Computers and Chemical Engineering 35 (2011) 1–24

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Review

Review of adaptation mechanisms for data-driven soft sensors

Petr Kadlec[a,*], Ratko Grbić[b], Bogdan Gabrys[a]

[a] Smart Technology Research Centre, Computational Intelligence Research Group, Bournemouth University, Poole BH12 5BB, United Kingdom
[b] Faculty of Electrical Engineering, University of Osijek, Osijek, Croatia

ARTICLE INFO

Article history:
Received 19 May 2010
Received in revised form 28 July 2010
Accepted 31 July 2010
Available online 18 August 2010

Keywords:
Data-driven soft sensing
Process industry
Adaptation
Incremental learning
Online prediction
Process monitoring
Soft sensor case studies
Review

ABSTRACT

In this article, we review and discuss algorithms for adaptive data-driven soft sensing. In order to be able to provide a comprehensive overview of the adaptation techniques, adaptive soft sensing methods are reviewed from the perspective of machine learning theory for adaptive learning systems. In particular, the concept drift theory is exploited to classify the algorithms into three different types, which are: (i) moving windows techniques; (ii) recursive adaptation techniques; and (iii) ensemble-based methods. The most significant algorithms are described in some detail and critically reviewed in this work. We also provide a comprehensive list of publications where adaptive soft sensors were proposed and applied to practical problems. Furthermore in order to enable the comparison of different methods to standard soft sensor applications, a list of publicly available data sets for the development of data-driven soft sensors is presented.

Contents

* Corresponding author.
E-mail address: pkadlec@bournemouth.ac.uk (P. Kadlec).

0098-1354/$ – see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.compchemeng.2010.07.034

1. Introduction

Data-driven soft sensors have been developed and implemented for a long time (Fortuna, 2007; Kadlec, Gabrys, & Strandt, 2009). They gained on popularity with the increasing availability of recorded data in the process industries and availability of computational power to process the data. The collected data, also referred to as *historical data*, can be exploited by statistical and machine learning techniques to obtain additional information that can be used to make decisions towards more efficient and safe process operation. This kind of information can, for instance, be an instant prediction of the variables that are related to the product quality, which can be achieved using *o* nline prediction soft sensors (Fortuna, 2007; Gonzalez, 1999; Kadlec et al., 2009), or the estimation of current process state, which can be achieved using *process monitoring and fault detection* soft sensors (Kourti, 2002; Venkatasubramanian, Rengaswamy, Kavuri, & Yin, 2003b; Venkatasubramanian, Rengaswamy, Yin, & Kavuri, 2003c; Venkatasubramanian, Rengaswamy, & Kavuri, 2003a). However, this task is not trivial because historical data are often data rich but information poor (Dong & McAvoy, 1996) and therefore the model building on its basis is a challenging task.

The first generation of data-driven soft sensors relied on offline modelling using the recorded historical data. In such a case, the collected historical recordings are used for the model identification. This step may for instance include the identification of optimal weights of an Artificial Neural Network (ANN) (see e.g. Yang & Chai, 1997) or principal components of a Principal Component Analysis (PCA)-based soft sensor (see e.g. Wold, Geladi, Esbensen, & Ohman, 1987). However, in order to guarantee the success of the offline soft sensors, there are several conditions that have to be fulfilled. Most critically, the historical data has to contain all possible future states and conditions of the process. This includes not only the states in which the process can be operated but also states related to environmental changes, changes of the process input materials, etc. Even if the collected historical data contains all the required process states, another difficulty is to select a model type, and its parameters, in such a way that the model can comprehend all the different conditions. This results in high model complexity, which in turn demands and a large number of historical data for the model development. Additionally, most of the processes are exhibiting some kind of time-varying behaviour and thus require a strategy for online adaptation (Gallagher, Wise, Butler, White, & Barna, 1997). The most common causes for such a behaviour are:

- changes of process input (raw) materials;
- process fouling;
- abrasion of mechanic components;
- catalyst activity changes;
- production of different product quality grades;
- changes in external environment (e.g. weather, seasons).

Especially the last point from the above list is very difficult to estimate during the model design phase and therefore for processes sensitive to the external environment adaptive soft sensing techniques should be used.

As a result of these facts, it can often be observed that the performance of static models starts to deteriorate during their online operation (Kadlec et al., 2009).

The above issues, were identified already in the mid-1990's and first works on the next generation of data-driven soft sensors started to appear (see e.g. Wold, 1993 for one of the first publications on adaptive soft sensing).

To be able to cope with the effects listed above, adaptive soft sensors rely on various techniques for their online adaptation. The first adaptive soft sensors were based on the moving window techniques and recursive updates of the Least Squares (LS), Principal Component Analysis (PCA) and Partial Least Squares (PLS) methods (see Dayal & MacGregor, 1997b; Qin, 1998; Wold, 1993 for some early examples of these methods). Another difficulty of the operation of adaptive soft sensors is that the model has to be steadily supplied with a feedback about its performance as shown in Fig. 1. In such an environment, the soft sensor is first developed using some data pre-processing and predictive techniques selected from a repository of available methods (e.g. LS, ANN, and PLS). The selected techniques are, together with the available expert process knowledge, applied to the historical data and as result a soft sensor is obtained. During the operational, or online, phase the soft sensor is adapted using a maintenance mechanism which relies on: (i) online data; (ii) expert knowledge; and (iii) feedback about its performance.

Furthermore, equipping a soft sensor with adaptive capability requires two tasks to be performed. In the first instance, the need for adaptation has to be recognised by monitoring the performance of the model. This can be achieved by comparing the model output with information acquired from the laboratory analysis. In practical scenarios, this step is often ignored and the models are adapted at periodical intervals or constant forgetting factors are implemented as shown later in this work. Once the need for adaptation has been identified, the actual adaptation is to be performed.

In the case when data pre-processing is applied to deal with data issues like outliers, missing values, etc., the model adaptation has also to include the adaptation of the pre-processing methods. Furthermore, dealing with these issues has to be extended beyond the development phase in order to ensure (during the online phase) that the models are adapted with useful data only, and thus to

Nomenclature

Methods

AKL	Adaptive Kernel Learning
ANN	Artificial Neural Networks
ARMA	Auto-Regressive Moving Average
ARMAX	AutoRegressive Moving Average with eXogenuous input
ARX	AutoRegressive with eXogenous input
CPCA	Consensus Principal Component Analysis
DISSIM	DISSIMinalirity analysis
EDISSIM	Extended DISSIMinalirity analysis
EM	Ensemble Methods
EWMA	Exponentially Weighted Moving Average
EWPCA	Exponentially Weighted Principal Component Analysis
EWPLS	Exponentially Weighted Partial Least Squares
FCM	Fuzzy C-Means
FIR	Finite Impulse Response
FMWPCA	Fast Moving Window Principle Component Analysis
IIR	Infinite Impulse Response
ILLSA	Incremental Local Learning-based Soft sensing Algorithm
ISVR	Incremental Support Vector Regression
JITL	Just-In-Time Learning
kNN	*k*-Nearest Neighbours
KPCA	Kernel Principal Component Analysis
LOOCV	Leave-One-Out Cross Validation
LR	Linear Regression
LS	Least Squares
LS-SVR	Least Squares Support Vector Regression
LT	Lanczos Tridiagonalization
MA	Moving Average
MAD	Median Absolute Deviation from median
MBPCA	Multi-Block Principal Component Analysis
MLP	Multi-Layer Perceptron
MLR	Multiple Linear Regression
MSPCA	Mult-Scale Principal Component Analysis
MWKPCA	Moving Window Kernel Principal Component Analysis
NFS	Neuro-Fuzzy System
NIPALS	Nonlinear Iterative Partial Least Squares
OC	Offset Compensation
OLMS	OnLine MultiScale
PCA	Principle Component Analysis
PCR	Principle Component Regression
PLS	Partial Least Squares
RBFN	Radial Basis Function Network
RNN	Recurrent Neural Network
RPCA	Recursive PCA
SLR	Stepwise Linear Regression
SNR	Signal to Noise Ratio
SOM	Self-Organizing Network
SPE	Squared Prediction Error
SVD	Singular Value Decomposition
SVM	Support Vector Machines
SVR	Support Vector Regression
WT	Wavelet Transformation

Application types

OP	Online Prediction
IC	Inferential Control
PM	Process Monitoring

prevent the soft sensor from adapting to data disturbances. This is particularly important for soft sensors acting as backup sensors for their hardware counterparts. In such a case, it has to be prevented that the soft sensors are adapted when the hardware sensor delivers faulty measurements (Gonzalez, 1999). For process monitoring soft sensors, the monitoring metrics and confidence limits have also to be included into the adaptation process.

For the purpose of evaluation and quality monitoring of the sensor, usually squared error-related metrics, including, e.g. Mean Squared Error (MSE), Root Mean Squared Errors (RMSE), etc., as well as correlation coefficient-based error measures are applied. Additionally, in order to ensure that the calculated error values covers the recent model performance only, temporal weighting of the squared error can be used (see e.g. Ruta, 2006 for such a error metric). If visual performance analysis is required, the 4-plot analysis can be used Fortuna (2007).

Another aspect playing a role in the development and maintenance of adaptive soft sensors is the implementation of process knowledge into the model. Traditionally, the application of expert knowledge in soft sensors is limited to the selection of critical process variables (Casali et al., 1998; Fortuna, Graziani, & Xibilia, 2005a; Kampjarvi et al., 2008; Lin, Recke, Knudsen, & Jorgensen, 2007) and lagged variables in case of Moving Average (MA) models (Casali et al., 1998; Fortuna, Graziani, & Xibilia, 2005b). In the case of adaptive soft sensors, any available expert knowledge about the dynamics of the process can be used for selecting appropriate values for model attributes related to the adaptive operation. Examples of such attributes are the size of moving windows, or the forgetting factors.

The aim of this article is to review the discussed mechanisms for soft sensor adaptation. We track the development of adaptive soft sensing from its beginning until today and discuss techniques across the different soft sensor applications including online prediction, process monitoring and fault detection. The adaptation methods are classified into three categories and outline the strength and weaknesses of each of the categories.

The article is structured as follows. In Section 2 a theoretical background from machine learning perspective is provided. This enables the categorisation of the adaptation methods as well as the discussion of their advantages and disadvantages. Section 3 briefly outlines the most common statistical soft sensing methods, which will be required for further discussion of the adaptation methods. Section 4 reviews the most commonly used adaptive techniques for soft sensing. The review includes several adaptive methods based on the Linear Regression (LR), Principle Component Analysis (PCA), Partial Least Squares (PLS), Adaptive Kernel Learning (AKL) framework, and Ensemble Methods (EM). Section 4 is followed by a comprehensive list of case studies summarising adaptive soft sensors for various problems in the process industry in Section 5. Section 6 outlines different publicly available data sets and process simulators, which can be used for the evaluation of adaptive as well as non-adaptive soft sensing techniques. Finally, this work is concluded in Section 7, where also recommendations for the selection of appropriate techniques in different situations are provided.

2. Theoretical aspects of adaptive soft sensing

2.1. Theory and terminology

In this section the adaptive behaviour of models, i.e. soft sensors, is going to be discussed from the point of view of machine learning research. From this position, the relation between the input (or easy-to-measure) data space $\mathcal{X}$ and the output (or target or difficult-

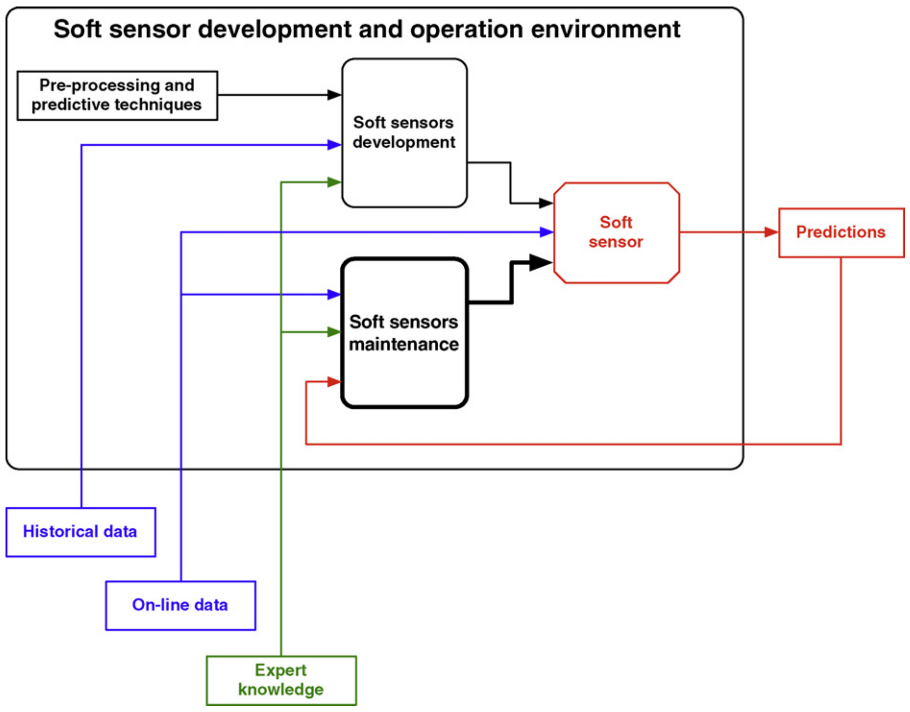

Fig. 1. Environment for the online operation of adaptive soft sensors.

to-measure) space $\mathcal{Y}$ can be described by an unknown function ϕ:

$$\mathcal{Y} = \phi(\mathcal{X}). \quad (1)$$

This relation is straightforward with online prediction soft sensors, where the function ϕ represents a relation between the easy-to-measure variables and the variable of interest (e.g. process quality). This applies also to process monitoring and fault detection soft sensors with the only difference being the target variable, which in this case is limited to two states only. The binary target variable can be either in a state of *process within specification* or *process out of specification*. The goal of model development is therefore to train a model $f()$, which is an accurate approximation of the mapping function ϕ and allows one to map the input space onto the target space:

$$\mathcal{Y} = f(\mathcal{X}) + \epsilon \quad (2)$$

where ϵ is the residual error of the predictions (in the case of regression it is assumed to be normally distributed with mean value zero). The function $f()$ is a regression model in the case of online prediction soft sensors and classification model in the case of typical process monitoring soft sensors.

In the offline soft sensor development scenario, the model $f()$ is trained on a set of historical data $\mathcal{D}^{hist} := \{X^{hist}, \mathbf{y}^{hist}\}$, where $X^{hist} \in \mathbb{R}^{n,m}$ are n input data points from the m-dimensional space $\mathcal{X}^{hist}$ and $\mathbf{y}^{hist} \in \mathbb{R}^{n,1}$ are the corresponding difficult-to-measure values from $\mathcal{Y}$ (in this work assumed as univariate but the concept can be applied to multivariate case in a straightforward way). The target function is unknown and it is the task of the learning algorithm L to find an approximation to this function given the historical data set $\mathcal{D}^{hist}$ and a (randomly initialised) predictor function f^{init}:

$$f \leftarrow \mathcal{L}(f^{init}, \mathcal{D}^{train}). \quad (3)$$

The predictor can be for example a linear model, such as a Multiple Linear Regression (MLR), a Principal Component Regression (PCR) model with a given number of principal components, or a nonlinear Multi-Layer Perceptron (MLP) with a given number of hidden units and randomly initialised weights. The outcome of the learning process is the trained predictor $f()$, which, together with other steps, like data pre-processing, is the actual soft sensor.

The model $f()$ can be applied to deliver predictions for the incoming online data points $\mathbf{x}^{online}$:

$$\hat{y} = f(\mathbf{x}^{online}), \quad (4)$$

where $\hat{y}$ stands for the predicted target value. However, in order to achieve accurate predictions, there is a critical assumption that the mapping function ϕ observed in the training data remains valid also for the online data, i.e.:

$$\mathcal{Y}^{online} = \phi(\mathcal{X}^{online}). \quad (5)$$

As discussed in Section 1, this is often not the case because the underlying process generating the data is changing over time, which makes the mapping function ϕ time variant, i.e. $\phi \rightarrow \phi_t$, which in turn requires to make the model also time variant or adaptive: $f_t()$. Before discussing the available mechanisms for adaptation of the models, some terminology for adaptive modelling is introduced.

The following types of models are introduced: (i) *dynamic*; (ii) *adaptive*; (iii) *incremental*; and (iv) *recursive*.

Dynamic models are trying to capture the influence of preceding data point(s) on the current sample for which the prediction has to be delivered. A popular modelling techniques are the Moving Average (MA) and Auto-Regressive Moving Average (ARMA) model structures (Fortuna, 2007) (see Fig. 2 for such a system). Another particularly popular modelling technique for dynamic predictive models are Recurrent Neural Networks (RNN) (Mandic & Chambers, 2001). These are topologically similar to MLP neural networks, with the difference that RNN have additional (delayed) feedback connections that enable to model the dynamics between consecutive data points. The important point is that these models are not changed during the online phase, which is the reason for missing the subscript t in $f()$ in Fig. 2. In practice it means for example that the connection weights of the RNN are set during the training phase and not changed afterwards. Examples of soft sensors based on these techniques were published in Su, Fan, and Schlup (1998); Wang,

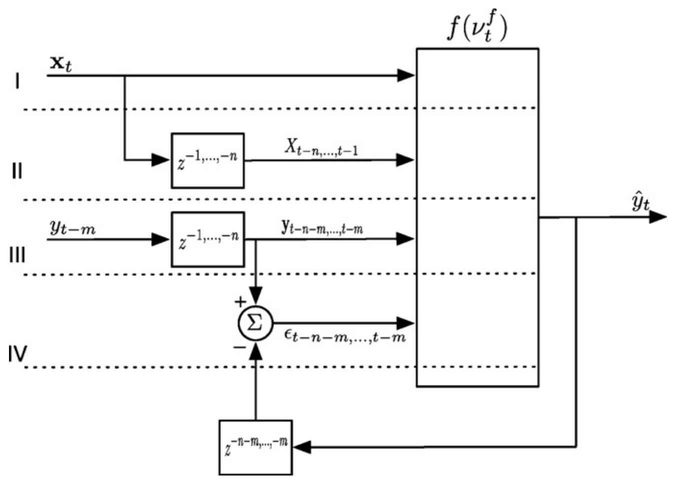

Fig. 2. A representation of different dynamic systems, dependent on the input the following types of dynamic systems can be distinguished: (i) Moving Average (MA) using inputs I and II; (ii) AutoRegressive (AR) system using input II; (iii) AutoRegressive with eXogenous input (ARX) using input I and II; AutoRegressive Moving Average with eXogenuous input (ARMAX) using inputs I, II, and III (and optionally IV). In the case of a nonlinear function *f*() the resulting models are referred to as NMA, NAR, NARX, and NARMAX.

