Local Learning-Based Adaptive Soft Sensor for Catalyst Activation Prediction

Petr Kadlec and Bogdan Gabrys
Smart Technology Research Centre, Computational Intelligence Research Group,
Bournemouth University, Poole BH12 5BB, U.K.

DOI 10.1002/aic.12346
Published online August 17, 2010 in Wiley Online Library (wileyonlinelibrary.com).

This work presents an algorithm for the development of adaptive soft sensors. The method is based on the local learning framework, where locally valid models are built and maintained. In this framework, it is possible to model nonlinear relationship between the input and output data by the means of a combination of linear models. The method provides the possibility to perform adaptation at two levels: (i) recursive adaptation of the local models and (ii) the adaptation of the combination weights. The dataset used for evaluation of the algorithm describes a polymerization reactor where the target value is a simulated catalyst activity in the reactor. This dataset is also used to evaluate the performance of the proposed algorithm. The results show that the traditional recursive partial least squares algorithm struggles to deliver accurate predictions. In contrast to this, by exploiting the two-level adaptation scheme, the proposed algorithm delivers more accurate results. © 2010 American Institute of Chemical Engineers *AIChE J,* 57: 1288–1301, 2011
Keywords: soft sensor, adaptive predictive modeling, polymerization process, local learning, ensemble methods

Introduction

Soft sensors have been established as useful tools, in the process industry which can provide additional information about the underlying processes. One can distinguish two fundamentally different soft sensor types, namely model-driven and data-driven soft sensors.[1] There are several other terms that can be used for these two types of models, such as parametric vs. nonparametric models, white box vs. black box models, and phenomenological vs. empirical models. The model-driven soft sensors use the chemical and physical principles underlying the process. The focus of this work is on the other type of soft sensors, i.e., the data-driven models. These models rely on data modeling techniques and are trained on data collected during the operation of the process. Although the field of application of soft sensors is broad, the dominant application type is the on-line prediction.[1] In the case of this application type, the soft sensor is trained using historical data before it is launched in the real-life environment where its task is to provide an on-line prediction on the basis of the incoming input data stream. The goal of the model is to provide a prediction of one or more difficult-to-measure variables based on some easy-to-measure variables. The latter variable type are process measurements such as temperatures, pressures, and material flows, which are automatically measured for process control and management purposes. On the other hand, the difficult-to-measure variables are often critical and have to be obtained either through expensive measuring devices or through manual chemical analysis carried out off-line in laboratories. Reviews of this type of soft sensor application have been published in Refs. 1–3.

The most common data-driven techniques applied for soft sensor modeling are the principal component regression[4] and partial least squares (PLS) method (see Recursive Partial

Correspondence concerning this article should be addressed to P. Kadlec at pkadlec@bournemouth.ac.uk.

Least Squares section). These techniques gained popularity because of their statistical background, ease of interpretability of the model, and because they deal efficiently with data colinearity, which is common among industrial datasets. Examples of soft sensor applications based on PCA/PLS have been discussed in Refs. 5–8. Multilayer perceptron (MLP)[9] is another predictive technique widely applied as a soft sensor.[10] The popularity of the MLPs originates in their ability to model nonlinear functions. As a soft sensor, MLPs are commonly applied as on-line prediction models (see e.g., Refs. 11 and 12). Furthermore, one can find a range of other computational learning methods like support vector machines[13] and neuro-fuzzy systems[14] applied as on-line prediction soft sensors.[15,16]

A particular drawback of many of the current soft sensors is their nonadaptive nature.[1] Traditionally, the models are not adaptive, and once deployed into the real-life operation, the model does not change (see Refs. 3 and 17). In contrast to this, the environment, in which the soft sensors are applied, is often changing.[18] The combination of the static model and the changing data leads to performance deterioration because the model usually represents the out-of-date state of the process as it was observed during the training phase. Recently, this issue has been recognized and several approaches to dealing with it were presented. Examples of incremental (or recursive) techniques and soft sensors based on them are as follows: (i) recursive PLS (RPLS; see the next section for more details); (ii) recursive exponentially weighted PLS[19]; and (iii) recursive PCA.[20] An example of a RPLS-based soft sensor can be found in Ref. 21. Another set of adaptive techniques is based on the moving window technique. Examples of adaptive soft sensors based on the moving window technique can be found in Refs. 22 and 23, where a batch process monitoring soft sensor based on the moving window PCA[22] and a process monitoring soft sensor based on the fast moving window PCA[23] have been published.

Despite the availability of several adaptation schemes, these methods are often difficult to handle. The selection of the parameters like the forgetting factors in the case of the recursive methods or the length of the adaptation window in the case of the moving windows-based techniques has a critical influence on the performance of the adaptive methods. Another possible issue with the adaptation is that some of the techniques might be inappropriate for the given problem. Such a case was shown in Ref. 24, where the moving window techniques lead to a deterioration of the performance of the soft sensor. Another particular issue of the moving windows-based methods also require the storage and access to a set of historical data, which could be difficult in some applications. In this work, it is also demonstrated that despite the adaptation, the traditional RPLS on its own does not deliver satisfactory predictions for this industrial problem.

To address the above-mentioned issues, the presented algorithm is based on the local learning framework. Local learning models consist of a set of simple models, which are trained on limited partitions of the historical data.[25] The way in which the global data is split into the local partitions depends on the algorithm. A common approach for this purpose is the k-means algorithm.[26] To obtain the (global) model's prediction, it is necessary to combine the predictions of the local experts. For this purpose, the presented algorithm stores performance maps, which allow to estimate the prediction performance of the local models. This framework allows to perform the model adaptation at two levels. On one hand, at the level of the local experts, the presented implementation is achieved by exploiting the adaptation capability of the RPLS method. The other level of adaptation is achieved by adapting the local expert's performance maps, which results in the updates of their combination weights. Additional advantage of this method is that both combination methods can be performed on sample-by-sample basis without the need to store any past data.

Another problem of current soft sensor development is the lack of the possibility to benchmark the developed algorithms on a set of standard datasets. In general machine learning research, this is widely done using the UCI dataset collection.[27] In soft sensing, the situation is different because very often the soft sensing algorithms are developed for very specific purposes, e.g., a particular process or parts of it. More than that, for understandable reasons, the data used for the testing of the algorithms in vast majority of the cases are protected by strict nondisclosure agreements. An exception to this rule are the two datasets, of a debutanizer column and a sulfur recovery unit provided for public use in Ref. 2. To further encourage other researchers and process operators to share their datasets with the public, we provide, with agreement of Evonik Industries AG, a dataset for general purpose soft sensors development and benchmarking.

The rest of this article is organized as follows: (Recursive) Partial Least Squares section briefly outlines the PLS algorithm and its recursive version RPLS. The novel adaptive soft sensor is discussed in detail in Incremental Local Learning Soft Sensing Algorithm section. In The Data Set section, a description of the data for public use is given. This dataset is also used for the evaluation of the presented algorithm in Experiments section. Finally, this work is concluded in the last section.

Recursive Partial Least Squares

In this section, the most common soft sensing technique and its adaptive (recursive) version are outlined. The PLS method was originally proposed in Ref. 28. The goal of the algorithm is to project the scaled and mean-centered input data $X \in \mathbb{R}^{n \times m}$ (where n is the number of available data points and m is the number of easy-to-measure variables) and output data $Y \in \mathbb{R}^{n \times p}$ (where p is the number of difficult-to-measure variables that are supposed to be predicted) to separate latent variables:

$$X = TP^T + E \tag{1}$$

$$Y = UQ^T + F, \tag{2}$$

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

$$T = [\mathbf{t}_1, \ldots, \mathbf{t}_l] \qquad \text{with } \mathbf{t}_i \in \mathbb{R}^{n \times 1} \tag{3}$$

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_l] \qquad \text{with } \mathbf{u}_i \in \mathbb{R}^{n \times 1}. \tag{4}$$

The latent vectors, which are orthonormal to each other (i.e., $\mathbf{t}_i^T \mathbf{t}_j = 0 \; \forall_{i \neq j}$) are a more compact description of the

input data. The column vectors $\mathbf{p} \in \mathbb{R}^{m \times 1}$ and $\mathbf{q} \in \mathbb{R}^{p \times 1}$ of the loading matrices P and Q represent the contributions of the input and output variables to the latent vectors $\mathbf{t}$ and $\mathbf{u}$, respectively. Equations 1 and 2 are also called the outer model of the PLS method.[29]

Subsequently, the PLS method builds a regression model between the latent scores (the PLS inner model):

$$U = TB + R, \tag{5}$$

where $B \in \mathbb{R}^{l \times l}$ is a diagonal matrix of regression weights that is calculated such as to minimize the regression residuals R. The estimates $\widetilde{Y}$ of Y are consequently:

$$\widetilde{Y} = TBQ^T. \tag{6}$$

There are several ways to calculate the required vectors $\mathbf{t}$, $\mathbf{p}$, $\mathbf{u}$, $\mathbf{q}$, and $\mathbf{b}$. A particularly popular algorithm for the calculation of PLS is the NIPALS algorithm.[30] The NIPALS algorithm calculates one latent vector after the other in an iterative way. After each iteration, the explained covariance is removed from the data:

$$X_{i+1} = X_i - \mathbf{t_i}\mathbf{p_i}^T \tag{7}$$

$$Y_{i+1} = Y_i - \mathbf{u_i}\mathbf{q_i}^T, \tag{8}$$

which is followed by the calculation of the next, i.e., $(i + 1)$th vectors for the PLS outer and inner models using the new data matrices X_{i+1} and Y_{i+1}. The number of calculated latent dimensions is usually established using cross-validation or some other parameter optimisation techniques.

