Smart process analytics for predictive modeling

Weihe Sun, Richard D. Braatz∗
Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

A R T I C L E   I N F O
Article history:
Received 3 May 2020
Revised 1 October 2020
Accepted 20 October 2020
Available online 21 October 2020

Keywords:
Process analytics
Data analytics
Machine learning
Intelligent systems
Predictive modeling
Model identification

A B S T R A C T
While data analytics tools are changing how manufacturers make critical decisions and designs, the selection of the best method requires a substantial level of expertise. In practice, methods are chosen based on familiarity or on cross-validation results from a large candidate model pool which over-fits data. A Smart Process Analytics framework is presented which empowers the users to focus on goals rather than on methods and automatically transforms manufacturing data into intelligence. The method selection is based on domain knowledge, the specific data characteristics, and nested cross-validation procedures. The approach is demonstrated in case studies for experimental datasets from a variety of process systems. For a four-stage evaporator, a state-space identification method is selected that has half the long-term prediction error than a recurrent neural network. For a combined cycle power plant, machine learning methods are selected that have up to 30% lower mean-squared error than partial least squares.

© 2020 Elsevier Ltd. All rights reserved.

1. Introduction

Specific challenges are encountered when applying the advances in data analytics and machine learning algorithms to manufacturing processes. The first challenge is the high diversity in data quantity and quality. The second challenge is that a substantial level of expertise is required to select the best process data analytics technique for a specific application, to avoid overfitting and ensure that an accurate process understanding is obtained. In reality, it is hard for analysts to be knowledgeable about all techniques. In actual practice, analysts typically default to using familiar techniques, which introduces additional biases into the model. Alternatively, simply trying every possible model and selecting the one that gives the best cross-validation errors is also not optimal.

The aforementioned challenges motivate the development of a systematic procedure – Smart Process Analytics (SPA) – for robust and automated method selection and model construction of manufacturing data, which allows analysts to focus on objectives rather than spending extensive time and effort in learning and selecting among methods, making advanced data analytics and machine learning more widely accessible and easier to use. The focus of this paper is predictive modeling, although the general idea can be extended to other tasks such as process monitoring or classification. To the best of our knowledge, no previous work has been done in the automated method selection and model construction on manufacturing data. For method selection, the basic concept was demonstrated by Severson et al. (2018), which takes a bottom-up approach that makes the decision about which technique to use based on both the characteristics of the data and expert domain knowledge on machine learning and manufacturing processes. For model construction, a rigorous cross-validation procedure is implemented based on the specific scenario. SPA is designed to provide models with high robustness, accuracy, and even interpretability when required.

The rest of this article is organized as follows. Section 2 describes a framework for automated method selection for predictive modeling, which is followed by a discussion of the importance of the proposed systematic procedure rather than performing an extensive model search purely on cross-validation. Section 3 describes methodologies for data interrogation. Section 4 discusses the principles and justifications for the data-based method selection. Section 5 discusses model construction and evaluation procedures, which are important but usually overlooked in the literature. Case studies with various data characteristics that demonstrate the procedure and its effectiveness are provided in Section 6, followed by the conclusions in the final section.

2. A Framework for predictive modeling

The prediction model considered in this paper has the general form of

\[ y = f(x) + \epsilon \]  (1)
where \( y \) represents a univariate response variable, \( \mathbf{x} \in \mathbb{R}^m \) is a vector of predictors, \( f(.) \) is the prediction model, and \( \varepsilon \) is the error term.

The simplest linear static model is

\[
y = w_0 + w_1 x_1 + \cdots + w_m x_m
\]

where \( w_0, \ldots, w_m \) are the model parameters. The parameter \( w_0 \) represents the intercept of the regression model, which allows for any fixed offset. It can be convenient to define an additional dummy variable \( x_0 = 1 \) so that

\[
y = w^\top \mathbf{x} + \varepsilon
\]

where \( \mathbf{w} = [w_0, \ldots, w_m]^\top \in \mathbb{R}^{m+1} \) is a vector of model parameters (aka weights).

The most basic way of reconstructing the model parameters in Eq. 3 is via ordinary least squares (OLS), which minimizes the mean squared error (MSE) of the predictions,

\[
\min_{\mathbf{w}} \frac{1}{N} \sum_{i=1}^{N} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2
\]

where \( N \) is the total number of training data points, and \( i \) is the index for a specific observation.

Given the training predictor matrix \( \mathbf{X} \in \mathbb{R}^{N \times m_h} \) and the training output vector \( \mathbf{y} \in \mathbb{R}^{N} \), OLS has a unique analytical solution

\[
\hat{\mathbf{w}}_{\text{OLS}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}
\]

provided that the matrix \( \mathbf{X}^\top \mathbf{X} \) is invertible.

The Gauss–Markov theorem states that OLS produces the best linear unbiased estimation under certain conditions. Those conditions include: (1) the underlying relationship between the predictor variables \( \mathbf{X} \) and the output variable \( y \) should be linear, (2) the design matrix \( \mathbf{X} \) has full rank (identification condition), and (3) errors are uncorrelated, homoscedastic, and have zero mean.

Alternatives are required when one or more of the assumptions are violated, which include: (1) nonlinear relationship between the output and predictors, (2) strong multicollinearity between predictors, (3) serial correlations in the model errors, which implies unexplained dynamics in the response, and (4) heteroscedasticity of the model errors. Moreover, in cases with a large number of predictors, OLS performs poorly in terms of model robustness and interpretation.

In the aforementioned situations, other predictive modeling techniques give more robust and accurate prediction results, which leads to the development of the automated method selection and model construction framework described in this article.

2.1. A systematic approach for robust and automated predictive modeling

A bottom-up approach for method selection is presented for providing an accurate and efficient procedure for automated method selection. The procedure contains two main steps: (1) data interrogation to assess data characteristics and (2) method selection based on the extent of the key characteristics and on expert domain knowledge.

A systematic procedure for data interrogation is proposed to directly assess the characteristics of the data, namely nonlinearity, multicollinearity, dynamics, and heteroscedasticity, as shown in Fig. 1. These four characteristics correspond to violations of the assumptions discussed above. The nonlinearity between the predictors and the response, and the multicollinearity between the predictors, are first assessed. The dynamics and heteroscedasticity are then analyzed after the first static model is built since those are residual characteristics rather than input or output characteristics.

Then the method selection is repeated if there are significant dynamics and heteroscedasticity. The detailed procedure of how to interrogate the data is given in Section 3.

The information obtained during data interrogation is used to select a best-in-class process data analytics tool, which can be graphically illustrated in the form of a triangle. Here only three key characteristics are considered: nonlinearity, multicollinearity, and dynamics. Depending on the degrees of each characteristic, the classes of techniques most suitable to the dataset are indicated on the data analytics triangle, as shown in Fig. 2. The triangle is labeled with methods that are appropriate for data with the specific characteristics. The vertices of the triangle represent the techniques that are suitable if the dataset only possesses one specific characteristic. For example, if a dataset only has the characteristics of dynamics, then canonical variate analysis (CVA), state-space autoregressive exogenous model (SSARX), and multivariable output error state space (MOESP) are the most suitable methods. Methods on the edges represent cases where the dataset has two of the three characteristics. Methods in the center are suitable for the dataset with all three characteristics. While these techniques can be more powerful, a much higher level of data analytics expertise and a larger quantity of data are required for these methods to be applied reliably, and the tendency of overfitting is higher when the complexity of the model is higher. As such, the simplest method able to describe the significant characteristics of the dataset should be used, which occurs by using the proposed SPA framework. Heteroscedasticity is treated separately, as discussed in Section 3.

The methods included in the data analytics triangle are selected based on theoretical analysis, case study performance, and domain knowledge of manufacturing processes. The methods selected for multicollinearity includes ridge regression (RR), which adds a 2-norm penalty to the least-squares objective, and partial least squares (PLS), which performs regression in a projected space in which the multidimensional directions in the input space best explain the maximum multidimensional variance directions in the output space. The sparse versions of those methods, which employ 1-norm penalties on the least-square objective to bias regression coefficients to zero, are elastic net and sparse PLS, respectively.

For nonlinear models, algebraic learning via elastic net (ALVEN) is selected for interpretable nonlinear model construction, which uses the combination of sparse regression with nonlinear feature construction, and support vector machine (SVR) and random forest (RF) are non-interpretable nonlinear models with greater approximation capabilities. For linear dynamic problems, linear system identification using CVA, MOESP, and SSARX algorithms are selected to construct linear subspace models. For nonlinear dynamic systems, recurrent neural networks (RNNs) and dynamic ALVEN (DALVEN) are selected for non-interpretable and interpretable nonlinear model constructions, respectively. All of these methods are discussed in more detail, with associated references, in Section 4.

2.2. Why not try every possible model and pick the “best” one

An important question concerns whether all of these regression analyses are needed for method selection. Software programs are available that run every possible model and provide the users with a list of models to choose from based on cross-validation results, where they include candidate models ranging from simple linear regression to advanced black-box models such as deep neural networks. Intuitively, it might be expected that if the cross-validation procedure is rigorously implemented, those programs will provide the users with the optimal results. However, crossvalidating every possible model comes at a cost: a large pool of candidate models increases the degrees of freedom, which increases the chance of overfitting (Arlot and Celisse, 2010). This problem becomes more severe when only a limited amount of data is availi-
able or when the cross-validation procedure is not appropriate for a given dataset.

This issue can be illustrated by a simple example. The underlying true process is a fourth-order polynomial with Gaussian noise $\varepsilon \sim N(0, \sigma^2)$:

$$y = x + 0.5x^2 + 0.1x^3 + 0.05x^4 + \varepsilon.$$  \hspace{1cm} (6)

Four types of approaches are compared: (1) biased second-order model, (2) unbiased fourth-order model, (3) 3-fold cross-validation on polynomials from order 1 to 10, and (4) 3-fold cross-validation on polynomials of order 2 and 4. The training dataset has 30 samples, and the testing dataset has 1000 samples (the testing dataset was chosen to be very large to ensure that the estimates of the testing error for each model is highly accurate). The noise level $\sigma^2$ varies from 1 to 1000. For each noise level, the simulation is repeated 3000 times, where x is sampled from $x \sim N(0, 5)$. The medians of testing MSEs in 3000 runs for each noise level are shown in Fig. 3.

When the noise level is low, the unbiased model gives the optimal performance. The biased model gives the largest error at low noise levels because the bias in the model dominates the MSE when the noise level is low. Cross-validation gives worse performance than the unbiased model for all low noise levels, because cross-validation has the possibility of picking the wrong order as long as the data have some noise.

Because the quality and quantity of data are limited, the true model structure is not known during model building, and a common practice by practitioners is to select the model with the lowest error among many models. The use of higher degrees of freedom, however, allows the picking of non-representative model structures, that is, model structures that just happen to fit the training data but give high prediction errors. When the noise level is high, the biased low-order model gives the best predictions, because accurate values of the model parameters cannot be estimated from the data for polynomials of higher order.\(^1\) This bias-variance tradeoff (Arlot and Celisse, 2010; Luxburg and Scholkopf, 2011) is well-known in applied statistics and statistical learning theory, but not taken into account by many practitioners and numerous AutoML software packages (Truong et al., 2019; Weaver, 2019) in terms of the increased variance when selecting among many models.