Kruger, and Irwin (2005). Although these techniques can be combined with the adaptation methods discussed later, in their plain form they do not support online adaptation of the predictive model or its structure, and are therefore out of the scope of this work. This type of models can be described using the following expression (where *f*() can be a linear or nonlinear function):

$$\hat{y}_t = f(\nu_t^f), \tag{6}$$

where ν_t^f is the input of the model. Dependent on the model structure used, the input can be formed by: (i) input vectors $\mathbf{x}_t, \dots, \mathbf{x}_{t-N_x}$; (ii) past values of the target variable $y_{t-1}, \dots, y_{t-N_y}$; (iii) prediction residuals $d_{t-1}, \dots, d_{t-N_d}$. For example in the case of a moving average model, the input has the following format: $\nu_t^f := [\mathbf{x}_t, \dots, \mathbf{x}_{t-N_x}]$. An issue of these techniques is that it increases the pressure on dealing with high-dimensional data. Normally, in the context of soft sensing, there is a large number of easy-to-measure variables (Kadlec et al., 2009), which gets even more critical when using delayed inputs. For this reason, a mechanism for effective dealing with this issues should be provided. Also, using the residual as the model input can lead to problems with the stability of the model.

The term *adaptive* is a general description of models that posses the ability to automatically change either one or both of the following aspects during the online phase: (i) their attributes (e.g. latent variables in the case of PLS, connection weights in the case of ANN, etc.); and/or (ii) their structure (e.g. number of latent variables for PLS, the number of hidden units in the case of ANNs, etc.). As such the models need to be equipped with a mechanism, which allows them to accommodate the online data and the feed-back information about their performance. Additionally to the model description shown in Eq. (6), there is an adaptation function *g*() in the case of adaptive systems. This function changes, i.e. adapts, the model during its runtime:

$$f_t = g(f_{t-1}, \nu_t^g). \tag{7}$$

The input ν_t^g can contain the same or different data as ν_t^f. The function f_{t-1} in the input data is optional and is present only in the case of recursive adaptive techniques (see below). Furthermore, there are different types of adaptive algorithms that can be described by different types of the function *g*(), some

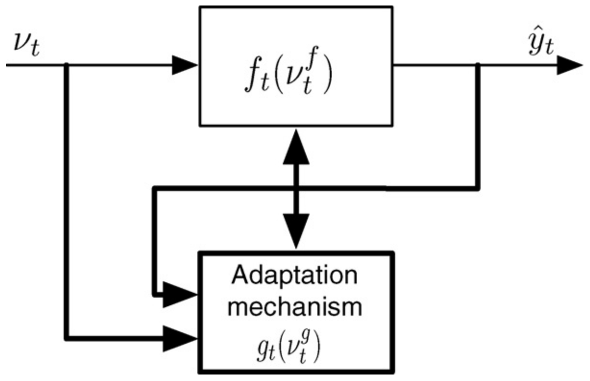

Fig. 3. A representation of an adaptive system having an additional mechanism *g*() that is responsible for the adaptation of the function $f_t()$.

examples of these are: (i) Moving window: $f_t = g(\nu_t^g)$, with $\nu_t^g := [\mathcal{D}^{MW} := \{x_{t-i}, y_{t-i}\}_{i=1}^{N-1}]$ (see Section 2.3.1); (ii) recursive methods: $f_t = g(f_{t-1}, \nu_t^g)$. A schematic representation of an adaptive system is shown in Fig. 3.

In order to be able to describe an algorithm as *incremental*, there are some additional requirements that have to be fulfilled together with the above defined adaptivity of the system (Bouchachia, 2009). The most critical one is that incremental algorithms do not posses any possibility to store the stream of incoming online data and the system has to strictly adapt on sample-by-sample basis, which is contrasting with the *b* lock-wise adaptation methods. Often, incremental systems are also required to start their operation *from scratch* without any specific training phase (see e.g. Kadlec & Gabrys, 2010a for such an algorithm). In soft sensing, this requirement can be relaxed since in most applications there is a set of historical data available. Nonetheless, incremental algorithms should be able to start their operation with a limited amount of historical data and continue improving the model during the online operation. This type of system is restricted to the following inputs:

$$\nu_t^f = [\mathbf{x}_t, y_{t-1}, f_{t-1}]. \tag{8}$$

$$\nu_t^g = [\mathbf{x}_{t-1}, y_{t-1}, f_{t-1}]. \tag{9}$$

The last type of the discussed algorithm types are *recursive* techniques. In general recursion refers to self-reference, i.e. an algorithm that calls itself. Nevertheless, in the literature dealing with adaptive soft sensors the term *recursive* is often used more loosely and algorithms are described as recursive even if they do not strictly adhere to the description in Eq. (8) (e.g. Li, Yue, Valle-Cervantes, & Qin, 2000). The recursive adaptation mechanism can also be described by Eq. (7). However, in this case the input f_{t-1} is not optional and has to be present. A recursive algorithm can also be incremental, which is the case when the adaptation fulfils the criteria discussed above.

2.2. Stability–plasticity dilemma

When dealing with adaptive systems, one inevitably has to deal with the *stability–plasticity* dilemma (Carpenter & Grossberg, 1998), sometimes also called *learning-forgetting* dilemma (Duda, Hart, & Stork, 2001). It relates to finding an optimal trade-off between learning new and forgetting old information. Adaptive learning systems with generalisation capability need a mechanism for forgetting old information as their capacity is limited and as such they suffer from *negative interference*. This refers to forgetting of past useful knowledge while learning new information (Schaal & Atkeson, 1998), also referred to as the plasticity of the model. The extreme manifestation of this problem is *catastrophic forgetting* (French, 1999), where new knowledge is added to the model at the cost of completely removing the previously learnt knowledge. This can happen when the forgetting factor is too high, which leads to removing the past knowledge and memorising only the last data point at each adaptation step. Similar effect can be observed when using too short adaptation windows in the case of the moving window technique. In the opposite case, when the plasticity of the system is too low (and consequently the stability too high) the model may fail to adapt to abrupt or other fast process changes. According to French (1999), the stability–plasticity problem can be tackled by: (i) having representative validation data set; (ii) memorising all training data samples (which is often not possible due to the limited capacity of the model); or (iii) incorporation of strong prior knowledge (e.g. having a knowledge about the dynamics and time constants of the process).

Practical approaches tackling this problem are based on variable moving window sizes and variable forgetting factors, where the plasticity of the system is adjusted according to the amount of estimated changes in the data.

2.3. Concept drift theory

In this section, different types of adaptive algorithms relevant for soft sensing are discussed. In machine learning, one of the research field dealing with adaptive data-driven techniques is called *c* oncept drift. The early works dealing with concept drift go back to Widmer and Kubat (1996). In the cited work, concept drift was described as: "The change of the target concept, which is caused by changes in some hidden context." In terms of process industry data, the hidden context can be any of the effects discussed in Section 1, or in other words, anything that has an influence on the process and/or the data and consequently on the function ϕ_t (see Eq. (1)). As for the type of the changes, it can be distinguished between: (i) *abrupt* changes, where the concept changes immediately from one state to another; and (ii) *gradual* drifts, with slower changes.

Dealing with concept drift issues consists of two tasks: (i) *c* oncept drift detection, which can be done by the identification of symptoms like poor model performance, etc.; and (ii) *concept drift handling*, which is the actual model adaptation (Gama, Medas, Castillo, & Rodrigues, 2004). There are three different approaches to adaptation in order to handle concept drifts (Tsymbal, 2004). These are:

- Instance selection – Moving windows techniques – (Maloof & Michalski, 2000);
- Instance weighting – Recursive adaptation techniques – (Klinkenberg, 2004);
- Ensemble methods (Scholz & Klinkenberg, 2007).

The following sections provide some examples of different approaches to concept drift handling while paying particular attention to techniques that are popular in adaptive soft sensing.

2.3.1. Moving window methods

This technique refers to a case where the model is adapted using a set of selected data points. The set is selected to maximise the relevance for the current concept (e.g. process state). In vast majority of cases a fixed number of most recent data points is used as these are assumed to be the most relevant to the current concept.

As new samples are acquired, the window slides along the data so that newest samples are included and the oldest ones are excluded from the model. The model can be recalculated either: (i) sample-wise, i.e. with each data point coming in (Liu, Kruger, Littler, Xie, & Wang, 2009a); or (ii) block-wise after accumulating a certain number of data points (Lee, Joung, Lee, Park, & Woo 2005b). In the latter case, the number of points that need to be collected for the recalculation of the window is called the *s* tep size (see Fig. 4). Furthermore, one can distinguish between two types or re-training the model. In the first case the model is trained from scratch using the set of data contained in the window (see Fig. 4). The advantage of this method is that the learning algorithm applied during the training phase can be re-applied during the online phase and there is no need to develop any special online training technique. This also means that it can be combined with any existing soft sensing algorithm (e.g. ANN, PCR, etc.). Consequently, this method is very popular for practical adaptive models. The other type of re-training, shown in Fig. 5, is using a two step procedure that includes "downdating" to remove the oldest data point from the model and "updating" to includes the latest data samples into the model (Liu et al., 2009a). This method is often more computationally efficient because it exploits incremental adaptation techniques for the adaptation purposes.

Nevertheless, there are also some drawbacks of the method. One of them is the fact that both of the moving windows methods require to store all of the data within the window. For large windows and memory limited applications this can be problematic. Another, perhaps more critical issue, is that there are two parame-

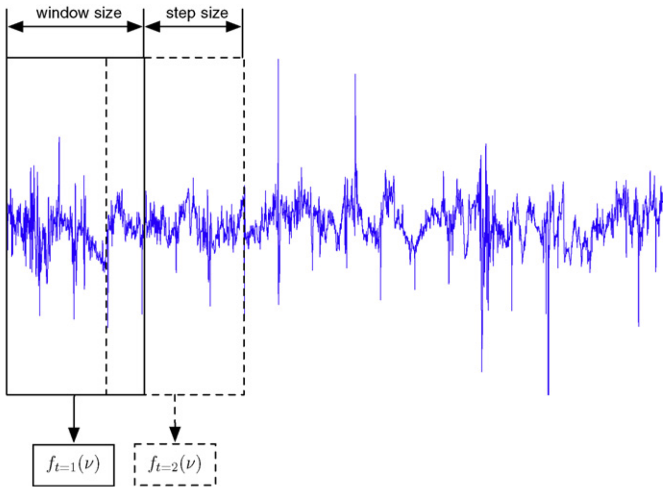

Fig. 4. Block-wise moving window approach, where the function $f_t()$ is retrained from scratch each time the required amount of data is collected.

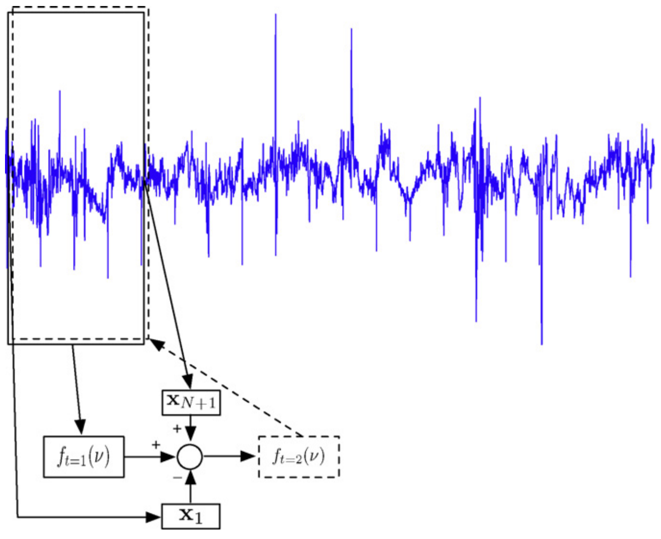

Fig. 5. Sample-wise moving window approach, where the function $f_t()$ is adapted in two steps by first removing the oldest data point from the model, followed by adding the newest data point to the model.

ters with critical influence on the model performance, namely the size of the adaptation window (window size) and the adaptation intervals between the updates (step size). Inappropriate setting can lead to performance degradation instead of improvement (Kadlec, 2009). In the ideal case, the window size should correspond to the length of the concept in the data or in other words to the span, where the function ϕ_t remains time invariant. If this requirement is not fulfilled, there is a danger that the soft sensor adapts to noise if the window size is too short or, in the case of too long, it can lead to limited adaptation capability. Furthermore, from the viewpoint of concept drift detection, which is one of the two required steps of an adaptive system, the issue of using fixed window size and step size, avoids the concept drift detection task as the window of fixed size is moving at constant speed. Unless, there is a guarantee that the window size and the adaptation intervals are set correctly and also that the process dynamics do not change, this approach will inevitably lead to a sub-optimal performance. This problem can be solved by applying an adaptive window size technique, however these methods can be found only rarely in the context of adaptive soft sensing. Some examples, where techniques for adaptive moving window size estimation were proposed are He and Yang (2008) and Zhao, Wang, and Jia (2007). The general idea of these methods is to decrease the window size if a concept drift is detected.

2.3.2. Recursive adaptation methods

The recursive methods are using the current model, e.g. in form of the covariance matrix (Dayal & MacGregor, 1997b), and the new data point (in sample-wise operation) or block (in block-wise operation) to update the model. The adaptation usually involves down-weighting the previous model by using a forgetting factor.

Recursive adaptation methods are available for the most popular statistical soft sensing algorithms including the Least Squares (Dayal & MacGregor, 1997b), Principle Component Analysis (Li et al., 2000), and Partial Least Squares methods (Qin, 1998).

These methods have similar issues with parameter estimation as the moving window technique. Focusing on the previous example, where the weights are assigned according to the age of the sample, the critical parameter is the speed of the temporal decay of the sample weights. Similarly to the moving window technique, it also does not provide any mechanism for concept drift detection unless the decay rate itself is adapted according to the speed of the concept change. Nevertheless, there are some techniques for adaptive setting of the forgetting factor. A frequently used method for dynamic setting of the forgetting factor in the recursive linear least squares framework was published in Fortescue, Kershenbaum, and Ydstie (1981). In this case, the value of the forgetting factor is modified depending on the product of the Hotelling's T^2 value and the Squared Prediction Error (SPE) in such a way that the information content of the estimator remains constant. Fortescue et al. show that the forgetting factor is flexible enough to enable the compensation of slow as well as abrupt changes in the studied process. Another approach to adaptive forgetting factor setting was shown in Choi, Martin, Morris, and Lee (2006).

The role of the forgetting factor λ is specifically shown in Fig. 6. In this case, the factor control the amount of knowledge transferred between the old model f_{t-1} and the new model f_t. Also indicated, is the importance of process knowledge, which has often to be used to control the forgetting factor.

2.3.3. Ensemble methods

The last approach for model adaptation discussed here is based on the ensembles learning framework (see Kadlec (2009) for the application of ensembles in soft sensing). In the ensemble framework, there is a set of models deployed and maintained. Each of these models provides a prediction of the difficult-to-measure variable. In order to obtain the final prediction of the ensemble, the

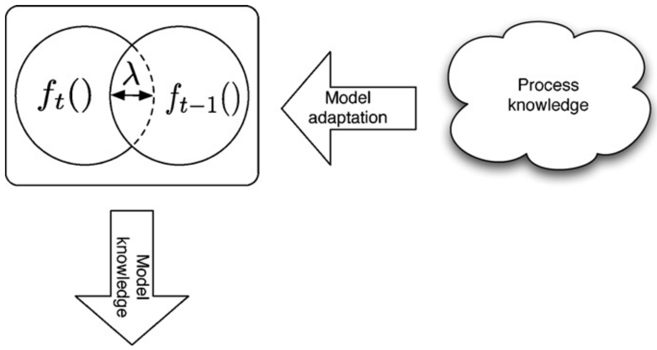

Fig. 6. Model adaptation and the role of process knowledge.

predictions of the particular models have to be combined. The simplest and very common combination technique is the average building, where a mean value of the ensemble members' predictions is built. However as shown in Krogh and Vedelsby (1995) the average building is not an optimal combination method and the weighted combination should be preferred. In this case each ensemble member is assigned a combination weight and the final prediction is a weighted sum:

$$\hat{y} = \sum_{i=1}^{p} \omega_i f_i(\mathbf{x}), \tag{10}$$

where p is the number of ensemble members, $f_i(\mathbf{x})$ is the prediction of the ith ensemble member for the current input data vector $\mathbf{x}$, and ω_i is its combination weight. An overview of an ensemble-based adaptive system is shown in Fig. 7.

The weighted combination method is also particularly interesting for the adaptation purposes because it provides more flexibility. In this case the concept drift detection and handling can be performed at two levels: (i) on the level of the models $f_i()$; and (ii) on the level of the combinations by modifying the combination weights ω_i. The former adaptation type can be achieved by applying either the instance weighting, or the instance selection techniques discussed above. Additional mechanism, which can be deployed at this level is launching new models and adding them to the ensemble if a new data concept is observed (e.g. in case of abrupt process

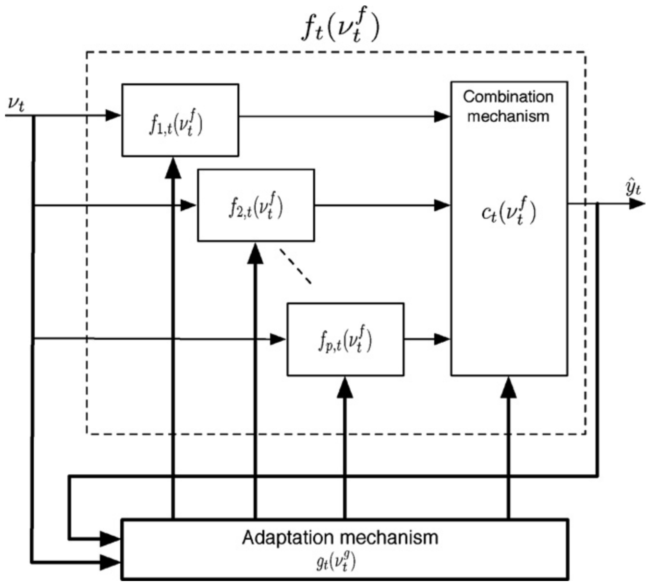

Fig. 7. Ensemble-based adaptation supporting the adaptation of the models $f_{i,t}()$ as well as the combination function $c_t()$.

change). As for the adaptation of the combination weights, there are many possibilities for the implementation of adaptation mechanisms. Examples of such mechanisms are discussed in Section 4.4.