In the off-line modeling scenario, the matrices P, T, Q, U, and B are calculated during the training phase based on the batch of historical data. However, as discussed in the Introduction section as well as in Ref. 29, this approach is often problematic because the process and its data are changing over the time. An advantage of the PLS algorithm is that it has the ability to incrementally integrate new data. The RPLS method has been introduced in Ref. 31 and further clarified in Ref. 29. It can be used to adapt the model in several ways: (i) on sample-by-sample basis; (ii) by integrating a new batch of data; or (iii) by merging two PLS models.

In the first case, which is exploited in this work, the model update is achieved by merging the old model represented by the matrices P, B, and Q with the latest data samples $\mathbf{x}_i$, $\mathbf{y}_i$:

$$X^{\text{new}} = \begin{bmatrix} \lambda P^T, \\ \mathbf{x}_i \end{bmatrix} \quad Y^{\text{new}} = \begin{bmatrix} \lambda BQ^T, \\ \mathbf{y}_i \end{bmatrix}. \tag{9}$$

The forgetting factor λ defines the strength of the adaptation. The lower value this factor has the smaller the influence of the previous model, which results in a faster adaptation to the new data. After the expansion, the PLS algorithm can be applied to X^{new}, Y^{new} as usual (see Eqs. 1 and 2). To be able to perform the above updates, the number of latent variables has to be selected to be equal to the rank of X.[29] Practically, this condition can be assured by finding a number of latent variables a, which fulfills:

$$\|E_a\| \leq \epsilon \quad \text{with } \epsilon \geq 0. \tag{10}$$

In general, the number of latent variables a required for the recursive operation can differ from the optimal number of latent variables required for the modeling.

Incremental Local Learning Soft Sensing Algorithm

The proposed soft sensing algorithm, (incremental local learning soft sensing algorithm) (ILLSA), can be split into the following steps:

- Construction of receptive fields
- Training of local models/experts
- Building of receptive field descriptors
- Combination of local experts
- On-line operation and adaptation of the model.

The above steps will be discussed in detail in the following sections.

Receptive fields construction

The aim of this step is to divide the historical data into partitions called receptive fields, which represent different states of the process (in this work referred to as data concepts). The notion of a concept is linked to the area where a model, called landmarker, provides constant performance. The decrease of performance of the landmarker is interpreted as a new data concept that triggers the building of a new receptive field.

Provided the historical dataset $\mathcal{D}^{\text{hist}}$, the first step of the algorithm is training the landmarker using samples from an initial window $\mathcal{D}^{\text{init}}$, which is a subset of the historical data:

$$\mathcal{D}^{\text{init}} = \left\{X^{\text{init}}, \mathbf{y}^{\text{init}}\right\} = \left\{(\mathbf{x}_i^{\text{init}}, y_i^{\text{init}})\right\}_{i=a}^{a+n^{\text{init}}}, \tag{11}$$

where a is the index of the first sample in the current receptive field and n^{init} is the length of the initial window (an input parameter of the algorithm).

Provided the initial set, the landmarker f^{lm} can be trained and the residual vector $\mathbf{r}^{\text{init}}$ of the landmarker's prediction on the training data can be calculated as follows:

$$\mathbf{r}^{\text{init}} = \mathbf{y}^{\text{init}} - f^{\text{lm}}(X^{\text{init}}), \tag{12}$$

where $f^{\text{lm}}(X^{\text{init}})$ are the predictions of the landmarker.

The next step is shifting the window one step forward $(s = 1)$, while keeping its size constant:

$$\mathcal{D}^{\text{shifted}} = (X^{\text{shifted}}, \mathbf{y}^{\text{shifted}}) := \left\{(\mathbf{x}_i, y_i)\right\}_{i=k+s}^{k+s+n^{\text{init}}} \quad \text{with } s = 1, \tag{13}$$

and calculating the new residual values $\mathbf{r}^{\text{shifted}}(s)$ of the landmarker's prediction using the shifted data window:

$$\mathbf{r}^{\text{shifted}}(s) = \mathbf{y}^{\text{shifted}} - f^{\text{lm}}(X^{\text{shifted}}) \tag{14}$$

Following this, the two residual vectors ($\mathbf{r}^{\text{init}}$ and $\mathbf{r}^{\text{shifted}}(s)$) are tested for a statistically significant difference using the t-test.[32] As long as the null hypothesis remains valid, it can be assumed that the performance of the landmarker on the data within the shifted window is comparable with the performance on the training data and thus that the data samples, within the

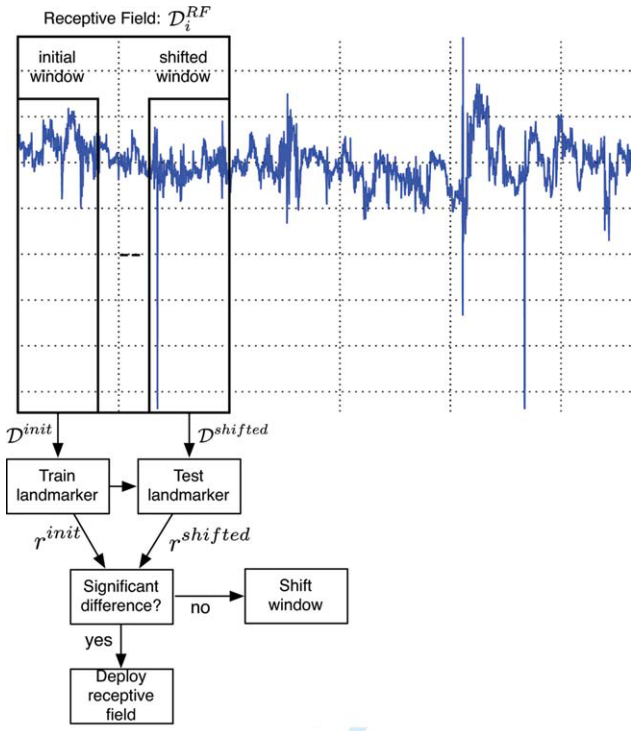

Figure 1. The receptive field construction process based on the detection of states in the data.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

shifted window $\mathcal{D}^{\text{shifted}}$, belong to the same process state as the samples from the initial window $\mathcal{D}^{\text{init}}$. This procedure is repeated, i.e, the window is shifted, as long as the null hypothesis of the significance test remains valid:

$$s_i^{\text{final}} = \underset{s\in[1,\ldots,n^{\text{train}}-k]}{\text{argmin}} \; (t\text{-}test(\mathbf{r}^{\text{init}}, \mathbf{r}^{\text{shifted}}(s)) == 1), \tag{15}$$

where n^{train} is the number of samples in the historical dataset and s^{final} corresponds to the first sample for which the t-test rejects the null hypothesis, and thus there is a significant difference in the residuals. The significance level of the t-test is an important parameter because it has a strong influence on the size of the receptive fields. However, in practical situations, the significance level can be fixed to 0.05 because the same effect can be achieved by manipulating the size of the initial window size n^{init}, which is already an input parameter of the algorithm.

Finally, the receptive field is built on the basis of the following data samples:

$$\mathcal{D}_r^{\text{RF}} = \{(\mathbf{x}_i, y_i)\}_{i=k}^{k+n^{\text{init}}+s_i^{\text{final}}-1} \tag{16}$$

and the algorithm can move to the next receptive field by constructing a new initial window. This is constructed by taking the last n^{init} samples of the previous receptive field (i.e., using the last shifted window of the previous receptive field), which results in a partial overlap of the receptive fields (Figure 2):

$$\mathcal{D}^{\text{init}} := \mathcal{D}^{\text{shifted}} = \{(\mathbf{x}_i, y_i)\}_{i=a+s^{\text{final}}}^{a+s^{\text{final}}+n^{\text{init}}}, \tag{17}$$

$$a = a + s^{\text{final}}. \tag{18}$$

The procedure of receptive field building is graphically illustrated in Figure 1.

By choosing a linear modeling technique (e.g., linear regression), the resulting receptive fields represent partitions of the historical with linear relation between the input and output variables. This is important if linear predictive methods, such as the PLS as it is the case in this work, are used for building the local experts at later stages of this algorithm.