SPA and AutoML software share the same objective of providing a tool that enables non-machine learning experts to analyze their data correctly and efficiently, and ideally, the software should involve minimum decision-making procedures to automate the data analytics procedure. Numerous AutoML packages have been developed in recent years which include AutoWEKA (Kotthoff et al., 2017), Auto-sklearn (Feurer et al., 2015), TransmogrifiAI (TransmogrifiAI, 2020), Auto-Keras (Jin et al., 2019), H2O-Automl (H2O.ai, 2020), MLjar (MLjar, 2020), and TPOT (Le et al., 2019).

The main differences between the SPA and available AutoML software are in the design principles and the approach taken. SPA is designed specifically for manufacturing data, which combines the expert knowledge in industrial manufacturing processes to the method selection and model construction, whereas other AutoML software is designed for general applications and do not account for valuable domain-specific information such as the characteristics of spectral data. The key characteristics of process data are explicitly addressed by SPA for process design and valuation purposes. The current AutoML software uses various techniques (e.g., Bayesian optimization, genetic programming) to select between different models, but the general principle is to select among a large number of candidate models or pipelines, which SPA avoids as that principle is known to produce low-quality models that overfit data unless the data size is huge. Such huge data sets are rare in manufacturing operations. In addition, the candidate mod-

\(^1\) In the limit of 0 signal-to-noise ratio, the mean of the response variable is the optimal model.
els in AutoML packages might not be suitable for specific process data analytics problems, as further discussed in subsequent sections. Finally, due to the specific characteristics of process data, the appropriate model selection strategy (aka cross-validation method) is best selected to account for those data characteristics.

In this case study, using cross-validation to select the best among a large number of models did not provide the best model quality in an average sense. Moreover, the performance variation at each noise level by cross-validation of many models is higher than the other three approaches, due to higher degrees of model complexity. Here, for each noise level, the simulation was repeated for 100 times, and the standard deviation of the testing MSE was calculated. This whole procedure was repeated 3000 times and the median of the testing MSE standard deviation is shown in Fig. 4. The cross-validation over many orders/models results in much higher variability in model performance than the other models. The biased low-order model has the smallest number of model parameters so its parameter estimates and associated prediction error have lower variation, and the produced predictions are much less sensitive to noise.

When the selection orders are constrained to 2 and 4, the average performance of cross-validation is significantly improved for all values for the noise level. Selecting among only the two models results in nearly the same prediction error at the biased model at high noise levels, and the prediction error turns towards the unbiased model at low values of the noise level. Also, the performance variation is much lower than the unbiased model only when the noise level is high.

Some points illustrated by this case study are:

- The selection of the “best” model based on cross-validation can produce very low quality models when many models are considered.
- The selection of the “best” model based on cross-validation from a small number of models can produce accurate models for all noise levels. As such, an effective strategy is to pre-select from a small number of models known to be suitable for the type of application/data.
- A simple biased model is a better choice than an unbiased model when the noise level is sufficiently high that there is not enough signal to support the construction of the unbiased model.
- The best model for a specific problem depends on the characteristics of the data.

In the SPA framework, expert knowledge of process data analytics has been built into both data interrogation and method selection procedures. We considered only data analytics methods that have strong theoretical foundations and demonstrated performance in practice. Moreover, data interrogation suggests whether the signal in the data supports a more complex model. To further narrow the selection, rigorous cross-validation is built in to determine the optimal tradeoff between variance and bias.

The candidate models listed in the data analytics triangle account for both interpretability, accuracy, and robustness, which are the most important factors when considering building a manufacturing model. This approach is not the only solution and does not guarantee the optimal performance for a specific application, but rather provides rigorous guidance of how to apply process analytics and serves as a benchmark for more advanced future algorithm development.

3. Data interrogation

This section discusses methods for data interrogation in SPA for predictive modeling which guide the method selection in Section 4.

3.1. Nonlinearity

The first step is to assess whether nonlinearity between the response and predictors is significant. If not, a linear model should be used. For systems with only a few predictors, a quick way to check nonlinearity is to plot each predictor versus the response. For larger systems, a more efficient procedure uses linear and nonlinear correlation analyses.

The first step is to check linear relationship via pairwise linear correlation analysis. The Pearson’s correlation coefficient is calculated as

$$\rho_{x,y} = \frac{\text{cov}(x,y)}{\sigma_x \sigma_y}$$

(7)

where $\sigma$ is the standard deviation. The correlation coefficient is a measure of a linear relationship and ranges from −1 to 1 with a higher absolute value indicating a stronger linear correlation. In the second step, the nonlinear correlation is assessed by the combination of two nonlinearity tests: the quadratic test and maximal correlation analysis. These two tests are recommended to be used together since each method has its strengths and weaknesses.

3.1.1. Quadratic test

The quadratic test is a statistical test to check whether the response depends quadratically on the predictor. Quadratic functions are the second-order Taylor series approximation for a smooth nonlinear function, which provides a reasonable starting point to check nonlinearity.

The null hypothesis is $H_0 : y = w_1x + w_0$ and the alternative hypothesis is $H_a : y = w_2x^2 + w_1x + w_0$. The test statistic is defined as
\[ F = \frac{\text{SSE}(0) - \text{SSE}(\alpha)}{df_0 - df_a} \cdot \frac{\text{SSE}(\alpha)}{df_a} \]  

(8)

where \( \text{SSE}(0) = \sum (y_i - \hat{y}_i)^2 \) is the sum of squared errors between the observed and estimated values \( y \) using the null hypothesis model, \( \text{SSE}(\alpha) = \sum (y_i - \hat{y}_i)^2 \) is the sum of squared errors of the alternative model, and \( df_0 \) and \( df_a \) are the degrees of freedom for the null hypothesis and alternative hypothesis, respectively. The test statistic follows the \( F \)-distribution \( F_{df_0, df_a} \), and the null hypothesis is rejected if the associated \( p \)-value is below the significance level \( \alpha \), which indicates nonlinearity. With a large number of predictors, it is possible that some relationships will be flagged as being statistically significant by random chance and a failure to consider such false positives can lead to low specificity. The correction of the significance level is important and discussed in detail elsewhere (Nadon and Shoemaker, 2002). The Bonferroni correction is the default choice in SPA.

3.1.2. Maximal correlation analysis

The second approach is the maximal correlation analysis, which allows arbitrary nonlinear transformations for each pair of predictor and response. The maximal correlation (Rényi, 1959) is defined by

\[ \sup_{\theta, \phi} \text{corr}(\theta(x), \phi(y)) \]

(9)

where \( \theta \) and \( \phi \) are the sets of all Borel measurable functions with zero means, i.e., \( E(\theta) = E(\phi) = 0 \), and \( \text{corr}(\theta(x), \phi(y)) \) is the Pearson’s correlation coefficient between \( \theta(x) \) and \( \phi(y) \). The maximal correlation ranges from 0 to 1, with a higher value indicating a higher correlation between the transformed variables. The maximal correlation does not require transformation functions to be formed from a particular parametrized family and is a general measure of nonlinear correlation. However, this measure also requires a large number of data points to justify statistical significance. The alternating conditional expectation (ACE) algorithm developed by Breiman and Friedman (Breiman and Friedman, 1985) has been used extensively to calculate the maximal correlation.

Having both the linear correlation and the maximal correlation close to zero or one means that the variables are uncorrelated or linearly correlated, respectively. Having the linear correlation close to zero and the maximal correlation close to 1 indicates the variables are nonlinearly correlated.

For a large number of predictors, nonlinearity assessment results obtained by the quadratic test and maximal correlation analysis can be collected into matrices and compared with the linear correlation matrix to assess whether nonlinear correlations between predictors and the response are significant.

3.1.3. Bilinear test

Both of the above nonlinearity assessment tests assess pairwise nonlinear dependency between the predictor and the response. The bilinear test is a statistical test to check whether the response has a significant dependency on the interaction of two predictors. The null hypothesis is \( H_0 : y = w_1 x_1 + w_2 x_2 + w_0 \) and the alternative hypothesis is \( H_a : y = w_1 x_1 + w_2 x_2 + w_2 x_1 x_2 + w_0 \). The test statistic and the decision rule are the same as for the quadratic test.

3.2. Multicollinearity

The second step is to assess multicollinearity, which occurs when there are high inter-correlations among the predictors. Multicollinearity is common in manufacturing data due to insufficient experiments with the number of predictors near to or even more than the number of samples, and/or can be caused by high intrinsic correlations between several variables driven by the same latent factors or poor experimental design. When the multicollinearity is absolute, regularization or variable selection is necessary. High multicollinearity is also a problem because it increases the variance of the estimated coefficients, which are sensitive to minor changes in the model and difficult to interpret. Therefore, multicollinearity assessment is important to guide appropriate method selection.

Several methods are available for assessing multicollinearity; SPA uses the variable inflation factor (VIF) (James et al., 2013). The VIF quantifies how much the variance of the estimated coefficient for each predictor is inflated when multicollinearity exists. A higher value of VIF indicates a higher degree of multicollinearity.

The calculation of VIF for the \( k \)-th predictor \( x_k \) is

\[ \text{VIF}_k = \frac{1}{1 - R_k^2} \]

(10)

where \( R_k^2 \) is the \( R^2 \)-value by regressing the \( k \)-th predictor on the remaining predictors. As a rule of thumb, VIF greater than 10 is a sign of significant multicollinearity.

3.3. Dynamics

If the response variable is a measured time series, then the dynamic information (aka serial correlation) between the response variable and past response and predictors might be important for prediction. However, a dynamic model is not necessarily required in fitting a particular set of time series measurements. The sole cause of serial correlations in the response might be serial correlations in the predictors, in which case a static model would be appropriate. The assumption for OLS is only violated when the errors from the static model are serially correlated. Whether a dynamic regression model is needed can be assessed by analyzing static model residuals. Serial correlation in the residuals indicates that the data contain significant dynamic behavior that is not described by the static model.

Serial correlation can be assessed through the autocorrelation function (ACF) (Kočenda and Cerný, 2015),

\[ \text{corr}(\varepsilon_t, \varepsilon_{t-1}) \]

(11)

where \( \varepsilon \) is the lag variable. The ACF shows the extent to which previous measurements can be used to predict the current response. A dynamic model is needed when the autocorrelation in the model residual is significant.

3.4. Heteroscedasticity

Heteroscedasticity refers to non-constant variance of the model error across the range of values of the predictors or response. Heteroscedasticity of the error can result from incorrect model specification – such as nonlinearity in the data not captured by the model – or can be intrinsic variability of the response variable. For model selection, it is important to be able to design the data interrogation to account for the effects of nonlinearity when assessing heteroscedasticity.

There are several methods to test heteroscedasticity, including the Goldfeld-Quandt test, Breusch-Pagan test, and White test, and the reader is referred to (Breusch and Pagan, 1979; Goldfeld and Quandt, 1965) for more details. Although the statistical tests are straightforward, they should not be solely relied on, and visual inspection is still the most reliable way.

A common way to address heteroscedasticity is to apply a variance-stabilizing transformation to the response variable using transformation functions, including log, square root, and reciprocal
so that the transformed data have uniform variance. Log transformation is applied when the spread of the error variance is close to proportional to the conditional mean of the response variable. Alternatively, the Box-Cox transformation can be used to automatically determine the order for a power transformation on the response variable. Heteroscedasticity is not explicitly included in the data analytics triangle since such transformations can be applied before applying the data analytics triangle.

4. Method selection: The data analytics triangle

After assessing the degrees of the nonlinearity, multicollinearity, and dynamics, methods are selected based on the information and the modeling objective of the user. The candidate models in the data analytics triangle are suitable for data with the specific characteristics.