A drawback of the ensemble concept drift methods is the higher model complexity, which is caused by the need to maintain a set of models and their combination weights.

In adaptive soft sensing these techniques are represented by Neuro-Fuzzy Systems (NFS), Incremental Local Learning-based Soft sensing Algorithm (ILLSA), and other hybrid methods (Liu, 2007). The ILLSA method is reviewed as a representative method in Section 4.4.

2.4. Adaptation of nonlinear methods

As discussed earlier, adaptation is useful in situations, where the relationship between the input and output variables changes. An alternative way of dealing with this problem is using nonlinear techniques covering variables dependence in a wider operating region. With the increasing available computational power, nonlinear techniques are gaining on popularity in data-driven soft sensing Kadlec et al. (2009). However, these techniques are successful in dealing with the time-varying processes only as long the historical data contains information about the region where the data drifts. Since this is often not the case, the nonlinear soft sensors still have to be periodically updated in order to keep their desired precision. One of the most popular nonlinear modelling types are ANNs (see e.g. Fortuna et al., 2005b). However, ANN-based soft sensors are not well suited for adaptation because: (i) their re-training requires large amount of data; (ii) of the difficulties with model structuring (e.g. optimal number of hidden units); and (iii) the fact is that ANNs are prone to local minima and overfitting. For these reasons it is difficult to use these methods in practical scenarios requiring the implementation in common automation equipment such as programmable logic controllers or microcontrollers.

An alternative to ANNs are Support Vector Machines (SVM) (see Section 3.4). Recently, their adaptive version named Online Kernel Learning algorithm (OKL) (see Section 4.3.6), which can successfully cope with time-varying process nature by using a growing and pruning techniques, was proposed in Liu, Yang, Wang, and Li (2008).

The ensemble framework proposed in Section 2.3.3 can also be applied to deal with nonlinear process behaviour.

3. Offline data-driven soft sensing

This section describes the most common techniques for data-driven soft sensing using multivariate methods. The purpose of this section is to introduce background knowledge for the discussion of the adaptive versions of these techniques in Section 4.

3.1. Linear Regression—Least Squares

The Least Squares (LS) is a simple method for the calculation of the coefficients $\mathbf{a}$ of a linear regression model. The offline version of the Least Squares algorithm has the following form:

$$\hat{\mathbf{y}} = X\mathbf{a} \quad \text{with} \quad \mathbf{a} = (X^T X)^{-1}(X^T \mathbf{y}), \tag{11}$$

where $\hat{\mathbf{y}}$ are the predictions and $\mathbf{a} = [a_1, \ldots, a_m]^T$ a vector of the linear regression coefficients.

3.2. Principal Component Analysis

The offline Principle Component Analysis (PCA) algorithm requires the matrix of the input data X to be normalised to zero mean and unit variance:

$$X' = (X - \mathbf{1}_n \mathbf{b}^T)\Sigma^{-1} \quad \text{with} \quad \mathbf{b} = \frac{1}{n} X^T \mathbf{1}_n \quad \text{and} \tag{12}$$

$$\mathbf{1}_n = [1, 1, \ldots, 1]^T \in \mathbb{R}^{n\times 1}, \quad \Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_m) \tag{13}$$

where σ_i is the standard deviation of each of the input variables. The scaling has to be done in order to assure equal influence of the input variables in the model. The normalised data X' is then transformed to l-dimensional ($l \le m$) scores $T \in \mathbb{R}^{n\times l}$ using the following relation:

$$X' = TP^T + E, \tag{14}$$

where $P \in \mathbb{R}^{m\times l}$, $E \in \mathbb{R}^{n\times m}$ are the loadings and residuals matrices respectively.

There are several ways to calculate the matrix P. In the case of the covariance approach, the correlation matrix C of the input data X has to be calculated:

$$C = \frac{1}{n-1}(X')^T \cdot X'. \tag{15}$$

Next the eigenvalues and eigenvectors of this matrix are derived by solving:

$$V^{-1} C V = eig(C), \tag{16}$$

where $eig(C)$ is the diagonal eigenvalues matrix and V the eigenvector matrix. The eigenvalues λ_i are then sorted in descending order such that $\lambda_1 > \lambda_2 > \ldots > \lambda_m$. The columns of matrix P are then formed by the eigenvectors $\mathbf{v}_i$ corresponding to the l highest eigenvalues:

$$P = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_l] \quad \text{with} \quad \mathbf{v}_i \in \mathbb{R}^{mx1}. \tag{17}$$

For the calculation of the eigenvalue decomposition either the Singular Value Decomposition (SVD) (Jollie, 2002, p. 44), the NIPALS algorithm (Geladi & Kowalski, 1986) or other methods (see e.g. Li et al. (2000)) can be used.

Having the matrix T, one can build a regression model using the LS algorithm:

$$\hat{\mathbf{y}} = T\mathbf{a}. \tag{18}$$

Applying the fact that the vectors in T are orthogonal, the previous equation can be simplified to:

$$\mathbf{a} = (T^T T)^{-1} T^T \mathbf{y} = L^{-2} T^T \mathbf{y}, \quad \text{with} \quad L \in \mathbb{R}^{l\times l}. \tag{19}$$

where L is a diagonal matrix with elements equal to $\sqrt{\lambda_i}$ (see (Jollie, 2002, p. 168)).

3.3. Partial Least Squares

The Partial Least Squares (PLS) was originally proposed in Wold (1966). Since the pioneering work of Wold, the potential of the PLS has been recognised and it became a popular method in chemometrics. The reason for the popularity resides in the fact that the method is able to deal with large dimensional co-linear data as well as the fact that the resulting model takes into account the covariance between the input and output data. This results in more accurate prediction of the difficult-to-measure variable than in the case of PCA, which exploits the variations in the input space only.

Similarly to the PCA case, the goal of the PLS algorithm is to project the scaled and mean-centered input data $X' \in \mathbb{R}^{n\times m}$ 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:

$$X' = TP^T + E \tag{20}$$

$$Y' = UQ^T + F, \tag{21}$$

where $T \in \mathbb{R}^{n\times l}$ (with $l \le m$ 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 the so-called latent vectors:

$$T = [\mathbf{t}_1, \ldots, \mathbf{t}_l] \quad \text{with} \quad \mathbf{t}_i \in \mathbb{R}^{n\times 1} \tag{22}$$

$$U = [\mathbf{u}_1, \ldots, \mathbf{u}_l] \quad \text{with} \quad \mathbf{u}_i \in \mathbb{R}^{n\times 1}. \tag{23}$$

The latent vectors, which are orthonormal to each other (i.e. $\mathbf{t}_i\mathbf{t}_j^T = 0\,\forall_{i\neq j}$) represent a more compact description of the input data achieved by removing the collinearity from the data. The column vectors $\mathbf{p}_i \in \mathbb{R}^m$ and $\mathbf{q}_i \in \mathbb{R}^p$ 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. Eqs. (20) and (21) are also called the outer model of the PLS method (Qin, 1998).

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

$$U = TB + R, \tag{24}$$

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

$$\hat{Y} = TBQ^T. \tag{25}$$

There are several ways to calculate the required vectors **t**, **p**, **u**, **q**, *and* **b**. A particularly popular algorithm for the calculation of PLS is the Nonlinear Iterative Partial Least Squares (NIPALS) algorithm (Geladi & Kowalski, 1986). After the initialisation: $i = 1; E_0 = X; F_0 = Y$; and $\mathbf{u}_1 = \mathbf{y}_1$ (i.e. the first target variable), the NIPALS algorithm proceeds in an iterative way with the following calculation:

$$\mathbf{w}_i = \frac{E_{i-1}^T\mathbf{u}_i}{\mathbf{u}_i^T\mathbf{u}_i} \tag{26}$$

$$\mathbf{t}_i = \frac{E_{i-1}\mathbf{w}_i}{||E_{i-1}\mathbf{w}_i||} \tag{27}$$

$$q_i = \frac{F_{i-1}^T\mathbf{t}_i}{||F_{i-1}^T\mathbf{t}_i||} \tag{28}$$

$$\mathbf{u}_i = F_{i-1}q_i \tag{29}$$

$$\mathbf{p}_i = E_{i-1}^T\mathbf{t}_i \tag{30}$$

$$b_i = \mathbf{u}_i^T\mathbf{t}_i, \tag{31}$$

where i is the index of the latent variable. Eqs. (26)–(29) are performed for each latent variable until convergence is reached. After each iteration the residuals are deflated:

$$E_{i+1} = E_i - \mathbf{t}_i\mathbf{p}_i^T \tag{32}$$

$$F_{i+1} = F_i - \mathbf{u}_i\mathbf{q}_i^T, \tag{33}$$

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 E_{i+1} and F_{i+1}. The number of calculated latent dimensions is usually established using cross-validation or some other parameter optimisation technique. In contrast to the NIPALS algorithm from Geladi, where the variables w_h and p_h were normalised, it is useful for the derivation of the recursive version of the PLS to normalise the latent variable t_i instead (Qin, 1998).

3.4. Support Vector Machines

Due to their theoretical background in the statistical learning theory (Vapnik, 1998), Support Vector Machines (SVM) increasingly gain more attention of soft sensor developers (see e.g. Liu et al., 2009a; Yan, Shao, & Wang, 2004). The general Support Vector Regression (SVR) algorithm was originally proposed in Drucker, Burges, Kaufman, Smola, and Vapnik (1997), Smola and Scholkopf (2004).

The first step of the SVR algorithm is constructing a linear function in a high-dimensional space using a kernel function $\varphi()$:

$$\hat{y} = \omega^T\varphi(\mathbf{x}) + b, \tag{34}$$

where $\varphi()$ is a kernel function mapping the m-dimensional data to a high-dimensional (m_k) space, i.e. $\varphi() := \mathbb{R}^m \rightarrow \mathbb{R}^{m_k}$. The goal is to optimise the parameters ω and b in order to fulfil the following condition:

$$|y - \omega^T\varphi(\mathbf{x}) - b| < \epsilon, \tag{35}$$

where ϵ is the precision parameter.

After some modifications (see Smola & Scholkopf, 2004 for details) the dual representation of model (34) can be expressed as:

$$\hat{y} = \sum_i(\alpha_i - \alpha_i^*)k(\mathbf{x}_i, \mathbf{x}) + b. \tag{36}$$

where x_i are support vectors, $k()$ can be any function fulfilling the Mercer condition, i.e. $k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$, and b is a constant that can be calculated by applying the Karush–Kuhn–Tucker conditions (Smola & Scholkopf, 2004).

The above optimisation problem (i.e. finding $\mathbf{x}_i$, α_i, α_i^*, and b) has to be solved using quadratic programming techniques. The complexity of the problem is independent of the dimensionality of the input space, however it grows with the number of training samples. Therefore, for large data sets the calculation of the support vector can become computationally infeasible.

3.5. Kernel PCA

Kernel PCA (KPCA) emerged recently as a method for dealing with nonlinear processes. The basic idea of KPCA is to first map the input space into a feature space via nonlinear mapping (kernel function) and then to compute the principal components in that feature space (Scholkopf, Smola, & Muller, 1998). The method does not involve nonlinear optimization procedure, which is often involved in other nonlinear approaches (e.g. Dong & McAvoy, 1996).

The KPCA model is built from covariance matrix of the process data transformed into the feature space $\Phi(X) = [\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)]$:

$$C_\Phi = \frac{1}{n-1}\sum_{i=1}^{n}(\Phi(\mathbf{x}_i) - \mathbf{b}_\Phi)(\Phi(\mathbf{x}_i) - \mathbf{b}_\Phi)^T, \tag{37}$$

where C_Φ is the sample covariance matrix in the feature space, $\Phi()$ is a nonlinear mapping in high-dimensional feature space and is the sample mean in the feature space, and $\mathbf{b}_\Phi = (1/n)\Phi(X)1_n$ is the sample mean in the feature space. In order to obtain the principal components, the eigendecomposition of C_Φ is carried out.

4. Adaptive data-driven soft sensing

This section outlines the most popular mechanisms for adaptive soft sensing algorithms. The methods are classified according to the different types of concept drift handling algorithms discussed in Section 2.3. The descriptions focus on the explanations of the adaptation mechanisms rather then the descriptions of the modelling methods, which can be found in Section 3.

4.1. Adaptive data pre-processing

Data that are collected during process operation often contain a lot of different impurities as a result of different disturbances, malfunctions, degradation and errors in sensors and data acquisition system. In the measured data these impurities are manifested as high frequency noise, sudden abrupt changes (outliers), drifts, offsets, trends, low frequency disturbances and missing values. To remove these undesirable impurities which negatively affect model quality, data pre-processing is incorporated in model building procedure (Fortuna, 2007; Kadlec et al., 2009; Warne, Prasad, Rezvani, & Maguire, 2004).

In online applications, to achieve successful model adjustment and precise estimation of the quality variable, data need to be pre-processed in an online manner. This consists of several operations such as outlier detection, data de-noising, missing value replacement and data scaling. While in offline model building procedure this step can be repeated and done in cooperation with plant experts, in an online application this is generally not possible. Also, in online applications *future* samples are not known in advance like in the offline model building.

4.1.1. *Online data de-noising*

Measurement noise causes errors in model estimation and has therefore to be dealt with by increasing the Signal to Noise Ratio (SNR) of the data (Martens & Naes, 1991). The easiest way to achieve this goal is to smooth the data using linear filters. The reason for the popularity of linear filters lays in the fact that they can be easily implemented offline as well as online. A data variable $\mathbf{x}_k$ is smoothed by using a weighted sum of previous measurements in a window of finite length (Finite Impulse Response—FIR):

$$x^d_{t,k} = \frac{1}{N}\sum_{i=1}^{N} a_i x_{t-i,k} \quad \text{with} \quad \sum_{i=1}^{N} a_i = 1, \tag{38}$$

where N is the filter length, a_i are the filter coefficients and x^d_t is the de-noised sample. If all previous measurements are used then an infinite length (Infinite Impulse Response—IIR) filter is obtained. An example of a IIR filter is the Exponentially Weighted Moving Average (EWMA) filter that is recursively implemented as:

$$x^d_{t,k} = \alpha x_{t,k} + (\alpha - 1)x^d_{t-1,k}, \tag{39}$$

where α is an adjustable smoothing parameter with values between 0 and 1 (Nounou & Bakshi, 1999).

A drawback of linear filters is their single scale nature. So there is a trade off between the noise removal and accurate feature representation that have to be dealt with as explained in Nounou and Bakshi (1999). This issue can be addressed by using Wavelet Transformation (WT) (Donoho, 1995), which is known as powerful tool for signal de-noising. Nounou and Bakshi (1999) proposed an OnLine MultiScale (OLMS) rectification technique, which is based upon wavelets in a moving window of dyadic length. The authors pointed out that the multiscale treatment of the data efficiently removes noise from the broader frequency region without smoothing out important signal features and gave some practical issues in OLMS rectification.

4.1.2. *Online outlier detection*

Outliers can be defined as samples that are not consistent with the majority of the data (Pearson, 2002). It is very important to detect such samples in online applications because they can negatively affect model parameter adaptation. However, while this samples can be erroneous readings, they can also be normal samples that carry important information about rare process states and require further examination (Martens & Naes, 1991; Pearson, 2001).

Generally, there are two ways to treat outliers. The first one relies on detection and replacement of outliers with some reasonable values while the second one uses advantages of robust techniques for model parameters estimation which are less affected by presence of outliers in the data set. A commonly used method for outliers detection is visual inspection of the data set. However, in online applications this method is not feasible and there is a need for automatic detection of outlying samples. Simple approach to outlier detection is 3σ edit rule (Fortuna, 2007), which works under assumption that a sequence of data is approximately normally distributed (Pearson, 2001). In online applications, it can be implemented in a moving window with incoming samples being marked as outlier if the following condition is fulfilled:

$$|x_{t,k} - b_{t,k}| > 3\sigma_{t,k} \quad \text{with} \quad b_{t,k} = \frac{1}{N}\sum_{i=1}^{N} x_{t-i,k},\ \sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_{t-i,k} - b_{t,k})^2, \tag{40}$$

where N is the window size, $b_{t,k}$ and $\sigma_{t,k}$ are the mean and standard deviation of the samples in the window. In recursive implementations, the values of the mean and standard deviation can be updated as the new samples arrive according to expressions shown later in Eqs. (42) and (43). However, in practice the stipulation of normal distribution of the data is often not fulfilled, which results in problems with this outlier detection method. A more efficient univariate approach for outlier detection is the Hampel identifier (Chiang, Pell, & Seasholtz, 2003; Lin et al., 2007; Pearson, 2002). Hereby, the mean and standard deviation are replaced with the median and the Median Absolute Deviation from median (MAD) that is defined as follows:

$$MAD(\mathbf{x}_k) = 1.4826\, median_i \left(\left|x_{t-i,k} - median(x_{t-(N-1),k}, \ldots, x_t)\right|\right) \quad \text{with} \quad i = t-(N-1), \ldots, t. \tag{41}$$

Soft sensors are often based on a large number of easy-to-measure variables (Kadlec et al., 2009), which is often dealt with by means of multivariate statistical methods. These methods can also be used for outlier detection. The online version of the PCA algorithm (see e.g. Section 4.2.1) can also be used for online outlier detection. Under the assumption that the PCA model is developed from normal process data, online outlier detection by PCA is based upon monitoring of two statistics, namely the SPE and Hotelling's T^2, which can be calculated for the new data samples (Qin, 2003). The violations of confidence limits set by these indices can be caused by outliers or abnormal process situation. If necessary the monitoring metrics can be adapted in order to take into account the changing nature of the process (see Section 4.5 for details). However, in this case it has to be assured that the data used for the updates are meaningful and do not contain any disturbances.