The outcome of this stage is a set of receptive fields $\mathcal{D}^{\text{RF}}$, each corresponding to a concept of the historical data:

$$\mathcal{D}^{\text{RF}} := \{\mathcal{D}_i^{\text{RF}}\}_{i=1}^{r}, \tag{19}$$

where r is the number of built receptive fields. An example of the outcome from the data that was split into three receptive fields can be found in Figure 2.

The proposed partitioning algorithm was compared with the k-means[26] algorithm in preliminary experiments. It leads not only to a higher performance of the overall method but also has the advantage that it does not require the a priori setting of the number of partitions/clusters as it is the case with k-means.

Local experts training

After identifying the receptive fields, one model, called *local expert* f^{LE}, is trained for each of the receptive fields. In the proposed algorithm, the local experts are based on the PLS technique (see Section 2 for more details). This has several advantages including the possibility of incremental adaptation shown in Recursive Partial Least Squares section. Another benefit of this predictive method is that it automatically reduces the dimensionality of the input data. This property is critical because process industry data are often highly colinear.

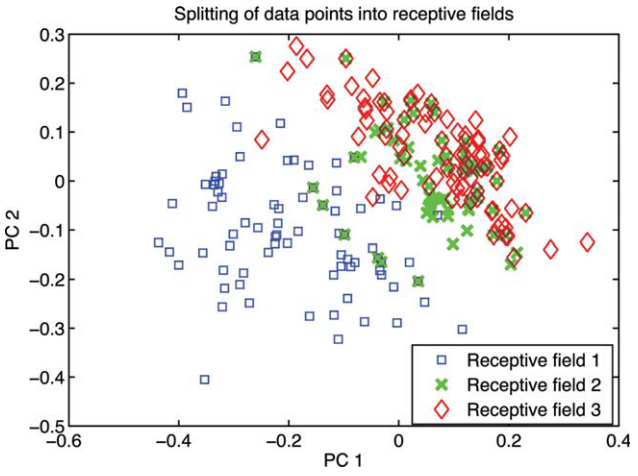

Figure 2. An example of splitting the training data into three receptive fields (displayed after projection to two-dimensional space by means of principal component analysis).

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

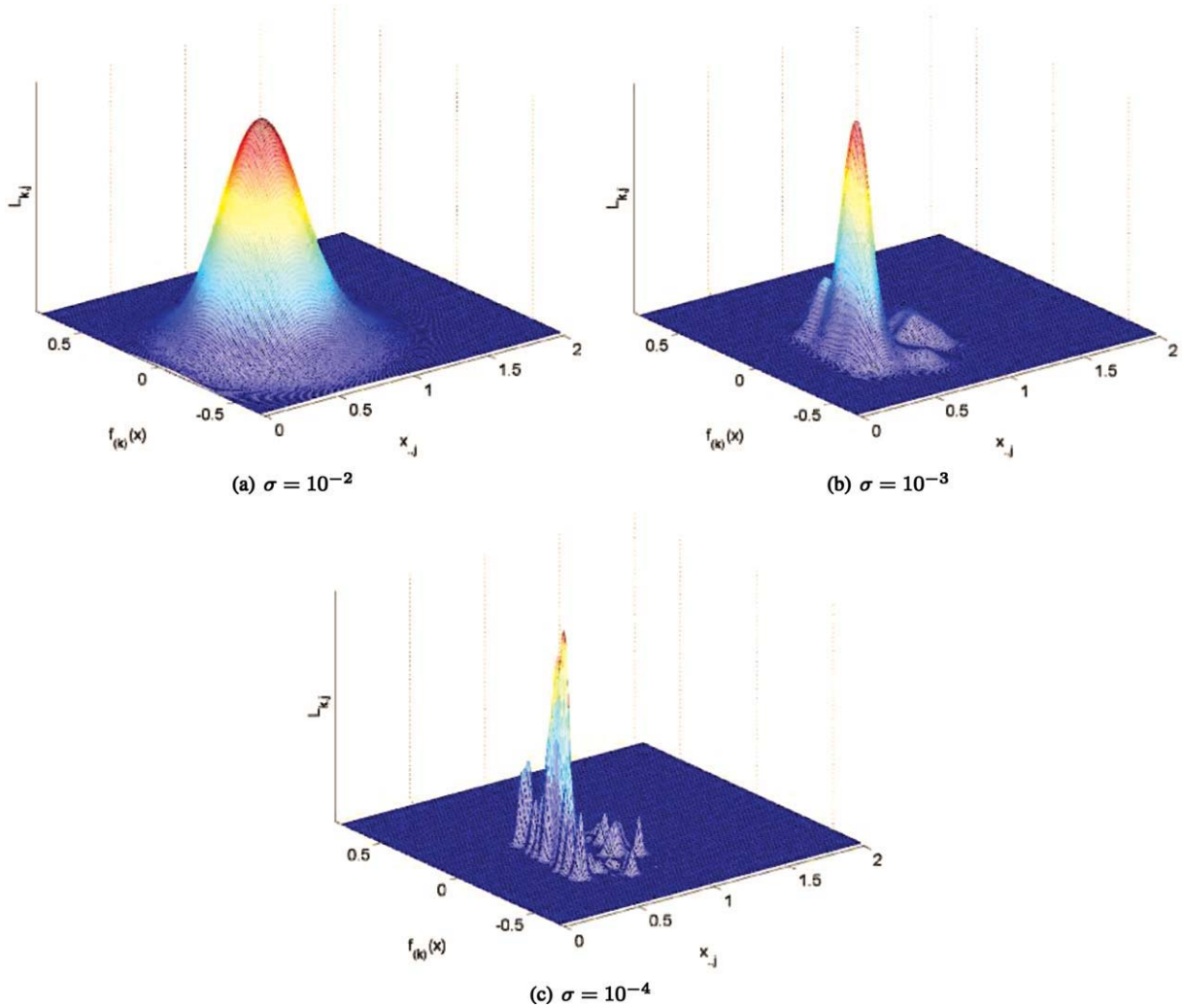

Figure 3. Local expert descriptor $L_{k,j}$ with different settings of σ.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

After this step, there is a set of trained local experts $\mathcal{F}^{\mathrm{LE}}$:

$$\mathcal{F}^{\mathrm{LE}} := \left\{ f_{(k)}^{\mathrm{LE}} \right\}_{k=1}^{r}. \tag{20}$$

Local experts descriptor building

The next step toward the final model is building descriptors for the local experts, which will be later used to estimate the prediction accuracy of the local experts for the given input sample. As such, the aim of the descriptors is to describe the area of expertise of each local expert. This is approached by building a set of two-dimensional probability density functions $L_{k,j}$, where j is the index of the PLS latent variable and k is the index of the local expert (or receptive field). The descriptors are constructed using a weighted two-dimensional Parzen window method [33]:

$$L_{k,j} = \frac{1}{\left\| \mathcal{D}_k^{\mathrm{RF}} \right\|} \sum_{t_{k,j} \in \mathcal{D}_k^{\mathrm{RF}}} w_k(t_{k,j}) \Phi(\mu, \Sigma), \tag{21}$$

where $\| \mathcal{D}_k^{\mathrm{RF}} \|$ is the number of samples in the kth receptive field, $t_{k,j}$ is the value of jth latent vector corresponding to the data sample $\mathbf{x}$ (i.e., $\mathbf{x} = \mathbf{t}_k \mathbf{p}_k^T$ and $\mathbf{t}_k = [t_{k,1}, \ldots, t_{k,l_k}]^T$), $w_k(t_{k,j})$ is the weight of the same sample point (for more details see below), and $\Phi(\mu, \Sigma)$ is a two-dimensional Gaussian kernel function with mean value at the positions defined by $\mu := \{t_{k,j}, y_i\}$ and variance matrix Σ (a diagonal 2×2 matrix with the kernel width σ at the diagonal positions—an input parameter of the algorithm). The kernel width defines the size of the neighborhood, which is influenced by each sample. For simplicity reasons, the variance is kept equal in both dimensions, but the approach can be easily extended to a more general case with different kernel widths along the two dimensions. Figure 3 shows an example of the descriptor for three different settings of the kernel size σ. One can see that with smaller kernel sizes, the descriptor gets more detailed, which involves a potential danger of low generalization capability and overfitting of the descriptor. On the other hand, two large kernels can lead to too strong generalization and loss of detail. The peaks in the figures indicate the area of the input–output space where the given local expert performs better than the remaining local experts.

The weights w_k for the construction of the descriptors (see Eq. 21) are set proportionally to the inverted quadratic prediction error of the local experts for the sample $\mathbf{x}_i$:

$$w_k(\mathbf{x}_i) = \exp(-(f_{(k)}^{LE}(\mathbf{x}_i) - y_i)^2). \quad (22)$$

Weighting the contribution of each sample by the prediction performance of the corresponding local experts assures that the descriptors model the local experts' area of expertise in the input–output space and as such can be later sampled to estimate the local experts' performance given the input data and its prediction.

