Evolving Systems (2014) 5:239–257
DOI 10.1007/s12530-014-9108-y

ORIGINAL PAPER

Self-adaptive and local strategies for a smooth treatment of drifts in data streams

Ammar Shaker · Edwin Lughofer

Received: 4 October 2013 / Accepted: 10 April 2014 / Published online: 1 May 2014

Abstract In this paper, we are dealing with a new concept for handling drifts in data streams during the run of on-line, evolving modeling processes in a regression context. Drifts require a specific attention in evolving modeling methods, as they usually change the underlying data distribution making previously learnt model parameters and structure outdated. Our approach comes with three new stages for an appropriate drift handling: (1) drifts are not only detected, but also quantified with a new extended version of the Page-Hinkley test; (2) we integrate an adaptive forgetting factor changing over time and which steers the degree of forgetting in dependency of the current drift intensity in the data stream; (3) we introduce local forgetting factors by addressing the different local regions of the feature space with a different forgetting intensity; this is achieved by using fuzzy model architecture within stream learning whose structural components (fuzzy rules) provide a local partitioning of the feature space and furthermore ensure smooth transitions of drift handling topology between neighboring regions. Additionally, our approach foresees an early drift recognition variant, which relies on divergence measures, indicating the degree of divergence in local parts of the feature space separately already before the global model error may start to rise significantly. Thus, it can be seen as an attempt regarding drift prevention on global model level. The new approach is successfully evaluated and compared with fixed forgetting and no forgetting on high-dimensional real-world data streams, including different types of drifts.

Keywords Data stream learning · Adaptive local and global drift handling · Drift and forgetting intensity · Component divergence · Early drift recognition · Evolving granular models

A. Shaker
Department of Mathematics and Computer Science, Philipps-University Marburg, Marburg, Germany
e-mail: shaker@Mathematik.Uni-Marburg.de

E. Lughofer (✉)
Department of Knowledge-based Mathematical Systems, Johannes Kepler University of Linz, Linz, Austria
e-mail: edwin.lughofer@jku.at

1 Introduction

1.1 Motivation

In today's industrial systems, adaptive data-driven learning methodologies (Gama 2010) play a more and more important role, as they are able to cope with dynamic and continuously evolving environments (Sayed-Mouchaweh and Lughofer 2012). This keeps the quality of the system models permanently up-to-date and on a high level. In particular, the methodologies typically employed are able to adjust the models to new system states and operation modes on-the-fly. Incremental learning concepts and evolution of model components play a key role during model adaptation, thus the models equipped with these technologies are also called evolving (soft computing) models or in a broader sense evolving intelligent systems (EIS) (Angelov et al. 2010). Examples of realizations of such models are different types of incremental classifiers [such as incremental SVMs, (Diehl and Cauwenberghs 2003; Shilton et al. 2005), incremental discriminant analysis (Hisada et al. 2010) or incremental decision and pattern trees (Utgoff 1997; Shaker et al. 2013)], incremental clustering approaches (see Bouchachia 2011, for a compact survey), growing Gaussian mixture models (Bouchachia

and Vanaret 2011; Song and Wang 2005), evolving connectionist systems (a kind of adaptive neural networks) (Kasabov 2007) or evolving fuzzy systems (Lughofer 2011). In some cases, it is sufficient to refine some parameters in the model architecture, e.g. to update the neuron weights or covariance matrices. This is usually done within incremental optimization steps (see Lughofer 2011, Chapter 2) in order to achieve convergence requirements and thus stability of the models. In other cases, additionally the evolution of structural components (e.g. the dynamic growing leaves in a tree or the birth of new fuzzy rules) is often requested to expand the models into a feature space so far unexplored. This assures the coverage of new system states and operation modes, preventing extrapolation cases and thus a downtrend in model accuracy. The interactions with the user during on-line process streams should be kept at a low level to minimize effort and costs. Hence, the incremental training and evolution steps are often conducted fully unsupervised or, in the field of classification, at least semi-supervised. The latter is achieved with the help of single-pass active learning methodologies (Lughofer 2012), requiring feedback on true class labels on exquisitely selected samples.

Drifts refer to a gradual evolution of the underlying data distribution and therefore to a change in the input or target concept over time (Widmer and Kubat 1996; Tsymbal 2004). Especially in dynamical systems, drifts are usually occurring regularly due to the permanently changing nature of system states. This shifted the problem of dealing with drifts to become the main concern of the learning process in such systems and in non-stationary environments in general (Sayed-Mouchaweh and Lughofer 2012). Drifts could happen due to many reasons, such as a complete change in the system behavior, a change in the environment conditions or due to the dynamic change (a shift) in the target concept. (e.g. consider the behavior of different customers in an on-line shop, or the weather change due to drastic global changes).

