Statistical Machine Learning in Model Predictive Control of Nonlinear Processes: An Overview of Recent Results

Mohammed S. Alhajeri [a,b], Aisha Alnajdi[d,e], Zhe Wu [c] and Panagiotis D. Christofides[1][a,d]

[a] Department of Chemical and Biomolecular Engineering, University of California, Los Angeles, CA, 90095-1592, USA.

[b] Department of Chemical Engineering, Kuwait University, P.O.Box 5969, Safat 13060, Kuwait

[c] Department of Chemical and Biomolecular Engineering, National University of Singapore, 117585, Singapore

[d] Department of Electrical and Computer Engineering, University of California, Los Angeles, CA 90095-1592, USA.

[e] Department of Electrical Engineering, Kuwait University, P.O.Box 5969, Safat 13060, Kuwait

Abstract

Machine learning (ML) has received an increased level of attention in modeling of nonlinear systems in recent years. While supervised learning methods such as recurrent neural networks (RNNs) have demonstrated their capability in fitting training data, a fundamental challenge that hinders the implementation of ML modeling to real-word chemical processes is to characterize the generalization ability on previously unseen data. This article summarizes our recent work that utilizes statistical learning theory to develop generalization error bounds for a variety of machine learning models that are developed to model nonlinear processes and barrier functions, and discusses closed-loop stability properties in the context of ML-based model predictive control (MPC).

Keywords

Machine learning, Generalization error, Recurrent neural networks, Physics-informed neural networks, Long-short term memory networks, Model predictive control, Nonlinear systems, Barrier functions, Operational safety.

Introduction

Machine learning models are developed to find statistical patterns in a training set that generalize to unseen data outside the training set. While the training error of machine learning models can be made sufficiently small as demonstrated in many computer science research work, the generalization error that characterizes the model's ability to adapt properly to previously unseen data of the same distribution remains a major challenge that should receive more attention. Statistical learning theory, a framework for machine learning drawing from the fields of statistics and functional analysis, provides an efficient tool to analyze the generalization performance of ML models. A lot of research efforts have been made in the computer science community to derive theoretical generalization error bounds for different ML models, such as feedforward neural networks (FNN) and recurrent neural networks (RNN) (Bartlett et al., 2017; Golowich et al., 2018; Cao and Gu, 2019). However, as these works primarily focused on classification problems using single-output networks, how to adapt the results to ML modeling of multi-input multi-output nonlinear processes in the context of regression problems remains questionable.

In addition, machine learning models have been incorporated in the designs of model predictive control (MPC) to provide state predictions in the real-time optimization of control actions (Wu et al., 2019). Despite the success of applications of ML-based MPCs to many chemical processes, theoretical study of closed-loop stability properties based on the generalization ability of ML models is still in its infancy. Based on the above, this work summarizes our recent research on the generalization error of ML models and closed-loop stability of ML-based MPC using statistical machine learning theory. A goal of this work is to highlight the derivations of generalization error bounds for a variety of neural networks that are commonly used to model nonlinear dynamic systems, and provide probabilistic closed-loop stability analysis for ML-based MPC. Additionally, we also discuss the generalization ability of neural networks for the barrier functions that are used to ensure process operational safety in control of chemical processes.

Class of Nonlinear Process Systems

We consider a class of multi-input multi-output (MIMO) nonlinear continuous-time systems represented by the following state-space form:

$$\dot{x} = F(x,u) := f(x) + g(x)u \tag{1}$$

[1] Corresponding author. Email: `pdc@seas.ucla.edu`.

where $x = [x_1, ..., x_n]^T \in \mathbf{R}^n$ is the state vector, $y = [y_1, ..., y_q]^T \in \mathbf{R}^q$ is the output vector, and $u = [u_1, ..., u_m]^T \in \mathbf{R}^m$ is the manipulated input vector. $F(x,u)$ represents a nonlinear vector function of x and u which is assumed to be sufficiently smooth. The constraints on control inputs are given by $u \in U := \{u_i^{min} \leq u_i \leq u_i^{max}\}$. The functions $f(\cdot)$, and $g(\cdot)$ are nonlinear vector and matrix functions of $n \times 1$ and $n \times m$ dimensions, respectively. We assume $f(0) = 0$ such that the origin is a steady-state of Eq. 1. A stabilizing feedback controller is assumed to exist for the nonlinear system of Eq. 1 under which the origin is rendered exponentially stable for the states in an open neighborhood around the origin D. Based on the stabilizability assumption, we assume there exists a Lyapunov function $V(x)$ and a Lyapunov-based controller $u = \Phi(x) \in U$ such that an estimate of initial states D that satisfy $\frac{\partial V}{\partial x} F(x,u) < 0$ can be characterized. The closed-loop stability region for the nonlinear system of Eq. 1 is then determined as follows: $\Omega_\rho := \{x \in D \mid V(x) \leq \rho\}$, $\rho > 0$, which is a level set of $V(x)$ within D. Throughout this manuscript, we use $|\cdot|$ to denote the Euclidean norm of a vector, and set subtraction is denoted by "$\backslash$", i.e., $A \backslash B := \{x \in \mathbf{R}^n \mid x \in A, x \notin B\}$. Additionally, there exist positive constants M_F, L_x, L'_x such that the following inequalities hold for all $x, x' \in \Omega_\rho$ and $u \in U$:

$$|F(x,u)| \leq M_F, \ \ |F(x,u) - F(x',u)| \leq L_x |x - x'| \tag{2a}$$

$$\left| \frac{\partial V(x)}{\partial x} F(x,u) - \frac{\partial V(x')}{\partial x} F(x',u) \right| \leq L'_x |x - x'| \tag{2b}$$