The final descriptor $\mathcal{L}$ is a set of two-dimensional descriptors:

$$\mathcal{L} := \left\{(L_{k,j})\right\}_{k=1;\, j=1}^{r;\, l_k}, \quad (23)$$

where l_k is the number of PLS latent variables for the kth local expert and r is the number of receptive fields (as well as local experts).

Local experts combination

After the training phase, the soft sensor consist of the trained local experts $\mathcal{F}^{\mathrm{LE}}$ and receptive field descriptors $\mathcal{L}$. From the set of predictors $\mathcal{F}^{\mathrm{LE}}$, each of its members f^{LE} is making predictions of the target value given the input samples. To obtain the final predictions y^{final} during the on-line phase, these predictions have to be combined.

In what follows, the combination method will be described in the Bayesian framework. In general terms, the combined prediction is a weighted sum of the local experts' predictions:

$$y^{\mathrm{final}} = \sum_{k=1}^{r}\left(v_k\left(\mathbf{t}_k, f_{(k)}^{\mathrm{LE}}(\mathbf{x})\right) f_{(k)}^{\mathrm{LE}}(\mathbf{x})\right), \quad (24)$$

where $f_{(k)}^{\mathrm{LE}}(\mathbf{x})$ are the predictions given in the on-line data sample and v_k is the combination weight of the kth local expert's prediction. These weights are read from the local expert descriptors. Because the descriptors store maps of the local experts' performance in the input–output space, they can be interpreted as an effective estimation of the prediction performance of the local experts for the given on-line data sample. The weights can be expressed as the posterior probability of the kth local expert given the sample $\mathbf{x}$ and the prediction of the local expert for this sample $f_{(k)}^{\mathrm{LE}}(\mathbf{x})$:

$$v_k(\mathbf{t}_k, f_{(k)}^{\mathrm{LE}}) = p(k|\mathbf{t}_k, f_{(k)}^{\mathrm{LE}}) = \frac{p(\mathbf{t}_k, f_{(k)}^{\mathrm{LE}}|k)p(k)}{\sum_k p(\mathbf{t}_k, f_{(k)}^{\mathrm{LE}})}, \quad (25)$$

where $p(k)$ is the a priori probability of the kth local expert (in this implementation, equal for all local experts, but in general it can be used to prioritize between them), $\sum_k p(\mathbf{t}_k, f_{(k)}^{\mathrm{LE}})$ is a normalization factor, and $p(\mathbf{t}_k, f_{(k)}^{\mathrm{LE}}|k)$ is the likelihood of $\mathbf{t}_k$ and the local expert, which can be calculated by reading the descriptor of the kth local expert:

$$p(\mathbf{t}_k, f_{(k)}^{\mathrm{LE}}|k) = \prod_{j=1}^{l_k} p\left(t_{k,j}, f_{(k)}^{\mathrm{LE}}|k\right) = \prod_{j=1}^{l_k} L_{k,j}(t_{k,j}, f_{(k)}^{\mathrm{LE}}(\mathbf{x})\Big). \quad (26)$$

Equation 26 shows that the descriptors $L_{k,j}$ are read at positions that are given by the scalar value $t_{k,j}$ (which is again the value of the jth latent vector of the kth PLS model for the current data sample) and at the position of the predicted output $f_{(k)}^{\mathrm{LE}}(\mathbf{x})$ of the same local expert. Sampling the descriptors at the positions of the predicted outputs may be potentially ineffective because the predicted value does not necessarily need to be similar to the correct target value. However, as the correct target values are not available during the on-line phase at the time of the prediction, this is the only way to read the values from the descriptors. Furthermore, the rationale for this approach is that the local expert is likely to make the correct prediction if it generates a prediction that conforms with an area that was occupied by a large number of accurate predictions in the past.

As the multiplication of the contributions from the different variables in Eq. 26 requires linear independence of the variables, it is important to operate in the space of the latent variables that results from the local PLS preprocessing where the colinearity of the variables is dealt with.

Soft sensor adaptation

In the proposed method, two levels of adaptation can be distinguished: (i) adaptation at the level of the local experts and (ii) adaptation of the combinations weights. Neither of the adaptations requires the storage of any past data sample as they both work on a sample-by-sample basis.

Local expert adaptation. The adaptation of the local experts is done by the means of the algorithm shown in the Recursive Partial Least Squares section. Furthermore, the adaptation strength λ of the local experts is modified according to the amount of the responsibility of the local expert to the overall prediction. These values are represented by the weights v_i of the local experts. The implemented adaptation strategy is a winner-takes-all approach, where the model with the highest weight is adapted much stronger than all other models:

$$\lambda_j = \begin{cases} 0.5 & j = i \\ 1 & \text{otherwise} \end{cases} \qquad \text{with } i = \underset{i=1\ldots r}{\operatorname{argmax}}\ v_i \text{ and } j = 1\ldots r \quad (27)$$

This approach ensures that the model which contributes the most to the prediction is strongly focused on the prediction of the current data. At the same time, this method deals automatically with the setting of the forgetting factor for the local experts, which, in general, is difficult in the case of the traditional RPLS.

Local expert descriptor adaptation. The descriptors can also be modified each time a correct target value y is received. Provided this value, a feedback about the performance of the local experts in the form of the quadratic error $e_{(k)}$ of the kth local expert's prediction $f_{(k)}^{\mathrm{LE}}$ is calculated as follows:

$$e_k(\mathbf{x}) = (y - f_{(k)}^{\mathrm{LE}}(\mathbf{x}))^2. \quad (28)$$

The error is further mapped on a performance index u_k:

$$u_k = \exp\left(-\frac{e_k - \mathrm{med}(\mathbf{e})}{\mathrm{med}(\mathbf{e})\log(2)}\right), \quad \text{with } \mathbf{e} = [e_1, \ldots, e_r] \quad (29)$$

where med($\mathbf{e}$) is the median squared error across the local experts and r is the number of local experts. The above

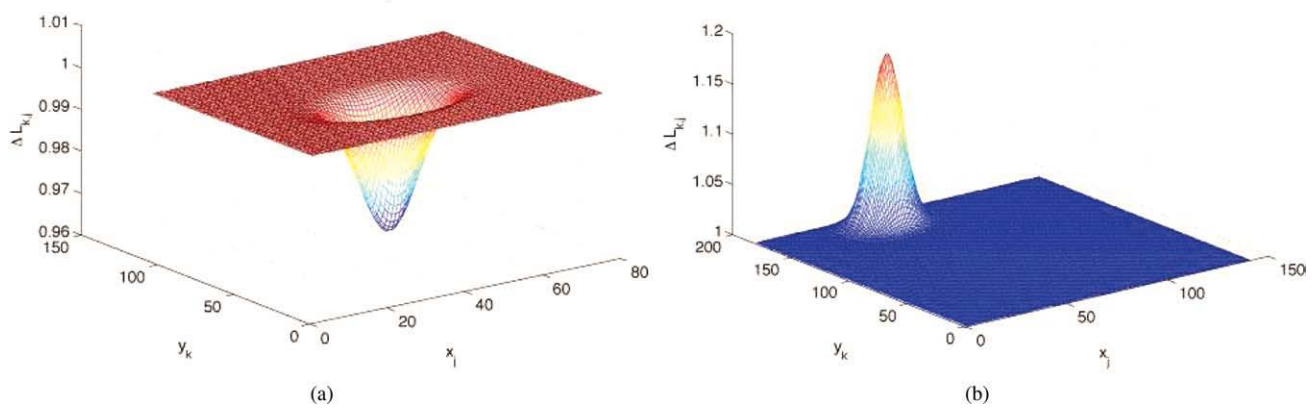

Figure 4. Adaptation masks $\Delta L_{k,j}$ for the modification of the local expert descriptors.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

mapping transforms the prediction error in such a way that the best performing local expert receives a weight equal to 1 and the weights of the remaining local experts decay exponentially with the increasing error, while the median error is mapped to the value 0.5.

This mapping function, together with Eq. 30, leads to a decrease of the neighborhood of the current sample within the receptive field descriptors (see Figure 4a for the adaptation mask in this case) for local experts with weak performance. Contrary to this, descriptors of local experts whose performance is better than the average are increased in the neighborhood of the current sample $\mathbf{x}$ (see Figure 4b for an example of such an adaptation mask). Overall, this approach leads to an increase of areas within the descriptors where the local expert performs well and to a decrease of such areas that can be better predicted by another local expert.

Equations 30 and 31 describe the adaptation process:

$$\Delta L_{k,j} = \Phi(\mu, \Sigma)(u_k - 0.5), \quad \text{with } \mu = \left\{t_{k,j}, f_k^{\mathrm{LE}}(\mathbf{x})\right\}, \quad (30)$$

where $\Delta L_{k,j}$ is a two-dimensional Gaussian adaptation mask for the jth latent variable and the kth receptive field (or local expert). Further on Σ is the variance matrix for the Gaussian kernel. These values of the 2×2 matrix (consisting of σ^{adapt} at the diagonal positions) define the size of the neighborhood of the current sampling point that is being modified by the adaptation mask (an input parameter of the algorithm).