The influence of outliers can also be suppressed by using nonlinear filters, such as MT filter (Pearson, 2002) or robust OLMS techniques (Nounou & Bakshi, 1999).

4.1.3. *Online data scaling*

Measured process variables have often different magnitudes so it is important to scale variables before the model building or adaptation. This is often done through normalization (variables centering and scaling to unit variance). Since many processes are time-varying, the means and standard deviations of the process variables change over the time. This can be achieved using the following update equations:

$$\mathbf{b}_t = \frac{n-1}{n}\mathbf{b}_{t-1} + \frac{1}{n}\mathbf{x}_t, \tag{42}$$

$$\sigma_{t,k}^2 = \frac{n-2}{n-1}\sigma_{t-1,k}^2 + \frac{1}{n-1}(x_{t,k} - b_{t,k})^2, \tag{43}$$

where n is the total number of data points (in the case of moving window approach n is the window size, i.e. $n=N$), t is the current time stamp, $\mathbf{b}_t$ is a vector of means of all m variables in $\mathbf{x}$, and $\sigma_{t,k}^2$ is the variance of the k-th variable at time t. Consequently, the incoming data point can be normalised using the updated values:

$$\mathbf{x}'_t = (\mathbf{x}_t - \mathbf{b}_t)\Sigma_t^{-1} \quad \text{with} \quad \Sigma_t = diag(\sigma_{t,1}^2, \ldots, \sigma_{t,m}^2). \tag{44}$$

If required the correlation matrix can be recursively calculated as:

$$R_t = \frac{n-2}{n-1}\Sigma_t^{-1}\Sigma_{t-1}R_{t-1}\Sigma_{t-1}\Sigma_t + \frac{1}{n-1}\mathbf{x}_t\mathbf{x}_t^T. \tag{45}$$

A possible issue of the above method is the influence of the new data point diminishes with increasing n, i.e. the number of data. If this is an issue the method can be used as a moving window of size $N<n$. However, in this case a mechanism for removing the oldest data from the model needs to be implemented (see e.g. Wang et al., 2005). Another possibility is using exponential weighting of the data:

$$\mathbf{b}_t = \alpha\mathbf{b}_{t-1} + (1-\alpha)\mathbf{x}_t \tag{46}$$

$$\Sigma_t = \alpha\Sigma_{t-1} + (1-\alpha)\mathbf{x}_t\mathbf{x}_t^T \tag{47}$$

$$R_t = \alpha R_{t-1} + (1-\alpha)D_t^{-\frac{1}{2}}\mathbf{x}_t\mathbf{x}_t^T D_t^{-\frac{1}{2}} \tag{48}$$

$$\text{with} \quad D_t = \alpha D_{t-1} + (1-\alpha)diag(\mathbf{x}_t\mathbf{x}_t^T), \tag{49}$$

where α plays the role of a forgetting factor.

4.2. Moving window-based techniques

In this section the moving window techniques (outlined in general terms in Section 2.3.1) are discussed in further detail.

4.2.1. Block-wise moving window techniques

The block-wise moving windows adaptation is performed by retraining the model periodically after a given number of data samples have been collected. This number corresponds to the step size discussed in Section 2.3.1. After each of these intervals the model is retrained by using the N latest data points, where N is also referred to as the window size. The procedure can be summarised as follows:

$$f_t = \mathcal{L}(f^{init}, \mathcal{D}^{MW}) \quad \text{with} \quad \mathcal{D}^{MW} = \left\{\mathbf{x}_i, y_i\right\}_{i=t-N+1}^{t}, \tag{50}$$

where f_t is the new model, f^{init} is the modelling technique (e.g. PCR, PLS, etc.), $\mathcal{L}$ is the training algorithm for the methods (see Section 2.1), and $\mathcal{D}^{MW}$ is the set of latest N data points for the training of the new model. In general the model training should be combined with a parameter estimation method, such as cross-validation, because the optimal parameters can change from step to step.

4.2.2. Fast Moving Window PCA

Wang et al. (2005) have proposed an optimised version of the moving windows PCA called Fast Moving Window Principal Component Analysis (FMWPCA) . The algorithm combines the moving window method with the recursive adaptation of the PCA and thus corresponds to the type of moving window methods presented in Fig. 5. Under common circumstances, when the size of the window exceeds three times the number of input variables, the recursive recalculation of the PCA models is computationally more efficient than the block-wise moving window PCA discussed above.

The algorithm consists of two steps: (i) decremental adaptation (downdating), where the latest data point in the window is removed from the model (Eqs. (51)–(53)); and (ii) incremental update (updating), where the new data point is incorporated into the model (Eqs. (54)–(59)). During both steps, the means, variances and correlation matrices are updated.

The decremental phase, where the oldest data sample $\mathbf{x}_{t-N}$ is removed from the model:

$$\mathbf{b}_t^- = \frac{N}{N-1}\mathbf{b}_{t-1} - \frac{1}{N-1}\mathbf{x}_{t-N}, \tag{51}$$

$$\mathbf{x}_t^- = (\mathbf{x}_t - \mathbf{b}_t)\Sigma_{t-1}^{-1}, \tag{52}$$

$$\begin{aligned} R^- = {} & R_{t-1} - \Sigma_{t-1}^{-1} - \Sigma_{t-1}^{-1}(\mathbf{b}_{t-1} - \mathbf{b}_t)(\mathbf{b}_{t-1} - \mathbf{b}_t)^T\Sigma_{t-1}^{-1} \\ & - \frac{1}{N-1}\mathbf{x}_t^-(\mathbf{x}_t^-)^T. \end{aligned} \tag{53}$$

This phase is followed by the incremental phase, where the new data point $\mathbf{x}_t$ is included into the model:

$$\mathbf{b}_t = \frac{N-1}{N}\mathbf{b}_t^- + \frac{1}{N}\mathbf{x}_t, \tag{54}$$

$$\Delta\mathbf{b} = \mathbf{b}_t - \mathbf{b}_t^-, \tag{55}$$

$$\begin{aligned} \sigma_{t,i}^2 = {} & \sigma_{t-1,i}^2 + \Delta b_i^2 - (b_{t-1,i} + b_{t,i})^2 \\ & + \frac{(x_{t,i} - b_{t,i})^2 - (x_{t-N,i} - b_{t-1,i})^2}{N-1}, \end{aligned} \tag{56}$$

$$\Sigma_t = diag(\sigma_{t,1}, \ldots, \sigma_{t,m}), \tag{57}$$

$$\mathbf{x}_t = (\mathbf{x}_t - \mathbf{b}_t)^2\Sigma_t^{-1}, \tag{58}$$

$$\begin{aligned} R_t = {} & \Sigma_t^{-1}\Sigma_{t-1}R_t^-\Sigma_{t-1}\Sigma_t^{-1} + \Sigma_t^{-1}\Sigma_{t-1}\Delta\mathbf{b}\Delta\mathbf{b}^T(\Sigma_{t-1}^-)^{-1} \\ & + \frac{1}{N-1}\mathbf{x}_t\mathbf{x}_t^T. \end{aligned} \tag{59}$$

The algorithm has to store the data within the moving window to be able to remove them from the model (see Eq. (52)). Another issue of the approach is its limitation to linear modelling of the data constrained by the linear PCA method. This particular problem was targeted in Liu et al. (2009a), where the same algorithm was extended to nonlinear modelling using the kernel PCA (see Section 4.2.3). The FMWPCA method was further developed by He and Yang (2008), who proposed to use the rank-r SVD in order to improve the computational efficiency and skip the requirement for storing the full correlation matrix.

4.2.3. Moving Window Kernel PCA

In the Moving Window Kernel PCA (MWKPCA) proposed by Liu et al. (2009a), the adaptation of KCPA model is done in two-steps, which are similar to the Fast Moving Window PCA discussed in Section 4.2.2. The first step also refers to removing the oldest sample from window and the second step to adding newly available sample. Combining these two steps, the mean vector $\mathbf{b}_{\Phi,t}$ in the feature space and the covariance matrix $C_{\Phi,t}$ of the new window are obtained as:

$$\mathbf{b}_{\Phi,t} = \mathbf{b}_{\Phi,t-1} + \frac{1}{N-1}[\Phi(\mathbf{x}_t) - \Phi(\mathbf{x}_{t-N})] \tag{60}$$

$$\begin{aligned} C_{\Phi,t} = {} & C_{\Phi,t-1} - \frac{N-1}{(N-2)^2}(\Phi(\mathbf{x}_{t-N}) - \mathbf{b}_{\Phi,t-1})(\Phi(\mathbf{x}_{t-N}) - \mathbf{b}_{\Phi})^T \\ & + \frac{1}{N-1}(\Phi(\mathbf{x}_t) - \frac{N-1}{N-2}\mathbf{b}_{\Phi,t-1} - \frac{N-1}{N-2}\Phi(\mathbf{x}_{t-N}))\cdot(\Phi(\mathbf{x}_t) \\ & - \frac{N-1}{N-2}\mathbf{b}_{\Phi,t-1} + \frac{1}{N-2}\Phi(\mathbf{x}_{t-N}))^T, \end{aligned} \tag{61}$$

where N is window length. Adapting the KPCA model is thus reduced to the recalculation of the eigenvalues and eigenvectors of $C_{\Phi,t}$. Liu et al. (2009a) proposed numerically efficient procedure for the calculation of the decomposition, which is based upon the eigendecomposition of the Gram matrix and has the complexity $O(n^2)$. Additionally, Liu et al. (2009a) propose a method for the determination of the number of retained principal components as well as a technique for the adaptation of statistical confidence limits of the SPE and T^2 metrics.

4.3. Instance weighting techniques

This section discusses the instance weighting adaptation methods (for the introduction to this type of adaptive methods see Section 2.3.2). The dominating methods in this category are the recursive versions of LS, PLS, and PCA algorithms as well as the incremental versions of Support Vector Machines (SVMs).

4.3.1. Recursive LS

The general linear Least Squares algorithm shown in Section 3.1 can be used incrementally to incorporate new incoming data. An overview of the derivation of the incremental adaptation of LS algorithm can also be found in Jang, Sun, and Mizutani (1997). Further details can be found in Goodwin and Sin (2009).

The goal of the recursive operation is to accommodate the new incoming data sample $\mathcal{D}_t^{online}$ into the model parameters $\mathbf{b}_t$:

$$\mathbf{b}_t = \left(\begin{bmatrix} X_{t-1} \\ \mathbf{x}_t \end{bmatrix}^T \begin{bmatrix} X_{t-1} \\ \mathbf{x}_t \end{bmatrix} \right)^{-1} \left(\begin{bmatrix} X_{t-1} \\ \mathbf{x}_t \end{bmatrix}^T \begin{bmatrix} \mathbf{y}_{t-1} \\ y_t \end{bmatrix} \right). \tag{62}$$

The solution of the above problem is:

$$P_t = P_{t-1} - \frac{P_{t-1}\mathbf{x}_t\mathbf{x}_t^T P_{t-1}}{1+\mathbf{x}_t^T P_{t-1}\mathbf{x}_t^T} \quad \text{with} \tag{63}$$

$$\mathbf{b}_t = \mathbf{b}_{t-1} + P_t\mathbf{x}_t(y_t - \mathbf{x}_t^T\mathbf{b}_{t-1}), \tag{64}$$

where P_t corresponds to the inverted covariance matrix:

$$P_t = \left(X_t^T X_t\right)^{-1}. \tag{65}$$

The problem of the presented incremental technique is that the influence of all samples on the model parameters is equal. Therefore for large data sets, the influence of the new appearing data goes towards zero (Jang et al., 1997):

$$\lim_{t\to\infty} P_t = 0. \tag{66}$$

This can be solved by introducing a forgetting factor into Eq. (11):

$$\mathbf{b} = (X^T W X)^{-1}(X^T W \mathbf{y}), \quad \text{with} \quad W = diag(\lambda^{(n-1)}, \ldots, \lambda^(1), 1), \tag{67}$$

where $0 < \lambda^1 < 1$ is the weighting factor for the data samples. Consequently, Eq. (63) becomes:

$$P_t = \frac{1}{\lambda}\left(P_{t-1} - \frac{P_{t-1}\mathbf{x}_t\mathbf{x}_t^T P_{t-1}}{\lambda + \mathbf{x}_t^T P_{t-1}\mathbf{x}_t^T}\right). \tag{68}$$

Further details of this method can be found in Ljung (1987).

4.3.2. Kernel-based recursive LS

Another method for the implementation of the RLS algorithm was presented in Dayal and MacGregor (1997b). In their work the adaptation is implemented by updating the covariance matrix of the data:

$$X_t^T X_t = \lambda(X_{t-1}^T X_{t-1}) + \mathbf{x}_t^T\mathbf{x}_t, \tag{69}$$

where $\mathbf{x}_t$ is the new incoming data. In order to achieve the recursive adaptation of the coefficient vector $\mathbf{b}$ with the new data point $\mathbf{x}_t$, Eq. (11) is rearranged to:

$$\mathbf{b}_t = \mathbf{b}_t + (X_t^T X_t)^{-1}\mathbf{x}_t^T(y_t - \mathbf{x}_t\mathbf{b}_{t-1}). \tag{70}$$

The above equations ensures that the amount of updates of the coefficient vector is equivalent to the prediction error of the previous model (i.e. $\mathbf{b}_{t-1}$). However, the problem of the presented RLS is that it requires to memorise the whole data matrix X. This can be avoided by inserting:

$$L_t = (X_t^T X_t)^{-1} = (\lambda L_{t-1} + \mathbf{x}_t^T\mathbf{x}_t)^{-1}. \tag{71}$$

into Eq. (70), after which one obtains the following recursive algorithm:

$$\mathbf{b}_t = \mathbf{b}_{t-1} + \frac{L_{t-1}\mathbf{x}^T}{\lambda + \mathbf{x}L_{t-1}\mathbf{x}^T}(y_t - \mathbf{x}_t\mathbf{b}_{t-1}) \tag{72}$$

$$L_t = \frac{L_{t-1} - (L_{t-1}\mathbf{x}_t^T\mathbf{x}_t/\lambda + \mathbf{x}_t L_{t-1}\mathbf{x}_t^T)L_{t-1}}{\lambda}. \tag{73}$$

From the above equations one can see that it does not require storing the data matrix X, the only data that have to be maintained is the matrix $L \in \mathbb{R}^{m,m}$.

4.3.3. Recursive PCA

In order to be able to update the PCA model recursively, the incoming data $\mathbf{x}_t$ has to be normalised in the same way the matrix was normalised in the offline case. For this purpose, it is necessary to recursively update the statistics of the data variables (see Section 4.1.3).

Li et al. (2000) derive the required recursive update equation for the mean $\mathbf{b}$ vectors, variance matrix Σ, and correlation matrix R of the incoming data. The previously mentioned values are required for the recursive calculation of the RPCA algorithm. The incremental version (i.e. only single data point provided for each adaptation step) of the recursive equations has the following form:

$$\mathbf{b}_t = \lambda\mathbf{b}_{t-1} + (1-\lambda)\mathbf{x}_t \tag{74}$$

$$\boldsymbol{\sigma}_{t,k}^2 = \lambda(\boldsymbol{\sigma}_{t-1,k}^2 + (b_{t,k} - b_{t,k})^2) + (1-\lambda)(x_{t,k} - b_{t,k})^2 \tag{75}$$

$$\begin{aligned} R_t &= \lambda\Sigma_t^{-1}(\Sigma_{t-1}R_{t-1}\Sigma_{t-1} + (\mathbf{b}_t - \mathbf{b}_{t-1})(\mathbf{b}_t - \mathbf{b}_{t-1})^T)\Sigma_t^{-1} \\ &\quad + (1-\lambda)\mathbf{x}_t\mathbf{x}_t^T \end{aligned} \tag{76}$$

For the case of block-wise adaptation and further details of the derivation are in Li et al. (2000). As one can see in the above equations, the only values needed for the calculation of the updates are the past mean and variance vectors and the correlation matrix together with the new data.

Li et al. also discuss two algorithms for the calculation of the principal components, which are numerically more efficient than the standard SVD approach. In the incremental scenario (i.e. when only one data point is available for each adaptation step) a rank-one algorithm for the updates of the correlation matrix on sample-by-sample basis is discussed. For this scenario, the algorithm is shown to be numerically more efficient than the Lanczos Tridiagonalization (LT) and the standard SVD approach. In contrast to this for the block-wise operation, the LT algorithm for the decomposition of the matrix R is shown to be more computationally efficient.

The authors also review the approaches for the selection of optimal number of principal components, which are applicable in the recursive modelling scenario.

Finally, Li et al. also describe how to recursively update the process monitoring statistics Q and Hotellings T^2 and their upper limits. In both cases the calculation of the statistics requires only the current data sample $\mathbf{x}_t$.

The problem of this method is however that it requires the storage of the full correlation matrix, which for larger data sets can require too much memory. An approach which deals with this problem was proposed for the block-wise adaptation in Choi et al. (2006).

4.3.4. Recursive PLS

The recursive version of the PLS algorithm was originally published in Helland (1992). However, the original suffered from issues like numerical stability problems in the case of rank deficient data, which was dealt with in Qin (1998). Therefore, in this work Qin's approach is discussed. The recursive algorithm can be used to adapt the model in several ways: (i) incremental adaptation on sample-by-sample basis; (ii) blockwise adaptation; or (iii) by merging two PLS models.

In the first case, the model update is achieved by merging the old model represented by the matrices P, B, and Q with the new data sample $\mathbf{x}_t, \mathbf{y}_t$:

$$X_t = \begin{bmatrix} \lambda P_{t-1}^T \\ \mathbf{x}_t \end{bmatrix} \quad Y_t = \begin{bmatrix} \lambda B_{t-1} Q_{t-1}^T \\ \mathbf{y}_t \end{bmatrix}. \tag{77}$$

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_t, Y_t as usual (see Eqs. (20) and (21)). In order 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 (Qin, 1998). Practically, this condition can be assured by finding a number of latent variables r, which fulfils:

$$||E_r|| \leq \epsilon \quad \text{with} \quad \epsilon \geq 0. \tag{78}$$

In general, the number of latent variables r required for the recursive operation can differ from l, the optimal number of latent variables required for the modelling.