Generalization error of ML models

Generalization error measures the machine learning model's ability to adapt properly to new, previously unseen data, which is drawn from the same distribution as the one used to train the model. A theoretical analysis of generalization error is of significant importance as it provides a fundamental understanding on how good the model performs on unseen data that will be collected in real-world systems. In this section, we will discuss the derivations of generalization errors using statistical learning theory for several popular neural networks that are often used to model the nonlinear dynamic system of Eq. 1. Before we present the results on generalization error bounds, we first introduce the definitions of generalization error as follows. Given a data distribution D, and a function h that predicts y (output) based on x (input), the generalization error is given by $\mathbf{E}[L(h(x),y)] = \int_{X \times Y} L(h(x),y)\rho(x,y)dxdy$, where $\rho(x,y)$ denotes the joint probability distribution for x and y, and Y and X represent the vector space for all possible outputs and inputs, respectively. $L(\cdot,\cdot)$ is the loss function, e.g., mean squared error (MSE) for regression problems. Since the distribution may be unknown, the following empirical error is often used as an approximation measure for the generalization error: $\hat{\mathbf{E}}_S[L(h(x),y)] = \frac{1}{m}\sum_{i=1}^{m} L(h(x_i),y_i)$, where $S = (s_1, ..., s_m)$, $s_i = (x_i, y_i)$ includes m data samples drawn from the data distribution D.

Recurrent nerual networks (RNNs)

As discussed in the introduction, RNN models are a powerful tool for modeling dynamic systems using time-series data. To simplify the discussion, we consider a single-hidden-layer RNN model with the following form to approximate the nonlinear dynamics of Eq. 1.

$$\mathbf{h}_t = \sigma_h(U\mathbf{h}_{t-1} + W\mathbf{x}_t), \ \ \mathbf{y}_t = \sigma_y(Q\mathbf{h}_t) \tag{3}$$

where the RNN input and output at the t^{th} time step are denoted by $\mathbf{x}_t \in \mathbf{R}^{r_x}$ and $\mathbf{y}_t \in \mathbf{R}^{r_y}$, respectively. $\mathbf{h}_t$ denotes the hidden state, and W, U, and Q are the weight matrices connecting different layers. The (nonlinear) activation functions are denoted by σ_h and σ_y. Specifically, σ_h is often chosen to be a nonlinear activation function that may take different forms (e.g, *tanh* or *ReLU*), while σ_y typically uses a linear element-wise activation function for regression problems. Without loss of generality, we have the following assumptions for the development of RNN models: 1) the RNN inputs are bounded, i.e., $|\mathbf{x}_{i,t}| \leq B_X$, for all $i = 1, ..., m$ and $t = 1, ..., T$, 2) the Frobenius norms of the weight matrices are bounded, i.e., $\|W\|_F \leq B_{W,F}, \|Q\|_F \leq B_{Q,F}, \|U\|_F \leq B_{U,F}$, 3) all the datasets (i.e., training, validation, and testing) are drawn from the same distribution, and 4) σ_h is a 1-Lipschitz continuous activation function, and is positive-homogeneous in the sense that $\sigma_h(\alpha z) = \alpha\sigma_h(z)$ holds for all $\alpha \geq 0$ and $z \in \mathbf{R}$. Consider a hypothesis class $\mathcal{H}$ of RNN models $h(\cdot)$ that map a d_x-dimensional input $\mathbf{x} \in \mathbf{R}^{d_x}$ to a d_y-dimensional output $\mathbf{y} \in \mathbf{R}^{d_y}$. The predicted output of the RNN model and the loss function are denoted by $\mathbf{y}_t = h(\mathbf{x}_t)$ and $L(\mathbf{y}_t, \bar{\mathbf{y}}_t)$, respectively, where $L(\mathbf{y}, \bar{\mathbf{y}})$ calculates the squared difference between the predicted output $\mathbf{y}$ and the true output $\bar{\mathbf{y}}$. The following lemma gives the generalization error bound for a general class of machine learning models.