Finally, the descriptors can be adapted using the modification masks in the following way:

$$L_{k,j}^{\mathrm{new}} = L_{k,j} \cdot \Delta L_{k,j}, \quad (31)$$

where $\cdot$ is the Hadamard matrix product.

Model transparency and interpretability. One of the advantages of the proposed algorithm is the possibility to interpret the predictions despite increased complexity of the model. In fact, it provides similar possibilities to interpret the results as the frequently used PLS-based soft sensors. Because the final prediction is a convex combination of predictions of the PLS-based local experts (see Eq. 24), it is straightforward to combine the contribution plots of the local experts in the same and thus to obtain the influence of the data variables (together with the influence of the particular local experts) on the final prediction. There are several ways for calculating the contributions plots (see e.g., Refs. 34 and 35). Assuming that $c_{(k),j}$ is the contribution of the kth local expert and the jth variable to the prediction, the contribution of the same variable using overall model is simply:

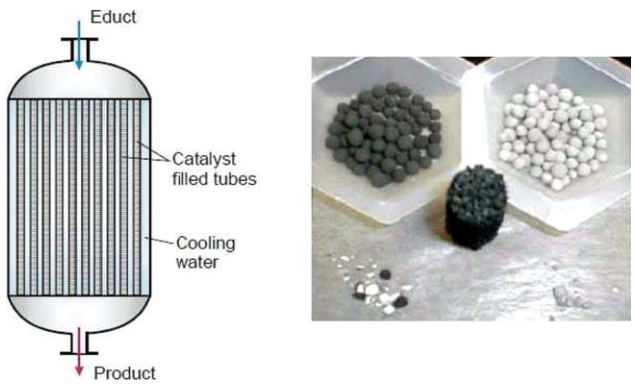

Figure 5. The reactor and product related to the catalyst activation data.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

Table 1. The Variables of the Catalyst Activation Dataset

No.	Description	Units
1	Time	days
2	Measured flow of air	kg/hr
3	Measured flow of combustible gas	kg/gr
4	Measured concentration of combustible component in the combustible gas (mass fraction)	n.a.
5	Total feed temperature	°C
6	Cooling temperature	°C
7	Temperature at length 1/20 of reactor length	°C
8	Temperature at length 2/20 of reactor length	°C
9	Temperature at length 4/20 of reactor length	°C
10	Temperature at length 7/20 of reactor length	°C
11	Temperature at length 11/20 of reactor length	°C
12	Temperature at length 16/20 of reactor length	°C
13	Temperature at length 20/20 of reactor length	°C
14	Product concentration of oxygen (mass fraction)	n.a
15	Product concentration of combustible component (mass fraction)	n.a

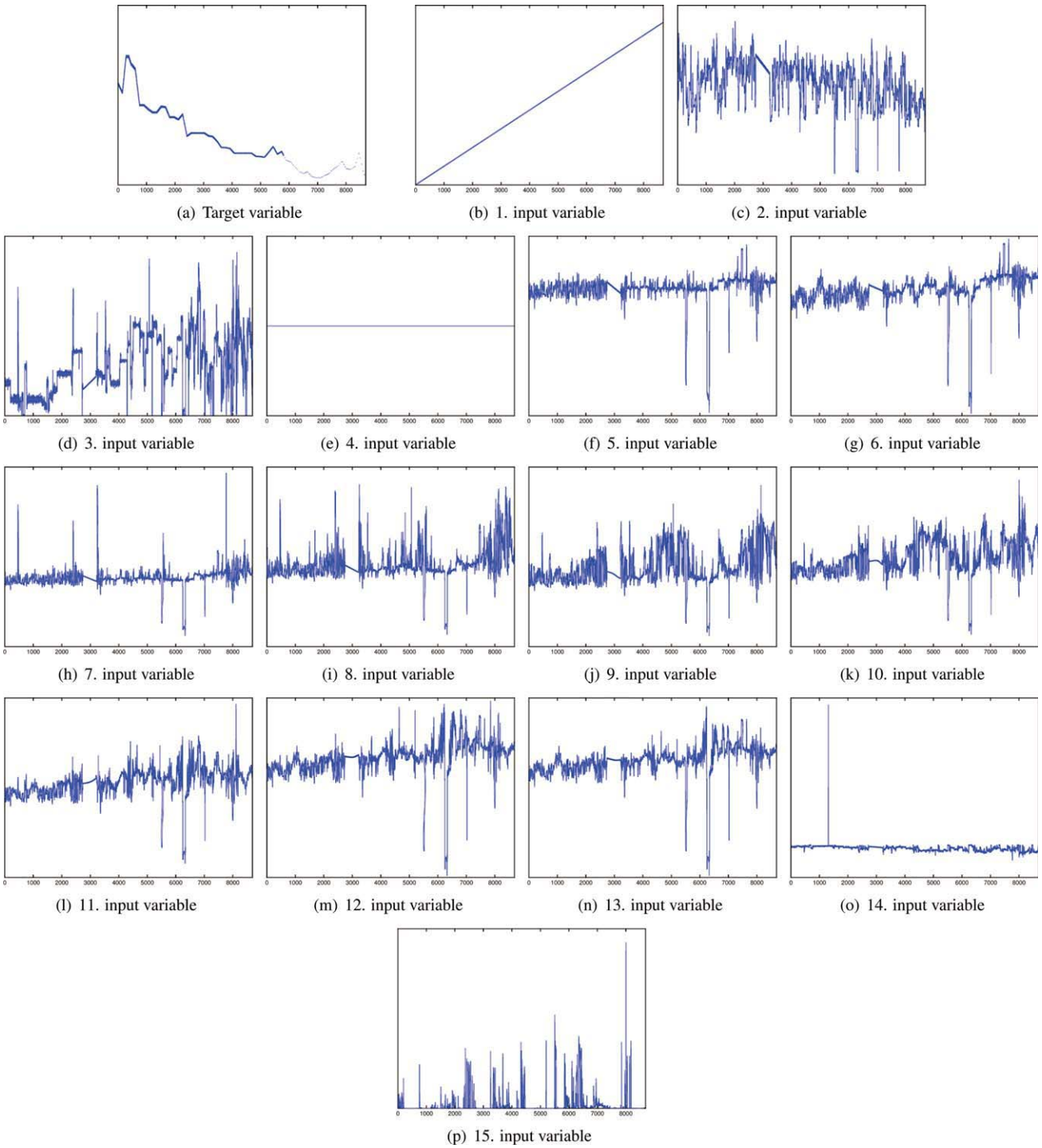

Figure 6. The catalyst activation dataset.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

$$c_j = \sum_k v_k c_{(k),j}. \tag{32}$$

The dataset

In this work, a dataset for benchmarking of adaptive soft sensing algorithms is presented and made available for public use. The dataset describes a polymerization reactor. The reactor consists of approximately 1000 tubes filled with catalyst, which is used to oxidize a gaseous feed. It is cooled with a coolant that is supposed to be at constant temperature. The reaction speed has a strong nonlinear relationship with the temperature. Its exothermal reaction is counteracted by the cooling and leads to a maximum temperature somewhere along the length of the tube. As the catalyst decays, this becomes less pronounced and moves further downstream.

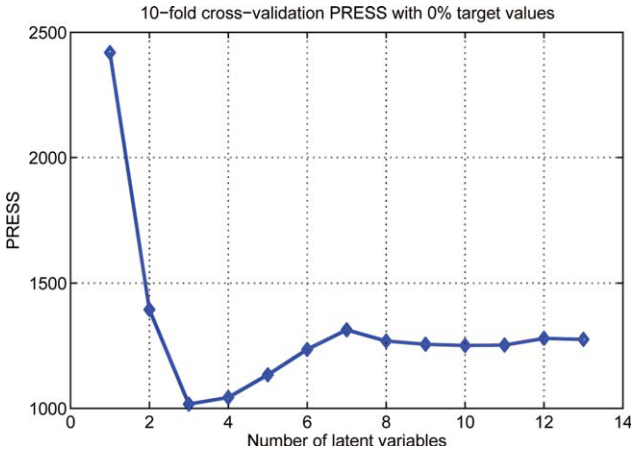

Figure 7. The PRESS for different numbers of latent variables of the RPLS algorithm.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

The catalyst activity usually decays within some time to zero, with a period of 1 year taken as example here. The process takes input from other larger processes so that the feed will vary over time. The operators react to this by choosing appropriate operating conditions. However, the catalyst decay is much slower than these effects. The process is equipped with measurements to log all the variations of the feed and the operating conditions. In addition, there are measurements showing concentrations, flows, and several temperatures along the length of a characteristic tube to identify the state of the processes. The reactor and the product are shown in Figure 5.

The goal is to predict the activity of the catalyst in the multitube reactor, which is simulated for the purpose of the training and evaluation of the soft sensors. The dataset covers 1 year of operation of the process plant. Many of the input variables show high level of colinearity and there are a large number of outliers, which can be found in as many as 80% of the variables.