Another advantage of the PLS algorithm is that it does not require the normalisation of the incoming online data $\{\mathbf{x}_t, \mathbf{y}_t\}$ as it is the case with the RPCA. The RPLS algorithm will still work even if the new data is not scaled to the unit variance. In order to deal with the changing mean value of the variables, an intercept term can be added to the data. This can be done expanding the input data vectors: $\mathbf{x}_t \to [\mathbf{x}_t, 1]$. In order to be able to run the recursive version with mean value adjustment, the intercept has to also be taken into account in the offline training data matrix X, which has to have the following form $\left[X, \frac{1}{\sqrt{n-1}}\mathbf{1}_n\right]$ (where $\mathbf{1}_n$ is a vector of ones of the length n).

A similar procedure can be followed for the block-wise adaptation. In this case there can be a new PLS model developed for the current block of data $\{X_t, Y_t\}$, which yields the matrices P_t, B_t, and Q_t. These can be merged with the previous model:

$$X_t = \begin{bmatrix} \lambda P_{t-1}^T \\ P_t^T \end{bmatrix} \quad Y_t = \begin{bmatrix} \lambda B_{t-1} Q_{t-1}^T \\ B_t Q_t^T \end{bmatrix}. \tag{79}$$

The new model can be again developed by applying the PLS algorithm to X_t and Y_t.

4.3.5. Recursive exponentially weighted PLS

Dayal and MacGregor proposed a computationally efficient recursive PLS version in Dayal and MacGregor (1997a) and explained it in further detail in Dayal and MacGregor (1997b). In their work, they show the method to deliver more accurate models than the RLS method discussed in Section 4.3.2. The computational efficiency origins from the application of the kernel algorithm for the PLS computation (Lindgren, Geladi, & Wold, 2005) rather than the NIPALS method. The proof of the computational efficiency of Dayal's and MacGregor's method was also presented in Dayal and MacGregor (1997a).

The method is based on the calculation of covariance matrices of the scaled and centred data $C^{xx} = X^T X$ and between the input and output data $C^{xy} = X^T Y$. The kernel matrix is calculated as:

$$Y^T X X^T Y = \left(C^{xy}\right)\left(C^{xy}\right)^T. \tag{80}$$

The first step of the method is the calculation of the PLS weight vector $\mathbf{q}_i$, which corresponds to the largest eigenvalue of the kernel matrix (using e.g. the SVD method). This step is repeated for each of the required latent variables ($i = 1, \ldots, l$) as follows:

$$\mathbf{q}_i \approx (Y^T X X^T Y)_i \mathbf{q}_i \tag{81}$$

$$\mathbf{w}_i = X^T Y \mathbf{q}_i \tag{82}$$

$$\mathbf{w}_i = \frac{\mathbf{w}_i}{||\mathbf{w}_i||} \tag{83}$$

When calculating the first latent variable, i.e. $i = 1$, then:

$$\mathbf{r}_1 = \mathbf{w}_1, \tag{84}$$

otherwise

$$\mathbf{r}_i = \mathbf{w}_i - \sum_{j=1}^{i-1} \mathbf{p}_j^T \mathbf{w}_i \mathbf{r}_j. \tag{85}$$

With the vector $\mathbf{r}_i$ available, the loadings vectors of the input and output data $\mathbf{p}_i$ and $\mathbf{q}_i$ can be calculated:

$$\mathbf{p}_i^T = \frac{\mathbf{r}_i C^{xx}}{\mathbf{r}_i^T C^{xx} \mathbf{r}_i} \tag{86}$$

$$\mathbf{q}_i^T = \frac{\mathbf{r}_i C^{xy}}{\mathbf{r}_i^T C^{xx} \mathbf{r}_i} \tag{87}$$

The last computation is the deflation of the input–output covariance matrix:

$$C_i^{xy} = C_{i-1}^{xy} - \mathbf{p}_{i-1}^T \mathbf{q}_{i-1}^T (\mathbf{t}_{i-1}^T \mathbf{t}_{i-1}). \tag{88}$$

One of the advantages of this algorithm is that it requires only the deflation of one of the covariance matrices. After finishing the calculation of all l latent variables, the regression coefficients of the PLS can be calculated as:

$$\mathbf{b} = W(P^T W)^{-1} Q^T = R Q^T, \tag{89}$$

where $W = [\mathbf{w}, \ldots, \mathbf{w}_l]$, $Q = [\mathbf{q}, \ldots, \mathbf{q}_l]$, $P = [\mathbf{p}, \ldots, \mathbf{p}_l]$, $R = [\mathbf{r}, \ldots, \mathbf{r}_l]$.

The recursive operation of the algorithm can be achieved by simply updating the covariance matrices C^{xx} and C^{xy} with the new data points:

$$C_t^{xx} = \lambda C_{t-1}^{xx} + \mathbf{x}_t^T \mathbf{x}_t \tag{90}$$

$$C_t^{xy} = \lambda C_{t-1}^{xy} + \mathbf{x}_t^T \mathbf{y}_t \tag{91}$$

Dayal and MacGregor also demonstrate how to keep the incoming data centred and scaled by using the following equations for the mean and variance updates:

$$\mathbf{b}_t = \frac{\mathbf{x}_t^a}{N_t} \tag{92}$$

$$\sigma_{t,k} = \frac{(x_{t,k}^a)^2 - N_t b_{t,k}^2}{N_t - 1}, \tag{93}$$

where $\mathbf{x}_{k,t}^a$ is the weighted cumulative sum of $\mathbf{x}$ (i.e. $\mathbf{x}_t^a = \lambda \mathbf{x}_{t-1}^a + \mathbf{x}_t$), N_t is the effective memory length of the data, which is also calculated recursively as $N_t = \lambda N_{t-1} + 1$. The sum of squares $(x_{t,k}^a)^2$ can be read from the covariance matrix C^{xx}.

Table 1
List of reviewed case studies of adaptive soft sensors. Process type: *B* atch, *C* ontinuous; Application type: *O* nline *P* rediction, *P* rocess *M* onitoring, *I* nferential *C* ontrol; Adaptation attributes: *R* ecursive adaptation, *M* oving *W* indow adaptation, *E* nsemble method, *S* ample-*W* ise adaptation, *B* lock-*W* ise adaptation.

No.	Application	Process type [B,C]	Application type [OP,PM,IC]	Adaptation attributes	Prediction method	Ref.
1	Paper machine—inferential control	C	IC	R, SW	EWPCA,EWPLS	Wold (1993)
2	Simulated Continuous Stirred Tank Reactor—inferential control, Mineral flotation circuit—10 Response variables prediction	C	OP	R, SW	KPLS	Dayal and MacGregor (1997b)
3	Catalytic reformer—Research octane number prediction	C	OP	R + MW, SW + BW	PLS	Qin (1998)
4	Polymerisation process monitoring	B	PM	R, BW	HPCA	Raennar et al. (1998)
5	Semiconductor processing—rapid thermal annealing process monitoring	B	PM	R, SW + BW	PCA	Li et al. (2000)
6	Waste-water treatment—Plant fault detection	C	PM	R, BW	MSPCA	Lennox and Rosen (2002)
7	Refinery—FCCU simulation; Distillation—Butane purification;	C	PM	R, BW	RPLS	Wang et al. (2003)
8	Manufacturing—Continuous sheet production monitoring	C	PM	R, SW	EWPCA	Lane et al. (2003)
9	Polymerisation (PVC) process—Plant monitoring	B	PM	MW, BW	Multi-Way PCA	Zhao and Chai (2004)
10	Waste-water treatment—fault detection and identification	C	PM	MW, BW	CPCA	Lee and Vanrolleghem (2004)
11	Simulated pH neutralisation process; Propylene polymerisation problem—Melt flow rate prediction	C	OP	R, SW + BW	NPLS	Li et al. (2005)
12	Simulated linear ARMA process, Refinery—Fluid catalytic cracking unit simulation	C	PM	MW, SW	PCA	Wang et al. (2005)
13	Waste-water treatment—full scale anaerobic filter	C	OP	MW, BW	ANN-ARX	Lee et al. (2005b)
14	Biological waste-water treatment- sequencing batch reactor process monitoring	B	PM	R, SW	Multi-Way MSPCA	Lee et al. (2005a)
15	Crude oil distillation—Evaporation temperature of kerosene prediction	C	OP	E, SW	NFS	Macias et al. (2006)
16	Purified Terephthalic acid—Average crystal size prediction	C	OP	R (SW) + MW (BW)	PLS + OC	Mu et al. (2006)
17	Industrial fired heatermonitoring	C	PM	R, BW	PCA	Jin et al. (2006)
18	Simulated continuous stirred tank reactor—Process monitoring	C	PM	R, SW + BW	PCA	Choi et al. (2006)
19	Tennessee Eastman Process—Fault classification	C	PM	R, SW	AKL	Wang et al. (2006)
20	High pressure polyethylene plant—Polyethylene melt index prediction	C	OP	E, SW	PCA + FCM + RLS	Liu (2007)
21	Fermentation (Penicillin) process—Process fault detection and diagnosis	B	PM	MW, SW	EDISSIM	Zhao et al. (2007)
22	Grinding mill—Particle size estimation	C	OP	R, SW	LS-ARMAX	Sbarbaro et al. (2008)
23	Simulated fermentation process—penicillin production	B	OP	R, SW	SVR	Liu et al. (2008)
24	Polymerisation process—P-xylene purity modelling	C	OP	E + R, SW	FCM + SVR	Fu et al. (2008)
25	Complex refining process—condensate fractionation process	C	PM	MW, BW	PCA, MBPCA	AlGhazzawi and Lennox (2008)
26	Tennessee Eastman Process	C	PM	R, SW	JITL + LSSVR	Ge and Song (2008)

27	Simulated random process, CSTR	C	PM	MW, SW	PCA	He and Yang (2008)
28	Polymerisation (HDPE) process–Melt index prediction	C	OP	R, SW	PLS + OC	Ahmed et al. (2009)
29	Distillation column (o-xylene)–Isopropylbenzene impurity prediction	C	OP, IC	R, SW	SLR	Ma et al. (2009)
30	Industrial drier–Residual humidity prediction, Thermal oxidiser–NOx prediction, Polymerisation reactor–Catalyst activation prediction	C	OP	E + R + MW, SW + BW	MLP, SVR, etc.	Kadlec and Gabrys (2009)
31	Simulated nonlinear process, distillation column–butane purification	C	PM	MW, SW	KPCA	Liu et al. (2009a)
32	Refinery process–Fluidized Catalytic Cracking Unit	C	OP	R, SW	LS-SVR	Liu et al. (2009b)
33	Rubber mixing plant–Viscosity prediction	C	OP	R, SW	AKL	Yang et al. (2009)
34	Floatation process–mineral flotation flurry contest prediction	C	OP	E,R, SW	PLS	Haavisto and Hytyniemi (2009)
35	Fermentation (Penicillin) process–Process monitoring; Polymerisation (Polyester) process–Process monitoring	B	PM	MW, BW	kNN + PCA	Facco et al. (2010)
36	Polymerisation reactor–Catalyst activation prediction	C	OP	E, SW	PLS	Kadlec and Gabrys (2010)
37	Polymerisation reactor–Catalyst activation prediction	C	OP	E, SW	PLS	Kadlec and Gabrys (in press)

4.3.6. *Adaptive Kernel Learning*

Originally, an incremental version of the SVM (see Section 3.4) was published in Cauwenberghs and Poggio (2001). The adaptive operation of Incremental Support Vector Machines (ISVM) consists of two steps, namely adding new support vectors to the model and removing of support vectors from the model. The underlying ideas were taken over and the *Adaptive Kernel Learning* framework was proposed in Wang et al. (2006). It is an adaptive algorithm based on the kernel framework with the focus on online prediction and fault detection soft sensors. The framework is based on the Least-Squares SVM (Suykens, 2002). The authors propose to use the Tikhonov cost function (Rifkin, Yeo, & Poggio, 2003) for the optimisation problem and show that this cost function can be used for both regression (i.e. online prediction) and classification (i.e. fault detection) problems. They also demonstrate an implementation of the kernel algorithm with the attention paid to *forward* learning, where new nodes are inserted into the model, as well as *backward* learning, which is a pruning mechanism for the removal of redundant nodes from the model.

4.4. *Ensemble-based methods*

In this category, the number of methods is rather limited. A representative method, which demonstrates well the possibilities of the this type of adaptation methods is the Incremental Local Learning Based Algorithm (ILLSA) (Kadlec & Gabrys, 2010b), which is in detail discussed in the next section.

4.4.1. *Incremental Local Learning Soft sensing Algorithm*

This method is based on the weighted combination of predictions of several models:

$$\hat{y} = \sum_{i=1}^{p} \omega_{t,i}(\mathbf{x}, f_{t,i}) f_{t,i}(\mathbf{x}). \tag{94}$$

The predictive models $f_{t,i}()$ are called local experts because they are trained using sub-sets of the historical data. The partitioning algorithm used in Kadlec and Gabrys (in press) splits the data in such way that the subsets represent parts of the data with a linear relationship between the input and output data. Consequently, it is possible to use a linear modelling technique for the local experts. The local experts themselves are based on RPLS (see Section 4.3.4).

Eq. (94) indicates that the combination weights $\omega_{t,i}$ depend on the current data sample $\mathbf{x}_t$ and the prediction of the corresponding local expert $f_{t,i}$. In ideal case, the weights should reflect the estimated prediction accuracy of the local experts (Krogh & Vedelsby, 1995). In ILLSA, this is approached by storing two dimensional maps that model the relative (in respect of the other local experts) prediction accuracy of the models. A drawback of this method is that the map has to be stored for each input-output variable pair, however if combined with dimensionality reduction of the PLS, this problem can be partly relaxed. Eq. (94) also shows that the combination weights are time dependent. The adaptation of the combination weights corresponds to changing the responsibility of the local experts for the final prediction, and can be achieved by a simple multiplication of the maps by an adaptation map (i.e. through a matrix element-wise multiplication) that reflects the relative performance of the local expert for the online data point. Additionally, ILLSA exploits the flexibility of the ensemble adaptation by linking the forgetting factor of the RPLS-based local experts to their weights. By assigning the local expert with the highest weight (i.e. the one most responsible for the prediction) a high forgetting factor, it can be achieved that the local expert is focused (i.e. adapted) on the data, where it is expected to deliver accurate predictions.

The same framework was exploited in Kadlec and Gabrys (2010), where a soft sensing method which is able to start making predictions with very limited amount of historical data was presented.

4.5. Adaptation of monitoring metrics

In process monitoring and fault detection applications it is necessary to update the monitoring metrics. The most commonly used metrics for the monitoring of industrial processes are: Hotelling's T^2; (ii) and the SPE. These metrics can be calculated using the following equations:

$$T^2 = \boldsymbol{\tau}^T \Lambda \boldsymbol{\tau} \quad \text{with} \quad \Lambda = diag(\lambda_1, \ldots, \lambda_l) \tag{95}$$

$$SPE^x = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2 \tag{96}$$

$$SPE^y = ||\mathbf{y} - \hat{\mathbf{y}}||_2^2, \tag{97}$$

where τ stands for the latent vectors in the case of PLS model, or principal components in the case of PCA. The SPE values can be calculated for the input data vectors $\mathbf{x}$, as well as output vectors $\mathbf{y}$ if necessary. These metrics need to be recalculated each time the underlying model changes (He & Yang, 2008; Li et al., 2000; Wang, Kruger, & Lennox, 2003;). Based on the assumption that the T^2 follows the F-distribution (Qin, 2003) and the SPE the central χ^2-distribution (Nomikos & MacGregor, 1995), the updates of the metrics are followed by the recalculation of the confidence limits T^2_β and SPE_β respectively:

$$T^2_\beta = \frac{l(n^2-1)}{n(n-l)} F_{l,n-l,\beta} \tag{98}$$

$$SPE_\beta = \theta_1 (1 + \frac{c_\alpha \sqrt{2\theta_2 h_0^2}}{\theta_1} + \frac{\theta_2 h_0 (h_0 - 1)}{\theta_1^2})^{1/h_0} \tag{99}$$

$$\text{with} \quad h_0 = 1 - \frac{2\theta_1\theta_3}{3\theta_2^2} \quad \text{and} \quad \theta_i = \sum_{j=l+1}^{m} \lambda_j^i, \qquad i = 1, 2, 3. \tag{100}$$

In a block-wise adaptive operation, the monitoring metrics can be updated using the exponential weighting approach (Jin, Lee, Lee, & Han, 2006):

$$T_t^2 = \lambda T_{t-1}^2 + (1-\lambda) T_{t-2}^2 \tag{101}$$

$$SPE_t = \lambda SPE_{t-1} + (1-\lambda) SPE_{t-2}. \tag{102}$$

5. List of adaptive data-driven soft sensors

This section provides an overview of published adaptive data-driven soft sensors, which are, together with their main attributes, summarised in chronological order of their appearance in the literature in Table 1. Interested reader can find further details of the listed publications in Appendix A.

6. Data sets for testing of adaptive soft sensors

In order to be able to objectively assess the performance of different soft sensing methods, it is necessary to benchmark these methods on publicly available data sets. For this purpose we provide a review of several available data sets and their characteristics. This should facilitate the exchange of these data sets and the compatibility of presented results. A summary of all data set can be found in Table 2.

(1) Polyester resin production plant—Activity number and viscosity prediction Facco, Doplicher, Bezzo, and Barolo (2009): This data set describes a polyester resin production plant. The main part is a $12m^3$ stirred tank reactor, which is used for the production of different resins. Other parts of the process are a distillation column for the removal of water from the product, a water cooled condenser, a scrubber, and vacuum pump maintaining vacuum in the reactor. There are 34 easy-to-measure variables measured every 30s. The run length of each of the batches varies and therefore the number of samples changes from batch to batch and is in the range between 4500 and 7500. There are 27 batches available. As for the difficult to measure variables, these are the activity number N_A and the viscosity of the product μ. These variables are measured with much lower sampling rate. In fact, there are only 15-20 measurements of quality variables available, which makes a online prediction soft sensor for this process highly desirable. More details about this process can be found in Facco et al. (2009) and Facco, Bezzo, and Barolo (2010).

(2) Fermentation process (Penicillin production) Birol, Undey, and Cinar (2002)[1]: The PenSim simulator provides a simulation of a fed-batch fermentation process for penicillin production. The main component of the process is a fermentor, where the biological reaction takes place. The simulator implements a control loop for the control of pH and fermenter temperature in the reactor. The simulator is very flexible and provides several settings including the settings of the controller, process duration, sampling rates, etc. An example of typical settings can be found in Birol et al. (2002), Facco et al. (2010), Lee, Yoo, and Lee (2004).