Such situations (drifts) are different from cases when new operation modes or system states arise. The latter should be usually included into the models with the same weight as previous samples gathered from other modes/states. This is important to extend the models, but to keep the relations in states seen before untouched (so, they remain still valid for future predictions). Drifts, however, usually mean that the older learned relations (as parts of a model) are not valid any longer and thus should be incrementally out-weighted, ideally in a smooth manner to avoid catastrophic forgetting (Moe-Helgesen and Stranden 2005; French 1999). In most cases (gradual drifts), it requires a more intense update of some model components (thus with increased plasticity) (Klinkenberg 2004) than in usual (non-drift) situations, where the sequential life-long learning concept is pursued to assure convergence and stability (Hamker 2001). In some cases (abrupt drifts also termed as shifts), a new model component will be induced by the evolution criterion, while older ones may become obsolete. It is then a matter of the incremental, evolving learning engine to prune the component upon ageing (Angelov 2010) or (statistical) influencing concepts (Pratama et al. 2014) in order to keep the model as compact as possible. Shifts, however, usually affect the model performance in terms of error and accuracy less than gradual drifts (older learned components are often not used any more, while in drifts they are used but misplaced, not reflecting the current model tendency). In this paper, we will deal with gradual drifts (which may appear with different intensities, indeed) and provide a generic framework for handling them in adaptive and local manner.

1.2 State-of-the-art

Different possibilities how to handle drifts were studied in the literature in the past see for instance (Delany et al. 2005; Gama et al. 2004; Ramamurthy and Bhatnagar 2007), which (1) are mostly designed for classification problems, (2) are using a forgetting mechanism which applies globally to the whole model definition space and (3) do not account for a fuzzy drift intensity level, but rather operate an a crisp detection signal and perform a reaction once the signal becomes untypical. The latter means that either a drift is detected ($\rightarrow$ detection signal receives a value of 1) or not detected ($\rightarrow$ detection signal receives a value of 0). For instance, in the recently published approach (Lindstrom et al. 2013), a global trigger is used based on the classifier score for estimating the confidence of a prediction (kind of uncertainty level), which then initiates a complete re-building mechanism of the classifier. Furthermore, little attention have been paid to the local nature of drifts and to regression problems as is the focus in this paper. An exception of the latter point is the adaptive forgetting strategy implicitly contained in IBLStreams (Shaker and Hüllermeier 2012), which is an incremental instance-based learner for classification and regression problems. In IBLStreams a percentage (pdiff) of the instance-base is removed (forgotten), where pdiff is the difference between the minimal classification error achieved and the classification error for the last 20 instances. On the other hand, instance-based learners do not provide a model with interpretable facets and always have to set up local models from scratch.

An approach which uses a global model and integrates local drift handling is proposed in (Ikonomovska et al. 2009). It relies on a regression tree, which partitions the feature space into local pieces in a crisp manner. Thus, samples recorded during drift phases affect a certain branch

of the tree, which is then adapted and traversed with increased flexibility. However, the approach lacks of a kind of uncertainty concept in terms of respecting drifts which may affect different branches of the tree with various intensities at the same time (no smooth handling of drifts).

In the context of evolving fuzzy systems (EFS), most of the approaches are integrating life-long learning concepts, i.e. they are using all the data samples equally weighted in the update mechanism and do not take care about their age—see (Lughofer 2011) for a recent survey, further approaches published since 2011 can be found at the 'Evolving Systems' journal at Springer[1]. Exceptions are eTS+ (Angelov 2010) and FLEXFIS++ (Lughofer 2012), which both integrate forgetting mechanisms over time in order to outdate older samples and thus put more emphasis on newer ones based on the concepts of rule age and rule utility, see (Lughofer and Angelov 2011). Other approaches can be found in Soleimani et al. (2010), Kalhor et al. (2010) and Dovzan and Skrjanc (2011) employing forgetting factors, but solely concentrating on the linear consequent parameters (no antecedent parameters are addressed). In these approaches (1) no local drift reaction was considered, i.e. the intensity of the drift was assumed to be the same over the whole input space; and (2) no adaptive forgetting factor was supported, i.e. a fixed value has to be chosen a priori, which may be not an appropriate one for different data streams—except in Lughofer and Angelov (2011): there, the drift intensity (detection) was rudimentarily estimated based on an unsupervised criterion (rule ages), but without the inclusion of the target concept, without supervising the decrease of the model error and without using any statistical hypothesis test or divergence measures.

1.3 Our approach—overview

Drift intensities are monitored by a new enhanced version of the famous Page-Hinkley-test (Mouss et al. 2004) (Sect. 3), which is not only able to detect drifts, but also able to quantify the change of model errors in terms of its magnitude and duration, yielding a continuous drift intensity level. We believe that the model error is an essential trigger when to react on a drift and when not, much more than an unsupervised criterion. The smoothness of our approach is achieved by three factors: (1) the forgetting factor plays a role of data out-weighting over time (following a differentiable smooth monotonic decreasing weight function); (2) forgetting factors are adaptively changed according to the current drift intensity levels (Sect. 4); (3) several local model components may achieve different importance levels (and thus different forgetting factor values), which are also continuously changing over time (Sect. 5); this achieves maximal flexibility in terms of reacting in those parts where the drift really happens more strongly than in other parts.

The Takagi–Sugeno (TS) fuzzy systems architecture (Takagi and Sugeno 1985) allows us to join both concepts, locality and smoothness of forgetting over neighboring regions by providing local components in form of fuzzy rules, which are usually overlapping, to some small extent at least. Thus, each fuzzy rule receives an own forgetting level, depending on the severeness of the drift in the corresponding region it models; neighboring rules then receive a similar forgetting level, allowing a smooth transition of forgetting intensity from one region to the other. Furthermore, when the drift intensity is very small or becomes very small again after a drift and forgetting phase, automatically the conventional life-long learning (with all convergence properties) will be achieved. The integration of the localized adaptive forgetting factors into the incremental learning algorithms of evolving TS fuzzy systems will be demonstrated in Sect. 7.

