Augmenting Adaptation with Retrospective Model Correction for Non-Stationary Regression Problems

Rashid Bakirov, Bogdan Gabrys and Damien Fay
Data Science Institute, Bournemouth University, United Kingdom
Email: {rbakirov,bgabrys,dfay}@bournemouth.ac.uk

***Abstract*—Existing adaptive predictive methods often use multiple adaptive mechanisms as part of their coping strategy in non-stationary environments. We address a scenario when selective deployment of these adaptive mechanisms is possible. In this case, deploying each adaptive mechanism results in different candidate models, and only one of these candidates is chosen to make predictions on the subsequent data. After observing the error of each of candidate, it is possible to revert the current model to the one which had the least error. We call this strategy retrospective model correction. In this work we aim to investigate the benefits of such approach. As a vehicle for the investigation we use an adaptive ensemble method for regression in batch learning mode which employs several adaptive mechanisms to react to changes in the data. Using real world data from the process industry we show empirically that the retrospective model correction is indeed beneficial for the predictive accuracy, especially for the weaker adaptive mechanisms.**

I. Introduction

With the current advances in data storage, database and data transmission technologies, mining streaming data has become a critical part of many processes. Many models which are used to make predictions on streaming data are static, in the sense that they do not learn on current data and hence remain unchanged. However, there exists a class of models, online learning models, which are capable of adding observations from the stream to their training sets. In spite of the fact that these models utilise the data as it arrives, there can still arise situations where the underlying assumptions of the model no longer hold. We call such settings dynamic environments, where changes in data distribution [1], change in features' relevance [2], non-symmetrical noise levels [3] are common. It has been shown that many changes in the environment which are no longer being reflected in the model contribute to the deterioration of model's accuracy over time [4]–[7]. This requires constant manual retraining and readjustment of the models which is often expensive, time consuming and in some cases impossible - for example when the historical data is not available any more. Various approaches have been proposed to tackle this issue by making the model adapt itself to the possible changes in environment while avoiding its complete retraining. Here we define *adaptation* as changes in model parameters and or structure in response to changes in the data stream.

Typically there are several possible ways or *adaptive mechanisms* (AMs) to adapt a given model. Recently attention has focused on selective or flexible deployment of differing AMs, which has been shown to result in the superior performance [8]. In this scenario, the adaptation is achieved by deploying one of multiple AMs, which changes the state of the existing model. At this time it is not known whether the choice of AM is optimal for the incoming data. However after the subsequent data is fully observed, it becomes possible to retrospectively identify the optimal AM. It is then possible to revert the current model to the model which is created by deploying this optimal AM. We call this strategy a retrospective model correction. This work provides analysis of retrospective correction in a system with multiple adaptive mechanisms. To the best of our knowledge, this idea hasn't been explored in the context of adaptation. Here we investigate whether, and for which cases retrospective correction is beneficial for predictive accuracy.

We focus on the batch prediction scenario, where data arrives in large segments called batches. This is a common industrial scenario, especially in the chemical, microelectronics and pharmaceutical areas [9]. For the experiments a regression algorithm which uses an ensemble of locally weighted experts to make a prediction was constructed. The local experts approach is a popular (described for example in [10]) way of aggregation of multiple models' predictions. Using local experts allows one to investigate a number of adaptive mechanisms, such as adding/removing experts and changing experts' local weights. In addition, the local expert models may include their own adaptation mechanisms. At the time when the true values become available, the model can then be adapted.

The main finding of this work is that in our settings, in the majority of cases the retrospective model correction strategy improves the predictive accuracy of the model. The magnitude of improvement differs depending on the adaptive mechanism and the dataset, with weaker AMs benefiting the most.

The paper is structured as follows; related work is presented in the Section II, Section III presents mathematical formulation of the framework of a system with multiple adaptive elements in batch streaming scenario. Section IV introduces our algorithm, which was used for the experimentation, describes the adaptive mechanisms which form the adaptive part of the algorithm and gives a short overview of Recursive Partial Least Squares (RPLS) [11], a base model for our algorithm. Section V introduces the datasets from the chemical processing industry on which the experiment were performed, the experimental methodology and results. We conclude by giving our final remarks in Section VI.

II. Related Work

Recently, especially for industrial processes, many regression methods which explicitly consider the adaptation of the model, such as [8], [12]–[26], have been proposed. Adaptivity is usually achieved by building a predictive model using a) the latest historical data and/or b) the historical data which is the most similar to the current data. Adaptive methods often use multiple models to make the final prediction, either by the weighted combination of their outputs([12]–[18], [22]–[24], [26]); known as ensemble models, or, more rarely, selecting one of them ([19], [22], [25]). Most of the models, or experts, are built on the subsets of historical data which represent different degrees of relation to the current data.

Ensemble methods date as far back as 1960's, when it was shown that combining multiple predictive models may give better results than using single models [27]. One of the advantages of ensemble methods is their ability to model local dependencies in the data, a classical example being [10]. This is achieved by weighting the models' predictions on a data instance by the location of this instance in the input space. Local ensemble learning was applied to regression ensembles in [12], [23], [24], [26], [28], [29]. These methods first identify the disjoint segments of the historical input space where the process produced outputs described by a common model, sometimes also called *receptive fields*. Then they build a model for each receptive field using Partial Least Squares (PLS) [30] or Support Vector Regression [31]. The models therefore describe different regions of the input space. The final prediction is a weighted average of all of the experts. Here, for each new data instance, the weights of experts depend on the location of the observed instance and in some cases the prediction. The AM used in [28] is based on change of models' local weights depending on their error. This model was extended in [12] to include adaptation of the base models using RPLS and further in [29] further extends the model to include creation of additional experts. [24], [26] uses adaptation of base models and adaptive weighting with [23] additionally introducing adaptive offset correction. Another local ensemble method, with a moving window and weights change AMs is described in [13].

Also popular in the literature are global regression ensembles [14]–[17]. These typically assign weights to experts based on their general performance, irrespective of the location of the input. Global ensemble methods use similar AMs to those described above. For instance, [14] adapts to changes by creating new experts and changing their weights. [17] includes AMs such as adaptation of base models via a moving window, changing experts weights and adding new experts. [16] additionally employs a boosting like *instance weighting* mechanism for resampling the training data. Both [16] and [17] may remove experts as well. [18] describes a method which uses an ensemble of univariate regressors for multivariate regression. It includes weighting of models and forgetting factor AMs. [32] proposes a *time difference* ensemble based on the distance between the current input and historical inputs. This method can use either moving window or just-in-time (creation of a model from most relevant instances) approaches for adaptation. [32] also uses just-in-time model creation with global performance based adaptive weighting.