(3) Polymerisation process—catalyst activation (Kadlec & Gabrys, 2010b): This data set describes a polymerisation reactor consisting of approximately 1000 tubes filled with catalyst, which is used to oxidise a gaseous feed. The reaction speed has a strong nonlinear relationship to the temperature. Its exothermal reaction is counteracted by the cooling and leads to a temperature maximum somewhere along the length of the tube. As the catalyst decays, this becomes less pronounced and moves further downstream. The catalyst activity usually decays within some time to zero. The process takes input from other, larger processes, so that the feed will vary over the time. The operators react to this by choosing appropriate operating conditions. The catalyst decay is however much slower than these effects. The process is equipped with measurements to log all the variations of the feed and the operating conditions. There are measurements showing concentrations, flows and several temperatures along the length of a characteristic tube to identify the state of the processes. The task is to predict the activity of a catalyst in a multi-tube reactor. The target variable is a simulated activity of the catalyst inside the reactor. The data set covers one year of operation of the process plant. Many of the variables show high co-linearity of the features and a large number of outliers, which can be found in as many as 80% of the variables.

(4) Tennessee Eastman Process (Downs & Vogel, 1993)[2]: The simulated TEP is widely used for the evaluation of process monitoring and fault detection soft sensors. The main components of the process are: a reactor, condenser, compressor, separator, and a stripper. The process simulates an irreversible, exothermic, and first order reaction. The process simulation consists of 42 measured and 12 manipulated variables. The sampling rate of the measured variables is either 6, or 15 minutes dependent on the measurement type. There is also a set of 21 standard faults available for this process. A set of simulated data sets for a normal operation as well as the 21 standard faults of the TEP process is available. Each of the 22 sets is further split into training data set with 480 samples and a testing set containing 960 samples.

[1] http://en.bio-soft.net/bioengi/PenSim.html.
[2] http://brahms.scs.uiuc.edu.

Table 2
Publicly available data sets and their characteristics. Process type: *B* atch, *C* ontinuous; *n* is number of data points; *m* is number of easy-to-measure variables.

No.	Process	Process type [B,C]	*n*	*m*	Target variable(s)	Ref.
1	Polymerisation process	B	4500–7500 per batch	34	Activity number; viscosity prediction μ	Facco et al. (2009)
2	Fermentation (Penicillin) process (simulated)	B	Variable	6	9 Different variables	Birol et al. (2002)
3	Polymerisation process	C	8687	15	Catalyst activation (simulated)	Kadlec and Gabrys (in press)
4	Tennessee Eastman process	C	480 (train) and 960 (test) per fault, 21 faults	52	n.a.	Chiang, Russell, and Braatz (2001)
5	Debutanizer column	C	2393	7	Butane concentration	Fortuna (2007)
6	Sulfur Recovery Unit	C	10,080	5	H_2S and SO_2 concentrations,	Fortuna (2007)
7	Distillation column	C	190–300 (train) and 380–600 (test) per fault, 10 faults	19	n.a.	Seng Ng and Srinivasan (2010)

(5) Debutanizer column—Butane concentration (Fortuna, 2007)[3]: The debutanizer column is a part of refinery process and its task is to remove propane and butane from a stream of naphta. The target variable, the concentration of butane in the debutanizer bottoms, is measured by a chromatograph. The input data consists of seven variables including pressures, temperatures and flows in and around the column. Altogether there are 2393 measurements available.

(6) Sulfur Recovery Unit—H_2S and SO_2 concentration in exhaust gasses (Fortuna, 2007)[4]: The goal of the SRU is to remove sulfur from exhaust gas streams before they are released to the atmosphere. The input data of this data set are formed by five different gas flows in the SRU. There are two variables to be predicted, namely the residual concentrations of H_2S and SO_2, which are normally measured by hardware sensors. However because the gasses are very aggressive and damage the sensors, these have to be removed and maintained frequently. Therefore a soft sensor is desirable for these measurements.

(7) Distillation columns—process monitoring and fault detection (Seng Ng & Srinivasan, 2010)[5]: This data set describes a two meters tall distillation column with ten trays. There are 19 hardware sensors measuring mainly temperatures and flows in the column implemented at a sampling interval of ten seconds. The start-up of the process normally takes two hours. The data set provides among the normal operation of the plant ten different faults (abrupt and slow drifts) such as sensor fault, failure to open pump, too high a reflux ratio for the evaluation of process monitoring and fault detection soft sensors.

7. Conclusions

The aim of this article is to provide an overview of adaptive techniques and approaches for data-driven soft sensing research. To achieve this, we have discussed the theoretical framework for adaptive process modelling. Based on this framework, we have categorised the available methods into three categories. These are: (i) Moving Window-based (Instance selection) methods; (ii) Recursive (Instance weighting) methods: and (iii) Ensemble-based methods. In the literature, there are many publications proposing different algorithms for each of the categories. From the reviewed case studies, it is obvious that most of the adaptive soft sensors are based on Principle Component Analysis and Partial Least Squares methods, which were established as methods of choice even before the appearance of first adaptive soft sensors. Another reason for the popularity of these methods is their inherent ability to deal with the data collinearity problem that is very common in the process industry data. However, the main reason for their popularity is the fact that these methods can be easily combined with the moving windows and recursive adaptation strategies.

As for the question: *which adaptation method is the best one?*, this depends on the application problem. For process monitoring problems, the Recursive Principle Component Analysis and Recursive Partial Least Squares methods are preferable because they enable calculation of the monitoring metrics in a straightforward way. Also, these methods can be operated either in sample-wise or block-wise adaptation modes. If it is known that the process to be monitored shows a nonlinear behaviour then the kernel-based methods should be preferred. Especially, the Adaptive Kernel Learning framework might be particularly of interest. However, in general when using the kernel-based methods, the choice of the kernel function and its parameters may have critical influence on the soft sensor performance and should be considered carefully. The choice of adaptive methods for online prediction soft sensors is more flexible. In the case of simple linear processes probably the Recursive Partial Least Squares Method is the most established one. In systems with nonlinear behaviour, one should restrain to the kernel-based methods like Incremental Support Vector Regression or the Adaptive Kernel Learning Framework. Another interesting possibility in such a case is the application of local learning and ensemble methods, such as the Incremental Local Learning-based Soft sensing Algorithm, where the linear local models focus on partitions of data with linear relation between the input and output data. This has the benefit of relatively simple and transparent model structure together with large flexibility for the application of different adaptation methods. This method can also be used in a scenario with limited amount of data. In processes with multiple overlapping dynamics again the ensemble methods, where each of the ensemble members can adapt with different speed, can be of benefit. Another type of algorithm that can be used in a multiple dynamics system is the Multi-Scale Principle Component Analysis, which uses the Wavelet Transformation to split the data into several scales. In case of batch processes, the moving window adaptation in combination with Principle Component Analysis seems to be very popular. In particular this method can be used to relax the

[3] http://www.springer.com/engineering/book/978-1-84628-479-3.
[4] http://www.springer.com/engineering/book/978-1-84628-479-3.
[5] http://www.iace.eng.nus.edu.sg/research/Distillation_column/index.htm.

requirement for filling in missing data between current time and the end of the batch run.

One aspect that should be always kept in mind is to use the explicitly available process knowledge for the setting of the adaptation speed of the model (i.e. moving window size or forgetting factor). If such a knowledge is not available then techniques for the variable setting of adaptation speed have to be used in order to achieve a good balance between the stability and plasticity of the soft sensor.

Appendix A. List of case studies

(1) Wold (1993): This is to our best knowledge the first publication dealing with adaptive soft sensing. It extends the previously published Exponentially Weighted Moving Average (EWMA) to the PCA/PLS models. The EWMA updates rely on weighted sum of the previous samples and the current data point as described in Section 4.1.3. In this way the weights of the samples are exponentially decreasing with increasing temporal distance to the current data point. Wold uses the EWMA framework to forecast the latent vectors (PLS) and principal components (PCA) for one-step-ahead process monitoring. Wold also proposed a framework for the adaptation of the PCA/PLS models, however this requires storing all past data together with a memory loading matrix and an auxiliary data matrix spanning (together with its memory loading matrix) the data space. The algorithm is applied to paper machine process control.

(2) Dayal and MacGregor (1997b): The recursive PLS algorithm proposed in this work was discussed in detail in Section 4.3.5. Therefore here, only the case study presented in the paper is outlined. In the first instance, the algorithm is applied to a simulated continuous stirred tank reactor (similar simulated process was also used in Choi et al. (2006)) and its adaptive control. The dynamic controller relies on the prediction of the reactor conversion and temperature, which are both predicted using separate ARX models.

For this process, the authors simulate a disturbance, which is a random walk signal added to the output variables. The authors report that the best control performance was achieved by a locally linearised model based on the mechanistic model of the process. The results of the proposed recursive PLS method were second best, with some stability issues that are discussed by the authors. The proposed recursive method clearly outperformed a simple recursive least squares model, which had major issues with stability. As next the method was also applied to a real-world mineral flotation circuit. The goal was to develop a regression model for the prediction of ten output variables. For this process, not only an outstanding performance of the recursive PLS when compared to a RLS model was shown but also the benefits of the application of the variable forgetting factor was demonstrated.

(3) Qin (1998): The algorithm proposed in this work was discussed in detail in Section 4.3.4. The main contribution of this work is the clarification of RPLS method and the proposal of its block-wise version. Qin describes a moving window type of adaptation as well as a version with a forgetting factor of the recursive algorithm. The work also demonstrates how the block-wise algorithm can be efficiently used in cross-validation framework by reusing the partial models from the particular folds for building of the final model. The case study used to demonstrate the performance of the algorithm is the prediction of the octane number in catalytic reformer process. A high prediction performance and efficiency of the proposed block-wise recursive PLS is shown.

(4) Raennar, MacGregor, and Wold (1998): The proposed algorithm is an implementation of the hierarchical PCA method. The drawback of the original hierarchical PCA was that it required filling in the data between the current time and the end of the batch, a problem that is dealt with by Raennar et al. The difference of the new method is that it processes only one time slice rather than all of the blocks at once. The algorithm includes a weighting factor which plays the role of forgetting factor that should be set according to the dynamics of the process. The algorithm was successfully applied to the monitoring of a batch polymerisation process. The results presented are comparable to those presented in the original hierarchical PCA paper, which demonstrates the capability of the method to be used for process monitoring without the need to fill the missing data samples.

(5) Li et al. (2000): The main contribution of this paper is the presentation of two efficient algorithms for the eigendecomposition of the recursively adapted correlation matrix which were discussed in Section 4.3.3. The work also discusses different methods for the determination of the optimal number of principal components that are applicable online. Another contribution of the work is the outline of a complete approach for adaptive process monitoring including the updates of the monitoring statistics. The authors also apply the adaptive monitoring method to a rapid thermal annealing process in semiconductor processing. The experiments deal with two cases: (i) with a small amount of training data case; and (ii) a larger set of training batches. In both cases the benefit of the adaptive models were shown.

(6) Lennox and Rosen (2002): The main contribution of this work is the proposal of a multi-scale version of the PCA for process monitoring. The scale decomposition of the input variables is achieved using the wavelet transformation. In this way the process monitoring can take into account different time-scale of the signals. In the adaptive framework the adaptive PCA uses the recursive covariance matrix updates described in Section 4.3.5, which is applied to each of the scales. In the work of Lennox and Rosen, the algorithm is further extended to enable the usage of different number of PCs for each scale, which can also be changing. The method is applied to the monitoring of a waste-water treatment process. When compared to the simple adaptive PCA, the proposed method was faster in identifying slow changes in the process. An issue of the algorithm is the difficulty of the interpretation of the faults which becomes more complex due to the monitoring of different scale.

(7) Wang et al. (2003): This work provides a detailed discussion of adaptation aspects in process monitoring. The difficulties of monitoring of time-varying processes using static PLS approaches are outlined by the authors. In first instance the authors distinguish between non-stationary and time varying processes and demonstrate that in the case of non-stationary processes, where the relation between the variables does not change, it is the T^2 statistics, which causes violations of the control limits. On the other hand in the case of time varying processes, it is the SPE of the output variables that violates the control limits. As a result, the model needs to be adapted in the case of time varying processes. The method chosen by the authors is the RPLS described in Section 4.3.4. Consequently, the authors propose a method for the adaptation of the confidence limits of the monitoring statistics using a kind of moving window technique, which requires

the determination of the optimal window length. The authors also propose a recursive version of the multi-block PLS. The resulting model is demonstrated to be able to deal with both non-stationary as well as time varying processes. The method is applied to a simulated (Fluid Catalytic Cracking Unit) FCCU process and to industrial distillation process. In both cases number of different faults is dealt with using the proposed method.

(8) Lane, Martin, Morris, and Gower (2003): In this work, the recursive adaptation method discussed in Section 4.3.5 is applied to the PCA. In this way, the original Exponentially Weighted PCA from Wold (1993) is made to operate in a recursive way without the need to store the past data. Furthermore, the authors use the popular method for dynamic setting of the forgetting factor from Fortescue et al. (1981). Part of the proposed method is also the dynamic adjustment of number of principal components, which is based on keeping the amount of explained variance constant. The updates of the model are done only if the control statistics T^2 and SPE do not violate the control limits. In such a case the new control limits are calculated based on the updated model. Otherwise, an inspection of the data and of the process using contribution plots is required. The proposed method is applied to the monitoring of a polymer film production process.

(9) Zhao and Chai (2004): This work proposes a moving window version of the Multi-way PCA. By using this approach the algorithm relaxes the requirement of fixed run length of the batches. However, the batches still need to be synchronised. The authors also discuss the way in which the model deals with nonlinear aspects of the data. This is achieved by splitting the data into smaller, localised, partitions, which is followed by the development of models for each of the partitions. Another advantage is that the algorithm does not require to fill the data from the current time point until the end of the batch, as it compares only small time windows of data. The algorithm is applied to the monitoring of an industrial polymerisation process.

(10) Lee and Vanrolleghem (2004): This work proposes the application of adaptive Consensus PCA for the monitoring of batch processes that is based on the general moving window approach (see Section 2.3.1). The Consensus PCA is similar to the Hierarchical PCA (Raennar et al., 1998) in a sense that it partitions the available data into a number of blocks. Unlike, the hierarchical PCA, which applies the PCA to the whole block of data, it applies the PCA algorithm to each vector corresponding to one time point. This method has the advantage that it does not require the estimation of the future data from the current time to the end of the batch run. The consensus method delivers similar results as the multiway PCA with the additional advantage of providing simple means (SPE contributions plot) for the interpretations of the faults as shown using a waste-water treatment process case study.

(11) Li, Ye, Wang, and Zhang (2005): This work deals with the fact that the traditional PLS can not handle nonlinear relations between the input and output data space. The proposed approach to the problem includes nonlinearly transformed input data in the PLS input data space. This is achieved by using the outputs of hidden units of a Radial Basis Function Network (RBFN) together with a unit vector as additional input variables. Next, the nonlinear PLS is embedded into an adaptive framework. The performance monitoring of the model is achieved using a moving window. If a decrease in performance is noticed there are two adaptation steps required. On one hand the RBFN is adapted, which is done by adding a new hidden node to the network if the distance of the new data sample to the other nodes is larger than a pre-defined threshold. On the other hand, the PLS model itself need to be adapted, which is done by exploiting the recursive adaptivity of RPLS (see Section 4.3.4). The algorithm was applied to a simulated pH neutralisation process, which exhibits strong time varying behaviour that was effectively dealt with by the model. In another case study, the model was applied to a propylene polymerisation process showing strong nonlinearity.

(12) Wang et al. (2005): The details of the methods proposed in this publication can be found in Section 4.2.2. Apart from the Fast Moving Window PCA, which is a combination of a moving window approach and a recursive method, the authors propose an N-step ahead monitoring scheme, which is equivalent to using an N-samples old model for the monitoring of current data. They demonstrate the advantage of looking N-steps-ahead, instead of using the traditional one-step-ahead method for monitoring a process. There are two case studies dealing with simulated processes. The first experiment deals with a simple ARMA process of first order in which a ramp-like disturbance is injected. The second discussed process is a FCCU simulation (also used in Liu, Hu, Wang, & Li, 2009b). In this process, two drifts of two input variables were implemented. The first drift was assumed to be known and had to be compensated for by the model (without raising alarms) while the second drift was unknown and should trigger process fault alarms. The results have shown that the N-step ahead FMWPCA is in contrast to the traditional one-step-ahead MWPCA able to deliver the expected behaviour.

(13) Lee et al. (2005b): In this work ANN with different model structures (ARX, ARMAX) are applied to the prediction of some critical features in a waste-water treatment anaerobic filtering process. The soft sensors predicts pH values, gas (total, carbon dioxide, methane), production rates, and total oxygen demand. All of these values are normally measured by taking and analysing samples. The authors discuss the model development process in detail and point out the need for rigorous cross-validation to be able to select optimal model parameters. Nonetheless, the developed models' performance seems to be rather low and therefore a moving window adaptation (see Section 2.3.1) with fixed window and step size for periodic model retraining is applied. The adaptive models perform significantly better and the best version is identified as moving window ARX artificial neural network soft sensor.

(14) Lee, Park, and Vanrolleghem (2005a): In this paper there is an adaptive Multi-Scale Multi-Way PCA for fault detection and diagnosis of batch processes proposed. When disturbances and events occur in different time-scales multiscale PCA is better suited for extracting information in comparison with single-scale PCA (Bakshi, 1998). To overcome the problem of changing process conditions authors employed adaptive multiscale PCA model with the recursive updating of the covariance matrices by exponentially discounting the old data (Dayal & MacGregor, 1997b) at each scale. The number of significant principal components was calculated recursively using the Cumulative Percent Variance (CPV) method (Li et al., 2000). To prevent adopting to disturbances and failures, model updating was skipped when SPE and T^2 statistics of new batches were exceeding predetermined limits. The method was applied for monitoring sequencing batch reactor (SBR) process for biological waste-water treatment which is highly nonlinear and time-varying batch process. Results

show that multiscale multiway PCA method is more successful in detection faults than single-scale adaptive MPCA approach.