Finally, we will provide a new concept in the direction of drift prevention (Sect. 6), before a drift really starts to become severe, i.e. before affecting the model accuracy/error significantly; an issue, which, to our best knowledge, has been not handled in literature before. This will be based on internal local indicators employing a statistically motivated divergence measure of rules in the product space. It is expected that drifts can be recognized earlier by zooming into the local scale than by observing the overall global model error from "outside". Most of the concepts, especially adaptive global forgetting (Sects. 3, 4) and drift prevention (Sect. 6) can be directly used in combination with other model architectures integrating local model components (e.g. leaves, neurons).

In Sect. 8, we will test our new approach in high-dimensional streaming data sets and will show that the new adaptive local drift reaction strategy is (1) able to outperform fixed settings of forgetting factors [as used in Lughofer and Angelov (2011)] in case when no drifts appear throughout the stream, and (2) able to outperform adaptive setting of a global forgetting factor (the same value for all local regions) in case when drifts appear. The comparisons will be made in terms of predictive quality (measured by the normalized root mean squared error = RMSE) on separate test folds during the incremental learning procedure, according to the periodic hold-out procedure (Gama et al. 2009) [an established evaluation technique also implemented in MOA framework (Bifet et al. 2010)]. The applied data sets are streaming data from the UCI repository where we include two different types of artificial drifts, in one case even with different intensities in order to demonstrate the achievable bounds of our approach, and a

[1] http://www.springer.com/physics/complexity/journal/12530.

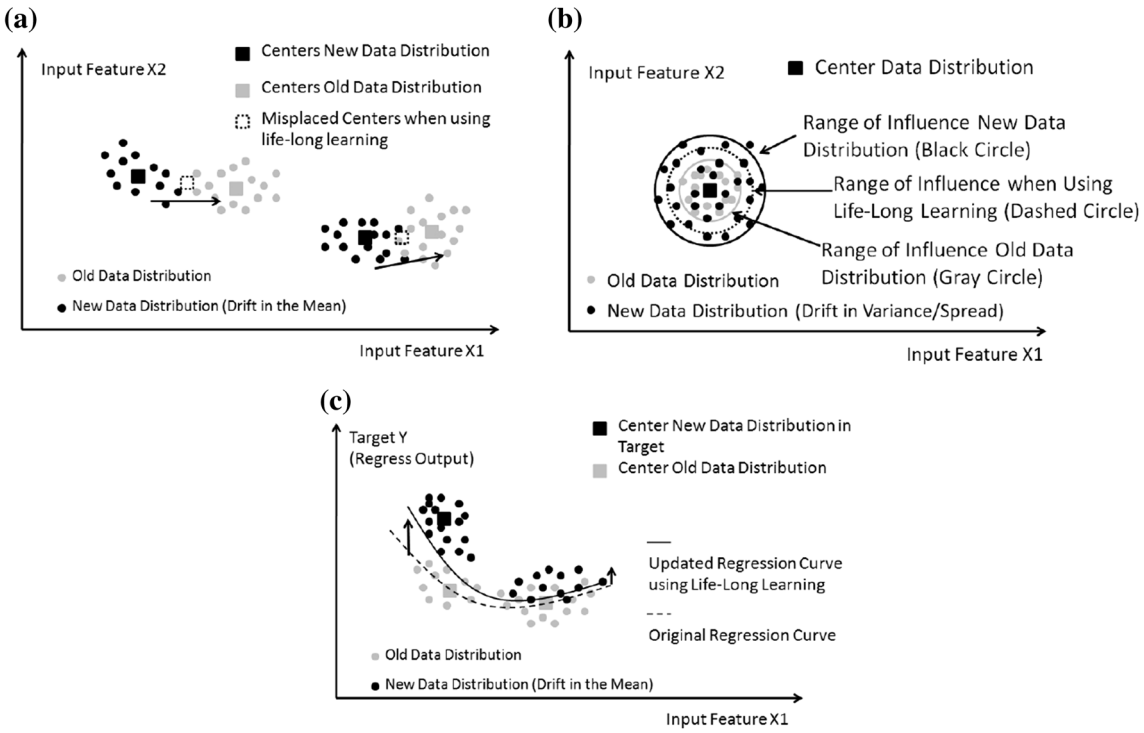

Fig. 1 Three drift concepts: **a** drift in the mean, **b** drift in the variance and **c** drift in parts of the target (correlation)

real-world on-line data set from a condition monitoring system in rolling mills, where a real occurring drift happened and its localization is known.

2 Problem statement

In principle, three different types of drifts may occur (Gama 2010):

- Drifts in the mean of the data distribution.
- Drifts in the variance of the data distribution.
- Drifts in the target concept, also referred as drifts in the correlation between inputs and outputs.

The effect of these three cases on the cluster/rule models as appearing in the inputs of the models are visualized in Fig. 1a–c.

All three types of drifts are causing a downtrend in the accuracy of model predictions when performing conventional adaptation in a life long learning concept. This is because structural components have been converged and are already quite 'stucked' according to the heavy support by many data stream samples used before. Hence, their adaptation do not offer sufficient flexibility for accounting severe changes in the process due to drifts, thus getting biased for the new drifted state. For instance, see Fig. 1c, where the approximation surface (solid line) is ending up in-between the two data clouds representing different drifted concepts of the target—this is simply because the surface tries to minimize the error between real measured values and their predictions (lying on the solid line) on all samples seen so far. In this case, the only reasonable option is to apply a forgetting mechanism to outdate older samples and to enforce the approximation surface to minimize the error only on the samples after the drift happened. Similar considerations can be made in cases as shown in Fig. 1a and b for cluster models: centers should be moved to the midpoints or other reliable prototype representations of data clouds appearing after the drift; ranges of influence resp. spreads of clusters should be updated more strongly in order to be able to properly cover the drifted data cloud.