From the analysis above we can see that there are many adaptive mechanisms which can be applied with ensemble methods. The mechanisms target different characteristics of the model; the error, the current location in the input space (or output space), and the temporal distance. The SABLE framework chosen here also includes such functionality. Most of the described work above have a common characteristic that whatever the AMs, they are applied at every time step in the same manner. In contrast the approaches proposed in [8], [16], [17], [22], [29], [32], [33] change the order of the adaptation. In particular, [29] creates new experts when existing ones are not built on the relevant data, [16], [17] create new experts when the predictive error on an instance is above a set threshold. In [32] the predictive accuracy is assessed to switch between two predictive models. Again the, predictive accuracy is used to choose between just-in-time model creation and offset update in [22]. [33] presents a plug and play architecture for preprocessing, adaptation and prediction which foresees the possibility of using different adaptation methods in a modular fashion, but does not address the method of AM selection. [8] introduces the Simple Adaptive Batch Learning Ensemble (SABLE) approach, the flexible adaptive framework which includes recursive update of the experts, update of their weights and creation/merger/removal of the experts. It explores several adaptation strategies, which are described in Table I.

Similarities may be found between retrospective correction and weight update backtrack mechanism, a key feature of Resilient Propagation neural network learning algorithm [34]. In fact, this mechanism can be considered as a special case of retrospective correction. To the best of our knowledge, the general notion and analysis of retrospective model correction strategy presented in the current work have not been given in the adaptation context before.

III. Formulation

A. Predictive model

We assume that the data is generated by a process which can be formulated as

$$y = \psi(\boldsymbol{x}) + E, \tag{1}$$

where ψ is an unknown function, E is unknown error, $\boldsymbol{x} = \{x_1, ..., x_m\}$ is an input data instance and y is the output value. Then we consider the predictive method at a time t as a function

$$\hat{y} = f_t(\boldsymbol{x}, \Theta_f), \tag{2}$$

$\hat{y}$ being the predicted output and Θ_f set of parameters of f.

B. Adaptation

We assume that we are operating in the batch streaming scenario, and receive a new input dataset, also called a batch, $\boldsymbol{x}_t$, at time t. The predictive model f_t generates prediction

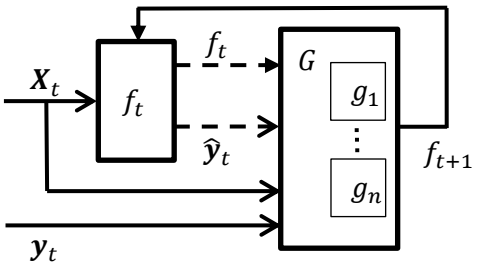

Fig. 1. Adaptation with multiple AMs. Optional inputs are shown with dashed lines.

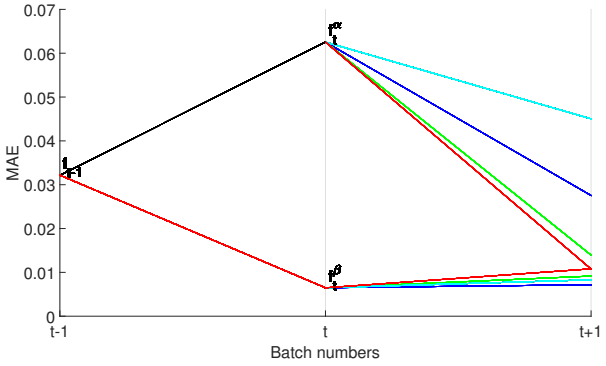

Fig. 2. Retrospective model correction effects. Each AM are is represented with a different color.

vector $\hat{\boldsymbol{y}}_t$. Subsequently observations of the output, $\boldsymbol{y}_t$, become available, so that $\boldsymbol{V}_t = \{\boldsymbol{x}_t, \boldsymbol{y}_t\}$. Then we consider an *adaptive mechanism* as an operator which generates an updated prediction function based on $\boldsymbol{V}_t$ and other optional inputs. This can be written as

$$g(f_t, \hat{\boldsymbol{y}}_t, \boldsymbol{V}_t, \Theta_g) : f_t \rightarrow f_{t+1}. \tag{3}$$

Here f_t and $\hat{\boldsymbol{y}}_t$ are optional arguments and Θ_g is the set of parameters of g.

We assume that we have multiple, different AMs $\{\emptyset, g_1, ..., g_h\} = G$ available. Datasets $\boldsymbol{V}_1, ..., \boldsymbol{V}_T$ arrive at timesteps $t = 1, ...T$. At each timestep any $g_t \subset G$ can be executed. We call a sequence $g_1, ..., g_T$ an *adaptation sequence*. In this formulation, we consider the combinations of different AMs as a separate new AM. Note that we also include the option of applying no adaptation at any particular stage denoted by the $\emptyset$. In this way, any combination of the AMs can be represented as a separate new AM. Figure 1 illustrates the proposed adaptation scheme.

C. Retrospective Model Correction

Deploying each possible AM, $g_t \in G$, at timestep $t-1$, produces different candidate models; $F_t = f_t^0, f_t^1, ..., f_t^H$. One of these will be chosen as an active model for prediction of the values for the next batch:

$$f_t = f_t^\alpha, \alpha \in \{0 \cdots H\}. \tag{4}$$

Once the true values of the batch, y_{t+1}, are known, it becomes possible to assess the performance of all the candidate models, using a desired criterion, for example mean absolute error (MAE), mean squared error (MSE) or others. Denoting this criterion ϵ, we can calculate:

$$\epsilon_t^0 = \langle f_t^0(x_t), y_t \rangle, \cdots, \epsilon_t^H = \langle f_t^H(x_t), y_t \rangle \tag{5}$$

and if there exists a model which outperformed the current model, replace the current model with it as:

$$f_t = f_t^\beta, \text{ where } \beta = \underset{i}{argmin}(\epsilon^i) \; i = 0 \cdots H. \tag{6}$$

Figure 2 shows a real example (from the Catalyst dataset, described in the Section V-B) of the effect retrospective model correction can have, where the graphs show the error after deploying different AMs. At time t, f^α results in a predictive MAE of 0.062. The subsequent predictions into $t+1$ result in MAE's varying from about 0.01 to 0.045 (depending on the next AM). However, by retrospectively changing the AM chosen at time t the predictions into $t+1$ results in MAE's less than or equal to 0.01.