(15) Macias, Angelov, and Zhou (2006): In this work there is an evolving Neuro-Fuzzy System based on the xTS algorithm (Angelov & Zhou, 2006) proposed. In this model, the antecedent parts of the rules describe the local validity of the rule in the input-output data space. The adaptive (evolving in terms of NFS) behaviour of the algorithm is achieved by online updates of the antecedent parts of the fuzzy rules using a recursive least squares algorithm. The model is applied to predict several critical parameters of a refinery process. These are: (i) temperature of heavy naphta when it evaporates 95% liquid volume; (ii) temperature of kerosine when it evaporates 95% liquid volume; (iii) Abel inflammability of kerosine; and (iv) temperature of gas oil when it evaporates 95% liquid volume. The algorithm is additionally able to extract easily readable rules for the prediction of the above values.

(16) Mu et al. (2006): In this work, the authors propose an adaptation strategy which alternatively applies sample-wise offset compensation and the recalculation of the PLS model. Although, the authors call the PLS adaptation to be a *recursive* method, the only recursive part are the updates of the mean and variance of the data (see Section 4.1.3). The PLS is recalculated using the re-scaled data after collecting a sufficient amount of data, i.e. in a moving window manner (refer to Section 2.3.1). The method is applied to an industrial process for the purification of terephthalic acid to predict the average size of crystals. For this process the method was shown to outperform the offline, dynamic, and recursive PLS methods.

(17) Jin et al. (2006): This paper proposes an adaptive process monitoring scheme based on recursive PCA (see Section 4.3.3). The authors focus on distinguishing between process alarms due to the two alarm causes. In the first case the monitoring model needs to be updated using the new incoming data. In the case of detected disturbances, these should be isolated and further analysed and excluded from model adaptation. In the case of violation of the monitoring statistics (T^2 and/or SPE) a set of fixed rules, which were created using process knowledge is used to distinguish between mode changes and disturbances. The rules are based on the observation of the development of the monitoring statistics. The update of T^2 and SPE statistics are done also in a recursive way. Furthermore, the statistics of the current data block are weighted using the DISSIM similarity index. The DISSIM metric is also used to recognise the end of the changes and thus to terminate the update process, which prevents the model from unnecessary updates. The algorithm is applied to the monitoring of an industrial fired heater and demonstrated to perform better than the plain RPCA (Li et al., 2000) method as well as the conventional PCA. The authors discuss four different rules for the detection of mode changes in the process (e.g. change of the set point of the outlet temperature).

(18) Choi et al. (2006):This work proposes a version of the adaptive PCA, which is based on recursive updates of the means and variances. The update formulas are derived for the case of sample-wise adaptation as well as for the case of block-wise adaptation. After the updates the eigendecomposition of the correlation matrix has to be done. An approach to achieve this was discussed in Li et al. (2000) (as well as in Section 4.3.3). The problem of the method from Li et al. is that it requires the storage of the full correlation matrix. Choi et al. propose a more optimal adaptation for the block-wise adaptation that requires the storage of the PCA loadings matrix only. Another contribution of the paper is the proposal of an adaptive forgetting for the mean and variance updates. Maintaining two different forgetting factors has the purpose to increase the flexibility of the algorithm. However, the flexibility can also be an obstacle for the implementation of the method. Finding the forgetting factors requires the selection of four different parameters. Although, the authors discuss a heuristic method for the selection of the parameter values in real-life application the finding of optimal values can be difficult. The authors also discuss a method for dealing with outliers. If a violation of the monitoring limits is noticed, the original input vector is replaced with its robust estimate. This action is performed only if low number of consecutive data points violate the monitoring limits, otherwise a process disturbance is reported. The method is applied to an illustrative case study, which demonstrates the ability of the algorithm to deal with different changes in the data. Furthermore the method is applied to the monitoring of a simulation of a continuous stirred reactor.

(19) Wang et al. (2006): This work proposes the Adaptive Kernel Learning framework (see Section 4.3.6), which applies the statistical learning theory to adaptive prediction and diagnosis in industrial processes. In order to control the complexity of the changing model, the algorithm employs a two step learning strategy, where in the first step old data is removed from the model (downdating) and in the next step the new data point is used to update the model (updating). The details of the algorithm can be found in Section 4.3.6. The algorithm is evaluated using two classification problems, where it is shown to be able to deal with difficult classification problem. The algorithm is then evaluated using the Tennessee Eastman Process (see Section 6) simulation where five commonly studied disturbances including abrupt changes and slow drifts are examined. The authors assign the different faults class labels and tried the fault detection as multi-class classification problem. The algorithm is shown to deliver in most cases the equal or better performance than other benchmark methods.

(20) Liu (2007): The proposed algorithm consists of several parts. The first step is dealing with data collinearity using the PCA. This is followed by clustering the scores of the data by the Gustafson-Kessel algorithm, which splits the data into distinct partitions in the scores space. At the same time these partitions are considered as separate rules of a Takagi-Sugeno model. The local models (i.e. the consequent part of the fuzzy rules) are identified using linear regression. The adaptation is also achieved in several steps including the adaptation of the clusters by re-calculating their centres and covariances as well as recursively updating the local predictive models. The algorithm is successfully applied to the prediction of melt index in a high pressure polyethylene plant, which has very nonlinear behaviour due to the production of different grades of polyethylene.

(21) Zhao et al. (2007): This work proposes an extension of the DISSIM process monitoring tool to batch process applications. The DISSIM index is made applicable to batch process data by using a time window which is growing with the increasing number of data of the monitored batch. By calculating the monitoring statistics for each of the window sizes, the monitored batch can be monitored using statistics derived at equivalent temporal positions of the training batches. The algorithm is applied to simple 2-dimensional time varying system with some superimposed disturbances (variable drifts and abrupt changes). The method is also

applied to the simulated penicillin production process (PenSim, see Section 6) also with some injected gradual and abrupt changes. The algorithm was demonstrated to deliver better results than the DISSIM method. More than that in the case of a disturbance, this can be analysed and the contribution plot of the dissimilarity metric can be used to identify the source of the disturbance which was also shown in the paper.

(22) Sbarbaro, Ascencio, Espinoza, Mujica, and Cortes (2008): This work presents a recursive linear algorithm. The predictions are calculated as sum of convolutions of the input variables with impulse response function. The advantages of this representation is that the input can be manipulated in flexible way (e.g. delayed) to obtain the predictions. In the cited work, also a recursive version of the algorithm is proposed. The recursive formula is similar to the Recursive Least Squares method and is obtained by minimising the squared prediction error and parameter changes which serves as a regularisation condition. The method is applied to a grinding plant particle size prediction. In the case study the impulse responses for the different input variables are obtained using process knowledge. The method is shown to outperform the traditional RLS algorithm.

(23) Liu et al. (2008): Here, the authors use the AKL framework proposed in Wang et al. (2006) and discussed in Section 4.3.6 for online prediction in industrial fermentation processes. In contrast to the space angle index used by Wang et al. (2006), the authors propose a different approach for finding of the support vectors. The approach is based on the thresholding the prediction error related to the current sample. If the error is found to be above the threshold, the current data point is added to the model as a support vector. In order to limit the model size, the authors used a pruning mechanism that removes samples with lowest Lagrange multipliers from the model. In a case study the authors apply the method to a simulated penicillin production process (see Section 6).

(24) Fu, Su, Zhang, and Chu (2008): The algorithm proposed in this work is an ensemble of SVR models (see Section 4.3.6). The training data is initially clustered by a Fuzzy C-Means algorithm, which is followed by building a separate regression model for each of the separate data partitions. To obtain the prediction for the online data, the model with the highest fuzzy membership is selected for delivering the prediction. Additional, if a target value is available during the online phase, the selected model is incrementally updated. In order to control the model complexity, the sample displacement method in the feature space is used. In this method the data point with the largest distance to the centre of the cluster (in the feature space) is decrementally removed from the model. The algorithm is applied to a simulated moving bed absorption separation process for p-xylene purity prediction. The model is compared to a plain SVR model and shown to perform better for this particular problem.

(25) AlGhazzawi and Lennox (2008): In this paper PCA is used for monitoring condensate fractionation process. The authors show that static PCA is not suitable for this application. To accommodate time-varying and non-stationary process behaviour, PCA model is re-calculated at each sampling instant using a moving window of process data (refer to Section 2.3.1). The adaptive PCA provides more accurate identification of abnormal conditions. Since the process is quite complex, it was divided in two logical sections, where Multi-Block PCA (MBPCA) models and their adaptive (moving window) version were developed. According to the process operators, MBPCA approach can help to quickly identify and isolate a section of the plant were abnormality occurred.

(26) Ge and Song (2008): In this paper, a new local model approach for monitoring multiple mode and time-varying processes is proposed. Local model building is based upon Just-In-Time Learning (JITL) strategy (Cheng & Chiu, 2004) and the Least Squares Support Vector Regression (LS-SVR) from Suykens (2002), to achieve online process modelling. In JITL strategy, when new sample is observed relevant data samples that match the current data sample are searched in the process database by using nearest neighbourhood criteria. Then, on the relevant dataset local model is built using LS-SVR. A two-step ICA-PCA information extraction strategy is used to analyse residuals between the real output and the local model prediction for the current sample. This adaptive approach is compared with linear JITL, recursive PCA and kernel PCA on numerical example from Dayal and MacGregor (1997b) and Tennessee Eastman benchmark process (see Section 6 on more information about the process).

(27) He and Yang (2008): In this work He and Yang proposed a further optimisation of the MWPCA from Wang et al. (2005). The method is also based on two-step updates of the correlation matrix of the data including the downdating and updating of the matrix (see Section 2.3.1). The authors show the application of the update formulas for the block-wise as well as the sample-wise operation. The contribution of their work is the application of the rank-r SVD instead of the rank-one modification or Lanczos tridiagonalisation for the eigendecomposition of the correlation matrix. This reduces the storage as well as the computational requirements. Furthermore, the authors propose an approach for the calculation of optimal moving window size. In their method the window size is varied between a minimal and maximal size dependent on the changes of the means and correlation of the data. Large changes lead to smaller window size and vice versa. The proposed technique is applied to two case studies: (i) an illustrative model with randomly generated transformation matrix; and (ii) a simulated Continuous Stirred Tank Reactor (CSTR). The first example is used for the demonstration how to select the parameters for the variable moving window method. Furthermore three disturbances (two slow drifts and a ramp) are added to the signals and studied. The process monitoring soft sensor is shown to be able to compensate for the slow drift and at the same time to recognise the process fault (ramp disturbance). In the second case study, the soft sensor is demonstrated to be able to deal with set-point changes in the process by the means of its adaptation and the changes of the moving window size.

(28) Ahmed, Nazir, and Yeo (2009): This proposes an adaptation algorithm which is similar to Mu et al. (2006). It also combines recursive PLS adaptation discussed in Section 4.3.4 with prediction offset compensation. Unlike in Mu et al. (2006), where the switching between RPLS and offset compensation is done periodically at fixed times, in this work the decision which of the two adaptation methods to use is based on the prediction error. If this is bellow a threshold then an RPLS is carried out, in the other case the prediction offset is updated. In addition to this scheme, the authors present another adaptation method, where the adaptation is skipped when the prediction error is bellow a lower threshold. The algorithm is applied to a continuous polymerisation (HDPE) production process for the prediction of melt index.

(29) Ma et al. (2009): In contrast to the common approach using the PCA/PLS for dealing with collinearity and data space reduction, Ma et al. use Stepwise Linear Regression (SLR) to determine the input variables for the model building. The authors argue that this method is more transparent and the resulting models are simpler to interpret. The used method uses partial F-test to estimate the significance of adding or removing of a process variable from the model. A drawback of the method is that it has to deal with the data collinearity separately. This is done using the condition number of the data which is the ratio between the maximal and minimal eigenvalue of its covariance matrix. For the adaptation of the model parameters of the linear regression model, the square root filter is used. This is conceptually similar to a Kalman filter but shows increased numerical stability in the presence of collinearity in the data. The technique was applied to a simulated isopropylbenzene impurity estimation. The developed soft sensor was successfully built in the inferential control loop of the process. The algorithm was also applied to a real-world industrial case study, namely an o-xylene distillation column.

(30) Kadlec and Gabrys (2009): This work presents a complex architecture for the development of adaptive soft sensors. The architecture supports different adaptation mechanisms, which can be implemented in soft sensors. These mechanisms range from low level adaptations of the models through ensemble level adaptation to high level adaptation methods. The work also shows an implementation of the architecture. The implementation not only supports the adaptive behaviour of the models but also the adaptation of the model structure according to the underlying data sets. The method was applied to three different data sets, an industrial drier process (residual humidity prediction), thermal oxidiser process (NOx emissions level) and the polymerisation reactor (see Section 6) and was able to deliver well performing models for all three data sets without any specific manual parameter settings for the particular data sets.

(31) Liu et al. (2009a): This work proposes a kernel-based version of the moving window PCA discussed in Section 4.2.3. The adaptation mechanism works in similar way as the Fast Moving Window PCA (Wang et al., 2005) with the significant difference that the updates of the mean and variance are done in the high dimensional kernel space. The adaptation is done in two steps. First removing the oldest samples (downdating) followed by taking the new sample into account (updating). Liu et al. also provide a numerically efficient algorithm for the eigen-decomposition of the covariance matrix. Furthermore, the work presents how to adapt the monitoring statistics T^2 and SPE as well as a method for the adaptive selection of an optimal number of principle components. For the process monitoring it employs the same N-step-ahead prediction scheme as Wang et al. (2005). There are two case studies demonstrating the performance of the algorithm. First one, a time varying non-linear system with three input variables. It is shown that the adaptive algorithm is able to change the monitoring statistics and significantly reduces the number of false alarms as well as the ability of the algorithm to detect slow drift-like disturbance. In a second case study an industrial process for butane purification was used. In this case too, the adaptive monitoring algorithm was able to detect faults in the process.

(32) Liu et al. (2009b): In this work, the authors propose a LS-SVR based algorithm for recursive online prediction with two stage recursive learning similar to Wang et al. (2006). The difference to the other recursive AKL algorithms is that Liu at al. propose a method for a variable moving window size that should reflect the changing dynamics of the data in a flexible way. The authors propose an online version of the Leave-One-Out Cross Validation (LOOCV) that enables finding the node (support vector) whose removal causes the lowest increase in the estimated generalisation error. The method for variable moving window size is based on the comparison of the general Root Mean Squared Error (RMSE) and the relative error. The degradation of model performance is identified when the relative error exceeds the general RMSE. In such a case, the pruning mechanism is triggered to remove a node from the model. The proposed method was applied to industrial FCCU process for the online prediction of gasoline, LPG and LDO yields.

(33) Yang, Liu, Fan, and Wang (2009): Here, the AKL algorithm (see Section 4.3.6) is used for the prediction of the viscosity in a rubber mixing process. The authors additionally point out that the AKL algorithm has only three input parameters and is easy to optimise due to the robustness of the model performance in respect of the values of the input parameters. The AKL-based soft sensor is shown to outperform a Recursive PLS-based soft sensor.

(34) Haavisto and Hytyniemi (2009): Effective monitoring and control of floatating process relies on online measurement of the mineral flotation flurry contest. Since X-ray fluorescence (XRF) analysers have very low sampling frequency, efficient process control cannot be performed since sudden content changes caused by process failures cannot be detected efficiently. Therefore, the authors developed a soft sensor that predicts mineral flotation slurry contents based on visual and near-infrared reflectance spectrum measurements that can be obtained with high sampling frequency. The soft sensor is based on a new PLS-based Recursive Multi-Model approach (RMM) with local Orthogonal Signal Correction (OSC). RMM consists of N local PLS models which are recursively calibrated as the new data from the XRF analyser are obtained. Adaptation of each local model is based on the recursive updating of the old data covariance matrix estimates with exponential forgetting factor. The kernel algorithm proposed by Dayal and MacGregor (1997b) is used for PLS model calculation. To maintain the localization, Gaussian weighing functions are used in the calibration and prediction. To improve prediction, each local model performs OSC data pre-processing. The final RMM prediction is calculated as the weighted average of the local response estimates. This technique is compared with recursive and non-adaptive PLS. Due to variations in the slurry properties a classical PLS model is not accurate during longer period. The RPLS has a considerably better prediction than the classical one. RMM have similar prediction capabilities as the RPLS but is found to be more efficient in predicting the sudden large content changes caused by rare process failures.

(35) Facco et al. (2010): The authors of this work propose a method for adaptive selection of reference batches for process monitoring purposes. The similarity is defined as the distance in the space of principal components. The reference set is collected by using the *k*-Nearest Neighbour technique in the PC space. The method has the advantages that it relaxes the requirement to use reference batches, which chronologically close to the current batch (i.e. a set of latest batches) and instead of this uses the most similar past batches. The problem of the method is that it requires the collection of minimal set of samples for the current batch in order to be able to built a model. The method also requires to store all past batches because they can potentially be used as reference in the future. The method is applied to the popular simulated penicillin production batch process and to a polymerisation process. In both cases it has led to performance improvement when compared to the moving window adaptation technique.

(36) Kadlec and Gabrys (2010): In this work an adaptive soft sensor for the prediction of catalyst activity is developed.

The adaptation algorithm is an ensemble-based method (see Section 4.4).The model works on the same principle as the one described in (Kadlec & Gabrys, 2010b). The difference to the previous work is that the model is able to start delivering predictions without any explicit training phase and can start making prediction from the second available data sample. The algorithm start with a single simple PLS local model, which is extended as the new data is coming in. Additionally, if the prediction ability of the local model(s) drops bellow a certain level, a new local model is started. The algorithm was shown to be able to deal with a highly time-varying data set.

(37) Kadlec and Gabrys (in press): The proposed model is based on simple locally valid models (PLS models), which are equipped with ensemble-based recursive adaptation capability (see Section 4.4). The final prediction of the set of available models is obtained as weighted combination, where the weights reflect the estimated performance of local models for the current data sample. In order to store the weights, performance maps of the local models are built and maintained. The adaptation is done at two levels: (i) at the level of the combination weights, which is done with every available performance feedback information; and (ii) through the recursive adaptation of the local models. In this case a winner-takes-all scenario is used as the model with the highest combination weight, i.e. the most responsible one for the current prediction, is adapted much stronger than the rest of the models. The soft sensor was applied to a polymerisation reactor for the prediction of the catalyst activity inside the reactor.