Thus, in literature, many incremental, adaptive learning approaches have been suggested in order to conduct a forgetting of older learned relations and to react on drifts appropriately (see Sect. 1.2 for an overview). However, most of these are performing a model-based drift handling concept by triggering a forgetting mechanism which applies to the whole model definition space to the same extent. On the other hand, often drifts may appear only severe in local parts of the feature space—an example between the difference of a global drift and a local one (in the target) is shown in Fig. 2. In local drift cases, it is beneficial to react appropriately to the actual drift intensity therein, otherwise catastrophic forgetting may get severe [as studied in Lughofer (2005)]. In the sense of the example shown in Fig. 2, a too weak forgetting would not be able to

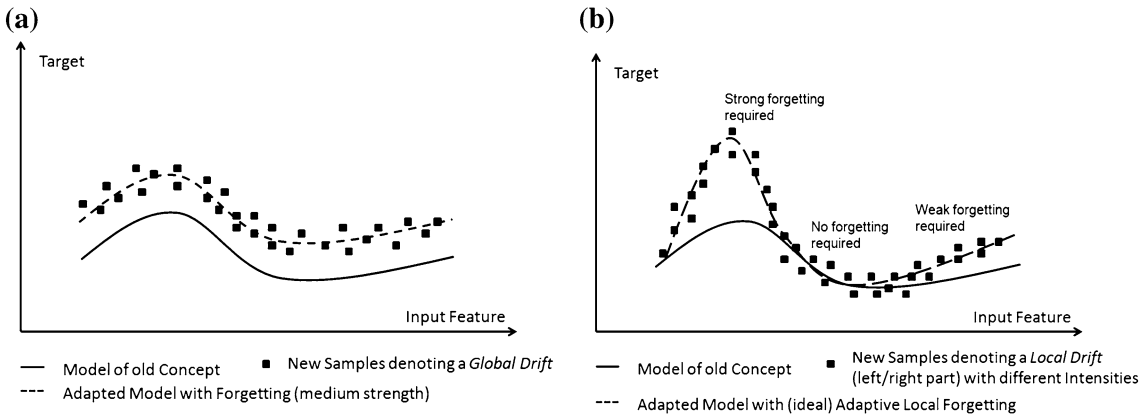

Fig. 2 **a** Global drift affecting the whole model space; **b** local drift affecting parts of the model space (*left-most* and *right-most*) with different intensities, the *dotted line* indicates the updated model with (ideal) adaptive local forgetting

react on the intense drift in the left part of the feature space, whereas a stronger forgetting would lead to unstable solutions in the middle part where no drift happens [deforming regression curves—see Lughofer (2005)]. This is also according to the well-known fact, that in usual cases when no drift happens, life-long learning employing the same weights for all samples over the whole stream, is known to generally out-perform learning with forgetting, see Hamker (2001). A reason for this lies in the more stable and faster convergence behavior of model components, usually coming close to the hypothetical batch solution (Sayed-Mouchaweh and Lughofer 2012), especially when parameters are modified by recursive approaches within incremental optimization scenarios [see Lughofer (2011), Chapter 2]. In the result section, we will see that, by using a fixed forgetting value, a permanent forgetting of parameters in our evolved rule-based models will lead to downtrends in accuracy during time frames when there is no drift present in the stream. Thus, a local adaptive forgetting concept according to the current drift intensity in a data stream and according to its locality is warmly recommended.

First, we will describe the mechanisms behind our new drift identification and quantification method (Sect. 3) and then demonstrate the adaptive drift reaction strategy (Sect. 4), which depends on the drift intensity level (outcome of the identification), both acting in global manner on model level and applicable to all sort of evolving models. Afterwards, we will present concepts which address the local nature of drifts, that is, the introduction of different (adaptive) forgetting factors in different regions of the model definition (feature) space. This is supported in a smooth way by the usage of evolving fuzzy systems providing a partitioning of the feature space into local granules (fuzzy rules) (Sect. 5). Finally, we will provide a concept for local drift indicators and quantifiers, which are performing a zooming stage into the feature space to realize whether some partial local regions become already affected by the drift (but still not affecting the model error) (Sect. 6). This can be seen as a kind of early drift recognition stage.