Lemma 1 (c.f. Theorem 3.3 in Mohri et al. (2018))**.** *Let $\mathcal{H}$ be the hypothesis class of ML models that map $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_t\} \in \mathbf{R}^{d_x \times t}$ (i.e., the first t-time-step inputs) to $\boldsymbol{y}_t \in \mathbf{R}^{d_y}$ (i.e., the t-th output), and $\mathcal{G}_t$ be loss function set with $\mathcal{H}$.*

$$\mathcal{G}_t = \{g_t : (\boldsymbol{x}, \bar{\boldsymbol{y}}) \to L(h(\boldsymbol{x}), \bar{\boldsymbol{y}}), h \in \mathcal{H}\} \tag{4}$$

where $\bar{\boldsymbol{y}}$ and $\boldsymbol{x}$ are the true output vector and the input vector of ML model, respectively. Then, given a dataset consisting of m i.i.d. data samples, the inequality below holds in probability for all $g_t \in \mathcal{G}_t$ over the data samples $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^T$, $i = 1, ..., m$:

$$\mathbf{E}[g_t(\boldsymbol{x}, \boldsymbol{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\boldsymbol{x}_i, \boldsymbol{y}_i) + 2\mathcal{R}_S(\mathcal{G}_t) + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{5}$$

Eq. 5 demonstrates that the upper bound for the generalization error depends on the training error (first term), the Rademacher complexity of $\mathcal{G}_t$ (second term), and a function of the samples size m and the confidence δ. Therefore, to derive a generalization error bound for RNN models, an upper bound for the Rademacher complexity of RNN hypotheses needs to be developed. The following lemma was developed in Wu et al. (2021) to show the generalization error bound for RNN models.

Lemma 2 (c.f. Theorem 1 in Wu et al. (2021))**.** *Given a dataset $S = (\boldsymbol{x}_{i,t}, \boldsymbol{y}_{i,t})_{t=1}^T$ with i.i.d. data samples, $i = 1, ..., m$, and the L_r-Lipschitz loss function class $\mathcal{G}_t$ associated with the RNN function class $\mathcal{H}_t$ that predicts outputs at the t-th* *time step, with probability at least $1-\delta$ over S, the following inequality holds for the RNN models:*

$$\mathbf{E}[g_t(\boldsymbol{x},\boldsymbol{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\boldsymbol{x}_i,\boldsymbol{y}_i)$$
$$+O\left(L_r d_y \frac{MB_X(1+\sqrt{2\log(2)t})}{\sqrt{m}}\right)+3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} \tag{6}$$

where $M = \frac{1-(B_{U,F})^t}{1-B_{U,F}} B_{W,F}B_{Q,F}$, and B_X is the upper bound for RNN inputs. d_y is the RNN output dimension.

With respect to the applicability of the generalization error bound of Theorem 1, in Wu et al. (2021), we provided a detailed computational study demonstrating the practicality and usefulness of the error bounds in the context of a chemical reactor example. Specifically, we successfully demonstrated the impact of sample size for training, network configuration, depth, training and tuning on the error bounds in both open-loop and closed-loop scenarios under MPC.

Physics-informed RNNs

While standard RNNs do not consider any domain-specific knowledge in model development and generally use fully-connected layers to capture input-output relationship using the given training dataset, it has been demonstrated in Wu et al. (2020) that a priori process structural knowledge can be utilized to improve RNN performance by developing a partially-connected architecture. Fig. 1 shows the difference between fully-connected and partially-connected RNNs, from which it can be observed that the connection between some neurons is removed in a partially-connected structure to resemble the underlying input-output relationship from a priori process structural knowledge. Partially-connected RNNs can be used to model a multiple-unit process in which upstream units affect downstream units but not in the opposite direction. For example, consider the nonlinear system of Eq. 1 for which the input vector u_1 affects the state x_1 only, and both u_1 and u_2 affect the state x_2, where $x = [x_1,\ x_2] \in \mathbf{R}^n$ and $u = [u_1 \in \mathbf{R}^{m_1},\ u_2 \in \mathbf{R}^{m_2}] \in \mathbf{R}^m$, $m_1+m_2 = m$. Wu et al. (2020) demonstrates that by using partially-connected architecture, the number of weight parameters can be significantly reduced to achieve a desired model accuracy compared to a fully-connected RNN model. Additionally, in Alhajeri et al. (2022), an Aspen simulation study of two CSTRs in series was carried out to demonstrate that the MPC using partially-connected RNN models achieved better closed-loop performances with a reduced computation time. To better understand the benefits of partially-connected RNNs in terms of higher modeling accuracy, a theoretical analysis of generalization error needs to be developed.

In a partially-connected structure, the connections between input and output should be carefully designed to be consistent with the a priori process structural knowledge. In particular, as shown in Fig. 1, u_2 does not affect x_1, and therefore, the weights associated with the links between u_2 and x_1 (dashed lines in Fig. 1) are assigned to be zero (i.e., $w_{i,j} = q_{j,l} = 0$). Since the Frobenius norm of matrix A is expressed as the square root of the matrix trace of $AA^{(H)}$, where $A^{(H)}$ is the conjugate transpose, more zero entries in the weight matrices will lead to lower bounds on their Frobenius norms (i.e., smaller $B_{W,F}$ and $B_{Q,F}$). As a result, a lower generalization error bound can be derived due to a smaller M (M is the product of the RNN weight matrices bounds in Eq. 6). By incorporating process structural knowledge into the development of partially-connected RNN models, the complexity of RNN hypothesis class is reduced compared to fully-connected RNNs, which leads to a tighter bound on the Rademacher complexity. Additionally, by revealing the correct direction for RNNs to find the optimal weight parameters, the training error (the first term in Eq. 6) is more likely to be minimized using the same hyperparameters (i.e., the number of layers and neurons) and the same training set of m i.i.d. data samples.