References

Ahmed, F., Nazir, S., & Yeo, Y. (2009). A new soft sensor based on recursive partial least squares for online melt index predictions in grade-changing HDPE operations. *Chemical Product and Process Modeling*, *4*(1), 33.

AlGhazzawi, A., & Lennox, B. (2008). Monitoring a complex refining process using multivariate statistics. *Control Engineering Practice*, *16*(3), 294–307.

Angelov, P., Zhou, X. (2006). Evolving fuzzy systems from data streams in real-time. In *Proceedings of the international symposium on evolving fuzzy systems*. (pp. 29–35).

Bakshi, B. R. (1998). Multiscale PCA with application to multivariate statistical process monitoring. *AIChE Journal*, *44*, 1596–1610.

Birol, G., Undey, C., & Cinar, A. (2002). A modular simulation package for fed-batch fermentation: Penicillin production. *Computers and Chemical Engineering*, *26*(11), 1553–1565.

Bouchachia, A. (2009). Adaptation in classification systems. In A. Hassanien, A. Abraham, & F. Herrera (Eds.), *Foundations in computational intelligence* (pp. 237–258). Springer.

Carpenter, G. A., & Grossberg, S. (1998). Adaptive resonance theory (art). In M. A. Arbib (Ed.), *The handbook of brain theory and neural networks table of contents* (pp. 79–82). Cambridge: MIT Press.

Casali, A., Gonzalez, G., Torres, F., Vallebuona, G., Castelli, L., & Gimenez, P. (1998). Particle size distribution soft-sensor for a grinding circuit. *Powder Technology*, *99*(1), 15–21.

Cauwenberghs, G., & Poggio, T. (2001). Incremental and decremental support vector machine learning. In *Advances in neural information processing systems*. The MIT Press., pp. 409–415

Cheng, C., & Chiu, M.-S. (2004). A new data-based methodology for nonlinear process modeling. *Chemical Engineering Science*, *59*(13), 2801–2810.

Chiang, L., Pell, R., & Seasholtz, M. (2003). Exploring process data with the use of robust outlier detection algorithms. *Journal of Process Control*, *13*(5), 437–449.

Chiang, L. H., Russell, E., & Braatz, R. D. (2001). *Fault detection and diagnosis in industrial systems*. Springer.

Choi, S. W., Martin, E. B., Morris, A. J., & Lee, I. B. (2006). Adaptive multivariate statistical process control for monitoring time-varying processes. *Industrial and Engineering Chemistry Research*, *45*, 3108–3118.

Dayal, B., & MacGregor, J. (1997). Improved PLS algorithms. *Journal of Chemometrics*, *11*(1), 73–85.

Dayal, B. S., & MacGregor, J. F. (1997). Recursive exponentially weighted PLS and its applications to adaptive control and prediction. *Journal of Process Control*, *7*(3), 169–179.

Dong, D., & McAvoy, T. J. (1996). Nonlinear principal component analysis—Based on principal curves and neural networks. *Computers and Chemical Engineering*, *20*(1), 65–78.

Donoho, D. L. (1995). De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, *41*(3), 613–627.

Downs, J., & Vogel, E. (1993). A plant-wide industrial process control problem. *Computers and Chemical Engineering*, *17*(3), 245–255.

Drucker, H., Burges, C. J. C., Kaufman, L., Smola, A., & Vapnik, V. (1997). Support vector regression machines. In *Advances in neural information processing systems*. Morgan Kaufmann Publishers., pp. 155–161.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Patten classification*. New York: John Wiley and Sons.

Facco, P., Bezzo, F., & Barolo, M. (2010). Nearest-neighbor method for the automatic maintenance of multivariate statistical soft sensors in batch processing. *Industrial and Engineering Chemistry Research*, *49*(5), 2336–2347.

Facco, P., Doplicher, F., Bezzo, F., & Barolo, M. (2009). Moving average PLS soft sensor for online product quality estimation in an industrial batch polymerization process. *Journal of Process Control*, *19*(3), 520–529.

Fortescue, T., Kershenbaum, L., & Ydstie, B. (1981). Implementation of self-tuning regulators with variable forgetting factors. *Automatica*, *17*(6), 831–835.

Fortuna, L. (2007). *Soft sensors for monitoring and control of industrial processes*. London: Springer Verlag.

Fortuna, L., Graziani, S., & Xibilia, M. (2005). Virtual instruments in refineries. *IEEE Instrumentation & Measurement Magazine*, *8*(4), 26–34.

Fortuna, L., Graziani, S., & Xibilia, M. G. (2005). Soft sensors for product quality monitoring in debutanizer distillation columns. *Control Engineering Practice*, *13*(4), 499–508.

French, R. (1999). Catastrophic forgetting in connectionist networks: Causes, consequences and solutions. *Trends in Cognitive Sciences*, *3*(4), 128–135.

Fu, Y., Su, H., Zhang, Y., & Chu, J. (2008). Adaptive soft-sensor modeling algorithm based on FCMISVM and its application in PX adsorption separation process. *Chinese Journal of Chemical Engineering*, *16*(5), 746–751.

Gallagher, N., Wise, B., Butler, S., White, D., & Barna, G. (1997). Development and benchmarking of multivariate statistical process control tools for a semiconductor etch process: Improving robustness through model updating. In *IFAC ADCHEM'97* Citeseer, (pp. 78–83).

Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Proceedings of the 17th Brazilian symposium on artificial intelligence advances in artificial intelligence (SBIA 2004). Vol. 3171* (pp. 286–295).

Ge, Z., & Song, Z. (2008). Online monitoring of nonlinear multiple mode processes based on adaptive local model approach. *Control Engineering Practice*, *16*(12), 1427–1437.

Geladi, P., & Kowalski, B. (1986). Partial least-squares regression: A tutorial. *Analytica Chimica Acta*, *185*, 1–17.

Gonzalez, G. D. (1999). Soft sensors for processing plants. In *Proceedings of the second international conference on intelligent processing and manufacturing of materials (IPMM'99)*.

Goodwin, G., & Sin, K. (2009). *Adaptive filtering prediction and control*. New York, NY, USA: Dover Publications, Inc.

Haavisto, O., & Hytyniemi, H. (2009). Recursive multimodel partial least squares estimation of mineral flotation slurry contents using optical reflectance spectra. *Analytica Chimica Acta*, *642*(1–2), 102–109.

He, X., & Yang, Y. (2008). Variable MWPCA for adaptive process monitoring. *Industrial and Engineering Chemistry Research*, *47*(2), 419–427.

Helland, H. E. (1992). Recursive algorithm for partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, *14*(1–3), 129–137.

Jang, J. S. R., Sun, C. T., & Mizutani, E. (1997). *Neuro-fuzzy and soft computing*. Upper Saddle River, NJ: Prentice Hall.

Jin, H., Lee, Y., Lee, G., & Han, C. (2006). Robust recursive principal component analysis modeling for adaptive monitoring. *Industrial and Engineering Chemistry Research*, *45*(2), 696–703.

Jolliffe, I. T. (2002). *Principal component analysis*. New York: Springer.

Kadlec, P. (2009). On robust and adaptive soft sensors. PhD. *AICHE Journal*, doi:10.1002/aic.12346.

Kadlec, P., & Gabrys, B. (2009). Soft sensors: Where are we and what are the future challenges? In *Proceedings of the IFAC international conference on intelligent control systems and signal processing (ICONS)*.

Kadlec, P., & Gabrys, B. (2010). Adaptive on-line prediction soft sensing without historical data. In *IEEE international joint conference on neural networks*. Barcelona: IEEE.

Kadlec, P., Gabrys, B. (in press). Local learning-based adaptive soft sensor for catalyst activation prediction. *AICHE Journal*.

Kadlec, P., Gabrys, B., & Strandt, S. (2009). Data-driven soft sensor in the process industry. *Computers and Chemical Engineering*, *33*(4), 795–814.

Kampjarvi, P., Sourander, M., Komulainen, T., Vatanski, N., Nikus, M., & Jämsä-Jounela, S. L. (2008). Fault detection and isolation of an on-line analyzer for an ethylene cracking process. *Control Engineering Practice*, *16*(1), 1–13.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis*, *8*(3), 281–300.

Kourti, T. (2002). Process analysis and abnormal situation detection: From theory to practice. *Control Systems Magazine, IEEE*, *22*(5), 10–25.

Krogh, A., & Vedelsby, J. (1995). Neural network ensembles, cross validation and active learning. *Advances in Neural Information Processing Systems*, (7), 231–238.

Lane, S., Martin, E., Morris, A., & Gower, P. (2003). Application of exponentially weighted principal component analysis for the monitoring of a polymer film manufacturing process. *Transactions of the Institute of Measurement and Control*, *25*(1), 17.

Lee, D., & Vanrolleghem, P. (2004). Adaptive consensus principal component analysis for on-line batch process monitoring. *Environmental Monitoring and Assessment*, *92*(1), 119–135.

Lee, D. S., Park, J. M., & Vanrolleghem, P. A. (2005). Adaptive multiscale principal component analysis for on-line monitoring of a sequencing batch reactor. *Journal of Biotechnology*, *116*(2), 195–210.

Lee, J., Yoo, C., & Lee, I. (2004). Fault detection of batch processes using multiway kernel principal component analysis. *Computers and Chemical Engineering*, *28*(9), 1837–1847.

Lee, M., Joung, J., Lee, D., Park, J., & Woo, S. (2005). Application of a moving-window-adaptive neural network to the modeling of a full-scale anaerobic filter process. *Industrial and Engineering Chemistry Research*, *44*(11), 3973–3982.

Lennox, J., & Rosen, C. (2002). Adaptive multiscale principal components analysis for online monitoring of wastewater treatment. *Water Science and Technology*, *45*(4), 227–235.

Li, C., Ye, H., Wang, G., & Zhang, J. (2005). A recursive nonlinear PLS algorithm for adaptive nonlinear process modeling. *Chemical Engineering and Technology*, *28*(2), 141–152.

Li, W., Yue, H. H., Valle-Cervantes, S., & Qin, S. J. (2000). Recursive PCA for adaptive process monitoring. *Journal of Process Control*, *10*(5), 471–486.

Lin, B., Recke, B., Knudsen, J., & Jorgensen, S. B. (2007). A systematic approach for soft sensor development. *Computers and Chemical Engineering*, *31*(5), 419–425.

Lindgren, F., Geladi, P., & Wold, S. (2005). The kernel algorithm for PLS. *Journal of Chemometrics*, *7*(1), 45–59.

Liu, J. (2007). On-line soft sensor for polyethylene process with multiple production grades. *Control Engineering Practice*, *15*(7), 769–778.

Liu, X., Kruger, U., Littler, T., Xie, L., & Wang, S. (2009). Moving window kernel PCA for adaptive monitoring of nonlinear processes. *Chemometrics and Intelligent Laboratory Systems*, *96*(2), 132–143.

Liu, Y., Hu, N., Wang, H., & Li, P. (2009). Soft chemical analyzer development using adaptive least-squares support vector regression with selective pruning and variable moving window size. *Industrial and Engineering Chemistry Research*, *48*(12), 5731–5741.

Liu, Y., Yang, D., Wang, H. & Li, P. (2008). Modeling of fermentation processes using online kernel learning algorithm. In *Proceedings of the 17th IFAC World Congress. Vol. 17* (pp. 9679–9684).

Ljung, L. (1987). *System identification: Theory for the user*. Englewood Cliffs, NJ: Prentice-Hall.

Ma, M., Ko, J., Wang, S., Wu, M., Jang, S., Shieh, S., & Wong, D. (2009). Development of adaptive soft sensor based on statistical identification of key variables. *Control Engineering Practice*, *17*(9), 1026–1034.

Macias, J. J., Angelov, P., & Zhou, P. X. (2006). A method for predicting quality of the crude oil distillation. In *Proceedings of the international symposium on evolving fuzzy systems* (pp. 214–220).

Maloof, M. A., & Michalski, R. S. (2000). Selecting examples for partial memory learning. *Machine Learning*, *41*(1), 27–52.

Mandic, D., & Chambers, J. (2001). *Recurrent neural networks for prediction: Learning algorithms, architectures and stability*. Wiley.

Martens, H., & Naes, T. (1991). *Multivariate calibration*. Wiley.

Mu, S., Zeng, Y., Liu, R., Wu, P., Su, H., & Chu, J. (2006). Online dual updating with recursive PLS model and its application in predicting crystal size of purified terephthalic acid (PTA) process. *Journal of Process Control*, *16*(6), 557–566.

Nomikos, P., & MacGregor, J. F. (1995). Multivariate SPC charts for monitoring batch processes. *Technometrics*, *37*(1), 41–59.

Nounou, M. N., & Bakshi, B. R. (1999). On-line multiscale filtering of random and gross errors without process models. *AIChE Journal*, *45*(5), 1041–1058.

Pearson, R. K. (2001). Exploring process data. *Journal of Process Control*, *11*(2), 179–194.

Pearson, R. K. (2002). Outliers in process modeling and identification. *IEEE Transactions on Control Systems Technology*, *10*(1), 55–63.

Qin, S. (2003). Statistical process monitoring: Basics and beyond. *Journal of Chemometrics*, *17*(8/9), 480–502.

Qin, S. J. (1998). Recursive PLS algorithms for adaptive data modeling. *Computers and Chemical Engineering*, *22*(4–5), 503–514.

Raennar, S., MacGregor, J. F., & Wold, S. (1998). Adaptive batch monitoring using hierarchical PCA. *Chemometrics and Intelligent Laboratory Systems*, *41*(1), 73–81.

Rifkin, R., Yeo, G., & Poggio, T. (2003). Regularized least-squares classification. *NATO Science Series*, *190*, 131–154.

Ruta, D. (2006). Crosstrained ensemble of neural networks for robust time series prediction. In *Proceedings of the NiSIS Symposium 2006*, available online: http://www.nisis.risk-technologies.com/Events/Symp2006/Papers/AB23_D.Ruta.pdf.

Sbarbaro, D., Ascencio, P., Espinoza, P., Mujica, F., & Cortes, G. (2008). Adaptive soft-sensors for on-line particle size estimation in wet grinding circuits. *Control Engineering Practice*, *16*(2), 171–178.

Schaal, S., & Atkeson, C. G. (1998). Constructive incremental learning from only local information. *Neural Computation*, *10*(8), 2047–2084.

Scholkopf, B., Smola, A., & Muller, K. R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*(5), 1299–1314.

Scholz, M., & Klinkenberg, R. (2007). Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, *11*(1), 3–28.

Seng Ng, Y., & Srinivasan, R. (2010). Multi-agent based collaborative fault detection and identification in chemical processes. *Engineering Applications of Artificial Intelligence*, *23*(6), 934–949.

Smola, A. J., & Scholkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222.

Su, H. B., Fan, L. T., & Schlup, J. R. (1998). Monitoring the process of curing of epoxy/graphite fiber composites with a recurrent neural network as a soft sensor. *Engineering Applications of Artificial Intelligence*, *11*(2), 293–306.

Suykens, J. A. K. (2002). *Least squares support vector machines*. World Scientific.

Tsymbal, A. (2004). The problem of concept drift: definitions and related work. Tech. Re TCD-CS-2004-15, Departament of Computer Science Trinity College, Dublin, https://www.cs.tcd.ie/publications/techreports/reports.

Vapnik, V. N. (1998). *Statistical learning theory*. New York: Wiley.

Venkatasubramanian, V., Rengaswamy, R., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis. Part II. Qualitative models and search strategies. *Computers and Chemical Engineering*, *27*(3), 313–326.

Venkatasubramanian, V., Rengaswamy, R., Kavuri, S. N., & Yin, K. (2003). A review of process fault detection and diagnosis. Part III. Process history based methods. *Computers and Chemical Engineering*, *27*(3), 327–346.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., & Kavuri, S. N. (2003). A review of process fault detection and diagnosis. Part I. Quantitative model-based methods. *Computers and Chemical Engineering*, *27*(3), 293–311.

Wang, H., Li, P., Gao, F., Song, Z., & Ding, S. (2006). Kernel classifier with adaptive structure and fixed memory for process diagnosis. *AIChE Journal*, *52*(10), 3515–3531.

Wang, X., Kruger, U., & Irwin, G. W. (2005). Process monitoring approach using fast moving window PCA. *Industrial and Engineering Chemistry Research*, *44*(15), 5691–5702.

Wang, X., Kruger, U., & Lennox, B. (2003). Recursive partial least squares algorithms for monitoring complex industrial processes. *Control Engineering Practice*, *11*(6), 613–632.

Warne, K., Prasad, G., Rezvani, S., & Maguire, L. (2004). Statistical and computational intelligence techniques for inferential model development: A comparative evaluation and a novel proposition for fusion. *Engineering Applications of Artificial Intelligence*, *17*(8), 871–885.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning*, *23*(1), 69–101.

Wold, H. (1966). Nonlinear estimation by iterative least squares procedures. In F. David (Ed.), *Research papers in statistics: Festschrift for J. Neyman* (pp. 411–444). Wiley.

Wold, S. (1993). Exponentially weighted moving principal components analysis and projections to latent structures. *Chemometrics and Intelligent Laboratory Systems*, *23*(1), 149–161.

Wold, S., Geladi, P., Esbensen, K., & Ohman, J. (1987). Multi-way principal components and PLS analysis. *Journal of Chemometrics*, *1*(1), 41–56.

Yan, W., Shao, H., & Wang, X. (2004). Soft sensing modeling based on support vector machine and bayesian model selection. *Computers and Chemical Engineering*, *28*(8), 1489–1498.

Yang, D., Liu, Y., Fan, Y., Wang, H. (2009). Online prediction of Mooney viscosity in industrial rubber mixing process via adaptive kernel learning method.In *Proceedings of the 48th IEEE conference on decision and control*. (pp. 404–409).

Yang, Y., & Chai, T. (1997). Soft sensing based on artificial neural network. In *Proceedings of the 1997 American control conference. Vol. 1*

Zhao, C., Wang, F., & Jia, M. (2007). Dissimilarity analysis based batch process monitoring using moving windows. *AIChE Journal*, *53*(5), 1267–1277.

Zhao, L., & Chai, T. (2004). Adaptive moving window MPCA for online batch monitoring. In *Proceedings of the fifth Asian control conference. Vol. 2*.