3 Smooth drift quantification (global)

Our approach relies on the evolution of a performance indicator over time in order to track the development of the model accuracy and to realize downtrends at an early stage. In fact, whenever the accuracy stays on a similar level over a certain amount of time, we assume the incremental modeling process in a stable state, at least from "outside" glance. For an appropriate indicator, we have chosen the Page-Hinkley (PH) test (Mouss et al. 2004), which considers a cumulative variable $m_{i,T}$ defined as the cumulative difference between the observed error values $x(t)$ and their mean $\bar{x}(t-1)$ till the current moment:

$$m_T = \sum_{t=1}^{T} (x(t) - \bar{x}(t-1) - \delta) \tag{1}$$

where δ corresponds to the magnitude of changes that is allowed. The PH test is then simply to monitor the difference:

$$PH_T = m_T - M_T \qquad M_T = \min_T m_T \tag{2}$$

In the classical case [as used in Mouss et al. (2004), Gama (2010)], a drift warning is provided whenever this difference exceeds a certain threshold. Please note that the mean $\bar{x}$ used in (1) can be incrementally updated by:

$$\bar{x}(t) = \frac{(t-1)\bar{x}(t-1) + x(t)}{t} \tag{3}$$

The primary goal of the PH approach is to check whether the null hypothesis H_0 (that the mean of the studied process

Fig. 3 Algorithm for predicting the change of the mean

Procedure ChangeDetection

Input: x_t, the $RMSE$ over the current data block, $X_0 = \{\}$, $\alpha, \delta, \Lambda, minsize$
Output: $drift\_indicator \in \{Up|, None|, Down\}$

1: {Initial Phase}
2: **if** $0 < t < minsize$ **then**
3: $X_t = X_{t-1} \cup \{x_t\}$
4: **return** $drift\_indicator = None$
5: **end if**
6: **if** $t == minsize$ **then**
7: $\overline{x}_t = \frac{1}{|X_t|} \sum_{x_t \in X_t} x_t$
8: $U_t = 0$
9: $L_t = 0$
10: **return** $drift\_indicator = None$
11: **end if**
12: {Preparing for an increasing mean}
13: $U_t = \alpha U_{t-1} + (x_t - \overline{x}_t - \delta)$
14: $MinU_t = min(U_t, t = 1, \ldots, t)$
15: $PHU_t = U_t - MinU_t$
16:
17: {Preparing for a decreasing mean}
18: $L_t = \alpha L_{t-1} + (x_t - \overline{x}_t + \delta)$
19: $MaxL_t = max(L_t, t = 1, \ldots, t)$
20: $PHL_t = MaxL_t - L_t$
21:
22: {Deciding the direction of change}
23: **if** $PHL_t > \Lambda$ **then**
24: **return** $drift\_indicator = Down$
25: **else if** $PHU_t > \Lambda$ **then**
26: **return** $drift\_indicator = Up$
27: **else**
28: Update $\overline{x}_t$ by (3)
29: **return** $drift\_indicator = None$
30: **end if**

has not changed) can be rejected. This goal is only helpful to indicate a possible change, but not helpful in quantifying this change neither in the magnitude nor in the duration. In Raquel Sebastião Margarida et al. (2011), they enhance the performance of Page-Hinkley by adding a forgetting factor, used to give less importance to the old instances than the new ones, its pseudo-code is shown in Fig. 3. This forgetting also stops both $MinU_t$ and $MaxL_t$ from growing in the negative / positive direction respectively, which causes numerical instabilities. The direction of change can be inferred from the Lines 23–29. The first lines indicate the initial phase which is required in order to affirm the error behavior of the current modeling problem and to obtain a stable setup of the indicator at the start of the learning process: we used $minsize = 10$ in all our empirical studies as start-up phase to set up an initial PH value. Basically, we want to adapt the algorithm in Fig. 3 to also be able to quantify the change in a monotone way i.e. if the mean of the initial process was $x_{initial}$ the drift indicator should last for $x_2 (x_2 > x_1 > x_{initial})$, more than it does for x_1. Quantification also subsumes a smooth transition between a drift and non-drift phase.

A possible problem of the drift indicator can be realized in Fig. 5 (part a) where the Down indicator is fired after sometime of firing the Up one. The change in this example is simulated by taking 1,000 instances from the first generator which generates the number a after that we take the output from a second generator which gives the number b. By taking $a = 0.1$ and different values for $b = 0.6/0.8/1$ we see that in (a) the bigger the input of the second

Fig. 4 Algorithm for quantifying the change of the mean (reflecting drift intensity)

Procedure ChangeDetectionQuantified

Input: x_t, the performance on data block t, $X_0 = \{\}$, $\alpha = 0.999$, $\delta = 0.005$, $\Lambda = 0.02$, $minsize = 10$
Output: $drift\_indicator \in \{Up, Down, None\}$
$drift\_accum$ degree of drift

1: {Initial Phase as in Figure 3}
2:
3: {Preparing for an increasing mean}
4: $U_t = \alpha U_{t-1} + (x_t - \overline{x}_t - \delta)$
5: $\boldsymbol{MinU_t = MinU_t + |MinU_t(1-\alpha)|}$
6: $MinU_t = min(MinU_t, U_t)$
7: $PHU_t = U_t - MinU_t$
8:
9: {Preparing for a decreasing mean}
10: $L_t = \alpha L_{t-1} + (x_t - \overline{x}_t + \delta)$
11: $\boldsymbol{MaxL_t = MaxL_t - |MaxL_t(1-\alpha)|}$
12: $MaxL_t = max(MaxL_t, L_t)$
13: $PHL_t = MaxL_t - L_t$
14:
15: {Deciding the direction of change}
16: **if** $PHL_t > \Lambda$ **then**
17: $drift\_accum = MaxL_t - L_t$
18: $drift\_indicator = Down$
19: $\boldsymbol{MaxL_t = MaxL_t - |(MaxL_t - L_t)(1-\alpha)|}$
20: **else if** $PHU_t > \Lambda$ **then**
21: $drift\_accum = U_t - MinU_t$
22: $drift\_indicator = Up$
23: $\boldsymbol{MinU_t = MinU_t + |(U_t - MinU_t)(1-\alpha)|}$
24: **else**
25: Update $\overline{x}_t$ by (3)
26: $drift\_accum = 0$
27: $drift\_indicator = None$
28: **end if**
29: **return**

generator becomes the earlier the indicator is, but the problem remains that a wrong Down indicator is fired after sometime.

