving average of regressed K values - this is analogous to the majority vote in the discrete domain. The sliding window is again set to the median process gain and averaged K values are similarly recorded 5 time steps after every setpoint change. Much like the SVC and MLP classifier, the SVR and MLP regressor can be used interchangeably with comparable results.

While the model mismatch threshold remains necessary for the SVR and MLP regressor, the classifier threshold does not apply in this framework. The classifier's dilemma of model oscillation stems from the sudden leaps in process gain from switching models. With only 1 regressor trained on the entire training set and the capability to slowly approach a desired process gain, this problem is avoided.

6 Results

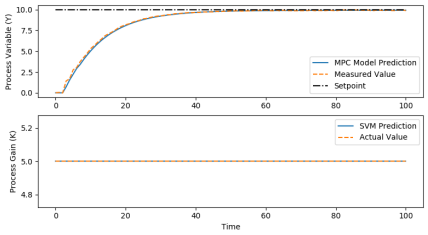
Case studies spanning 3 categories are examined using both classifiers and regressors, with the categories being start up, process disturbance, and variant control. Start up refers to a process's initial rise to the desired setpoint at steady state conditions from an offline state. This scenario is highly transient in nature and generally is one of the control regimes most susceptible to model mismatch. Process disturbance refers to an external influence, not explicitly accounted for in the control model, that drives the process away from a setpoint. Thus a robust control scheme is one that rejects disturbances and returns to the process setpoint, adapting the control model if necessary. Variant control denotes a multitude of disturbances and setpoint changes to see how the system behaves in fluctuating settings. This doesn't denote realistic control conditions, but does represent a worst case scenario to test the effectiveness of the proposed MPC machine learning methods. Each supervised learning model is tested in each case study and results are shown below, with each subfigure plotting the measured and predicted output compared to setpoint (top), and the predicted and the actual process gain (bottom).

6.1 Start Up

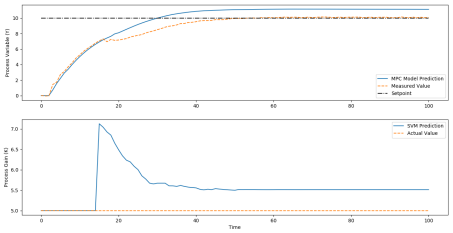
One start up case study is performed for each possible underlying process model for a total of 3 start up case studies. In each case the the process initialized at an offline state and then a process variable setpoint of 10 is specified. Figure 2 shows a comparison between the SVM classifier and the SVM regressor corresponding to each start up case.

The results indicate that the SVM classifiers are able to achieve setpoint equilibrium relatively quickly and are able to determine the correct process gain value once the moving window is fully populated. However, while the regressors are able to drive the measured process variable relatively close to the setpoint, the predicted process variable value corresponding to the MPC model and the MPC process gain maintain a constant offset from the true values. Although the measured process variable achieves the setpoint, the offset is problematic because it diminishes an MPC's stability. This behavior is likely due to the regressors being unable to deviate from local minima. A similar problematic behavior can be observed with MHE parameter estimation schemes when the measured variable settles around the setpoint [1]. Notably, the regressing method demonstrates this instability in case 3 where the process gain is set to its lowest value of 1 which exacerbates the regressor's unstable behavior.

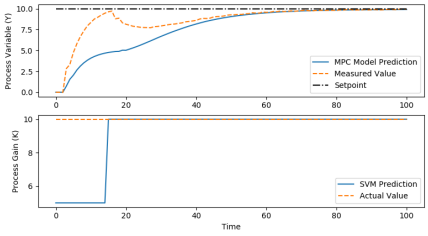
The MLP classifiers and regressors performed in manner identical to the SVM classifiers and regressors, meaning that Figure 2 is equivalently indicative of the MLP performance. Thus there is no clear advantage to using the more complex MLP methods over the simpler support vector methods.



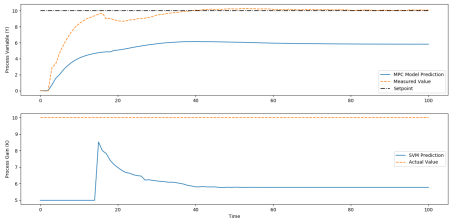
(a) SVM classification with initial $K = 5$.