4.1. Methods for multicollinearity

Many regularization methods have been developed to deal with multicollinear data by biasing the estimated coefficients towards zero to reduce the variance of the prediction. The right amount of bias–variance tradeoff can improve the overall prediction accuracy. This section provides an overview and comparison study of dense and sparse methods for dealing with multicollinearity. More detailed information about algorithms can be found in the corresponding references.

4.1.1. Dense methods

Regularization methods are primarily of two types: (1) shrinkage methods and (2) latent variable methods. Shrinkage methods directly penalize large regression coefficients in order to prevent overfitting and handle multicollinearity. Ridge regression (RR) (Hoerl and Kennard, 1970) is a widely used shrinkage method that applies a $l_2$-norm penalty on the vector of regression coefficients. By imposing the norm constraint when building models from multicollinear data, the regression coefficients are prevented from having large magnitudes that cancel each others’ effects on the output. With a $l_2$-norm penalty, highly correlated predictors tend to have similar regression coefficients, which is called the grouping effect (James et al., 2013).

Latent variable models use derived inputs, which are linear combinations of original predictors, as new predictors in the regression. Two widely applied latent variable models are principal component regression (PCR) (Massy, 1965) and partial least squares (PLS) (Rospal and Krämer, 2006). PCR uses the principal components as the new predictors. RR can be viewed as a continuous version of PCR (Hastie et al., 2009) that applies the shrinkage to all PCs, with a greater amount of shrinkage applied to directions with smaller variances. As in PCR, PLS constructs latent variables based on linear combinations of the original predictors, but instead maximizes the covariance between the predictor and the response. Research has shown that the variance aspect in PLS tends to dominate, which results in PLS tending to behave similarly to RR and PCR (Hastie et al., 2009).

Although RR, PCR, and PLS have very different procedures, their goal is the same and the performance is usually similar. All of the three methods bias the regression coefficients away from the OLS solution towards directions in the predictor space with large variations. The shrinkage paths by different methods have a similar pattern, although PLS and PCR are discrete and more extreme. A detailed theoretical analysis is provided by Frank and Friedman (1993). Comparison studies indicate similar performance for most practical datasets (Frank and Friedman, 1993; Hastie et al., 2009; Huang et al., 2002; Reis and Saraira, 2004; Yeniyi and Göktaş, 2002). RR and PLS are selected as the candidate models for a dense structure in the data analytics triangle.

4.1.2. Sparse methods

The aforementioned techniques construct dense models, that is in which all predictors have non-zero regression coefficients. However, most industrial processes have only a few relevant predictors that affect the final response variable, and the use of sparse regression methods can improve both prediction accuracy and model interpretability. When multicollinearity is present, LASSO (Tibshirani, 1996) tends to select one variable from the group of correlated features arbitrarily, and can select at most $N$ predictors in case of $N < m_e$. Elastic net (EN) (Zou and Hastie, 2005), which combines RR and LASSO, tends to select groups of correlated variables. Case studies have demonstrated the effectiveness of EN for the construction of sparse models for industrial processes (Chiu and Yao, 2013; Severson et al., 2015). Similarly, sparse PLS (SPLS) handles sparsity and multicollinearity simultaneously by fusing the PLS technique with LASSO (Chun and Keles, 2010). At the time of the writing of this article, no publication has included a detailed comparison between EN and SPLS, most likely largely because EN came from the machine learning community and SPLS came from the chemometrics community. Both methods are included in the data analytics triangle, which is discussed in the next section.

4.1.3. Comparison study between different methods for dealing with multicollinearity

This section provides a thorough comparison study to assess if there are any characteristics in the data that can further narrow down the selection between different multicollinearity methods. A total of 68 simulated case studies with different noise levels, multicollinearity degrees, and sparsity structures are tested. The true model has the form

$$y = w^T x + \sigma \epsilon$$

where $\epsilon \sim N(0, 1)$. For each example, the simulated data consist of a training set and an independent testing set. Models were fitted on training data using repeated 3-fold cross-validation with 20 repetitions, and the testing MSE is computed for the testing dataset. The notation / is used to describe the number of observations in the training set and the testing set, respectively. Each case is simulated 1000 times to ensure the statistical significance of the results. The useful predictors are those with non-zero coefficients, while the useless predictors are associated with zero coefficients in the true model. Two general situations are considered:

**Type 1 cases**: the number of data points is larger than the number of predictors.

---

**Case 1.** Correlated useful predictors with $20/200$: $w = [3.1, 5.0, 0, 2.5, 0, 0, 0, 0, 0, 0]_T$ and $\sigma = 0.5, 1, 3, 6$. The correlations between useful predictors are $\text{corr}(x_1, x_2) = 0.5, \text{corr}(x_1, x_3) = 0.7, \text{and} \text{corr}(x_2, x_3) = 0.3$.

**Case 2.** Correlated useless predictors with $20/200$: same as Case 1 except that the correlations are between useless predictors $\text{corr}(x_5, x_6) = \text{corr}(x_7, x_8) = 0.85$ and $\text{corr}(x_3, x_4) = \text{corr}(x_6, x_9) = 0.5$.

**Case 3.** Separately correlated predictors with $30/300$: $w = [3.1, 5.0, 0, 2.5, 0, 0, 0, 0, 0, 0, 0, 0]_T$ and $\sigma = 0.5, 1, 3, 6$. The correlations between useful predictors are $\text{corr}(x_1, x_2) = 0.5, \text{corr}(x_1, x_3) = 0.7, \text{and} \text{corr}(x_2, x_3) = 0.3$. The correlations between useless predictors are $\text{corr}(x_5, x_6) = 0.85, \text{corr}(x_6, x_9) = 0.5, \text{corr}(x_7, x_8) = 0.85, \text{corr}(x_{10}, x_{12}) = 0.9, \text{and} \text{corr}(x_3, x_4) = 0.5$.

---

2 The notation $i_j$ refers to a row vector of $k$ elements in which each element is equal to $i$.
Case 4. Pairwise correlated predictors with 30/300: \( w = [3, 1.5, 0, 0.25, 0.1] \) and \( \sigma = 0.5, 1, 3, 6 \). The pairwise correlation between predictors is \( \text{corr}(x_i, x_j) = 0.7^{i-j}, \forall i, j \in \{1, \ldots, 10\} \).

Case 5. Correlated useful predictors (high sparsity) with 30/300: \( w = [3, 1.5, 0.1] \) and \( \sigma = 0.5, 1, 3, 6 \). The correlation structure is \( \text{corr}(x_1, x_2) = 0.7 \).

Case 6. Correlated useless predictors (high sparsity) with 30/300: \( w = [3, 1.5, 0.1] \) and \( \sigma = 0.5, 1, 3, 6 \). The correlation structure is \( \text{corr}(x_1, x_2) = 0.7^{i-j}, \forall i, j \in \{3, \ldots, 20\} \).

Case 7. Separately correlated predictors (high sparsity) with 30/300: \( w = [3, 1.5, 0] \) and \( \sigma = 0.25, 0.5, 1, 3, 6 \). The correlation structure is \( \text{corr}(x_1, x_2) = 0.7^{i-j}, \forall i, j \in \{3, \ldots, 14\} \).

Case 8. Only one useful predictor with 30/300: \( w = [0, 1.5, 0] \) and \( \sigma = 0.2, 0.25, 0.5, 2, 5 \). The correlation structure is \( \text{corr}(x_1, x_2) = 0.7^{i-j}, \forall i, j \in \{1, \ldots, 10\} \).

Case 9. Pairwise correlated predictors (high sparsity) with 30/300: \( w = [3, 1.5, 0] \) and \( \sigma = 0.25, 0.5, 1, 3, 6 \). The correlation structure is \( \text{corr}(x_1, x_2) = 0.7^{i-j}, \forall i, j \in \{1, \ldots, 14\} \).

Type II cases: the number of data points is fewer than the number of predictors.

Case 1. Separate latent factors for useful and useless predictors with 20/200: \( w = [3_{10}, -4_{10}, 0] \) and \( \sigma = 1, 3, 6, 15 \). The three latent factors are \( t_i \sim N(0, 1), i = 1, 2, 3 \). The predictors are \( x_j = t_j + \alpha N(0, 1), j \in \{10(i-1) + 1, \ldots, 10i\} \) for \( \alpha = 0.01, 0.1, 0.2, 0.5 \), respectively.

Case 2. Latent variable model for slightly correlated useful predictors with 30/300: \( w = [3_{10}, 12_{20}, 0] \) and \( \sigma = 0.1, 0.5, 1, 5 \). The latent variable is \( t \sim N(0, 1) \). The predictors are \( x_j = [A^T + 0.5 N(0, 1)] \) \( \{3, 0, 0\} \) and \( \{0, 0, 0\} \) for \( j \in \{1, \ldots, 30\} \).

Case 3. Latent variable model for moderately correlated useful predictors with 30/300: same as Case 2 except that \( \sigma = 5, 10, 20, 30 \) and \( A \sim U[0, 1] \in R^{30 \times 3} \).

Case 4. Latent variable model for moderately correlated useful predictors with 30/300: same as Case 2 except that \( \sigma = 5, 20, 40, 80 \) and \( A \sim U[0.5, 1.5] \in R^{30 \times 3} \).

Case 5. Separate latent factor for useful and useless predictors (high sparsity) with 20/200: \( w = [3, 0] \) and \( \sigma = 0.5, 3, 6, 12 \). The three latent factors are \( t_i \sim N(0, 1), i = 1, 2, 3 \). The predictors are \( x_j = t_j + 0.5 N(0, 1) \) for \( j \in \{2, 3\} \) and \( x_j = t_j + 0.5 N(0, 1) \) for \( j \in \{4, \ldots, 16\} \) and \( x_j = t_j + 0.5 N(0, 1) \) for \( j \in \{17, \ldots, 30\} \).

Case 6. Latent variable model of low correlation (high sparsity) with 30/300: \( w = [0, 3_{10}, 15_{20}, 0] \) and \( \sigma = 0.05, 0.2, 0.35, 0.5 \). The latent variable is \( t \sim N(0, 1) \). The predictors are \( x_j = [A^T + 0.5 N(0, 1)] \) \( \{0, 0, 0\} \) for \( j \in \{1, \ldots, 30\} \) and \( \sigma = 0.5, 1, 2, 6 \) and \( A = U[0, 1] \in R^{30 \times 3} \).

Case 7. Latent variable model of medium correlation (high sparsity) with 30/300: same as Case 6 except that \( \sigma = 0.5, 1, 2, 6 \) and \( A = U[0, 1] \in R^{30 \times 3} \).

Case 8. Latent variable model of high correlation (high sparsity) with 30/300: same as Case 6 except that \( \sigma = 1.5, 15, 30 \) and \( A = U[0.5, 1.5] \in R^{30 \times 3} \).

The median MSE testing results for the Types I and II cases are summarized in Tables 1 and 2. EN and SPLS gave similar results in all cases. There is no specific pattern as to which sparse method is better based on correlation structure, sparsity, or noise level. Also, RR and PLS gave similar results for all cases with no specific pattern to guide the selection of one method over the other. EN and SPLS are closely related, in particular, that they are the result of combining a technique known as least-angle regression with RR and PLS, respectively (Zou and Hastie, 2005; Chun and Keles, 2010). Given this relationship, and that RR and PLS give similar model quality for most problems, perhaps it is not surprising that EN and SPLS also give similar performance.

Secondly, it is observed that when RR is better than PLS for a specific problem, EN tends to be better than SPLS, which can be explained by the fact that EN and SPLS are the sparse versions of RR and PLS. This result is observed in both the median MSE and its spread obtained by doing many Monte Carlo runs, such as seen in the median MSE of two Type I cases with its 95% confidence interval (estimated using bootstrap with 1000 resamplings) in Fig. 5.