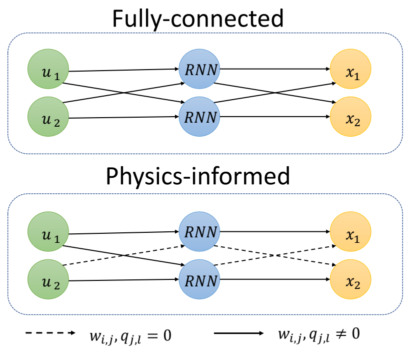

Figure 1: Schematic of standard fully-connected and physics-informed RNN structures.

Long short-term memory (LSTM)

LSTM networks are a variant of RNNs, and have been widely used to make predictions based on time series data, since they can address the vanishing gradient problem that is often encountered when training traditional RNNs. The LSTM is composed of a cell, an input gate, an output gate, and a forget gate, and is formulated by the following equations:

$$f_t = \sigma(W_f\mathbf{x}_t + U_f h_{t-1}) \tag{7a}$$
$$i_t = \sigma(W_i\mathbf{x}_t + U_r h_{t-1}) \tag{7b}$$
$$o_t = \sigma(W_o\mathbf{x}_t + U_o h_{t-1}) \tag{7c}$$
$$\tilde{c}_t = \tanh(W_c\mathbf{x}_t + U_c h_{t-1}) \tag{7d}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{7e}$$
$$h_t = o_t \odot \tanh(c_t) \tag{7f}$$

where $W_f, W_i, W_o, W_c \in \mathbf{R}^{d_h\times d_x}$ are the weights associated with the inputs, and $U_f, U_i, U_o, U_c \in \mathbf{R}^{d_h\times d_h}$ are the weights associated with the hidden states. $i_t, f_t, o_t \in \mathbf{R}^{d_h}$ represent the input, forget ,and output gates, respectively. $c_t \in \mathbf{R}^{d_h}$ is the cell state, and $\tilde{c}_t$ is the cell integrated with the input gate. σ is the nonlinear activation function *sigmoid*, and *tanh* is the hyperbolic tangent function. The output at time t is $\mathbf{y}_t = \sigma_{\mathbf{y}}(Qh_t)$. The following lemma derives the generalization error for LSTM-RNNs:

Lemma 3. *Let $\mathcal{G}_t$ be the family of loss functions associated with the hypothesis class $\mathcal{H}_t$ of vector-valued functions that map the LSTM-RNN inputs to the LSTM-RNN output* *at the t^{th} time step. Given a set of m i.i.d. data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}, i = 1, \ldots, m$, with probability at least $1-\delta$ over S, the following inequality holds for LSTM-RNN:*

$$\mathbb{E}[g_t(\mathbf{x},\mathbf{y})] \leq \frac{1}{m}\sum_{i=1}^{m} g_t(\mathbf{x}_i,\mathbf{y}_i) + O\left(L_r d_y \frac{\left(\sqrt{2}+1\right)M}{\sqrt{m}}\right) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}} \tag{8}$$

where $M = B_Q B_{W_c} B_{\mathbf{x}} \frac{1-\beta^t}{1-\beta}$, $\|Q\|_{1,\infty} \leq B_Q$, $\|W_c\|_F \leq B_{W_c}$, $\|x_{i,t}\|_2 \leq B_{\mathbf{x}}$ for all $i = 1, \ldots, m$, $t = 1, \ldots, T$, and $\beta = 1 + B_{U_c}$ with $\|U_c\|_F \leq B_{U_c}$.

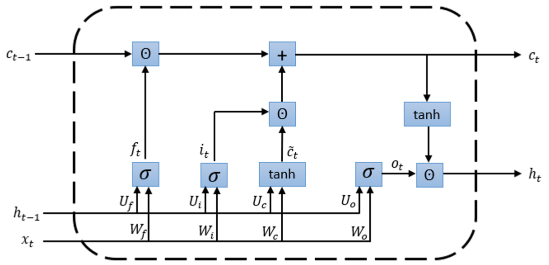

Figure 2: Schematic of an LSTM-RNN structure.