IV. Algorithm

To perform experiments, we are using an adaptive method, SABLE[1], which is inspired by the ILLSA method [12]. SABLE operates on batches of data. An expert can be created from each batch of data. The experts are trained using PLS (see Section IV-D) as a base method. PLS was chosen because it is widely used for predictions in process industry where our datasets originate from, features inherent dimensionality reduction and has a recursive version (i.e. RPLS). The final prediction is the weighted sum of predictions of all experts where the weights are calculated as described below. When comparing to ILLSA, the novel elements of SABLE are that it is a batch learning algorithm, the proposed batch learning routine, the mechanism of recalculation of expert weights and the ability to modify the ensemble by adding additional experts during algorithm's operation, as well as merging similar experts. We list the main parts and adaptive mechanisms of SABLE in the following sections.

A. Building of Experts' Descriptors

The relative performance of experts varies in different parts of the input/output space. In order to quantify this a *descriptor* is used. Descriptors of experts are distributions of their weights with the aim to describe the area of expertise of the particular local expert. They describe the mappings from a particular input, x^m, and output, y to a weight, denoted $D_{i,m}(x^m, y)$, where m is the m-th input feature[2] and i is the i-th expert. The descriptor is constructed using a two-dimensional Parzen

[1]SABLE was previously described in [8], Sections 4, 5, 6. To make this work self contained, we partially repeat the description of the algorithm again in this section.

[2]For the base methods which transform the input space, such as PLS, the transformed input arguments are used instead of original ones.

window method [35] as:

$$D_{i,m}(x^m, y) = \frac{1}{||\boldsymbol{V}_i^{tr}||} \sum_{j=1}^{||\boldsymbol{V}_i^{tr}||} w(\boldsymbol{x}_j)\Phi(x^m, y, \mu_j^m, \boldsymbol{\Sigma}) \quad (7)$$

where $\boldsymbol{V}_i^{tr}$ is the training data of the i-th expert, $||\boldsymbol{V}_i^{tr}||$ is the number of instances it includes, $w(\boldsymbol{x}_j)$ is the weight of sample point's contribution which is defined below, $\boldsymbol{x}_j$ is the jth sample of $\boldsymbol{V}_i^{tr}$, $\Phi(\mu_j^m, \boldsymbol{\Sigma})$ is a two-dimensional Gaussian kernel function with mean value $\mu = (x_j^m, y_j)$ and variance matrix $\boldsymbol{\Sigma} \in \Re^{2\times 2}$ with the kernel width, σ, at the diagonal positions. σ, is unknown and must be estimated as a hyperparameter of the overall algorithm.[3]

The weights $w(\boldsymbol{x}_j)$ for the construction of the descriptors (see Eq. 7) are proportional to the prediction error of the respective local expert:

$$w(\boldsymbol{x}_j) = \exp(-(\hat{y}_j - y_j)^2) \quad (8)$$

Finally, considering that there are M input variables and I models, the descriptors may be represented by a matrix, $\mathcal{D} \in \Re^{M\times I}$ called the *descriptor matrix*.

B. Combination of Experts' Predictions

During the run-time phase, SABLE must make a prediction of the target variable given a batch of new data samples. This is done using a set of trained local experts $\mathcal{F}$ and descriptors $\mathcal{D}$. Each expert makes a prediction $\hat{y}_i$ for a data instance $\boldsymbol{x}$. The final prediction $\hat{y}$ is the weighted sum of the local experts' predictions:

$$\hat{y} = \sum_{i=1}^{I} v_i(\boldsymbol{x}, \hat{y}_i)\hat{y}_i \quad (9)$$

where $v_i(\boldsymbol{x}, \hat{y}_i)$ is the weight of the i-th local expert's prediction. The weights are calculated using the descriptors, which estimate the performance of the experts in the different regions of the input space. This can be expressed as the posterior probability of the i-th expert given the test sample $\boldsymbol{x}$ and the local expert prediction $\hat{y}_i$:

$$v_i(\boldsymbol{x}, \hat{y}_i) = p(i|\boldsymbol{x}, \hat{y}_i) = \frac{p(\boldsymbol{x}, \hat{y}_i|i)p(i)}{\Sigma_{j=1}^{I} p(\boldsymbol{x}, \hat{y}_j|j)p(j)}, \quad (10)$$

where $p(i)$ is the *a priori* probability of the i-th expert[4], $\Sigma_{j=1}^{I} p(\boldsymbol{x}, \hat{y}_j)p(j)$ is a normalisation factor and $p(\boldsymbol{x}, \hat{y}_i|i)$ is the likelihood of $\boldsymbol{x}$ given the expert, which can be calculated by reading the descriptors at the positions defined by the sample $\boldsymbol{x}$ and prediction $\hat{y}_i$:

$$p(\boldsymbol{x}, \hat{y}_i|i) = \prod_{m=1}^{M} p(x^m, \hat{y}_i|i) = \prod_{m=1}^{M} D_{i,m}(x^m, \hat{y}_i). \quad (11)$$

Eq. 11 shows that the descriptors D_m are sampled at the positions which are given on one hand by the scalar value x^m of the m-th feature of the sample point $\boldsymbol{x}$ and on the other hand by the predicted output $\hat{y}_i$ of the local expert corresponding to the ith receptive field. Sampling the descriptors at the positions of the predicted outputs may be potentially ineffective because the predicted value is not necessarily similar to the correct target value. However the correct target values are not available at the time of the prediction. The rationale for this approach is that the local expert is likely to be more accurate if it generates a prediction which conforms with an area occupied by a large number of true values during the training phase.

To reduce the number of redundant experts, at time t those that deliver similar predictions on batch $\boldsymbol{V}_t$ are merged, making use of the linear base model. If the base model is non linear and merging is not straightforward, then a pruning strategy as in [28] can be considered. There, the weight vectors of all experts on a batch of data are pairwise compared and if their similarity is higher than a defined threshold, one of the experts is removed. Prediction vectors can also be used to measure the similarity between different experts.

C. Adaptive Mechanisms

The SABLE algorithm allows the use of adaptive mechanisms which are deployed as soon as the true values for the batch are available, before predicting on the next batch. It is also possible that none of them are deployed. The AMs are described in the following sections.