The third point is that, for lower noise level and higher sparsity, sparse methods perform better than dense methods, which holds for both Type I and Type II cases. Given the same sparsity in the true process, with an increasing level of noise, the difference between the sparse and dense methods is smaller (e.g., Case 1 in Type I shown in Fig. 6). This occurs due to limited data to characterize the correct underlying structure by the sparse methods, and the effect of averaging out noises and fluctuations by dense methods. Throwing out useful predictors will introduce a larger bias to the model. For complicated cases (e.g., Type II with \( N < m_x \)), sometimes dense methods give even better results as compared to the corresponding sparse methods because of insufficient data to correctly characterize the underlying sparse structure, as observed in Fig. 7.

Also, with higher sparsity, the sparse methods are expected to produce better models given the same noise level, e.g., see Case 5 of Type I in Fig. 8. Similar results are seen in the Type II cases.

All of the methods are compared to OLS for the Type I cases \( (N > m_x) \), which shows the effectiveness of regularization methods for data with multicollinearity (see Appendix A). OLS is not applicable to the Type II cases.

In summary, it is hard to pick a priori the best multicollinearity method for a specific problem based on prior knowledge, e.g., whether the underlying system has a sparse structure. Therefore, cross-validation is used to select the best method for a specific problem. In terms of the computational efficiency of the sparse methods, EN was observed to typically take less time compared to SPLS for the same problem, which could be a consideration for very big datasets.

4.2 Methods for nonlinear model construction

Two types of nonlinear models are considered in SPA: interpretable and black-box.\footnote{Interpretable model shows the explicit form of nonlinear dependence between the predictor and the response variable.} An interpretable model shows the explicit form of nonlinear dependence between the predictor and the response variable. In the process control literature, the word “black-box” is commonly used to refer to any model that is constructed purely from data. A drawback of such a definition is that there are data analytics methods that construct models purely from data while providing insight into the process, in other words, without the model being “black.” Perhaps for this reason, the process monitoring and other communities refer to models constructed purely from data as being “data-driven,” which is more precise and clearer term. Here the term “black-box” refers to data-driven models in which the internal model structure is not interpretable by a human, e.g.,
sponse variables, which is more robust when the number of training samples is limited. Black-box models, on the other hand, can construct highly accurate representations of complicated nonlinear networks, for which the model is truly “black” because the human cannot peer into the box/model to gain insights into the process.

Table 1
Median MSE of testing data by multicollinearity methods for Type I cases for 1000 repetitions.

<table>
<thead>
<tr>
<th>Noise</th>
<th>Case</th>
<th>EN</th>
<th>SPLS</th>
<th>RR</th>
<th>PLS</th>
<th>Case</th>
<th>EN</th>
<th>SPLS</th>
<th>RR</th>
<th>PLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>very low</td>
<td>1</td>
<td>0.011</td>
<td>0.013</td>
<td>0.013</td>
<td>0.014</td>
<td>6</td>
<td>0.030</td>
<td>0.026</td>
<td>0.061</td>
<td>0.060</td>
</tr>
<tr>
<td>low</td>
<td>0.044</td>
<td>0.048</td>
<td>0.051</td>
<td>0.052</td>
<td>0.113</td>
<td>0.099</td>
<td>0.187</td>
<td>0.179</td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.326</td>
<td>0.310</td>
<td>0.336</td>
<td>0.333</td>
<td>0.230</td>
<td>0.207</td>
<td>0.338</td>
<td>0.320</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>0.745</td>
<td>0.686</td>
<td>0.747</td>
<td>0.686</td>
<td>0.638</td>
<td>0.631</td>
<td>0.747</td>
<td>0.770</td>
<td></td>
<td></td>
</tr>
<tr>
<td>very low</td>
<td>2</td>
<td>0.023</td>
<td>0.025</td>
<td>0.027</td>
<td>0.026</td>
<td>7</td>
<td>0.005</td>
<td>0.005</td>
<td>0.007</td>
<td>0.007</td>
</tr>
<tr>
<td>low</td>
<td>0.087</td>
<td>0.092</td>
<td>0.099</td>
<td>0.100</td>
<td>0.071</td>
<td>0.076</td>
<td>0.096</td>
<td>0.094</td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.569</td>
<td>0.618</td>
<td>0.574</td>
<td>0.611</td>
<td>0.452</td>
<td>0.428</td>
<td>0.497</td>
<td>0.483</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>1.034</td>
<td>1.100</td>
<td>1.038</td>
<td>1.093</td>
<td>0.879</td>
<td>0.866</td>
<td>0.904</td>
<td>0.937</td>
<td></td>
<td></td>
</tr>
<tr>
<td>very low</td>
<td>3</td>
<td>0.009</td>
<td>0.010</td>
<td>0.012</td>
<td>0.011</td>
<td>8</td>
<td>0.022</td>
<td>0.022</td>
<td>0.029</td>
<td>0.029</td>
</tr>
<tr>
<td>low</td>
<td>0.037</td>
<td>0.035</td>
<td>0.045</td>
<td>0.044</td>
<td>0.034</td>
<td>0.034</td>
<td>0.044</td>
<td>0.045</td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.271</td>
<td>0.266</td>
<td>0.294</td>
<td>0.287</td>
<td>0.125</td>
<td>0.121</td>
<td>0.160</td>
<td>0.176</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>0.663</td>
<td>0.629</td>
<td>0.683</td>
<td>0.678</td>
<td>0.823</td>
<td>0.818</td>
<td>0.853</td>
<td>0.887</td>
<td></td>
<td></td>
</tr>
<tr>
<td>very low</td>
<td>4</td>
<td>0.011</td>
<td>0.012</td>
<td>0.014</td>
<td>0.014</td>
<td>9</td>
<td>0.018</td>
<td>0.021</td>
<td>0.028</td>
<td>0.028</td>
</tr>
<tr>
<td>low</td>
<td>0.044</td>
<td>0.048</td>
<td>0.051</td>
<td>0.053</td>
<td>0.071</td>
<td>0.078</td>
<td>0.097</td>
<td>0.096</td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.322</td>
<td>0.345</td>
<td>0.329</td>
<td>0.355</td>
<td>0.441</td>
<td>0.438</td>
<td>0.494</td>
<td>0.508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>0.727</td>
<td>0.744</td>
<td>0.72</td>
<td>0.738</td>
<td>0.854</td>
<td>0.849</td>
<td>0.884</td>
<td>0.921</td>
<td></td>
<td></td>
</tr>
<tr>
<td>very low</td>
<td>5</td>
<td>0.019</td>
<td>0.022</td>
<td>0.044</td>
<td>0.050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>low</td>
<td>0.072</td>
<td>0.076</td>
<td>0.143</td>
<td>0.156</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.456</td>
<td>0.422</td>
<td>0.592</td>
<td>0.583</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>0.914</td>
<td>0.863</td>
<td>0.968</td>
<td>0.926</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2
Median MSE of testing data by multicollinearity methods for Type II cases for 1000 repetitions.

<table>
<thead>
<tr>
<th>Noise</th>
<th>Case</th>
<th>EN</th>
<th>SPLS</th>
<th>RR</th>
<th>PLS</th>
<th>Case</th>
<th>EN</th>
<th>SPLS</th>
<th>RR</th>
<th>PLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>very low</td>
<td>1</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
<td>0.001</td>
<td>5</td>
<td>0.134</td>
<td>0.112</td>
<td>0.181</td>
<td>0.142</td>
</tr>
<tr>
<td>low</td>
<td>0.007</td>
<td>0.006</td>
<td>0.007</td>
<td>0.007</td>
<td>0.279</td>
<td>0.234</td>
<td>0.329</td>
<td>0.262</td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.029</td>
<td>0.026</td>
<td>0.028</td>
<td>0.026</td>
<td>0.575</td>
<td>0.496</td>
<td>0.617</td>
<td>0.530</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>0.176</td>
<td>0.156</td>
<td>0.171</td>
<td>0.156</td>
<td>1.018</td>
<td>0.980</td>
<td>1.052</td>
<td>1.080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>very low</td>
<td>2</td>
<td>0.204</td>
<td>0.239</td>
<td>0.211</td>
<td>0.235</td>
<td>6</td>
<td>0.004</td>
<td>0.128</td>
<td>0.463</td>
<td>0.521</td>
</tr>
<tr>
<td>low</td>
<td>0.212</td>
<td>0.246</td>
<td>0.218</td>
<td>0.242</td>
<td>0.062</td>
<td>0.362</td>
<td>0.498</td>
<td>0.557</td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.234</td>
<td>0.267</td>
<td>0.237</td>
<td>0.259</td>
<td>0.193</td>
<td>0.518</td>
<td>0.569</td>
<td>0.617</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>0.713</td>
<td>0.687</td>
<td>0.682</td>
<td>0.679</td>
<td>0.440</td>
<td>0.654</td>
<td>0.652</td>
<td>0.700</td>
<td></td>
<td></td>
</tr>
<tr>
<td>very low</td>
<td>3</td>
<td>0.029</td>
<td>0.023</td>
<td>0.029</td>
<td>0.025</td>
<td>7</td>
<td>0.032</td>
<td>0.088</td>
<td>0.088</td>
<td>0.095</td>
</tr>
<tr>
<td>low</td>
<td>0.082</td>
<td>0.067</td>
<td>0.082</td>
<td>0.069</td>
<td>0.119</td>
<td>0.142</td>
<td>0.147</td>
<td>0.145</td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.240</td>
<td>0.207</td>
<td>0.238</td>
<td>0.208</td>
<td>0.309</td>
<td>0.287</td>
<td>0.308</td>
<td>0.289</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>0.417</td>
<td>0.377</td>
<td>0.414</td>
<td>0.372</td>
<td>0.858</td>
<td>0.821</td>
<td>0.847</td>
<td>0.804</td>
<td></td>
<td></td>
</tr>
<tr>
<td>very low</td>
<td>4</td>
<td>0.006</td>
<td>0.006</td>
<td>0.008</td>
<td>0.008</td>
<td>8</td>
<td>0.029</td>
<td>0.039</td>
<td>0.042</td>
<td>0.041</td>
</tr>
<tr>
<td>low</td>
<td>0.073</td>
<td>0.062</td>
<td>0.072</td>
<td>0.063</td>
<td>0.346</td>
<td>0.322</td>
<td>0.342</td>
<td>0.318</td>
<td></td>
<td></td>
</tr>
<tr>
<td>medium</td>
<td>0.228</td>
<td>0.207</td>
<td>0.225</td>
<td>0.204</td>
<td>0.924</td>
<td>0.888</td>
<td>0.912</td>
<td>0.877</td>
<td></td>
<td></td>
</tr>
<tr>
<td>high</td>
<td>0.563</td>
<td>0.539</td>
<td>0.558</td>
<td>0.529</td>
<td>1.094</td>
<td>1.141</td>
<td>1.088</td>
<td>1.112</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 6. Median MSE for the four methods with different noise levels of Case 1 in Type I.

Fig. 7. Median MSE for the four methods for Case 8 in Type II.

Fig. 8. Median MSE for the four methods for Case 1 and its sparse version Case 5 in Type I.
nonlinear model type is selected by SPA. Moreover, all the methods considered in SPA for nonlinearity can deal with multicollinearity.