ML-based MPC

In this section, the ML models are incorporated into a Lyapunov-based model predictive controller (LMPC) to predict state evolution of the system of Eq. 1. Specifically, the optimization problem of LMPC using RNN/LSTM models is given as follows: (Wu et al. (2019))

$$\mathcal{J} = \min_{u \in S(\Delta)} \int_{t_k}^{t_{k+N}} (L_t(\tilde{x}(t), u))dt \tag{9a}$$

$$\text{s.t.} \quad \dot{\tilde{x}}(t) = F_{nn}(\tilde{x}(t), u(t)) \tag{9b}$$

$$u(t) \in U, \ \forall\, t \in [t_k, t_{k+N}) \tag{9c}$$

$$\tilde{x}(t_k) = x(t_k) \tag{9d}$$

$$\dot{V}(x(t_k), u) \leq \dot{V}(x(t_k), \Phi_{nn}(x(t_k))), \text{ if } x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}} \tag{9e}$$

$$V(\tilde{x}(t)) \leq \rho_{nn}, \ \forall\, t \in [t_k, t_{k+N}), \text{ if } x(t_k) \in \Omega_{\rho_{nn}} \tag{9f}$$

where $S(\Delta)$ denotes a set of piecewise constant functions with sampling period Δ, $\tilde{x}$ is the state trajectory predicted by the RNN-LSTM model, and N is the number of sampling periods in the prediction horizon. $L_t(\tilde{x}(t), u)$ is the objective function and is typically designed in a quadratic form, e.g., $L_t(\tilde{x}, u) = \tilde{x}^T Q \tilde{x} + u^T R u$, such that the minimum is achieved at the steady-state. The time-derivative of the Lyapunov function V is denoted by $\dot{V}(x, u)$ in Eq. 9e, i.e., $\dot{V}(x,u) = \frac{\partial V(x)}{\partial x}(F_{nn}(x,u))$, where F_{nn} represents the RNN of Eq. 3 or the LSTM model of Eq. 7. The MPC optimization problem is to minimize the integral of $L_t(\tilde{x}(t), u(t))$ of Eq. 9a, which represents the cost function over the prediction horizon, while satisfying the constraints of Eqs. 9b–9f. The LMPC computes the optimal input sequence $u^*(t)$ and delivers the first control signal $u^*(t_k)$ to the system to be implemented for the following sampling period. At the next sampling time, the LMPC receives new state measurement and will be solved again.

The RNN/LSTM model is used to forecast the evolution of the closed-loop state trajectory $\tilde{x}(t_k)$, and the initial conditions are updated according to Eq. 9d, where $x(t_k)$ is the current state measurement. The input constraints are given in Eq. 9c, and they are imposed over the entire prediction horizon. The constraints of Eqs. 9e-9f are designed to ensure the stability of the closed-loop system. Specifically, when $x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_{nn}}$, where $\Omega_{\rho_{nn}}$ is the target region and $\Phi_{nn}(x)$ is a stabilizing controller that can render the origin of the RNN/LSTM model exponentially stable, the constraint of Eq. 9e is triggered to drive the state towards the origin; however, when the state $x(t_k)$ enters $\Omega_{\rho_{nn}}$, the predicted closed-loop state will be maintained within this region for the duration of the prediction horizon. While the full proof of closed-loop stability of the nonlinear system of Eq. 1 under the LMPC of Eq. 9 can be found in Wu et al. (2021), we present some key propositions below to help readers understand the key steps. Specifically, we first assume there exists a stabilizing feedback controller $u = \Phi_{nn}(x) \in U$ that can render the origin of the RNN model of Eq. 3 or the LSTM model of Eq. 7 exponentially stable in an open neighborhood $\hat{D}$ around the origin. The stabilizability assumption implies the existence of a continuously differentiable control Lyapunov function $V(x)$ such that the following inequalities hold for all x in $\hat{D}$:

$$\hat{c}_1|x|^2 \leq V(x) \leq \hat{c}_2|x|^2, \tag{10a}$$

$$\frac{\partial V(x)}{\partial x} F_{nn}(x, \Phi_{nn}(x)) \leq -\hat{c}_3|x|^2, \tag{10b}$$

$$\left|\frac{\partial V(x)}{\partial x}\right| \leq \hat{c}_4|x| \tag{10c}$$

where $\hat{c}_1$, $\hat{c}_2$, $\hat{c}_3$, $\hat{c}_4$ are positive constants. Similarly to the characterization of the closed-loop stability region for the nonlinear system of Eq. 1, we can find a level set of Lyapunov function embedded in $\hat{D}$ as the closed-loop stability region for the RNN model F_{nn}: $\Omega_\rho := \{x \in \hat{D} \mid V(x) \leq \rho\}$, where $\rho > 0$. Since there exists model-plant mismatch, the following proposition is developed to demonstrate that the feedback controller $u = \Phi_{nn}(x) \in U$ is able to stabilize the nonlinear system of Eq. 1 with high probability provided that the modeling error is sufficiently small. The proof can be found in Wu et al. (2021) and is omitted here.