1) Batch Learning: The simplest AM augments existing data with the data from the new batch and retrains the model. Given predictions of each expert $f_i \in \mathcal{F}$ on $\boldsymbol{V}$, $\hat{Y} = \{\hat{\boldsymbol{y}}_1, ..., \hat{\boldsymbol{y}}_I\}$ and measurements of the actual values, $\boldsymbol{y}$, $\boldsymbol{V}$ is partitioned into subsets in the following fashion:

$$k = \underset{i}{argmin}(|\hat{y}_{i,j} - y_j|),\ i = 1 \cdots I,\ [\boldsymbol{x_j}, y_j] \in \boldsymbol{V}_k \quad (12)$$

for every instance $[\boldsymbol{x}_j, y_j] \in \boldsymbol{V}$. This creates subsets $\boldsymbol{V}_i, i = 1...I$ such that $\cup_{i=1}^{I} \boldsymbol{V}_i = \boldsymbol{V}$. Then each expert is updated using the respective dataset $\boldsymbol{V}_i$. This process updates experts only with the instances where they achieve the most accurate predictions, thus encouraging the specialisation of experts and ensuring that a single data instance is not used in the training data of multiple experts. This AM will be denoted as **AM1** in the description of the experiments below.

2) Batch Learning With Forgetting: This AM is similar to one described in Section IV-C1 but includes a forgetting factor (see Section IV-D) which reduces the weight of the experts historical training data, making the most recent data more important. This AM will be denoted as **AM2**.

3) Recalculation of Descriptors: This AM recalculates the local descriptors, $\mathcal{D}$, using the new batch as described in the Section IV-A. The previous weights are discarded. This AM will be denoted as **AM3**.

4) Creation of New Experts: New expert f_{new} is created from the new batch at time t. Then it is checked if any of the experts from $\mathcal{F}_{t-1} \cup f_{new}$, where $\mathcal{F}_{t-1}$ is the experts pool at the time $t-1$, can be merged or pruned. Finally the descriptors of all resulting experts are calculated as described in the Section IV-A. This AM will be denoted as **AM4**.

[3] In this research we assume an isotropic kernel weighted by the variance is sufficient for simplicity and also to reduce the number of parameters to be estimated.

[4] Equal for all local experts in our implementation, different values could be used to prioritize experts.

The listed AMs are different from each other in terms of how strongly they adapt the model. AM0, AM1 and AM3 have relatively weaker, and AM2 and AM4 stronger effects.

D. Recursive Partial Least Squares

In our experiments we use RPLS as a base algorithm. The advantages of this algorithm are that it can be updated without requiring the historical data and that the merging of two models can be easily realised. RPLS is extension of the Partial Least Squares algorithm, both of which are popular in process industry predictive modelling tasks. PLS projects the scaled and mean centered multidimensional input data $\boldsymbol{X} \in \mathcal{R}^{n \times m}$ and output data $Y \in \mathcal{R}^{n \times p}$, where n is the number of data instances, m is the number of input variables and p is the number of output variables, to separate latent variables as:

$$\boldsymbol{X} = \boldsymbol{T}\boldsymbol{P}^T + \boldsymbol{E} \tag{13}$$

$$\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{Q}^T + \boldsymbol{F}. \tag{14}$$

where $\boldsymbol{T} \in \mathcal{R}^{n \times l}$, with $l \leq m$ denotes the number of latent variables and $\boldsymbol{U} \in \mathcal{R}^{n \times l}$ represents the score matrices, $\boldsymbol{P} \in \mathcal{R}^{m \times l}$ and $\boldsymbol{Q} \in \mathcal{R}^{p \times l}$ represents the corresponding loading matrices, and $\boldsymbol{E}$ and $\boldsymbol{F}$ are the input and output data residuals. Then the score matrices $\boldsymbol{T}$ and $\boldsymbol{U}$ consist of so called latent vectors:

$$\boldsymbol{T} = [\boldsymbol{t}_1, ..., \boldsymbol{t}_l], \text{ where } \boldsymbol{t}_i \in \mathcal{R}^{n \times 1} \tag{15}$$

$$\boldsymbol{U} = [\boldsymbol{u}_1, ..., \boldsymbol{u}_l], \text{ where } \boldsymbol{u}_l \in \mathcal{R}^{n \times 1}. \tag{16}$$

where the column vectors $\boldsymbol{p} \in \mathcal{R}^{m \times 1}$ and $\boldsymbol{q} \in \mathcal{R}^{m \times 1}$ of the loading matrices, $\boldsymbol{P}$ and $\boldsymbol{Q}$, represent the contributions of the input and output variables to the mutually orthonormal latent vectors, $\boldsymbol{t}$ and $\boldsymbol{u}$, respectively. Equations 15 and 16 constitute the PLS outer model. Afterwards a regression model, which is also called the PLS inner model, between the latent scores is constructed:

$$\boldsymbol{U} = \boldsymbol{T}\boldsymbol{B} + \boldsymbol{R}, \tag{17}$$

where $\boldsymbol{B} \in \mathcal{R}^{l \times l}$ is a diagonal matrix of regression weights which minimizes the regression residuals, $\boldsymbol{R}$. Then the estimates of $\boldsymbol{Y}$, $\hat{\boldsymbol{Y}}$, can be expressed as:

$$\hat{\boldsymbol{Y}} = \boldsymbol{T}\boldsymbol{B}\boldsymbol{Q}^T, \tag{18}$$

There are different methods to calculate the required vectors $\boldsymbol{t}$, $\boldsymbol{p}$, $\boldsymbol{u}$, $\boldsymbol{q}$ and $\boldsymbol{b}$. One of the most popular ones, NIPALS [36], updates latent vectors in an iterative way. After each iteration, the explained covariance is removed from the data:

$$\boldsymbol{X}_{i+1} = \boldsymbol{X}_i - \boldsymbol{t}_i \boldsymbol{p}_i^T \tag{19}$$

$$\boldsymbol{Y}_{i+1} = \boldsymbol{Y}_i - \boldsymbol{u}_i \boldsymbol{q}_i^T. \tag{20}$$

The subsequent $(i+1)$-th vectors are calculated by the resulting new input and output data $\boldsymbol{X}_{i+1}$ and $\boldsymbol{Y}_{i+1}$. Recursive PLS, which uses NIPALS, updates the matrices $\boldsymbol{P}$, $\boldsymbol{T}$, $\boldsymbol{Q}$, $\boldsymbol{U}$ and $\boldsymbol{B}$ when the new data becomes available, on either a sample-by-sample (incremental) or a batch basis. In this work we are using batch adaptation. It works by applying PLS on the new batch and constructs new input and output matrices as follows:

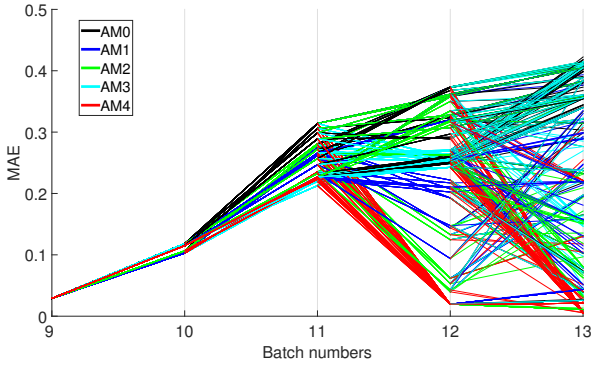

Fig. 3. Exhaustive 4-step ahead search. The graphs show the error after deploying different AMs.