Table I describes the physical values represented by the variables in this dataset.

Considering the measurements in Figure 6, one can identify some common issues of industrial datasets. These include:

- Missing values: some variables show missing values, e.g., the 4th variable is completely missing (Figure 6e), and there are also many missing values in the target variable (Figure 6a).
- Outliers: Many of the variables are affected by outliers, some of them quite severely, e.g., the 14th input variable (Figure 6o).
- Noise: some of the variables are very noisy, e.g., the 2nd input variable (Figure 6c).
- Changing sampling rate: the sampling rate of the target variable is lower for the last ca. 3000 samples (Figure 6a).
- Automatic value interpolation by the data recording system: All data around the 3000th sample are automatically interpolated values. This is due to a failure of the process information management system (Figures 6a–p).

Experiments

In this section, the proposed algorithm is applied to the dataset discussed in the Dataset section. The goal of the experiments is to analyze the adaptive behavior of the ILLSA method. For this purpose, there are six different scenarios with varying amount of data available for the adaptation of the method. This is achieved by purposefully removing a certain amount of target values which in turn limits the amount of data that can be used for the adaptation purposes. Nevertheless, all of the target data are used for the calculation of the prediction errors, which allows to compare the results between the particular scenarios. A special attention is paid to the performance of the particular local experts as well as the proposed combination method. By comparing the performance of the proposed algorithm to the standard RPLS method, the benefits of the method are further pointed out.

Methodology

To simulate the historical and the on-line data, the available dataset is split into two parts. The first 30% of the dataset form the historical data, which is used for the training of the models. The remaining 70% of the data simulate the on-line data and are delivered as a stream of samples. This way of splitting the data was chosen to be able to assess the adaptation capability of the proposed algorithm. For this reason, there is more data dedicated to the on-line phase than to the training phase, as is usually the case.

To evaluate the influence of the availability of the target data on the performance of the soft sensors, there are cases with 0, 5, 10, 25, 50, and 100% of available target values of the on-line data considered. The two extreme cases (i.e., 0 and 100%) represent a nonadaptive scenario and a scenario, where all the target values are available for the adaptation purposes, respectively. Intuitively, it can be expected that the performance of the soft sensor will increase with increasing percentage of target data allowing more frequent adaptation. In all of the above scenarios and the presented results, the models were provided with the input data, and only after delivering the prediction, they were given the target values for the adaptation. This procedure assures that the models are always tested with independent test data.

Manual data preprocessing

Before the experiments can be conducted, the dataset is manually preprocessed. In the experiments, we assume that only a limited amount of process information is available and restrict the preprocessing steps to the following:

- Downsampling of the first 5800 samples by factor 10 (i.e., after this step, the original 5800 samples are reduced to 580 data points).
- Removing variable numbers 3, 4, and 15 (as these variables are severely affected by outliers and missing values).
- Removing all data samples with missing target value.

The above preprocessing results into 194 training data points and 453 on-line data points.

Results

RPLS. The first set of experiments demonstrates the performance of the adaptive RPLS technique.

The optimal number of latent variables estimated by a 10-fold cross-validation on the training data is 3, which is surprisingly low. This fact demonstrates very high colinearity

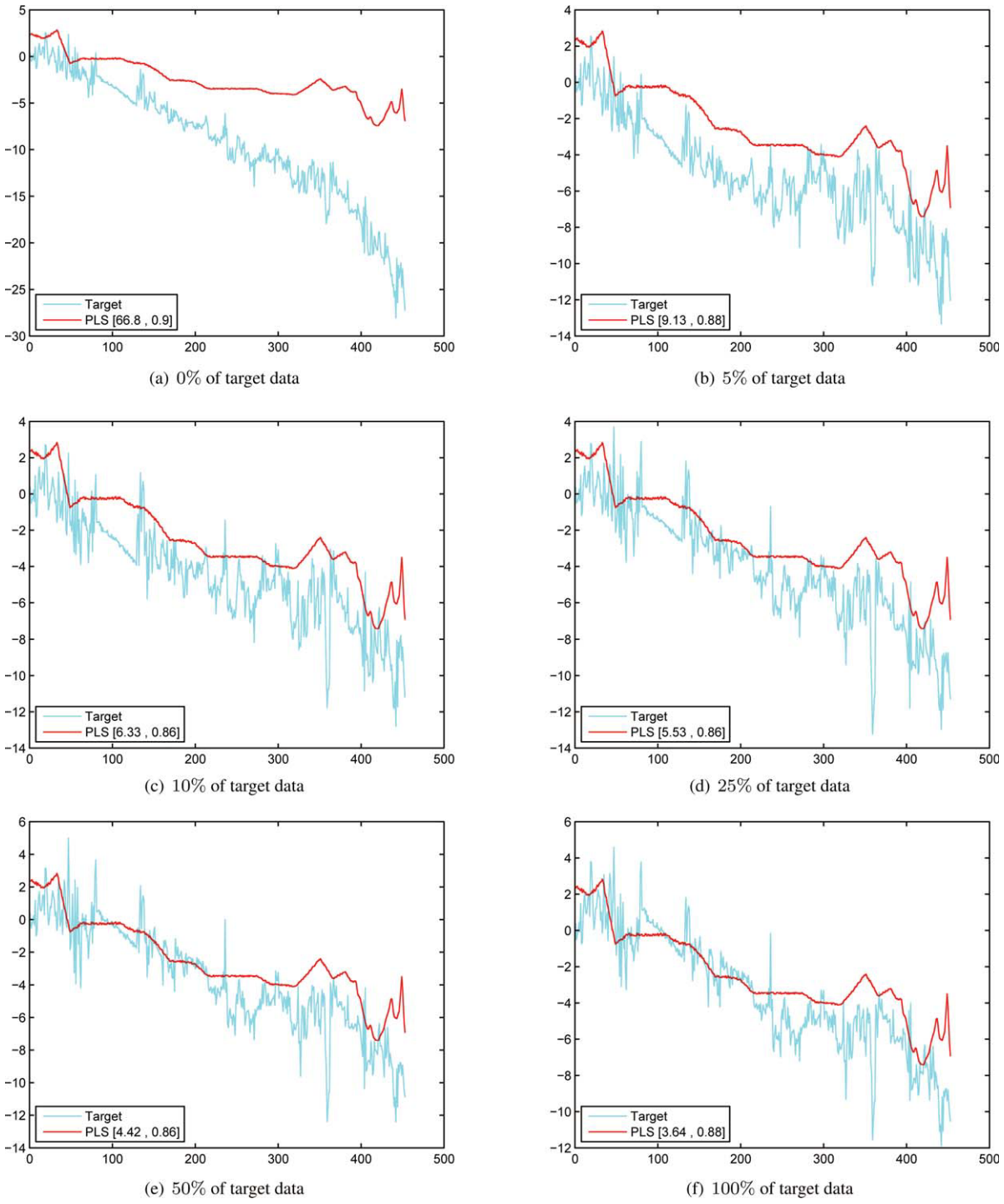

Figure 8. Predictions of the RPLS-based soft sensors.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

of the dataset. The PRESS (sum of squared prediction errors) results for the different numbers of latent variables are shown in Figure 7. The value of the forgetting factor was set to $\lambda = 1$.

The predictions for the different percentages of target values availability can be found in Figure 8. In the case without any target values, i.e., nonadaptive case, the RPLS model is not able to deliver any useful predictions. The plots also show the steady increase in performance achieved by more frequent adaptations as the predictions move closer to the correct target values. However, despite the increase in performance, the predictions are still not satisfactory.

ILLSA. For the ILLSA soft sensors, there are three critical parameters, which need to be optimized using the training data. The parameters are as follows:

- the size of the initial window n^{init} (see Eq. 11);

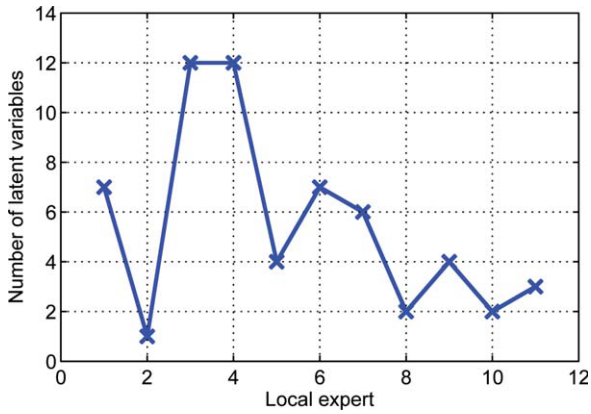

Figure 9. Number of latent variables of each of the 11 local experts.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

- the kernel size for the local expert descriptors σ (see Eq. 21); and
- the kernel size for the adaptation masks σ^{adapt} (see Eq. 30).

The above variables were optimized by the 10-fold cross-validation in the following ranges:

- $n^{\text{init}} = [30, \mathbf{50}, 100]$;
- $\sigma = [10^{-2}, \mathbf{10^{-3}}, 10^{-4}]$;
- $\sigma^{\text{adapt}} = [\mathbf{10^{-2}}, 10^{-3}, 10^{-4}]$.