Proposition 1. *Consider the RNN model trained using a set of m i.i.d. data samples $S = (\mathbf{x}_{i,t}, \mathbf{y}_{i,t})_{t=1}^{T}$, $i = 1, \ldots, m$ with the feedback controller $u = \Phi_{nn}(x) \in U$ that satisfies Eq. 10. If for all $x \in \Omega_\rho$ and $u \in U$, the modeling error can be constrained by $|F(x,u) - F_{nn}(x,u)| = E_M \leq \gamma|x|$, where γ is a positive real number satisfying $\gamma < \hat{c}_3/\hat{c}_4$, then the controller $u = \Phi_{nn}(x) \in U$ also renders the origin of the nonlinear system of Eq. 1 exponentially stable with probability at least $1-\delta$ for all $x \in \Omega_\rho$.*

The following proposition characterizes the deviation between the predicted state and the actual state in a finite period of time by accounting for the bounded modeling error assumed in Proposition 1.

Proposition 2. *Consider the nonlinear system $\dot{x} = F(x,u)$ of Eq. 1 and the RNN model $\dot{\hat{x}} = F_{nn}(\hat{x},u)$ with the same initial condition $x_0 = \hat{x}_0 \in \Omega_\rho$. There exists a class $\mathcal{K}$ function $f_w(\cdot)$ and a positive constant κ such that the following inequalities hold $\forall x, \hat{x} \in \Omega_\rho$:*

$$|x(t) - \hat{x}(t)| \leq f_w(t) := \frac{E_M}{L_x}(e^{L_x t} - 1) \tag{11a}$$

$$V(x) \leq V(\hat{x}) + \frac{\hat{c}_4\sqrt{\rho}}{\sqrt{\hat{c}_1}}|x - \hat{x}| + \kappa|x - \hat{x}|^2 \tag{11b}$$

Finally, the following proposition is developed to show probabilistic closed-loop stability of the nonlinear system of Eq. 1 under sample-and-hold implementation of the controller $u = \Phi_{nn}(x) \in U$.

Proposition 3. *Consider the nonlinear system of Eq. 1 with the controller $u = \Phi_{nn}(\hat{x}) \in U$ that meets the conditions of Eq. 10. Under the sample-and-hold implementation of control actions, i.e., $u(t) = \Phi_{nn}(\hat{x}(t_k))$, $\forall t \in [t_k, t_{k+1})$, where $t_{k+1} := t_k + \Delta$. there exist $\varepsilon_w > 0$, $\Delta > 0$ and $\rho > \rho_{min} > \rho_{nn} > \rho_s$ that satisfy*

$$-\frac{\tilde{c}_3}{\hat{c}_2}\rho_s + L_x^{'} M_F \Delta \leq -\varepsilon_w \tag{12}$$

and $\rho_{nn} := \max\{V(\hat{x}(t+\Delta)) \mid \hat{x}(t) \in \Omega_{\rho_s}, u \in U\}$, $\rho_{min} \geq \rho_{nn} + \frac{\hat{c}_4\sqrt{\rho}}{\sqrt{\hat{c}_1}} f_w(\Delta) + \kappa(f_w(\Delta))^2$, $\tilde{c}_3 = -\hat{c}_3 + \hat{c}_4\gamma > 0$, such that for any $x(t_k) \in \Omega_\rho \backslash \Omega_{\rho_s}$, with probability at least $1 - \delta$, the following inequality holds:

$$V(x(t)) \leq V(x(t_k)),\ \forall t \in [t_k, t_{k+1}) \tag{13}$$

and the state $x(t)$ of the nonlinear system of Eq. 1 is bounded in Ω_ρ for all times and ultimately bounded in $\Omega_{\rho_{min}}$.

Therefore, based on the above propositions, we can prove recursive feasibility and closed-loop stability of the nonlinear system of Eq. 1 by showing that 1) the stabilizing controller $u = \Phi_{nn}$ is a feasible solution to the MPC of Eq. 9 for all times, 2) the closed-loop state can be driven towards the origin, and ultimately bounded in the terminal set under $u = \Phi_{nn}$, and 3) due to the two Lyapunov-based constraints of Eqs. 9e-9f, the optimal solution solved by MPC will achieve an equally good closed-loop performance, if no better, than the stabilizing controller. Interested readers may refer to Wu et al. (2021) for the detailed proof of closed-loop stability for the nominal system of Eq. 1, and Wu et al. (2022) for the discussion of closed-loop stability for uncertain nonlinear systems with stochastic disturbances. Finally, it is important to point out that nonlinear model-based predictive control algorithms may lead to nonconvex optimization problems (whether ML models or first-principles models are used) which may be efficiently solved to local optimality with standard nonlinear optimization tools. There is no need to solve to global optimality as there is no time to accomplish this task in real-time for any practical application. This is how nonlinear MPC is implemented in practice with great success, and the computation of locally optimal solutions does not diminish at all the importance of developing further and implementing in practice nonlinear MPC with ML models.