$$\boldsymbol{X}_{new} = \begin{bmatrix} \lambda \boldsymbol{P}_0^T \\ \boldsymbol{P}_1^T \end{bmatrix} \tag{21}$$

$$Y_{new} = \begin{bmatrix} \lambda \boldsymbol{B}_0 \boldsymbol{Q}_0^T \\ \boldsymbol{B}_1 \boldsymbol{Q}_1^T \end{bmatrix}, \tag{22}$$

where the matrices $\boldsymbol{P}_0$, $\boldsymbol{B}_0$ and $\boldsymbol{Q}_0$ describe the old model and $\boldsymbol{P}_1$, $\boldsymbol{B}_1$ and $\boldsymbol{Q}_1$ the new one created from the most recent batch. $0 \leq \lambda \leq 1$ is the forgetting factor which determines how much influence the historic data will have, with $\lambda = 0$ meaning zero influence and $\lambda = 1$ meaning that the historical data has the same influence as the new batch. After constructing the new input and output data matrices, PLS is applied producing the updated matrices. The condition for this update is that the number of latent variables must be equal to the rank of $\boldsymbol{X}$. This condition can be practically met by finding a number of latent variables a for which the error on the training data is less than the defined threshold close to 0.

V. Experimental Results

A. Methodology

For the more general analysis of AMs behaviour and how they are affected by the correction mechanism we have performed the limited 4-step ahead exhaustive deployment of all AMs; at every batch we deploy separately all of the 5 AMs, use the resulting models to predict the values from the next batch, record the errors, then adapt the five models using 5 AMs for each of them, etc. for 4 batches ahead. After this the model which gives the best performance on the original batch was chosen as the basis for the next four steps. This process is visualised on the Figure 3. Thus 625 (5^4) MAE and standard deviations values and prediction vectors for every batch are obtained. These represent exhaustive list of the last 4 AMs (e.g. AM0, AM0, AM0, AM0; AM0, AM0, AM0, AM1;...) deployed to arrive at the current prediction. To assess the usefulness of the retrospective correction mechanism for the actual prediction on the dataset, we additionally use it for configurations of SABLE, shown in the Table I. On each batch

TABLE I
EXPERIMENTAL CONFIGURATIONS OF SABLE.

Configuration name	Description
Sequence0	Using AM0 on every batch. This means that only the first batch of data is used to create an expert.
Sequence1	Using AM1 on every batch.
Sequence2	Using AM2 on every batch.
Sequence3	Using AM2 on every batch.
Sequence4	Using AM4 on every batch.
XVSelect	Selecting AM based on the current data batch using the cross-validatory as described in [8].
Optimal	Selecting AM based on the subsequent data batch[5] as described in [8] .
Retrain	A new model is trained from the current batch and the old one is discarded. No data partitioning is used.

TABLE II
SABLE PARAMETERS FOR DIFFERENT DATASETS

Dataset	Batch size	Descriptor mesh grid size	Descriptor update weighing	RPLS forgetting factor	Expert generation kernel base size
Catalyst	50	50x50	0, 1	0.5	1
Catalyst	100	100x100	0, 1	0.25	1
Catalyst	200	100x100	0, 1	0.5	1
Oxidizer	30	50x50	0.25, 0.75	0.5	1
Oxidizer	50	50x50	0, 1	0.25	0.01
Oxidizer	100	50x50	0, 1	0.25	0.01
Drier	50	50x50	0, 1	0.25	0.01
Drier	100	50x50	0, 1	0.5	0.1
Drier	200	50x50	0, 1	0.25	0.01

one of the AMs described in the Section IV-C or none of them (which is denoted by **AM0**) can be deployed. We use the AM configurations presented in Table I for AM selection.

We have performed our experiments on three datasets (Catalyst, Oxidizer and Drier; described in detail in the next section) from process industry with different batch sizes. As explained in [33], there is a need for adaptation for the Oxidizer and particularly for the Catalyst datasets, which is not apparent with the Drier dataset. Choosing these datasets will allow us to test AM sequences on data which exhibits different behaviours. We chose three different batch sizes for each dataset, using different SABLE parameters for each of them (for brevity purposes we will show the batch size next to the dataset name to indicate which batch size was used for the experiment - i.e. Catalyst50 stands for Catalyst dataset with batch size of 50 samples). Batch sizes and parameters used are shown in the Table II. These parameter combinations were empirically identified as the best among the ones which have been tried.

To calculate the significance of differences between the predictions of different configurations, the significance test of difference of two estimators' errors relying on the sample covariance, as described in [37], Section 3.2 was used.

B. Datasets

1) Catalyst Activation Dataset: This data set was used for the NiSIS 2006 competition[6]. It includes 14 sensor measurements like flows, concentrations and temperatures from a real process. The target variable is the simulation of catalyst's activity inside the reactor. The description of the reaction speed is taken from literature showing a strong non-linear dependency on temperature. Further complicated processes like cooling and catalyst decay contribute to changes in the data. The data set covers one year of operation of the plant. Many of the features exhibit high co-linearity and contain high number of outliers. The data includes 5,867 data samples and is presented to an algorithm in batches of 100 samples. We have removed two features with mostly missing and 0 values during preprocessing. The number of latent vectors for PLS was experimentally set to 12. Below we present the results with batch sizes of 50, 100 and 200, denoted respectively as Catalyst50, Catalyst100 and Catalyst200.

2) Thermal Oxidizer Dataset: This dataset deals with the prediction of the concentration of exhaust gas during an industrial process where the task is to predict the concentrations of NO_x in the exhaust gases. The data set consists of 36 input features which are hard sensor measurements. They are physical values like concentrations, flows, pressures and temperatures measured during the operation of the plant. The data set consists of 2,820 samples. This dataset is also affected by issues like data outliers or missing values. We split it into batches of 50 samples, 55 batches in total. The number of latent vectors for PLS was experimentally set to 3. Below we present the results with batch sizes of 30, 50 and 100, denoted respectively as Oxidizer30, Oxidizer50 and Oxidizer100.