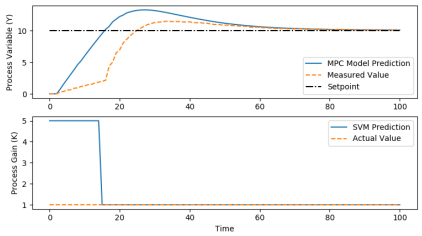
(b) SVM regression with initial $K = 5$.



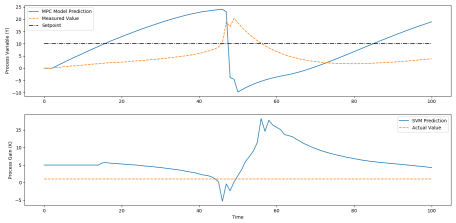
(c) SVM classification with initial $K = 10$.



(d) SVM regression with initial $K = 10$.



(e) SVM classification with initial $K = 1$.



(f) SVM regression with initial $K = 1$.

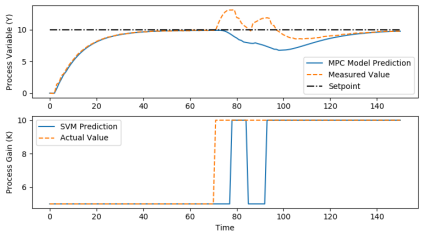
Figure 2: A comparison of SVM classification and regression for processes working to attain an operating setpoint of 10 with a fixed process gain value.

6.2 Process Disturbance

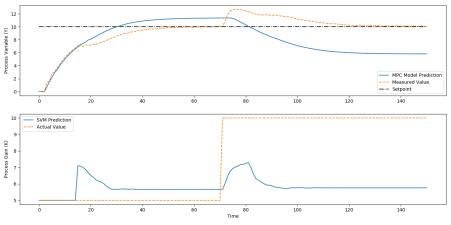
For process disturbance simulations, 2 scenarios are examined. In each case the process is initialized from start up and the underlying process gain is changed halfway through the simulation. The first case study undergoes a process gain change from $K = 5$ to $K = 10$ and the second changes the gain from $K = 10$ to $K = 1$. Results are depicted in Figure 3.

Each plot shows the models achieving equilibrium after start up and then reacting to the disturbance with the change in process gain. The classifiers perform well in these settings and show a return to equilibrium in both settings. Regression, however, doesn't perform as strongly. In the first case, the measured output is still driven back to the setpoint but the same undesirable constant offset is observed between the predicted and measured values. However, the sharp decrease in gain associated with case 2 highlights the SVR unstable characteristics as it predictions oscillate aggressively. Once again this correlates to the low process gain that exacerbates the regressor's unstable behavior. Again, this behavior can be analogous to poorly tuned MHE methods.

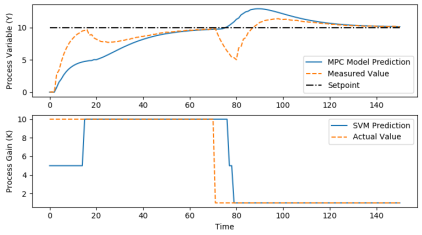
The MLP classifiers and regressors again responded in fashions that are identical to the support vector methods and can be equally represented by Figure 3. Thus the MLPs don't appear to improve performance by any appreciable margin.



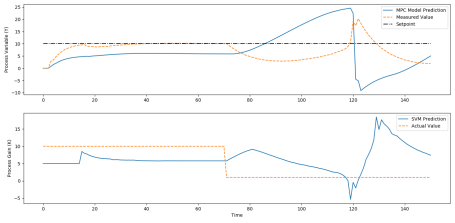
(a) SVC with $K = 5$ to $K = 10$.



(b) SVR with $K = 5$ to $K = 10$.



(c) SVC with $K = 10$ to $K = 1$.