Generalization error bound for barrier-function MPC

Barrier functions have been used to characterize the safety of dynamical systems by certifying whether a control law achieves forward invariance of a safe set. Some recent work have integrated control barrier functions (CBF) with Lyapunov functions to create a new function termed control Lyapunov barrier function (CLBF) and incorporate it in the design of MPCs to ensure process operational safety and stability simultaneously for safety-critical systems in chemical industry (Romdlony and Jayawardhana, 2016; Wu et al., 2019). Since machine learning tools such as feedforward neural networks (FNNs) can also be used to construct CBFs (Chen et al., 2022a), a generalization error bound on the resulting FNN-CBF is necessary for the derivation of probabilistic safety and stability guarantees for the control law designed using a CLBF. We first present the properties of a CBF in the following definition: (Wieland and Allgöwer (2007))

Definition 1. *Consider $\mathcal{D}$ which is a set of unsafe state values in state space, a continuously differentiable function $B(x) : \mathbf{R}^n \to \mathbf{R}$ is a CBF if the following conditions are met:*

$$B(x) > 0, \quad \forall\, x \in \mathcal{D} \tag{14a}$$

$$L_f B(x) \leq 0,\ \forall\, x \in \{z \in \mathbf{R}^n \backslash \mathcal{D} \mid L_g B(z) = 0\} \tag{14b}$$

$$\mathcal{X}_B := \{x \in \mathbf{R}^n \mid B(x) \leq 0\} \neq \emptyset \tag{14c}$$

For any initial state $x(t_0) = x_0 \in \mathcal{X}_0$, we say the system is safe in the sense that the closed-loop state stays within the safe region $\mathcal{X}_0$ for all times, where $\mathcal{X}_0 := \{x \in \mathbf{R}^n \backslash \mathcal{D}\}$, $\{0\} \in \mathcal{X}_0$ and $\mathcal{X}_0 \cap \mathcal{D} = \emptyset$, if there exists a constrained control law $u = \Phi(x) \in U$ that renders the origin of the closed-loop system of Eq. 1 asymptotically stable, and the closed-loop state trajectories do not enter the unsafe set D at all times, i.e., $x(t) \in \mathcal{X}_0$, $x(t) \notin \mathcal{D}$, $\forall\, t \geq 0$.

In Chen et al. (2022b), the CBF is developed from operating data in the state space that are labelled based on their safety status. This barrier function will then be synthesized using a feed-forward neural network, which typically consists of an input layer, some hidden layers, and an output layer. Specifically, the inputs to the FNN are the state vector $x \in \mathbf{R}^n$ of the nonlinear system of Eq. 1, and the output of the FNN predicts the barrier function value $\hat{B}(x) \in \mathbf{R}^n$. Training data points are collected from both the unsafe and the safe operating regions, where the target output values of $B(x)$ will satisfy the CBF conditions of Eq. 14a and Eq. 14c for the unsafe and the safe regions, respectively. More specifically, safe data points are labeled with a target output value of $B(x) = -1$, and unsafe data points are labeled with a target output value of $B(x) = +1$.

A general FNN model is considered, where m number of data samples are used to develop this model. The data samples are generated independently as per the data distribution over $X \times Y \in \mathbf{R}^{d_x} \times \mathbf{R}^{d_y}$, where d_x and d_y denote the dimension of the FNN input and output vectors respectively. The general structure of FNN model with inputs denoted as $\mathbf{x} \in \mathbf{R}^{d_x}$ and predicted output denoted as $\hat{\mathbf{y}} \in \mathbf{R}^{d_y}$ in terms of scalar or vector-valued functions and weight matrices for d total number of layers can be formulated as follows:

$$\hat{\mathbf{y}} = \sigma_d(W_d\sigma_{d-1}(W_{d-1}\sigma_{d-2}(...\sigma_1(W_1\mathbf{x})))) \tag{15}$$

where each W_l for $l = 1, ..., d$ layers represents the weight parameter matrix, and each σ_l represents the activation function in each layer. The number of layers d represents the depth of the network, and the width of the network h_{max} can be defined as the maximum number of neurons in a hidden layer (maximal column or row dimension of W_l), i.e., $h_{max} = \max_{l=1,...,d}\{h_l\}$, where h_l denotes the number of neurons in the l-th layer. Due to the unique dichotomous nature of $B(x)$, we choose a hyperbolic tangent sigmoid function $\sigma(z) = tanh(z) = \frac{2}{1+e^{-2z}} - 1$ as the activation function to polarize the output of the network and in turn, improve the prediction accuracy. This is because of the property of the $tanh(z)$ function approaching $+1$ as z approaches $+\infty$, and -1 as z approaches $-\infty$, thus polarizing the outputs of each layer and enforces the output of the FNN to approximate constant positive values (+1 for safe points), or constant negative values (-1 for unsafe points). The input and output of the FNN model are denoted by the bold face $\mathbf{x} \in \mathbf{R}^{d_x}$ and $\mathbf{y} \in \mathbf{R}^{d_y}$ respectively. For this particular application, $\mathbf{x}$ is the state vector of Eq. 1 ($x \in \mathbf{R}^n$), and $\mathbf{y}$ is the barrier function value ($B(x) \in \mathbf{R}^1$). Similar to the assumptions for training the RNN model of Eq. 3, we assume: 1) the FNN inputs are bounded, i.e., $|\mathbf{x}_i| \leq B_X$, for all $i = 1, ..., m$ samples, 2) the maximal 1-norm (l_1/l_∞) of the rows of weight matrices in the output and in the hidden layers are bounded by $\|W\|_{1,\infty} \leq B_W$, 3) all the datasets (i.e., training and testing) are drawn from the same underlying distribution, and 4) σ_l (where l denotes any hidden layers) is a 1-Lipschitz continuous activation function, and satisfies $\sigma_l(0) = 0$.
The following proposition was developed in Chen et al. (2022b) to derive the generalization error bound for the FNN model following the results for the RNN model of Eq. 3.