3) Industrial Drier Dataset: The target value of this dataset describes the laboratory measurements of the residual humidity of the process product. The dataset has 19 input features, most of them being temperatures, pressures and humidities measured in the processing plant. The original dataset consists of 1,219 data samples covering almost seven months of the operation of the process. It consists of raw unprocessed data as recorded by the process information and measurement system. Many of the input variables show problems common in industrial data like measurement noise, missing values or data outliers. We have removed 3 input features which mostly consisted of missing data. The data is presented in batches of 100 data samples. The number of latent variables for PLS was experimentally set to 16. Below we present the results with batch sizes of 50, 100 and 200, denoted respectively as Drier50, Drier100 and Drier200.

C. Correction Mechanism's Effect on Different AMs

Recall that at the timestep $t-1$ there exists H different candidate models, $F_t = f_t^0, f_t^1, ..., f_t^H$, and after the full observation of V_t, the state of the model is reverted to the result of retrospective correction, f_t^β. Let us assume that a single AM g is applied to each $f_t^i \in F_t$ resulting in $F_{t+1} = g(f_t^0), g(f_t^1), ..., g(f_t^H) = f_{t+1}^0, f_{t+1}^1, ..., f_{t+1}^H$.

[5]Note that this is not applicable in the real life situations, as at the time t, the data $\boldsymbol{V}_{t+1}$ is not yet known. This configuration's shows achievable results using one step ahead optimal AM and will be used as a benchmark.

[6]http://www.nisis.risk-technologies.com/(S(ftalbreakikhjt34pd2xggfu))/filedown.aspx?file=125

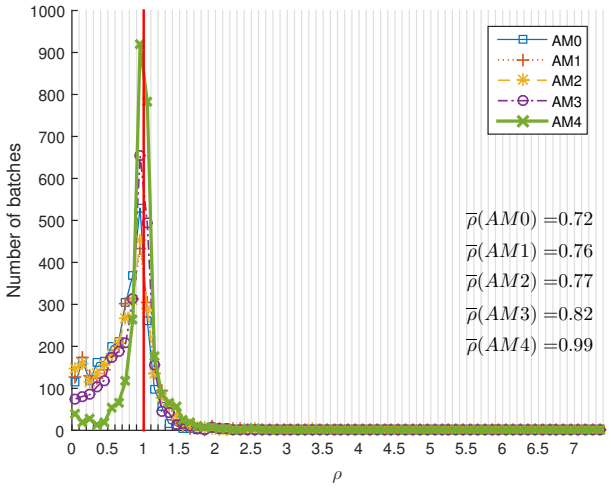

Fig. 4. Catalyst50 improvement ratio histograms

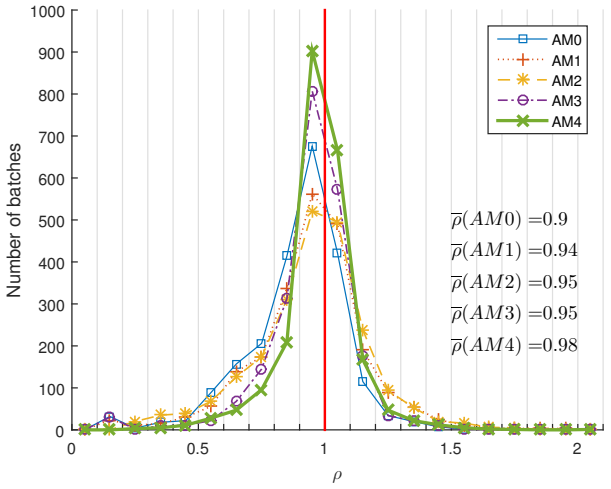

Fig. 5. Oxidizer30 improvement ratio histograms

Applying each $f_t^i \in F_t$ to the next batch, V_{t+1}, results in the errors $\epsilon_{t+1}^0 ... \epsilon_{t+1}^H$. We denote $g(f_t^\beta) = f_{t+1}^\beta$, its error $\epsilon_{t+1}^\beta = \langle f_{t+1}^\beta(x_{t+1}), y_{t+1} \rangle$, and the prior probabilities of g_i providing the lowest MAE ω_i. Then we consider the *improvement* ratio:

$$\rho = \frac{\epsilon_{t+1}^\beta}{\sum_{i=1}^{H} \omega_i \epsilon_{t+1}^i / H}. \tag{23}$$

This ratio compares the MAE when using the correction, to MAE when not. In other words it quantifies the improvement delivered by the retrospective correction.

Figures 4, 5, 6 show the histograms of the improvement ratio for each AM and average values of ρ. On the Catalyst50 (Figure 4), correction has noticeable positive impact for all of the AMs except AM4, as seen by the skew in the distribution towards values of $\rho < 1$. However, the improvement seen

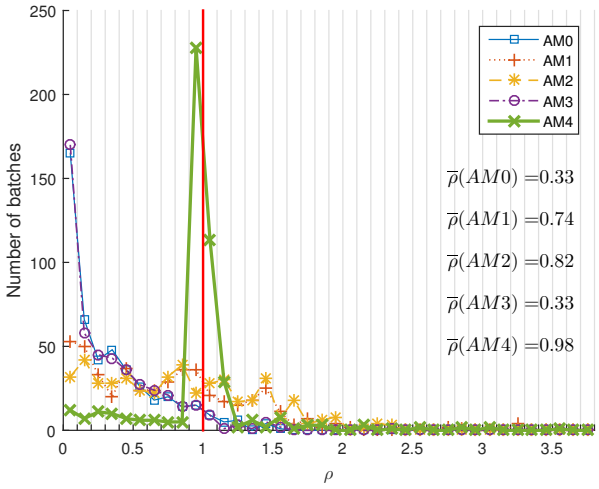

Fig. 6. Drier50 improvement ratio histograms

when using AM4 is not that apparent. The reason for this is the nature of AM4 which is able to actively totally renew the model by down weighting old experts and creating a new one, hence minimizing the effect of the existing model before the adaptation, which is what the correction mechanism modifies.

The result is not so clear for the Oxidizer30 dataset (Figure 5) but the distribution is still skewed to the left. This can be attributed to the low volatility of the data. Among all three datasets, Oxidizer30 is affected the least by the correction.

The distribution of ρ for Catalyst50 and Oxidizer30 is concentrated around 1. This means for the majority of cases for any AM, the improvement from the retrospective correction is minimal. However, the results for the Drier50 dataset (Figure 6)shows a somewhat different behaviour. Here the majority of batches for AM0 and AM3, and to lesser extent AM1 and AM2 are noticeably positively affected by the correction strategy. This difference is caused by the fact that this dataset is relatively stable and thus the most vulnerable to incorrect AM choices. Finally the histogram for AM4 is similar to the ones from previous datasets and is not noticeably affected by the correction. Finally, we note that varying the batch size did not have a significant effect on the conclusions drawn in the datasets above and so these results are not presented.