In order to avoid the aforementioned problem we apply the small forgetting factors not only on L_t and U_t but also on both $MinU_t$ and $MaxL$. This is shown in Lines 5 and 11 of our modified algorithm in Fig. 4. The quantification of the drift is shown on lines 15, 20 which measures the degree of change between the current and the previous process. In order to recover for the current change we try to drag $MaxL_t$ towards L_t by decreasing with a value proportion to its difference from L_t. Figure 5b shows the effect of these additional algorithmic lines: in fact, we can see that this approach is capable to detect the drift as early as the previous one, and the length of the indication period is relative to the change of the input. After some indication period, the process returns to the None status. Furthermore, we want to explicitly highlight that the quantification of the drift intensity is given by *drift_accum*, changing continuously over time.

4 Global drift reaction with adaptive forgetting

4.1 Adaptive forgetting

The drift indication process is applied on the normalized RMSE on separate test folds $t = 1, ...$ during the incremental process:

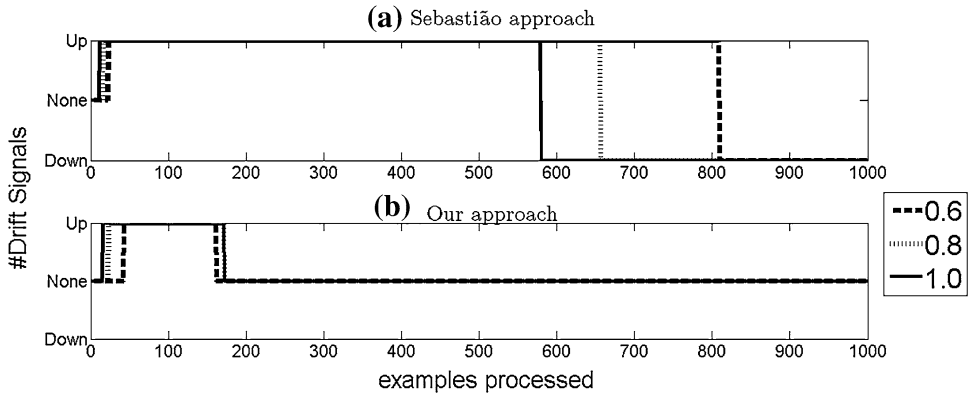

Fig. 5 Comparison of drift indicator approaches based on Page-Hinkley (2); **a** the approach by Sebastiao et. al in (2011) as shown in Fig. 3, **b** our new approach according to Fig. 4

$$RMSE_{normed}(t) = \frac{\sqrt{\sum_{j=1}^{N}(y(j) - \hat{y}(j))^2}}{(\max(y) - \min(y))} \tag{4}$$

substituting $x(t)$ in (1) and being used in the algorithm in Fig. 4, with N the number of data samples in one test fold (its value set depending on the size of the stream, see also Sect. 8). The normalization in (4) makes the method (especially its parameters) applicable for various targets with different ranges without a necessary re-tuning stage.

The initialization parameters are α, δ and Λ. δ is just a noise compensation level in the PH test (1) and can be usually set close to 0 (we used 0.005 in all our tests and found out low sensitivity). α is a forgetting level for the PH indicator itself an can usually set to a high value close to 1 (Sebastiao et al. 2013) (we used 0.999 in all our tests and found out low sensitivity). Λ is the real sensitive parameter in algorithm, as it steers the tradeoff for deciding the direction of the change based on the normalized RMSE; default setting is 0.02 and we performed a sensitivity analysis (Sect. 8.3).

After n steps, the forgetting factor for the model parameters and structure at time instance t termed as λ_t is adapted to reflect the way the current RMSE is changing (drift intensity). This adaptation is done based on the output of ChangeDetectionQuantifed procedure, namely the direction of change $drift\_indicator \in \{Up, None, Down\}$ and the degree of drift $drift\_accum$. We change the forgetting factor λ_t in the opposite direction of change, i.e a larger forgetting (which induces a smaller value of λ_t) when the performance drops (an Up indicator on the RMSE) and a smaller forgetting (which induces a higher value of λ_t) when the performance improves (a Down indicator on the RMSE). This is achieved by

$$\lambda_t = min(max(\lambda_{t-1} - direction(drift\_indicator)C_t, 0.9), 0.999) \tag{5}$$

$$C_t = \frac{drift\_accum_{t-1} - drift\_accum_t}{\rho * x_t} \tag{6}$$

where λ_t is the forgetting factor at time block t, C_t the amount of change in accumulated drift intensity and x_t the performance on data block [default $RMSE\_normed$ as in (4)].

The forgetting factor plays an essential role in order to ensure a smooth forgetting of the evolving model (see Sect. 7 how the concrete integration works). The sign of change is determined by the direction:

$$direction(x) = \begin{cases} -1 & x = Up \\ 0 & x = None \\ 1 & x = Down \end{cases}. \tag{7}$$

The range of forgetting factor values $\lambda \in [0.9..0.999]$, where 0.9 denotes a strong forgetting and 0.999 a very weak forgetting [achieved by the min–max operation in (5)], shows that the forgetting mechanism is highly sensitive to the values taken by the forgetting factor, being responsible for the tradeoff between plasticity and stability of the model. That's why ρ should be chosen to be large enough. Clearly, if there is an up-trend of the indicator, λ_t decreases (as C_t gets negative), hence the strength of forgetting is increased.