The bold entries above show the parameters identified as optimal during the cross-validation process using the training data.

The above settings result in building of 11 receptive fields and consequently 11 local experts. In each of the receptive fields, the local expert's number of latent variables is optimized locally using the same procedure as described in the RPLS section. This results in significant variation of the number of latent variables as shown in Figure 9. The figure indicates that the colinearity of the data is significantly changing, and therefore training a single model for the whole training dataset with a fixed structure (i.e., number of latent variables) results in suboptimal results.

Table 2. Mean Squared Errors and Correlation Coefficients (with 100% of Target Values) of the Local Experts Showing Big Differences in the Prediction Performance of the Particular Local Experts

Local Expert Number	MSE	Correlation Coefficient
1	2.8377	0.9110
2	20.7628	0.8909
3	3.3463	0.8655
4	3.1915	0.8428
5	0.3163	0.9704
6	0.6156	0.9468
7	0.4581	0.9595
8	1.4718	0.9013
9	1.4461	0.9344
10	3.1565	0.8415
11	0.3586	0.9673

In the next step, the predictions of all local experts are analyzed. Figure 10 shows the predictions for the on-line data of the 11 local experts with the availability of 100% target data. The thick line shows the correct target values to be predicted and the other lines show the predictions of particular local experts. What can be easily seen is that not all of the local experts are able to deliver accurate predictions. This fact is also reflected in Table 2, where one can observe high variance in the performance of some of the local experts.

To combine the predictions, it is not feasible to build a simple mean combination of the local experts' predictions. For the discussed example, this combination method leads to an MSE of 0.601 and correlation coefficient of 0.96. The MSE of mean combination is on one hand much better than the performance of the worst of the local experts, but on the other hand, it is also significantly lower than the performance of the best local expert. In contrast to this, the combination method proposed in this work modifies the combination weights according to the expected prediction performance of the local experts, which results in the weights shown in Figure 11. One can see that different parts of the on-line data are dominated by different local experts. For the

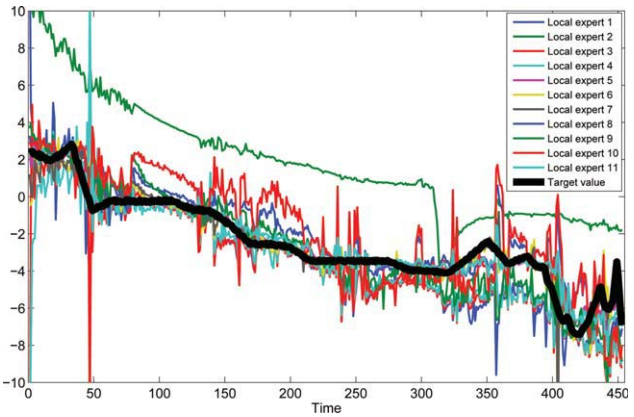

Figure 10. Predictions of all local experts.

As the local experts were trained on distinct parts of the historical data, they show diverse predictions on the test data, which is clearly demonstrated in this figure. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

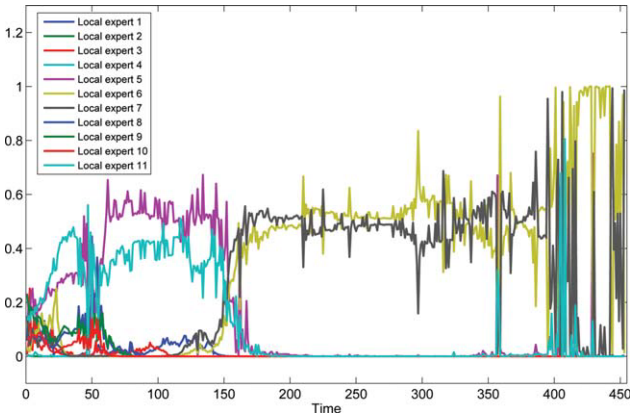

Figure 11. Combination weights *v* of all local experts.

One can observe increasing and falling weights of the local experts, which indicates the changing influence of the local experts during the online operation of the soft sensor. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

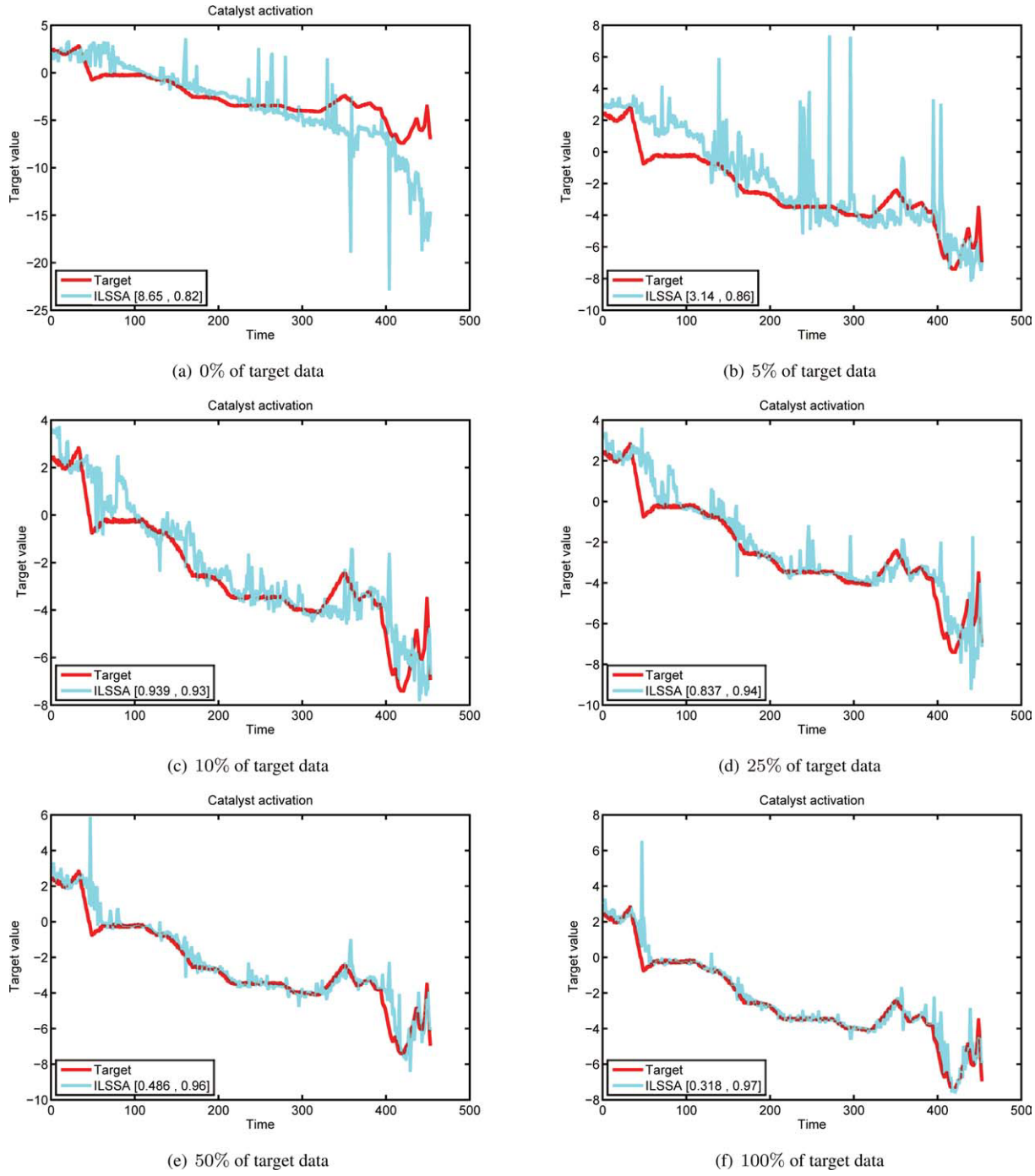

Figure 12. Predictions of the ILSSA-based soft sensors.

(a) 0% of target data; (b) 5% of target data; (c) 10% of target data; (d) 25% of target data; (e) 50% of target data; (f) 100% of target data. [Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

example discussed above, using the combination weights from Figure 11 leads to the predictions shown in Figure 12f. One can see that these combined predictions are very accurate despite the low performance of the local experts.

To be able to compare the results of the RPLS and ILLSA algorithms, Figure 12 shows the predictions as well as the MSE and correlation coefficient values for the cases with different amount of available target values.

A direct comparison of the results of the RPLS and ILLSA algorithms can be found in Figure 13 and in Table 3. From the table and the figure, it can be clearly seen that in terms of the MSE, the ILLSA outperforms the RPLS method with a large margin in the prediction performance.