D. Retrospective Model Correction Applied to SABLE Configurations

Configurations of SABLE (Table I) with and without correction were then applied to all of the datasets. The results of the experiments are described in the Tables III, IV, and V. These results suggest that for the majority of cases, the error values of the methods with added correction are significantly lower than those that don't use correction. The results of corrected ***Sequence3***, ***SimpleRetrain*** and ***Optimal*** are added for completeness. We observe that these results are generally consistent with our findings above, with few exceptions. Exceptions can occur as these are results of a single AM

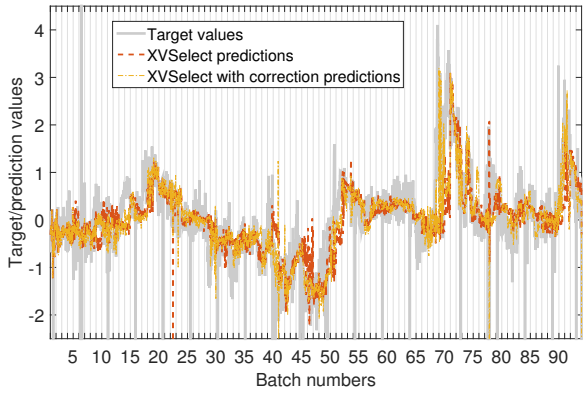

Fig. 7. Predictions on Oxidizer30

sequence, whereas the results in the Section V-C are based on many different sequences. As also noted in [8], ***XVSelect*** provides the most accurate predictions in most of the cases. Figure 7 shows the comparison of ***XVSelect***'s predictions on Oxidizer30, with and without retrospective correction.

VI. Conclusions and Discussion

The core aim of this paper was to investigate how retrospective model correction mechanisms affects the predictive accuracy using a range of popular AMs. We have conducted experiments on 3 datasets containing data from the process industry which display diverse properties and rates of change.

Using the AM which minimizes the MAE on the current batch (***Optimal***) always led to the lowest MAE. Thus, for our setup, it made sense using ***Optimal*** as a benchmark. It must be noted that it is possible that there exits an AM sequence which minimizes the error even further. However it must be sought in the set of all possible AM sequences, which is computationally prohibitive (H^t).

Our experiments show that retrospective model correction mechanism has a positive impact on the predictive accuracy of the model in many cases. The effects vary depending on the dataset and AM. We have established that the weaker AMs like AM0, AM1 and AM3 benefit most from the correction mechanism, while the most versatile AM4 is affected the least. The results of predictions on the datasets using SABLE, an algorithm with multiple AMs, are consistent with these findings. The configurations which involve the sequences of weaker AM and mixed sequences benefit from model correction the most.

Considering all of the above, we can conclude that while using an algorithm with multiple adaptive mechanisms, saving a limited history of model states and switching between them when needed, which is essentially what the retrospective model correction does, can be a simple and effective way of improving predictive accuracy of the models. Saving model states does not require saving the data and hence is inexpensive in terms of data storage.

Future research will investigate retrospective modelling combined with intelligent AM selection.

Acknowledgements

Research leading to these results has received funding from the EC within the Marie Curie Industry and Academia Partnerships and Pathways (IAPP) programme under grant agreement no. 251617. The authors would like to thank Evonik Industries AG for the provided datasets. MATLAB code by P. Kadlec and R. Grbić was used for the experiments.

References

[1] I. Zliobaite, "Combining Similarity in Time and Space for Training Set Formation under Concept Drift," *Intelligent Data Analysis*, vol. 15, no. 4, pp. 589–611, 2011.

[2] A. Fern and R. Givan, "Dynamic feature selection for hardware prediction," Purdue University, Tech. Rep., 2000.

[3] M. Schmidt and H. Lipson, "Learning noise," *Proceedings of the 9th annual conference on Genetic and evolutionary computation - GECCO '07*, pp. 1680–1685, 2007.

[4] J. C. Schlimmer and R. H. Granger, "Beyond incremental processing: Tracking Concept Drift," *AAAI-86 Proceedings*, pp. 502–507, 1986.

[5] W. N. Street and Y. S. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 377–382, 2001.

[6] R. Klinkenberg, "Learning drifting concepts : Example selection vs . example weighting," *Intelligent Data Analysis*, vol. 8, no. 3, pp. 281–300, 2004.

[7] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *The Journal of Machine Learning Research*, vol. Volume 8,, pp. 2755–2790, 2007.

[8] R. Bakirov, B. Gabrys, and D. Fay, "On sequences of different adaptive mechanisms in non-stationary regression problems," in *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–8.

[9] A. Cinar, S. J. Parulekar, C. Undey, and G. Birol, *Batch Fermentation: Modeling: Monitoring, and Control*. CRC Press, 2003.

[10] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Comput.*, vol. 3, no. 1, pp. 79–87, Mar. 1991.

[11] S. Joe Qin, "Recursive PLS algorithms for adaptive data modeling," *Computers & Chemical Engineering*, vol. 22, no. 4-5, pp. 503–514, Jan. 1998.

[12] P. Kadlec and B. Gabrys, "Local learning-based adaptive soft sensor for catalyst activation prediction," *AIChE Journal*, vol. 57, no. 5, pp. 1288–1301, May 2011.

[13] R. Grbić, D. Slišković, and P. Kadlec, "Adaptive soft sensor for online prediction and process monitoring based on a mixture of Gaussian process models," *Computers & Chemical Engineering*, vol. 58, pp. 84–97, Nov. 2013.

[14] H. Kaneko and K. Funatsu, "Adaptive soft sensor based on online support vector regression and Bayesian ensemble learning for various states in chemical plants," *Chemometrics and Intelligent Laboratory Systems*, vol. 137, pp. 57–66, Oct. 2014.

[15] ——, "Ensemble locally weighted partial least squares as a just-in-time modeling method," *AIChE Journal*, Nov. 2015.

[16] S. Gomes Soares and R. Araújo, "An on-line weighted ensemble of regressor models to handle concept drifts," *Engineering Applications of Artificial Intelligence*, vol. 37, pp. 392–406, Jan. 2015.

[17] ——, "A dynamic and on-line ensemble regression for changing environments," *Expert Systems with Applications*, vol. 42, no. 6, pp. 2935–2948, Apr. 2015.

[18] F. Souza and R. Araújo, "Online Mixture of Univariate Linear Regression Models for Adaptive Soft Sensors," in *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, May 2014, pp. 937–945.