4.2.1. Interpretable model: ALVEN

Algebraic learning via elastic net (ALVEN) is selected for interpretable nonlinear model construction. Given the training data, ALVEN constructs explicit nonlinear transformations of predictors and constructs a low-complexity linear model composed of selected useful nonlinear transformations (Sun and Braatz, 2020a; Sun, 2020). The two-step sparsity-promoting procedure in ALVEN combines univariate feature screening with sparse regression via EN, which is computationally efficient. ALVEN produces parsimonious models, to balance model complexity with descriptive capability. The interpretability is at the expense of approximation capability. For datasets with a large number of training samples from complex nonlinear systems, ALVEN might not describe data as well as a black-box nonlinear modeling method.

4.2.2. Black-box models

Black-box models have higher approximation accuracy, which comes at the expense of model complexity and uninterpretability. The nonlinear black-box models in SPA are Random Forest (RF) regression (Breiman, 2001) and Support Vector Regression (SVR) (Vapnik, 2000).

RF is an ensemble model that compromises several different decision trees using the bagging method. By combining individual regression trees, the ensemble model tends to be more accurate and have less variance compared to a single decision tree. Moreover, RF includes feature selection, and the feature importance can be computed to learn which predictor is most crucial for predicting the response variable. RF can also explicitly handle categorical variables as input without converting the categories into integer or real numbers.

Random Forest has some limitations. First, the final relationship between the predictors and the response is hard to interpret. Second, RF tends to overfit due to its high model complexity and many hyperparameters to tune with limited data. Thirdly, the RF prediction surface over the design space is discontinuous, which is difficult for using the model for optimal design or control. Finally, RF should never be used for extrapolation, where RF will give the upper or lower bound values of the response in the original set.

SVR has different options for kernel functions, which can generate nonlinear models through implicit nonlinear kernel mappings. SVR is also flexible and can produce nonlinear models with high approximation accuracy, which comes at the cost of high model complexity and uninterpretability. Same as RF, SVR should not be used for extrapolation.

4.3. Methods for dynamics

When the residuals are auto-correlated, the static model is not capable of capturing the dynamic behavior in the response variable. A model that uses both the current predictor and past system knowledge is then required to explain the variation in the response variable. System identification is a procedure to construct a model of a dynamic system from the measured data. System identification can be further classified into the linear and nonlinear system identification.

4.3.1. Linear system identification

Numerous methods are available for system identification of linear dynamical systems (Van Overschee and De Moor, 1996). The classical time series analysis autoregressive moving average with extraneous inputs (ARMAX) and the subspace methods are the most widely applied classes of algorithms. SPA selects the subspace algorithms for linear state-space model identification for their ease of application and salient numerical properties. Here only a brief overview is given. The reader is referred to reviews on subspace algorithms for more details (Chiang et al., 2001; De Moor et al., 1999; Qin, 2006).

Subspace identification algorithms are used to estimate the system matrices, which consist of two steps. The first step projects the measurements to subspaces to estimate the states or the extended observability matrix of the unknown system. The second step calculates the system matrices based on the estimated states or observability matrices using simple linear regression. The most popular subspace algorithms are numerical algorithms for subspace state-space system identification (N4SID) (Van Overschee and De Moor, 1996), MOESP (Verhaegen and Dewilde, 1992), and CVA (Larimore, 1990). N4SID and CVA use the state estimates while the MOESP uses the extended observability matrix to find the model parameters. The three subspace algorithms can be cast into a unified framework (Van Overschee and De Moor, 1995), equivalent to calculating the oblique projection and performing singular value decomposition (SVD) with different weighting matrices. Larimore (1990) states that the weighting used in CVA is optimal for the state order determination from finite sample sizes, which is argued by examples without any mathematical proofs (Juricek et al., 2002). Case studies in the literature have indicated that the methods have similar performance on average (Favoreel et al., 1999; Sotomayor et al., 2003).

Compared with ARMAX, subspace algorithms have several advantages. First is the simplicity in parameterization. The complexity of the identification for a large multi-input-multi-output system is the same as for single-input single-output systems. Moreover, the state-space model is globally identifiable, which is statistically well-conditioned. The subspace algorithms also avoid a prior parameterization of the state-space model, which instead is determined directly from the data. The hyperparameters, such as the state order, can be selected by specific criteria during the model-building procedure. The subspace algorithms are, therefore, suitable for automated multivariable system identification, which explains their incorporation into commercial process control software for the last twenty years (Qin and Badgwell, 2003).

Subspace algorithms can be implemented using the singular value decomposition (SVD) and QR decomposition, which do not require iterative nonlinear optimization and are numerically stable and accurate with an upper bound on the required computational cost. Therefore, the methods are well suited for large-scale systems. Lastly, subspace algorithms incorporate implicit model order reduction, which is useful for process data affected by noise and disturbances. Moreover, the estimated states that summarize the process are directly usable in dynamic process monitoring (Russell et al., 2000).

For systems with closed-loop controllers, traditional system identification methods become biased due to the correlation between future input and past noise. Closed-loop subspace algorithms have been proposed and are described in reviews (Lin et al., 2005; Qin, 2006). It has been shown that the modified subspace algorithms are effective for the closed-loop conditions. One major approach is system identification with pre-estimation. This method pre-estimates the Markov parameters of the predictor from a high-order ARX model to decouple the correlation between the noise and the controlled input. Typical methods include the ADAPTx formulation and implementation of CVA (Larimore, 2000; 2004), and SSARX which is a subspace identification method that uses an ARX estimation-based algorithm to compute the weighting (Jansson, 2003). A detailed comparison between these two methods with mathematical formulations is available (Qin, 2006).

The aforementioned algorithms have been incorporated into commercial software for system identification. SPA provides func-
tions that directly call the System Identification Toolbox in MATLAB (http://www.mathworks.com/products/sysid/) – which implements standard CVA, MOESP, and SSARX – and the system identification package ADAPTx by Adaptips (Larimore, 2003).

ADAPTx is commercial software for automated system identification for linear state-space models. ADAPTx implements a sophisticated formulation of CVA that can handle unstable systems with unknown feedback, stochastic seasonal components, and nonstationary noise (Larimore, 2000).

Case studies collectively show that no algorithm is consistently better than the others (see Appendix Appendix B). Therefore, the best model is selected from the System Identification Toolbox and ADAPTx based on user-defined criteria, such as the validation performance.

4.3.2. Nonlinear system identification

Although linear models are attractive for their simplicity and interpretability, they have limitations. Many real-world systems have nonlinear dynamics so that a linear model is only effective for a limited range of operating conditions. However, the nonlinear system identification is much more involved as compared to linear system identification. As a result, nonlinear system identification remains an active area of research with model structures including linear time-varying models (Larimore et al., 2015; Tóth, 2010; Verdult, 2002), local linear models (Nelles, 2001), Hammerstein-Wiener structures (Bai, 1998; Chen and Fassois, 1992; Palanthandalam-Madapusi et al., 2005; Rangan et al., 1995), nonlinear ARMAX (NARMAX) (Chen and Billings, 1989; Leontaris and Billings, 1985) models, and general nonlinear state-space forms (also called nonlinear grey-box models) (Schön et al., 2011). No method has been shown to be better than the others for all processes.

There are some limitations of the aforementioned methods:

- The linear time-varying models are applicable only to processes that operate near one state trajectory, such as in some batch and semibatch processes, and so are not practically applicable, for example, to batch processes that follow many varying trajectories or to continuous processes.
- The local linear model requires the separation of operating regimes and the selection of an interpolation rule. Also, this method only has limited approximation capability as compared to other methods.
- The Hammerstein-Wiener (aka sandwich) model can be considered as the combination of the Hammerstein model and the Wiener model. The Hammerstein model is a static nonlinearity block followed by a linear time-invariant (LTI) model, and the Wiener model is an LTI model followed by static nonlinearity to the output. The Hammerstein-Wiener model consists of a nonlinear static nonlinearity followed by an LTI model followed by a static nonlinearity. This model can also be combined with a nonlinear feedback model relatively easily. These types of models can be considered as nonlinear dynamical systems with external nonlinearity, with typical choices of nonlinearity being polynomial, saturation, or piecewise linear functions. The Hammerstein-Wiener type model can be estimated by gradient-based search and has been widely applied because it captures the nonlinear in many specific processes including distillation columns and pH neutralization (Kim et al., 2012). A drawback is that the general type of Hammerstein-Wiener model is not interpretable, and that the model cannot describe all forms of dynamic nonlinearity.
- Similar to the ARMAX model, the NARMAX model uses the past input and output with predefined nonlinear transformations to predict the current output, which requires the selection of the appropriate nonlinearity and also nonlinear optimization procedure. The MATLAB System Identification Toolbox provides several nonlinear transformations, but those transformations are not interpretable.
- The general nonlinear state-space model requires the specification of the nonlinear structure before parameter estimation, which is usually not feasible due to limited process knowledge. The MATLAB System Identification Toolbox has an implementation of the Hammerstein-Wiener model, the NARMAX model, and the general nonlinear state-space model by Ljung (1995).

SPA selects another two methods for nonlinear system identification. The selected methods have advantages and disadvantages as compared to the aforementioned methods, and can be considered complementary to what is offered by the nonlinear system identification toolboxes (Ljung, 1995; Ninness et al., 2013). The two selected methods – dynamic ALVEN (DALVEN) (Sun and Braatz, 2020a; Sun, 2020) and the recurrent neural network (RNN) (Elman, 1990) – are described below.

Interpretable model: DALVEN

DALVEN is an interpretable nonlinear system identification method, which has the form of a nonlinear autoregressive model with exogenous inputs (NARX) structure (Sun and Braatz, 2020a; Sun, 2020). Its model structures cannot describe every possible nonlinear dynamic relationship. However, the model structures include nonlinearities that commonly arise in chemical processes, and the predefined nonlinear mapping in DALVEN is interpretable and can be easily integrated with an optimizer for process design and/or control. After the DALVEN model is fitted, an ARIMA model can be applied to the residual to construct a time-series model if the residuals have significant remaining autocorrelation. This approach produces DALVEN model structures with quantified ARIMA error which can potentially improve the prediction accuracy. The ARIMA part might not be useful for specific applications such as system design and control, and is omitted in the default setting for SPA.

Same as the linear dynamics models constructed by the MATLAB System Identification Toolbox, DALVEN is also capable of performing k-step prediction through recursion. More details can be referred to (Sun and Braatz, 2020a; Sun, 2020).

Black-box model: RNN

Traditional feedforward neural nets have been used extensively in modeling nonlinear dynamical systems (Chen and Billings, 1992; Hunt et al., 1992; Narendra and Parthasarathy, 1990). For example, the auto-regressive models have been combined with the feedforward neural net to describe nonlinear dynamics (Narendra and Parthasarathy, 1990). The RNN is an alternative to the NARX model that uses a recurrent state to produce a more compact model and reduce the system memory requirements (Elman, 1990). RNNs have been very successful in language modeling, speech recognition, image captioning, and music composition, and have been shown to achieve state-of-the-art performance on a wide range of sequential data modeling tasks (Jozefowicz et al., 2016; Merity et al., 2016; Wu et al., 2016). Moreover, RNNs have been used in a number of system identification problems (Li, 2003; Parlos et al., 1994; Sun et al., 2020).