Comparing Table 2 with Table 3 reveals that the fifth local expert performs very well and delivers similar performance as the full model. This may in first instance invoke the

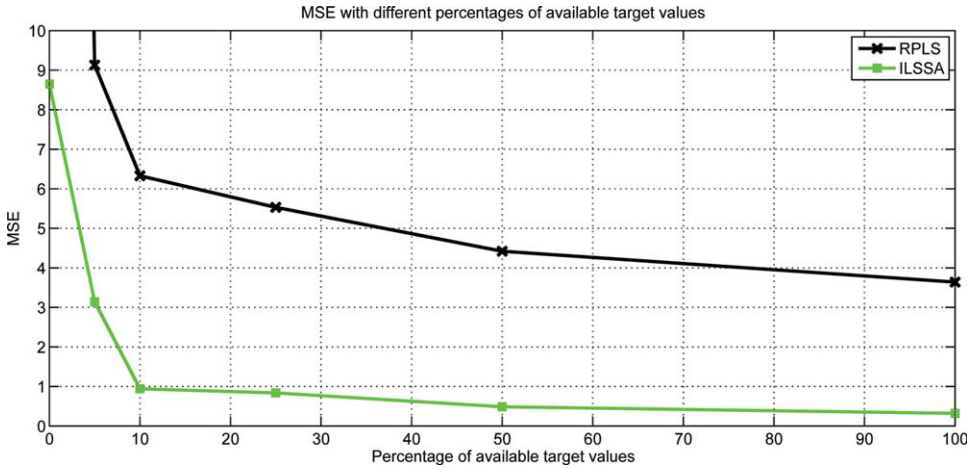

Figure 13. Overview of the MSE values achieved by the RPLS and ILLSA algorithm.

[Color figure can be viewed in the online issue, which is available at wileyonlinelibrary.com.]

idea that this sole local expert could replace the more complex model. However, considering the high variance of the predictions of the local experts, in practice it is very difficult to identify the local expert guaranteed to perform best on the independent online data. In this sense, the proposed algorithm provides a mechanism for dealing with the strong variance of the predictions of the particular local experts. Consequently, by using the proposed way of combining the local experts, it allows one to achieve performance that is very close to the best local expert.

Conclusions

The ILLSA soft sensing algorithm, the main contribution of this work, allows one to deal with several problems of algorithms like PLS, which are commonly applied for soft sensor development. By exploiting the local learning framework, it allows to model nonlinear relations between the input and target data. Moreover, it also supports several model adaptation possibilities, which is required in many soft sensing tasks. In this work, there are two different types of adaptation methods applied, which enable to maintain the soft sensor's performance in changing environment. In the presented experiments, the proposed method was shown to deliver predictions that are by a large margin more accurate than the predictions of the traditional RPLS soft sensor commonly used in practical applications.

Table 3. Performance comparison between the RPLS and ILLSA soft sensors

Percentage of Available Target Values	RPLS		ILLSA	
	MSE	Correlation coefficient	MSE	Correlation coefficient
0	66.8	0.90	8.65	0.82
5	9.13	0.88	3.14	0.86
10	6.33	0.86	0.939	0.93
25	5.53	0.86	0.837	0.94
50	4.42	0.86	0.486	0.96
100	3.64	0.88	0.318	0.97

Acknowledgments

The authors thank Evonik Industries AG for their substantial financial support and for providing the datasets. They also express their gratitude in particular to Monika Berendsen, Reinhard Dudda, and Sibylle Strandt from Evonik Industries AG for their numerous discussions. The authors also acknowledge Ratko Grbic from Faculty of Electrical Engineering, University of Osijek, Croatia for his implementation of the RPLS algorithm.

Literature Cited

1. Kadlec P, Gabrys B, Strandt S. Data-driven Soft Sensor in the process industry. *Comput Chem Eng*. 2009;33:795–814.
2. Fortuna L. Soft Sensors for Monitoring and Control of Industrial Processes. London: Springer Verlag, 2007.
3. Gonzalez GD. Soft sensors for processing plants. In *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials (IPMM'99)*, Vol. 1. IEEE, Honolulu, HI; 1999:59–69.
4. Jolliffe IT. *Principal Component Analysis*. New York: Springer, 2002.
5. Dong D, McAvoy TJ, Chang LJ. Emission monitoring using multivariate soft sensors. In *Proceedings of the American Control Conference*, Vol. 1. IEEE, Washington, DC; 1995; 761–765.
6. Lin B, Recke B, Knudsen J, Jrgensen SB. A systematic approach for soft sensor development. *Comput Chem Eng*. 2007;31:419–425.
7. Kaneko H, Arakawa M, Funatsu K. Development of a new soft sensor method using independent component analysis and partial least squares. *AIChE J*. 2009;55:87–98.
8. Rotem Y, Wachs A, Lewin DR. Ethylene compressor monitoring using model-based PCA. *AIChE J*. 2000;46:1825–1836.
9. Bishop CM. *Neural Networks for Pattern Recognition*. New York: Oxford University Press, 1995.
10. Fortuna L, Graziani S, Xibilia MG. Comparison of Soft-Sensor Design Methods for Industrial Plants Using Small Data Sets. In *IEEE Transactions on Instrumentation and Measurement* Vol. 58. 2009.
11. Fortuna L, Graziani S, Xibilia MG. Soft sensors for product quality monitoring in debutanizer distillation columns. *Control Eng Pract*. 2005;13:499–508.
12. Desai K, Badhe Y, Tambe SS, Kulkarni BD. Soft-sensor development for fed-batch bioreactors using support vector regression. *Biochem Eng J*. 2006;27:225–239.
13. Vapnik VN. *Statistical Learning Theory*. New York: Wiley, 1998.
14. Jang JSR, Sun CT, Mizutani E. *Neuro-Fuzzy and Soft Computing*. Upper Saddle River, NJ: Prentice Hall, 1997.
15. Zhong W, Pi D, Sun Y. SVM based soft sensor for antibiotic fermentation process. In *Proceedings of the IEEE International*

Conference on Systems, Man and Cybernetics, Vol. 1. Washington D.C. USA; 2003;160–165.
16. Macias JJ, Angelov P, Zhou PX. A method for predicting quality of the crude oil distillation. In *Proceedings of the International Symposium on Evolving Fuzzy Systems*. IEEE, Ambleside, UK. 2006; 214–220.
17. Zamprogna E, Barolo M, Seborg DE. Development of a soft sensor for a batch distillation column using linear and nonlinear PLS regression techniques. *Control Eng Pract*. 2004;12:917–929.
18. Gallagher N, Wise B, Butler S, White D, Barna G. *Development and benchmarking of multivariate statistical process control tools for a semiconductor etch process: improving robustness through model updating*. *IFAC ADCHEM'97*, Alberta, Canada. 1997:78–83.
19. Dayal BS, MacGregor JF. Recursive exponentially weighted PLS and its applications to adaptive control and prediction. *J Process Control*. 1997;7:169–179.
20. Li W, Yue HH, Valle-Cervantes S, Qin SJ. Recursive PCA for adaptive process monitoring. *J Process Control*. 2000;10:471–486.
21. Fujiwara K, Kano M, Hasebe S, Takinami A. Soft-sensor development using correlation-based just-in-time modeling. *AIChE J*. 2009;55:1754–1765.
22. Zhao L, Chai T. Adaptive moving window MPCA for online batch monitoring. In *Proceedings of the Fifth Asian Control Conference*. Vol. 2. Melbourne, Australia, 2004; (Online proceedings http://asc-c2004.ee.mu.oz.au/proceedings/papers/P190.pdf).
23. Wang X, Kruger U, Irwin GW. Process monitoring approach using fast moving window PCA. *Ind Eng Chem Res*. 2005;44:5691–5702.
24. Kadlec P. On robust and adaptive soft sensors, PhD Thesis. Bournemouth University, Bournemouth, UK, 2009.
25. Bottou L, Vapnik V. Local learning algorithms. *Neural Comput* 1992;4:888–900.
26. MacQueen J. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, Vol. 1. University of California Press. 1967;281–297.
27. Frank A, Asuncion A. UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml 2010.
28. Wold H. Nonlinear estimation by iterative least squares procedures. *Research Papers in Statistics*; 1966; New York: Wiley; 411–444.
29. Qin SJ. Recursive PLS algorithms for adaptive data modeling. *Comput Chem Eng*. 1998;22:503–514.
30. Geladi P, Kowalski B. Partial least-squares regression: a tutorial. *Anal Chim Acta*. 1986;185:1–17.
31. Helland Hans E. Recursive algorithm for partial least squares regression. *Chemom Intell Lab Sys*. 1992;14:129–137.
32. Gosset WS. The probable error of a mean. *Biometrika* 1908;6:1–25.
33. Parzen E. On estimation of a probability density function and mode. *Ann Math Stat*. 1962;33:1065–1076.
34. Ergon R. Informative PLS score-loading plots for process understanding and monitoring. *J Process Control*. 2004;14:889–897.
35. Kourti T. Process analysis and abnormal situation detection: from theory to practice. *IEEE Control Syst Mag*. 2002;22:10–25.

Manuscript received Feb. 9, 2010, and revision received May 10, 2010.