5 Local adaptive forgetting factors

Opposed to the previous strategy, where we introduced one unique forgetting factor for the whole model definition space, in this section we go a step further and demonstrate a possibility how to react onto local changes due to local drifts with different intensities. The basic idea of our approach is to distribute the degree of a drift $drift\_accum$ among the models' components (partial local regions) by incorporating their accumulated activation levels during the last test data block which was used for obtaining the normalized RMSE (4) as current prediction error. We therefore assume that the evolving model used for data stream regression has components which can be identified as local approximators. Local regions with higher

activation levels will get a higher portion of the forgetting factor than local regions with low activation levels; this is somewhat intuitive as these contributed more to the final model predictions and therefore also to the error over the last data block. Hence, in case of a drift it can be assumed that it is mainly present in regions which contributed most to an increased model error.

In case of fuzzy models, the partial error committed by a Rule i on an instance x_j, is the prediction error for the instance x_j, $Err_{x_j} = |y(j) - \widehat{y}(j)|/(\max(y) - \min(y))$ weighted by the average activation level of the fuzzy sets in the ith rule:

$$Err_{i,j} = \frac{\sum_{k=1}^{p} A_{ik}(x_j)}{p} Err_{x_j} \tag{8}$$

with p the dimensionality of the input space and A_{ik} the fuzzy set in the kth antecedent part of the ith rule. Thus, the activation level of the sets guarantee a smooth transition and division of the forgetting intensity over nearby lying local regions affected by the drift (as fuzzy sets are usually overlapping, especially when having infinite support), rather than triggering a crisp switch between 'drift on' and 'drift off' for the different local components in the regression models [as is the case in Ikonomovska et al. (2009)]. Summing $Err_{i,j}$ over all the data samples N in the current test block yields

$$Err_i = \sum_{j=1}^{N} Err_{i,j} \tag{9}$$

The forgetting factor for each rule i is calculated in the same way as in the global case except that the change is weighted by the normalized activation level A_i:

$$\begin{aligned} \lambda_{i,t} =& min(max(\lambda_{t-1} - direction(drift\_indicator) \\ & \times C_t A_{i,t}, 0.9), 0.999) \end{aligned} \tag{10}$$

$$C_t = \frac{drift\_accum_{t-1} - drift\_accum_t}{\rho * x_t} \qquad A_{i,t} = \frac{Err_i}{\sum_{k=1}^{C} Err_k} \tag{11}$$

with C the number of rules in the current evolved fuzzy system.

6 Early drift quantification towards drift prevention (local indicators)

The motivation here is that local drifts in the feature space can be only or earlier seen when inspecting drift indicators in the local parts than when using a global drift indicator on the model error level: all model components joined together contribute to the drift indicator and when a drift in one local region is already intense, it may be washed up and not seen in the global indicator based on overall global model outputs. Later on, when the drift gets more intense (e.g. affecting a larger part of the input space), the global error indicator will indeed rise, however the model performance has been already affected before. This finally results in a kind of "reaction after a drift has already become severe". Thus, when correcting partial local drifts earlier (through local drift quantifier), downtrends in model performance may be even prevented; in this sense, we may also talk about early drift recognition towards drift prevention.

A strategy which we pursue here is a specific drift indication criterion for the product space in a fuzzy system. The criterion for product space relies on multivariate Gaussians defined by:

$$\mu_i(\boldsymbol{x}) = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{c}_i)^T \Sigma_i^{-1} (\boldsymbol{x} - \boldsymbol{c}_i)\right) \tag{12}$$

with $\boldsymbol{c}_i$ the center and Σ_i the covariance matrix, and can therefore be applied for other architectures integrating these components (e.g. Gaussian mixture models, radial basis function networks, gaussian process models etc.). In the special case of fuzzy systems applying Gaussian membership functions [the most common choice in current EFS approaches (Lughofer 2011)], the contours of the rules in the product-space become axis-parallel ellipsoids, which can be interpreted as a specific case of a $p+1$-dimensional multivariate Gaussian distributions with diagonal co-variance matrices $\Sigma = diag(\sigma_1, \sigma_2, ..., \sigma_{p+1})$, thus:

$$\mu_i(\boldsymbol{x}) = \prod_{j=1}^{p+1} e^{-\frac{1}{2}\frac{(x_j - c_{ij})^2}{\sigma_{ij}^2}} = e^{-\frac{1}{2}\sum_{j=1}^{p+1}\frac{(x_j - c_{ij})^2}{\sigma_{ij}^2}} \tag{13}$$

Thus, a local drift in the antecedents goes in conformity with a changed local multivariate data distribution in the product-space. The degree of change depends on the (local) drift intensity and its impact is limited by the component (rule) evolution criterion. For instance, a local shift usually causes a new component (rule) to be evolved and this is not affecting the old data distributions (rules), thus usually not affecting the model error (older components are not activated any longer). In this sense, local shifts mostly "only" cause unnecessary complexity but not a downtrend in accuracy. This is a strong point compared to other model architectures which do not contain structural components for representing local models over different parts of the feature space (e.g. genetic programming for regression, statistical regression methods etc.): a local shift there already requires a global model adaptation in a more intense sense as affecting the overall model error.