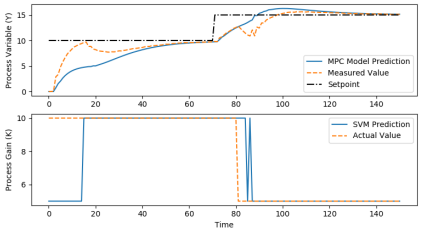


(d) SVR with $K = 10$ to $K = 1$.

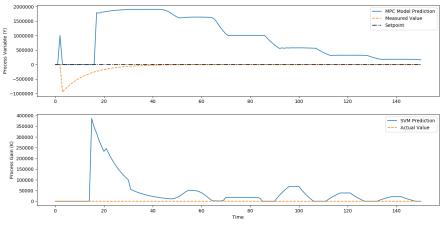
Figure 3: A comparison of SVM classification and regression for stable processes with disturbances in the form of process gain changes introduced.

6.3 Variant Control

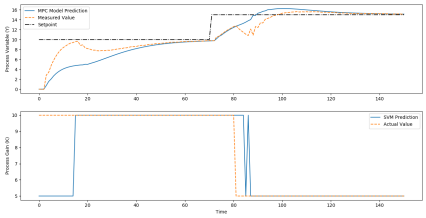
A more complicated situation is considered for the variant control case. The process is initialized from a start up condition with a process gain of 10 and a setpoint of 10. The setpoint is raised to a value 15 and the process gain is changed suddenly to 1. This represents a drastic change in model behavior during a setpoint change. Figure 4 summarizes the performance of each of the 4 methods tested.



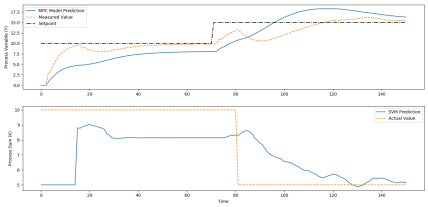
(a) SVM classification.



(b) Support vector regression.



(c) MLP classification.



(d) MLP regression.

Figure 4: A comparison of classification and regression for process whose model changes when the operating setpoint is raised.

Both classification methods quickly address the model mismatch and are able to achieve steady behavior about the new setpoint. Although a little fluctuation occurs at the process gain change. The SVM regressor diverges to a very large K_p value before slowly settling down to a more a appropriate value. However, this behavior is not observed with the MLP regressor which is able to respond adequately without the large diversion. Thus the MLP regressor appears to be more stable than the SVM regression method. Moreover, the SVM classification methodology is able to best match the process model and behaves as well as the more complex MLP classifier.

7 Conclusions and Future Work

This study has juxtaposed the effectiveness of several common machine learning approaches within the realm of MPC model mismatch. The regression-based methods considered are prone to model offset and instability that is exacerbated at low process gain values. The classification methods tend to more readily reject model offset and appear to be fairly resilient to disturbances over sufficient time scales. The support vector and neural network methods considered in this study also behave similarly, which indicates that neural networks might add unnecessary complexity. Future efforts will further test the resiliency of support vector methods in combination with more complex nonlinear process models that exhibit variations over appropriate time scales. Efforts will also be made to obtain training data that is more randomized.

References

[1] Moritz Diehl, Hans Joachim Ferreau, and Niels Haverbeke. Efficient numerical methods for nonlinear mpc and moving horizon estimation. *Nonlinear model predictive control*, 384:391–417, 2009.

[2] Eduardo F Camacho and Carlos Bordons Alba. *Model predictive control*. Springer Science & Business Media, 2013.

[3] Ian Lenz, Ross A Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, 2015.

[4] BAO Zhejing, PI Daoying, and SUN Youxian. Nonlinear model predictive control based on support vector machine with multi-kernel** supported by the state key development program for basic research of china (no. 2002cb312200) and the national natural science foundation of china (no. 60574019). *Chinese Journal of Chemical Engineering*, 15(5):691–697, 2007.

[5] Qi Miao and Shi-Fu Wang. Nonlinear model predictive control based on support vector regression. In *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, volume 3, pages 1657–1661. IEEE, 2002.

[6] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practice—a survey. *Automatica*, 25(3):335–348, 1989.

[7] James B Rawlings. Moving horizon estimation. *Encyclopedia of Systems and Control*, pages 1–7, 2013.