[19] H. Jin, X. Chen, J. Yang, and L. Wu, "Adaptive soft sensor modeling framework based on just-in-time learning and kernel partial least squares regression for nonlinear multiphase batch processes," *Computers & Chemical Engineering*, vol. 71, pp. 77–93, Dec. 2014.

[20] W. Shao, X. Tian, and P. Wang, "Local Partial Least Squares Based Online Soft Sensing Method for Multi-output Processes with Adaptive Process States Division," *Chinese Journal of Chemical Engineering*, vol. 22, no. 7, pp. 828–836, Jul. 2014.

TABLE III
CATALYST DATASET RESULTS

Batch size	**50**		**100**		**200**	
Configuration	**No Corr.**	**Corr.**	**No Corr.**	**Corr.**	**No Corr.**	**Corr.**
Sequence0	0.310	**0.031**	0.279	**0.045**	0.361	**0.072**
Sequence1	0.138	**0.026**	0.147	**0.042**	0.161	0.073
Sequence2	0.023	0.026	**0.031**	0.039	0.058	0.073
Sequence4	0.037	**0.021**	**0.031**	0.034	**0.052**	0.053
XVSelect	0.020	**0.018***	**0.029**	0.032	**0.049***	0.051
Sequence3	0.026		0.046		0.067	
Retrain	0.024		0.028*		0.052	
Optimal	*0.015*		*0.024*		*0.040*	

TABLE IV
OXIDIZER DATASET RESULTS

Batch size	**30**		**50**		**100**	
Configuration	**No Corr.**	**Corr.**	**No Corr.**	**Corr.**	**No Corr.**	**Corr.**
Sequence0	0.675	**0.503**	0.760	**0.530**	0.779	**0.644**
Sequence1	0.602	**0.496**	0.640	**0.491**	0.662	0.663
Sequence2	0.464	0.484	0.490	0.489	**0.564**	0.666
Sequence4	0.459	0.431	0.504	**0.470**	0.543	**0.528***
XVSelect	0.471	**0.415***	0.484	0.464*	0.570	0.553
Sequence3	0.473		0.533		0.595	
Retrain	0.459		0.499		0.565	
Optimal	*0.373*		*0.396*		*0.480*	

TABLE V
DRIER DATASET RESULTS

Batch size	**50**		**100**		**200**	
Configuration	**No Corr.**	**Corr.**	**No Corr.**	**Corr.**	**No Corr.**	**Corr.**
Sequence0	7.68E-04	**9.78E-06**	5.41E-04	**1.43E-05**	3.87E-04	**1.34E-04**
Sequence1	8.98E-06	1.02E-05	**8.09E-06***	8.96E-06	**5.06E-05**	5.41E-05
Sequence2	3.04E-05	3.02E-05	**1.75E-05**	1.79E-05	5.28E-05	**5.09E-05**
Sequence4	9.86E-05	**4.06E-05**	1.43E-05	1.34E-05	8.77E-05	**6.41E-05**
XVSelect	9.27E-06	**6.95E-06***	1.20E-05	**1.12E-05**	4.67E-05*	4.67E-05*
Sequence3	9.78E-06		1.44E-05		1.34E-04	
Retrain	5.86E-05		2.59E-05		5.38E-05	
Optimal	*3.40E-06*		*3.15E-06*		*4.67E-05*	

[21] W. Ni, S. D. Brown, and R. Man, "A localized adaptive soft sensor for dynamic system modeling," *Chemical Engineering Science*, vol. 111, pp. 350–363, May 2014.

[22] H. Jin, X. Chen, J. Yang, L. Wang, and L. Wu, "Online local learning based adaptive soft sensor and its application to an industrial fed-batch chlortetracycline fermentation process," *Chemometrics and Intelligent Laboratory Systems*, vol. 143, pp. 58–78, Apr. 2015.

[23] H. Jin, X. Chen, J. Yang, H. Zhang, L. Wang, and L. Wu, "Multi-model adaptive soft sensor modeling method using local learning and online support vector regression for nonlinear time-variant batch processes," *Chemical Engineering Science*, vol. 131, pp. 282–303, Jul. 2015.

[24] W. Shao and X. Tian, "Adaptive soft sensor for quality prediction of chemical processes based on selective ensemble of local partial least squares models," *Chemical Engineering Research and Design*, vol. 95, pp. 113–132, Mar. 2015.

[25] W. Shao, X. Tian, P. Wang, X. Deng, and S. Chen, "Online soft sensor design using local partial least squares models with adaptive process state partition," *Chemometrics and Intelligent Laboratory Systems*, vol. 144, pp. 108–121, May 2015.

[26] W. Shao, X. Tian, and P. Wang, "Soft sensor development for nonlinear and time-varying processes based on supervised ensemble learning with improved process state partition," *Asia-Pacific Journal of Chemical Engineering*, vol. 10, no. 2, pp. 282–296, Mar. 2015.

[27] J. Bates and C. Granger, "The combination of forecasts," *OR*, vol. 20, no. 4, pp. 451–468, 1969.

[28] P. Kadlec and B. Gabrys, "Soft sensor based on adaptive local learning," *Advances in Neuro-Information Processing*, no. i, 2009.

[29] ——, "Adaptive on-line prediction soft sensing without historical data," in *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, Jul. 2010, pp. 1–8.

[30] H. Wold, *Estimation of Principal Components and Related Models by Iterative Least squares*. New York: Academic Press, 1966, pp. 391–420.

[31] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support Vector Regression Machines," *Neural Information Processing Systems*, vol. 1, pp. 155–161, 1996.

[32] H. Kaneko, T. Okada, and K. Funatsu, "Selective Use of Adaptive Soft Sensors Based on Process State," *Industrial & Engineering Chemistry Research*, vol. 53, no. 41, pp. 15 962–15 968, Oct. 2014.

[33] P. Kadlec and B. Gabrys, "Architecture for development of adaptive on-line prediction models," *Memetic Computing*, vol. 1, no. 4, pp. 241–269, Sep. 2009.

[34] M. Riedmiller and H. Braun, "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm," in *IEEE International Conference on Neural Networks*, 1993.

[35] E. Parzen, "On Estimation of a Probability Density Function and Mode," *The Annals of Mathematical Statistics*, vol. 33, no. 3, pp. pp. 1065–1076, 1962.

[36] P. Geladi and B. R. Kowalski, "Partial least-squares regression: a tutorial," *Analytica Chimica Acta*, vol. 185, pp. 1–17, Jan. 1986.

[37] B. Mizrach, "Forecast comparison in L2," Working Papers, No. 1995-24, Department of Economics, Rutgers, The State University of New Jersey, Tech. Rep. 908, 1996.