The standard RNN architecture has an input layer $x$, a hidden layer $s$ (state layer), and output layer $y$. Input, state, and output layers are computed as

$$ s_{t+1} = \phi(A_s {s_t} + B x_t + b) $$
$$ y_t = C s_t + k $$

(13)

where $A, B, C, b, k$ are model parameters with appropriate dimensions, and $\phi$ is an element-wise activation function that can be either linear or nonlinear (e.g., logistic, hyperbolic tangent, and rectifier functions). The mathematical formulation is a nonlinear state-space model, which captures the nonlinear dynamic information in the system and has the universal approximation capabili-
ity of neural nets. RNNs are appropriate when the objective is to have the capability of modeling any form of nonlinear dynamics while not being interested in model interpretability. RNNs can be overfit due to their complexity, and so appropriate regularization should be used to mitigate this problem. For example, early stopping is used for RNNs in SPA to choose the number of training epochs (Prechelt, 1998). Also, RNN is only advised to be the candidate model when there is a large number of samples.

A drawback of the architecture (13) is that prediction errors are compounded, which leads to unrealistic predictions for a longer prediction horizon or an inaccurate initial condition. Feedback connections of the system can be added to the RNN to improve prediction accuracy by using the actual past output measurements as information to correct the system prediction, which can be computed as

\[
\begin{align*}
\hat{y}_{t+1} &= \phi(As_t + Bs_t + D\hat{y}_t + b) \\
y_t &= Cs_t + k
\end{align*}
\]

(14)

where \(D\) is the parameter matrix with the appropriate dimension and \(\hat{y}_{t+1}\) is the actual output measurement at \(t = 1\). The architecture of the RNN model with feedback connections is illustrated in Fig. 9.

The multi-step prediction can be implemented by feeding the output of the RNN back to the model. The intermediate states should also be the corresponding multi-step-prediction states from the RNN model (Fig. 10 for illustration of multiple-step-ahead prediction in SPA). The multi-step prediction function is available in the SPA software.

The loss function is typically defined as the mean squared one-step prediction error, with \(l_2\) regularization to prevent overfitting,

\[
L = \frac{1}{t} \sum_{t=1}^{N} ||y_t - \hat{y}_{t}\|^2 + \lambda (||A||^2 + ||B||^2 + ||C||^2)
\]

(15)

where \(y_t\) is the true output at time \(t\), \(\hat{y}_{t}\) is the one-step-ahead prediction by the RNN model, and \(\lambda\) is a weight decay parameter.

To train the RNN, backpropagation through time (BPTT) is used to update model weights in order to minimize the loss function (Werbos, 1990). However, BPTT has difficulties learning long-range dependencies because of the vanishing gradient problem (Bengio et al., 1994). To alleviate this problem, gating mechanisms have been developed, resulting in two popular RNN variations: long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014). LSTM and GRU have been reported to show salient performance (Cakir et al., 2015; Graves et al., 2013). The SPA provides regular, LSTM, and GRU units with different types of activation functions.

5. Hyperparameter selection for automated model construction

Cross-validation is the most widely applied method for prediction error estimation and hyper-parameter selection. Cross-validation estimates the expected out-of-sample prediction error by holding out a portion of data when training the model, and testing the model performance based on the holdout dataset. The hyperparameters that give the smallest validation error is selected, and the final model is rebuilt on all of the data. Cross-validation can be used with any loss function and data analytics method. There are different cross-validation schemes, and the appropriate method depends on many attributes which include but are not limited to: (1) the total number of samples and predictors in the dataset, (2) the presence of replicate samples in the dataset, (3) the specific objective of the analysis, (4) available time to do cross-validation, (5) the ordering of the samples in the dataset, and (6) the consequences of overly optimistic or pessimistic results. Cross-validation schemes used in SPA are discussed in Sections 5.1 to 5.7.

For dynamic models, cross-validation needs to be done in a slightly different way to ensure that the dynamic structure in the data is not distorted. For a relatively small sample size, the sample sizes across different folds will be mismatched, which is likely to systematically favor over-parasimonious models. Besides cross-validation, the information criterion is a widely applied technique, especially for state-space model identification. Several widely applied information criteria are discussed in Section 5.8.

5.1. Simple held-out validation set

The training data can be split into one training and one validation fold. The training dataset is used to train the model while the validation dataset is used to validate the trained model. The hyperparameters that give the smallest validation error are selected. The validation dataset can be either picked randomly from the available data or taken from the last portion of the data for time series modeling (see Section 5.7 for more details). This method is computationally efficient as compared to other cross-validation schemes described in Sections 5.2-5.7, but is only recommended when there are enough sample points.

5.2. K-fold cross-validation

K-fold cross-validation is usually implemented when the amount of data is limited, and the data are independent and identically distributed (i.i.d.). The training set is split into k smaller sets, and then the model is trained using k – 1 folds of the training data and validated on the remaining fold of data. This procedure is repeated \(k\) times, and the performance is averaged over \(k\) models (see Fig. 11).

5.3. Repeated k-fold and monte carlo (MC)-based cross-validation

Repeated k-fold repeats k-fold \(ntimes\), with different randomized splits in each repetition. Monte Carlo-based cross-validation is similar, where training samples are shuffled and then split into
a pair of training and validation datasets with a predefined portion of validation samples. The procedure is repeated n times, and the performance is averaged over all the validation errors. The Monte Carlo-based cross-validation is illustrated in Fig. 12. These two cross-validation approaches allow control of the number of repetitions to average over and also the number of validation data, which are good alternatives to k-fold cross-validation for limited samples. However, with an increasing number of repetitions, the computational cost is higher.

5.4. Grouped cross-validation

While i.i.d. data is a common assumption, this requirement may not be fulfilled in practice. As a rule of thumb, the data splitting should ensure independence of the training and validation data. For manufacturing processes, this independence can be violated if the process has a group structure, such as samples collected by different measurement devices over time, or ongoing sensor drift. In those cases, grouped cross-validation should be used to avoid an overly optimistic estimation of the true error and the selection of hyperparameters that fits the system bias. The cross-validation should estimate the prediction performance of the unseen groups. Therefore, all of the data collected from one group should be either in the training dataset or the validation dataset. There are different strategies based on the grouping principle, including group k-fold, leave-p-group-out, and Monte Carlo group split. The group k-fold is to split data into k-folds while making sure the data from one group is in one fold. The leave-p-group-out will retain pgrouops of data as the validation data set (see Fig. 13). The Monte Carlo group split is a randomized version of leave-p-group-out where the pgroups are randomly selected, which gives finer control over the number of repetitions.

5.5. Importance of held-out test set and nested cross-validation

Cross-validation has a risk of overfitting when the validation dataset is used for both hyperparameter selection and model evaluation. This approach may provide an optimistic estimate of the true generalization error (Cawley and Talbot, 2010). This problem is avoided by holding out a test dataset from the available data for final model evaluation to give a better estimation of the generalization error. The nested cross-validation is to split data into different training and held-out test datasets and then do regular cross-validation using the aforementioned schemes on the training set. This procedure is repeated, and the final generalization performance of an algorithm is the average over different held-out test datasets. This procedure can be combined with any cross-validation schemes as the inner cross-validation loop, and an illustration of k-fold nested cross-validation is shown in Fig. 14.

Nested cross-validation provides an unbiased estimate for the model error, and also an estimate of model stability by comparing variation in different folds. Large variations in the model performance or the selected hyperparameters indicate insufficient training data and bad generalization performance. Moreover, if the error estimate from the inner loop is overoptimistic as compared to the outer loop, the model could be overfit. A drawback of nested cross-validation is its computational cost. For this reason, this cross-validation method is left as an option based on the amount of time available for cross-validation.

5.6. The one-standard-error rule

Instead of choosing the hyperparameters with the minimum validation error, the one-standard-error rule can be used, which selects the most parsimonious model whose error is smaller than one standard deviation above the error of the best model. This rule takes into consideration that the validation performance curve is estimated with error, and therefore a conservative estimation is taken, which is considered to have higher robustness. An illustration of a PLS model with its single hyperparameter fit using this rule is in Fig. 15. The parsimonious model not only has better interpretability but is also less prone to overfitting.

5.7. Cross-validation for the dynamic model

For time series, conventional cross-validation is inappropriate, as leaving an arbitrary portion of data out for validation will distort the dynamic information in the data and can also cause data
leakage because the future data provides information of the past. Therefore, the holdout cross-validation set should be separated chronologically, where successive training datasets are supersets of the past data (see Fig. 16).

5.8. Information criteria

The Akaike information criterion (AIC) (Akaike, 1974) is often used as a criterion to select model structures for dynamic models and can also be used to compare different models, as long as the training data are the same. AIC requires less computation than cross-validation. AIC is an estimator of the expected Kullback-Leibler information of the true model with respect to the fitted model. A smaller value of AIC indicates that the estimated model is closer to the unknown system that generated the data. Minimizing AIC has also been shown to be asymptotically equivalent to minimizing the mean-squared forecast error (Shibata, 1980). However, AIC is not a measure of forecast accuracy, so the forecast error needs to be calculated separately on the holdout set.

The AIC for a $k^{\text{th}}$-order state-space model is calculated as

$$
\text{AIC}(k) = -2 \ln p(y; \mathbf{u}; \theta_k) + 2 M_k
$$

(16)

where $p(y; \mathbf{u}; \theta_k)$ is the likelihood function based on observations $y$ and $\mathbf{u}, \theta_k$ is the maximum likelihood parameter estimation, and $M_k$ is the number of parameters in the model. The model order is associated with the minimum value of AIC($k$) chosen as the final model.

In the case of relatively few training data, a small-sample version of AIC, called AICc, should be used (Cavanaugh, 1997; Sugiura, 1978):

$$
\text{AICc}(k) = -2 \ln p(y; \mathbf{u}; \theta_k) + 2f M_k
$$

(17)

where $f$ is the correction factor. AICc is recommended when the number of training samples divided by the number of parameters is smaller than 40 (Anderson and Burnham, 2004). When the number of training samples is large, the difference between AIC and AICc is negligible.

Another information criterion for method selection is the Bayesian information criterion (BIC) (Schwarz, 1978):

$$
\text{BIC}(k) = -2 \ln p(y; \mathbf{u}; \theta_k) + M_k \ln N
$$

(18)

The BIC tends to favor simpler models than AIC because of the higher penalization factor. Despite the similarity between BIC and AIC, BIC is motivated differently than AIC. BIC selects the best model from the candidate models with the highest posterior probability. Given their different objectives, neither AIC or BIC is clearly better than the other.

6. Case studies

In this section, several examples are used to illustrate the systematic procedure in SPA for tackling different predictive modeling
problems. Each example represents a specific data characteristic. The selected model is compared to other modeling techniques in SPA to illustrate the effectiveness.

A Python software implementation of the algorithms is available for download (Sun and Braatz, 2020b).

6.1. Limited fiber data describable by a linear model

First consider a simple single-input single-output dataset for testing manufactured fibers that has a linear input-output relationship (Montgomery, 2017). There are 15 measurements of the fiber diameter and the fiber strength, and the modeling goal is to predict the fiber strength (S) using the measured diameter (D). The scatter plot of the data is provided in Fig. 17.

The first step in SPA is data interrogation. Since there is only one input variable in the system, multicollinearity is not a consideration. Since the data are static measurements without particular ordering, dynamic analysis is not needed. As such, only nonlinearity needs to be considered. The nonlinearity assessment by the linear correlation, quadratic test, and maximal correlation is shown in Fig. 18. The maximal correlation is observed to be close to linear correlation, and the p-value of the quadratic test is above any reasonable significance level (e.g., 0.01). Therefore, a simple linear model using OLS is selected.

Several other models (RR, ALVEN, RF, SVR) are also trained for comparison. Because this dataset only has 15 samples, nested cross-validation is implemented. 5-fold cross-validation is used in the inner loop for other techniques, while OLS does not need cross-validation in the inner loop. The outer loop testing dataset has 20% of the total data and is repeated 100 times.