Proposition 4. *Consider the dataset S_s consisting of m i.i.d. data samples and the class of loss functions $\mathcal{G} = \{g_L : (\boldsymbol{x}, \boldsymbol{y}) \to L(h(\boldsymbol{x}), \boldsymbol{y}), h \in \mathcal{H}_h\}$ associated with the vector-valued FNN hypothesis class $\mathcal{H}_h$. With probability of at least $1 - \delta$, we have the following inequality:*

$$\begin{aligned}\mathbf{E}[g_L(\boldsymbol{x},\boldsymbol{y})] \leq & O\left(L_r d_y \frac{B_X (B_W)^d \sqrt{d+1+log(d_x)}}{\sqrt{m}}\right) \\ & + 3\sqrt{\frac{\log(\frac{2}{\delta})}{2m}} + \frac{1}{m}\sum_{i=1}^{m} g_L(\boldsymbol{x}_i, \boldsymbol{y}_i)\end{aligned} \tag{16}$$

where B_X is the upper bound on FNN inputs, B_W is the upper bound on FNN weight matrices, L_r is the local Lipschitz constant for the loss function $L(\cdot)$, d_x is the FNN input dimension, and d_y is the FNN output dimension.

Conclusions

This work presents an overview of recent research results on statistical machine learning modeling of nonlinear processes and closed-loop stability under machine-learning-based model predictive control. The theoretical results of the generalization error bounds for standard RNNs, physics-informed RNNs, and LSTMs were first discussed. Then, the machine learning models were incorporated into the design of MPCs that ensure closed-loop stability in probability. Finally, the recent work on statistical ML in barrier-function MPC was discussed, in which a generalization error bound was derived for the FNN-based barrier functions.

References

Alhajeri, M., J. Luo, Z. Wu, F. Albalawi, and P. D. Christofides (2022). Process structure-based recurrent neural network modeling for predictive control: A comparative study. *Chemical Engineering Research and Design 179*, 77–89.

Bartlett, P., D. J. Foster, and M. Telgarsky (2017). Spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1706.08498.*

Cao, Y. and Q. Gu (2019). Generalization bounds of stochastic gradient descent for wide and deep neural networks. *arXiv preprint arXiv:1905.13210.*

Chen, S., Z. Wu, and P. D. Christofides (2022a). Machine-learning-based construction of barrier functions and models for safe model predictive control. *AIChE Journal 68*, e17456.

Chen, S., Z. Wu, and P. D. Christofides (2022b). Statistical machine-learning-based predictive control using barrier functions for process operational safety. *Computers & Chemical Engineering 163*, 107860.

Golowich, N., A. Rakhlin, and O. Shamir (2018). Size-independent sample complexity of neural networks. In *Proceedings of the 31st Conference On Learning Theory*, pp. 297–299.

Mohri, M., A. Rostamizadeh, and A. Talwalkar (2018). *Foundations of machine learning*. MIT press.

Romdlony, M. Z. and B. Jayawardhana (2016). Stabilization with guaranteed safety using control Lyapunov–barrier function. *Automatica 66*, 39–47.

Wieland, P. and F. Allgöwer (2007). Constructive safety using control barrier functions. *IFAC Proceedings Volumes 40*, 462–467.

Wu, Z., F. Albalawi, Z. Zhang, J. Zhang, H. Durand, and P. D. Christofides (2019). Control Lyapunov-barrier function-based model predictive control of nonlinear systems. *Automatica 109*, 108508.

Wu, Z., A. Alnajdi, Q. Gu, and P. D. Christofides (2022). Statistical machine-learning–based predictive control of uncertain nonlinear processes. *AIChE Journal 68*, e17642.

Wu, Z., D. Rincon, and P. D. Christofides (2020). Process structure-based recurrent neural network modeling for model predictive control of nonlinear processes. *Journal of Process Control 89*, 74–84.

Wu, Z., D. Rincon, Q. Gu, and P. D. Christofides (2021). Statistical machine learning in model predictive control of nonlinear processes. *Mathematics 9*, 1912.

Wu, Z., A. Tran, D. Rincon, and P. D. Christofides (2019). Machine learning-based predictive control of nonlinear processes. part I: theory. *AIChE Journal 65*, e16729.