The results are shown in Table 3. The linear models are significantly better than all of the nonlinear models, in terms of both model accuracy and stability. The tested nonlinear methods’ testing errors are higher in both mean and variance. OLS and RR give nearly identical results. The final prediction results using OLS are plotted in Fig. 19, and the residual analysis is shown in Fig. 20. The residual analysis shows no significant nonlinearity and heteroscedasticity in the data, indicating that the OLS model is appropriate.

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Model prediction results for fiber data.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
</tr>
<tr>
<td>OLS</td>
<td>0.114</td>
</tr>
<tr>
<td>RR</td>
<td>0.116</td>
</tr>
<tr>
<td>ALVEN</td>
<td>0.113</td>
</tr>
<tr>
<td>RF</td>
<td>0.053</td>
</tr>
<tr>
<td>SVR</td>
<td>0.108</td>
</tr>
</tbody>
</table>

6.2. Sensor calibration model from biased spectral data with high multicollinearity

Next, consider a more complicated case of spectral data analytics. The modeling goal is to use attenuated total reflection-Fourier transform infrared (ATR-FTIR) spectroscopy to predict the potassium dihydrogen phosphate (KDP) concentration in the solution. A detailed description of the experiments is available elsewhere (Togkalidou et al., 2001). The dataset contains 116 samples with five different KDP concentrations and varying solution temperature. Each solution was cooled while spectral data were collected in the range between 4000 and 650 cm⁻¹ until crystals started to appear or to room temperature. Several measured spectra with different concentrations are plotted in Fig. 21. The construction of the best models from spectral data usually requires pre-processing to remove artifacts. For ATR-FTIR spectra collected from liquid solutions, it is well-established that data for frequencies in the blue-shaded regions shown in Fig. 21 are associated with water vapor and CO₂ inside the instrument and are subject to drift and/or noise (Bruun et al., 2006). Such drift and noise are observed in the spectra, which confirms that such data should be removed from the dataset before applying data analytics. Such a removal based on domain knowledge could be automatically included into the software to pre-process ATR-FTIR data collected from liquid solutions. Below SPA is demonstrated for the cases in which the data are included or excluded, to evaluate its performance for a prototypical real dataset in which the data are not pre-processed.

Beer’s Law, which holds for dilute solutions, states that absorbance is linearly related to the concentration. Beer’s Law also holds, although not always, for many concentrated solutions over the concentration range of interest in a particular application. According to the nonlinearity test, there is no significant nonlinearity between the absorbance and the concentration, indicating that Beer’s Law holds for this dataset. The test also shows that the correlation between temperature and concentration is linear, which agrees with a visual inspection that the ATR-FTIR data mostly vary linearly with temperature (that is, the slopes of lines fit to the five sets of data points in Fig. 22 are roughly the same). As such, SPA correctly selects to apply a linear model for this dataset.

From visual observation of Fig. 22 that the absorbance for one of the five concentrations (0.0296, in orange) showed a drift dur-
Fig. 20. Residual analysis for the OLS model of the fiber data.

Fig. 21. Example ATR-FTIR spectra for KDP solution: (left) raw spectra, (right) identified blue regions with spectral artifacts that should be removed in pre-processing. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Fig. 22. Absorbance at wavenumber 1074.2 cm⁻¹ of KDP solutions at different temperatures (label corresponds to KDP concentration). This frequency corresponds to the peak absorbance, which makes it easier to evaluate trends due to the highest signal-to-noise ratio.

ing the middle of the batch experiment that went from high to low temperature, that resulted in the absorbance at the lowest temperatures being biases. Such drifts are common in spectra and can be from a number of causes, such as a gaseous nitrogen tank used to purge the systems running out of gas, or a malfunctioning flow regulator on the nitrogen gas tank. This case study will include the biased data, to evaluate the effectiveness of SPA for such realistic data.

For multicollinearity, the absorbances within similar wavenumbers are highly correlated. Also, there are only 116 samples, whereas the number of predictors is 961 (after removing the noise region). Since spectral data always have multicollinearity, there is no need to run a formal multicollinearity test when building models from such data. The multicollinearity tests were run and did indicate that the data have significant multicollinearity, but are not shown here for brevity.

The data interrogation from SPA indicates that EN, SPLS, RR, or PLS should be applied to the data. Now the problem becomes whether the sparse methods (EN and SPLS) should be used instead of the dense methods (RR and PLS). In SPA, the dense methods are selected for spectral data because there are typically fluctuations and drift in some of the highly correlated absorbances, and the sample number is inadequate to find the correct underlying sparsity structure. To illustrate what can happen even when the best sparse methods are applied, as well as what can happen when
intentionally noisy data are included, SPLS and EN were applied to the original dataset and the regressed coefficients are plotted with the average spectra in Fig. 23. In general, the regressed coefficients are expected have similar shapes to the acquired spectra in terms of smoothness (Togkalidou et al., 2001). This smoothness occurs because, for well-acquired spectra, the wavelength channel should be as nearly as good as its neighboring channel for predictions, as neighboring wavelength channels are typically highly correlated. Therefore, observing noise in the regressed coefficient pattern (e.g., fluctuating between positive and negative in the neighborhood wavelengths), indicates that the model is overfitting to the tiny differences between neighboring wavelengths. As shown in Fig. 23, the regressed coefficients by SPLS selected the noisy wavelength regions as useful predictors, which indicates overfitting. For EN, the regressed coefficients are the same as RR ($l_1 = 0$), which also indicates a dense model is appropriate. This example illustrates a key point that, although EN is a sparse method when a sparse method is appropriate, EN can produce dense models if its internal selection of hyperparameters indicates that a dense model gives the best predictions. Although EN works well for this dataset even when biased and noisy data are included, for the reasons given above, only dense methods should be applied to spectral data.\(^5\)

As discussed in Section 4.1, when the number of training samples is limited, dense models average the effects of fluctuations. Hence, dense methods (RR and PLS) should be selected as defaults for spectral calibration.

Another observation from the EN results in Fig. 23(right) is that the model coefficients around 1000 cm\(^{-1}\) for EN have a very similar weighting as the mean spectra. That frequency range has clearly resolved narrow peaks, meaning that each peak is associated with a specific vibration in the solute molecule. The similarity in shape between the model coefficients and absorbances indicates that most of the change in the spectra with concentration is associated with scaling the size of the overall peak absorbance or a shift in the absorbance peaks to left or right. The EN model coefficients around 3300 cm\(^{-1}\) is a mirrored reflection of the mean spectra, see Fig. 23(right). That broad peak is for water, and the mirror image indicates that an increase in solute concentration corresponds to a linear decrease in the absorbance of light by water. That interpretation is consistent, as an increase in solute immediately implies a decrease in solvent for the fixed sample volume measured by the sensor.

Nested cross-validation with grouped cross-validation for the inner and outer loop is used for model construction based on

---

\(^5\) An exception is a dataset from a very low-cost spectral sensor that only provides measurement for a small number of wavelengths.
datasets. To evaluate SPA for a dataset with such characteristics, the raw data were used, without doing any pre-processing, e.g., to remove apparent outliers.

For this dataset, different numbers of training samples are used to assess the performance of different nonlinear models. The number of training samples varies from 100 to 5000, while the last 3000 points are always used for testing. First, the data were interrogated to assess nonlinearity and multicollinearity. The results for 200 training samples were used in the data interrogation, with the other cases not shown because the results were very similar. The nonlinearity tests (Fig. 26) shows that there is a significant nonlinearity in the relationship between the predictors and the output (the maximal and linear correlations are very different in magnitude for RH and the p-values below the significant level in the quadratic and bilinear tests). There is identifiable multicollinearity in the predictors (Fig. 27). All of the nonlinear models used in SPA can deal with multicollinearity.

The model construction procedures were run for the three selected nonlinear models (ALVEN, SVR, and RF). The 3-fold repeated cross-validation with 20 repetitions is used. For comparison purposes, four linear methods are also tested. The training, validation, and testing MSEs using different numbers of training samples are summarized in Table 4. The nonlinear models are observed to be suitable for the energy output prediction, which is in accordance with the data interrogation result. As expected from the discussion in Section 4.2, ALVEN gave the best performance (smallest predic-

---

6 The multicollinearity test is not needed for method selection when the nonlinearity test indicates that the data have a nonlinear relationship, but is run by SPA anyway to provide insight to the user into the dataset.
Fig. 26. Nonlinear test results for CCPP data. One of the values is much greater than 1.

Fig. 27. Multicollinearity test results for CCPP data.

Table 4
Mean squared errors for different static regression methods for CCPP data with different numbers of training samples.

<table>
<thead>
<tr>
<th>N_train</th>
<th>100</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALVEN</td>
<td>0.055</td>
<td>0.066</td>
</tr>
<tr>
<td>RF</td>
<td>0.049</td>
<td>0.102</td>
</tr>
<tr>
<td>SVR</td>
<td>0.046</td>
<td>0.064</td>
</tr>
<tr>
<td>EN</td>
<td>0.063</td>
<td>0.071</td>
</tr>
<tr>
<td>SPLS</td>
<td>0.063</td>
<td>0.072</td>
</tr>
<tr>
<td>RR</td>
<td>0.063</td>
<td>0.072</td>
</tr>
<tr>
<td>PLS</td>
<td>0.063</td>
<td>0.072</td>
</tr>
<tr>
<td>N_train</td>
<td>2000</td>
<td>5000</td>
</tr>
<tr>
<td>MSE</td>
<td>Train</td>
<td>Val-</td>
</tr>
<tr>
<td>ALVEN</td>
<td>0.062</td>
<td>0.063</td>
</tr>
<tr>
<td>RF</td>
<td>0.049</td>
<td>0.061</td>
</tr>
<tr>
<td>SVR</td>
<td>0.058</td>
<td>0.060</td>
</tr>
<tr>
<td>EN</td>
<td>0.070</td>
<td>0.070</td>
</tr>
<tr>
<td>SPLS</td>
<td>0.078</td>
<td>0.070</td>
</tr>
<tr>
<td>RR</td>
<td>0.070</td>
<td>0.070</td>
</tr>
<tr>
<td>PLS</td>
<td>0.078</td>
<td>0.070</td>
</tr>
</tbody>
</table>

with the observation from the scatter plot; the heavy-tailed distribution on the left is caused by the outliers that has relatively large negative residuals.

In summary, SPA correctly picks the nonlinear model for the CCPP data.

6.4. Dynamic model for a four-stage evaporator from data from replicated factorial experiments

This section presents a time-series modeling case study. The dataset was collected from a four-stage evaporator, which was designed to reduce the water content of milk products (Zhu et al., 1994). There are three input variables to the system \((x_1, x_2, x_3)\), which are feed flow (FF), vapor flow to the first evaporator stage (VF), and cooling water flow (CF), respectively. The task is to predict the dry matter content (DM) in the final product. A more detailed description is available (Zhu et al., 1994), and the dataset can be downloaded from (De Moor et al., 1999). The dataset has 6305 samples, with the raw data plotted in Fig. 30 and the scatter plot shown in Fig. 31); the first 4305 samples are used for training, and the remaining 2000 points are used for testing. The scatterplot for the predictor variables shows that most of the data are located near the extreme points of the dataset, which indicates a 2^4 factorial experimental design with replication. A weakness of the experimental design is that no center-points were included.

The data interrogation for nonlinearity indicates that the predictors and the output have weak static linear correlation and have a weak static nonlinear relationship (Fig. 32), and Fig. 33 indicates that multicollinearity is minimal. Collectively, these results suggest that either the output cannot be predicted or that a dynamic relationship exists between the inputs and output.

For time-series data, its static character is first assessed to determine whether a static model is able to capture most of the input-output relationship through residual analysis. This step should be taken because static models are sufficient for modeling some types of time-series data, such as in many multivariate statistical process control applications for continuous-flow oil refineries (Russell et al., 2000). Assessing dynamic content can avoid constructing an overly complex model when a static model captures almost all of the input-output behavior.

The model prediction results for a simple linear regression model are shown in Figs. 34, 35, and 36. The static linear model does not explain hardly any of the variations in the output, which is consistent with the linear correlation analysis. The same conclusion can be drawn from the residual analysis in Fig. 35. The resid-

---

Such plant-wide data are typically sampled relatively slowly and have relatively low signal-to-noise.
Fig. 29. CCPP residual analysis for testing data.

Fig. 30. Water evaporator data for the three input variables (FF, VF, and CF) and one output variable (DM).
ual has large magnitude and has the same variation pattern as the original data, showing that the variation is not explained by the static model. The distribution of the residuals are well described as being normal, as seen the Q-Q plot. The residuals are lumped

**Fig. 31.** Water evaporator data scatter plot.

**Fig. 32.** Nonlinearity test for water evaporator data.

**Fig. 33.** Multicollinearity test for water evaporator data.

**Fig. 34.** Comparison of model predictions and raw testing data for a static linear model of a water evaporator.
into two clouds when plotted with respect to the predicted output, which is in part due to the input variables always being at extreme values. The residuals are autocorrelated (Fig. 36), indicating that, a dynamic model should be constructed.

To further assess whether a nonlinear dynamic model is needed, the nonlinearity assessment is repeated for lagged variables. The results in Fig. 37 show that a linear dynamic model is sufficient. The linear and maximal correlations between lagged DM and the output are the same for each lag, meaning that the maximal correlation is high solely due to the linear correlation. The other inputs have a small linear and nonlinear correlations to the output. The p-values for the quadratic test indicate that a quadratic term is not statistically justified, which was also observed for the bilinear test (not shown here for brevity).

The SPA analysis indicates that a linear state-space model should be used for final model construction. The software packages in MATLAB and ADAPTx (SSARX and CVA, see Section 4.3) are used to construct state-space models. For comparison, DALVEN and RNN models were also constructed. AIC was used for model selection for the linear state-space model and RNN. For DALVEN, both the full nonlinear mapping version (denoted as DALVEN-full) and the partial mapping version (denoted as DALVEN) are tested, using one single-hold out dataset (20% of the training data), AIC, and BIC for model selection. The RNN automatically selects the final activation function from linear, relu, and sigmoid activation functions.
Fig. 38. Testing prediction MSE for water evaporator data for seven data analytics methods for 0 to 10 prediction steps.

Fig. 39. Comparison of predicted values with testing data for an SSARX model for a water evaporator.

Fig. 40. Water evaporator SSARX model residual analysis for testing data.

The final model performance results are compared using 1-step to 10-step ahead predictions, and the testing results are shown in Fig. 38. For a small prediction horizon, the methods had similar mean squared error with RNN being a bit higher. When the prediction horizon grows, the linear dynamic models outperform the nonlinear dynamic models. The results are as expected based on the data interrogation. The terms selected by DALVEN with relatively large parameters are all linear terms (the first 5 selected terms are $y_{t-1}, y_{t-2}, y_{t-3}, x_{3,t-1}, \text{and} x_{3,t-2}$). The RNN software automatically selected the linear activation function, which provides a linear mapping between input and state.

For a relatively small prediction horizon, the difference between the linear state-space model, DALVEN, and RNN is small. As the prediction horizon grows, errors are propagated through time, and the model structure mismatch results in the linear state space models producing much more accurate predictions than DALVEN and RNN. At the final time step, the linear state-space model has half the prediction error of the RNN.

The state-space model using the SSARX algorithm with 17 states has the lowest MSE and is selected as the final model. The prediction results and residual analysis are shown in Figs. 39, 40, and 41. The residual histogram is symmetric and approximately normal. The final model residual has some moderate autocorrelation that cannot be explained by the linear state-space model, which is
consistent with the earlier observation from the nonlinearity tests of there being some minor nonlinearity. At this point, it would be an engineering decision whether to attempt to squeeze out that last bit of autocorrelation by using a nonlinear dynamic model. As compared to the linear static model (see Fig. 36), the model predictions are substantially improved, and most of the variation can be explained by the linear state-space model.

In summary, SPA correctly picks the best model structure with a systematic analysis procedure for this water evaporator case study.

7. Conclusion

SPA is proposed as a robust and automated framework for method selection and model construction for predictive data-driven modeling of manufacturing processes. For method selection, SPA assesses the salient data characteristics by using data interrogation tests. This information is then used to select the best-in-class data analytics methods within the pre-selected candidate methods based on expert domain knowledge of process data analytics. The users’ objectives are also taken into consideration during method selection. For automated model construction, rigorous cross-validation procedures are implemented. This systematic framework facilitates consistent applications of best practices and continuous improvement of tools and decision making in manufacturing processes.

Case studies involving real manufacturing datasets with different data properties demonstrate the effectiveness of the procedure. For a four-stage evaporator, a state-space identification method is selected that has half the long-term prediction error than a recurrent neural network. For a combined cycle power plant, machine learning methods are selected that have up to 30% lower mean-squared error than partial least squares. Although this article only considers predictive modeling, SPA can be generalized and integrated with other applications such as classification and process monitoring. Future work will be based on the design and implementation of SPA for other application scenarios.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Weike Sun: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Visualization. Richard D. Braatz: Conceptualization, Methodology, Formal analysis, Investigation, Resources, Data curation, Writing - original draft, Writing - review & editing, Supervision, Funding acquisition.

Acknowledgement

This work is supported by the U.S. Food and Drug Administration. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the financial sponsor.

Appendix A. Case study results for linear static method comparisons

Below are the detailed linear method comparison results by RR, PLS, EN, SPLS, and also OLS if applicable.

A1. Type I cases: number of features < number of training samples

There are 9 cases in total of the Type I comparison for RR, PLS, EN, and SPLS. The box-plots for MSEs of testing data calculated over 1000 simulations are shown in Fig. 42. For each case, there are 4 noise levels. The final results by four linear methods are compared with the OLS model regressed on the correct features (OLS-part), which provides a lower bound on the MSE, and the OLS model regressed on all the features (OLS-full), which provides an upper bound on the MSE.

Fig. 43 plots the median of the testing MSEs for the simulated Type I examples and the 95% confidence interval estimated by using the bootstrap with $B = 1000$ resamplings on the 1000 MSEs.

A2. Type II cases: number of features $> \text{number of training samples}$

There are 8 cases in total of the Type II comparison. The box-plots for MSEs of testing data calculated over 1000 simulations are plotted in Fig. 44. For each case, there are 4 noise levels. The final results by four linear methods are compared with the RR/PLS models regressed on the correct features, which provide a lower bound on the MSE for each case.

Fig. 45 plots the median of the testing MSEs for the simulated Type II examples and the 95% confidence interval estimated by using the bootstrap with $B = 1000$ resamplings on the 1000 MSEs.

Appendix B. Case study results for linear dynamic method comparisons

Below are the comparison results of different linear dynamic methods in SPA. The datasets are all from the demonstration datasets in the ADAPTx software.

B1. Case 1

The dataset generated by a 2-input 2-output system (dem1 in ADAPTx software) is shown in Fig. 46. There are 500 observations, and the system involves feedback, state noise, and measurement noise. Four hundred points are used to train the model, while the last 100 points are used to test the trained model. The k-step ahead testing prediction MSE results are shown in Fig. 47. MOESP has the lowest mean squared error for testing data, with the other methods giving nearly the same MSE.

When the number of training samples is reduced to 300 and the remaining 200 is used for testing, the relative performance between different methods changed as shown in Fig. 48. In this case, MOESP has the worst MSE of the methods.

B2. Case 2

The dataset generated by a 2-input 2-output system (dem3 in ADAPTx software) is shown in Fig. 49. There are 500 observations, and the system involves feedback, state noise, and measurement noise. Four hundred points are used to train the model, while the last 100 points are used to test the trained model. The k-step ahead testing prediction MSE results are shown in Fig. 50. In this case, the largest MSE for testing data is for SSARX and the ADAPTx implementation of CVA.

When the number of training samples is reduced to 300, and the rest 200 is used for testing, the relative performance between different methods changed as shown in Fig. 51. MOESP has by far the largest MSE for the testing data whereas CVA-ADAPTx has the lowest.
B3. Case 3

The dataset generated by a 1-input-1-output system (Gasfuoi in ADAPTx software) is shown in Fig. 52. There are 296 observations, and the system involves feedback, state noise, and measurement noise. Two hundred points are used to train the model, while the last 96 points are used to test the trained model. The k-step ahead testing prediction MSE results are shown in Fig. 53. The CVA implementation in Matlab and auto have the lowest errors.

B4. Case 4

The dataset generated by a 1-input-1-output system (Gasfubi in ADAPTx software) is shown in Fig. 54. There are 296 observations, and the system involves feedback, state noise, and measurement noise. Two hundred points are used to train the model, while the last 96 points are used to test the trained model. The k-step ahead testing prediction MSE results are shown in Fig. 55. The CVA implementation in ADAPTx has the lowest MSE for testing data whereas SSARX has the highest.

B5. Case 5

This dataset for a single-output system with seasonality (dem5 in ADAPTx software) is shown in Fig. 56. The data are the number of recorded live births in each month. The abrupt jump in the series around sample 80 is the end of World War II. Although not a dataset from manufacturing, similar nearly periodic behavior appears in many chemical systems including batch and semibatch processes used to manufacture pharmaceuticals and semiconductor devices.

There are a total of 474 points in the dataset, where 374 points are used for training and the rest 100 points are used for testing. The testing results are shown in Fig. 57. In this case, ADAPTx has the highest MSE, followed by MOESP.
Fig. 42. Continued
Fig. 43. Type I case: testing MSE medians with 95% confidence interval for RR, PLS, EN, and SPLS.
Fig. 44. Type II case: testing MSE distributions for RR/PLS/EN/SPLS and RR/PLS with only useful predictors.
Fig. 44. Continued
Fig. 45. Type II case: testing MSE medians with 95% confidence interval for RR, PLS, EN, and SPLS.

Fig. 46. Case 1 data for linear dynamic model comparison.

Fig. 47. k-step ahead prediction MSE results for Case 1 testing set by different methods with 400 training samples. The label 'auto' indicates the option for automatic algorithm selection in N4SID in Matlab, which selects the weights in the subspace algorithm from MOESP, CVA, and SSARX.
Fig. 48. $k$-step ahead prediction MSE results for Case 1 testing set by different methods with 300 training samples.

Fig. 49. Case 2 data for linear dynamic model comparison.

Fig. 50. $k$-step ahead prediction MSE results for Case 2 testing set by different methods with 400 training samples.

Fig. 51. $k$-step ahead prediction MSE results for Case 2 testing set by different methods with 300 training samples.

Fig. 52. Case 3 data for linear dynamic model comparison.

Fig. 53. $k$-step ahead prediction MSE results for Case 3 testing set by different methods.
Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compchemeng.2020.107134

References


Fig. 54. Case 4 data for linear dynamic model comparison.

Fig. 55. k-step prediction MSE results for Case 4 testing set by different methods.

Fig. 56. Case 5 data for linear dynamic model comparison.

Fig. 57. k-step ahead prediction MSE results for Case 5 testing set by different methods.