Figure 6 shows our principal strategy of local drift indication through divergence of "sample clouds" between two consecutive blocks.

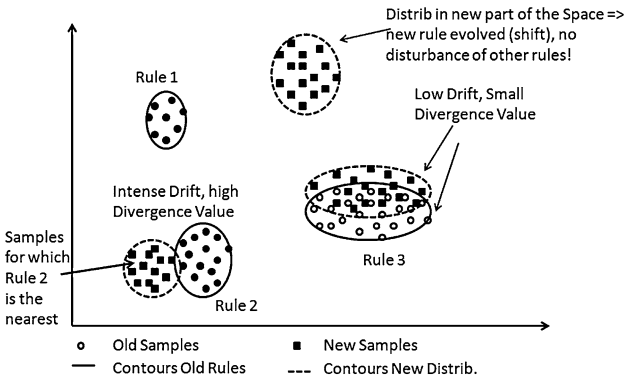

Fig. 6 Divergence of distributions from a new data block (marked by *rectangular* samples) to old data represented by rules (*solid ellipsoids*); the divergence is measured in terms of the nearest component (rule) to the new data samples (local divergence pointing to local drifts): in one case the drift is high (*bottom left*), in another it is low (*middle right*), in both cases the old component (rule) is expected to cover the new situation for sufficient flexibility, which cannot be granted by conventional update due to converged (stucked) situations—as also demonstrated in Fig. 1 → forgetting required

Whenever a new data blocks arrives, all the samples contained in it are assigned to those rules which are nearest (regular samples), except those ones which are candidates for forming new rules according to the applied rule evolution criterion. In particular, for each sample $\boldsymbol{x}$ in the new data block, it is first checked whether the rule evolution criterion is fulfilled, and if so, the sample is not assigned to any already available rule. The rule evolution criterion depends on the evolving fuzzy systems approach used: in case of eTS+ (Angelov 2010), for instance, it is the (recursively calculated) potential of the current sample. Whenever the evolution criterion is not fulfilled, the index of the rule which is nearest to $\boldsymbol{x}$, denoted as $i*$, is elicited:

$$i* = argmin_{i=1,\ldots,C} \|\boldsymbol{x} - \boldsymbol{c}_i\|_A \tag{14}$$

with A the norm inducing distance: if L^2 is used, it triggers the standard Euclidean distance, in case of L^1 we yield the Mahalanobis distance (Mahalanobis 1936), which also respects the rule spreads. Then, the set of new samples belonging to $i*$ is expanded, i.e. $S_{i*} = S_{i*} \cup \{\boldsymbol{x}\}$, with S_{i*} always set to the empty set before the next data block arrives. After that, all of the regular samples falling into a rule, let's say the ith, and thus contained in S_i are used for extracting the current mean and diagonal covariance matrix of the corresponding local distribution (as contained in the data):

$$\boldsymbol{c}_i(new) = \frac{\sum_{\boldsymbol{x} \in S_i} \boldsymbol{x}}{|S_i|} \qquad \sigma_i(new) = \frac{1}{|S_i| - 1} \sum_{\boldsymbol{x} \in S_i} (\boldsymbol{x} - \boldsymbol{c}_i)^2 \tag{15}$$

Then, the divergence degree to the old rule's distribution is measured within the product space (multivariate Gaussians).

For doing so, we use the Kullback–Leibler divergence (Kullback and Leibler 1951), which for two multivariate Gaussian distributions $G_1 = (\boldsymbol{c}_1, \Sigma_1)$ and $G_2 = (\boldsymbol{c}_2, \Sigma_2)$ is defined by the following compact form:

$$D_{KL}(G_1||G_2) = \frac{1}{2}\left(trace(\Sigma_2^{-1}\Sigma_1) + (\boldsymbol{c}_2 - \boldsymbol{c}_1)^T \Sigma_2^{-1}(\boldsymbol{c}_2 - \boldsymbol{c}_1) - p - ln\left(\frac{det(\Sigma_1)}{det(\Sigma_2)}\right)\right) \tag{16}$$

with p the dimensionality of the learning problem. In our case, (16) measures the divergence of the model G_2 [which is equal to the rule component $R_i = (\boldsymbol{c}_i(old), \sigma_i(old))$] to the actual distribution contained in the data G_1 [which is $R_i(new) = (\boldsymbol{c}_i(new), \sigma_i(new))$] estimated by (15)]. Note that for rule component G_2 (13) is assumed to be multiplied with $1/\sqrt{(2\pi)^p * det(\Sigma_2)}$ to represent the multivariate normal distribution the corresponding rule induces. Clearly, when $G_1 = G_2$, $D_{KL}(G_1||G_2)$ achieves its minimal value of 0, pointing to no drift. Thus, the higher the value of $D_{KL}(G_1||G_2)$ becomes, the more difference between the distributions is. Note that in case of fuzzy systems, we are operating with axis parallel Gaussians (rules), thus inversion, determinant and trace in (16) become fast computations on diagonal matrices. In this sense, the computational complexity is $O(6(p+1)+3) \approx O(p+1)$, thus linear in the dimensionality of the learning problem.

Furthermore, old components represented by the fuzzy rules and new components extracted from the current data buffer usually possess a different "significance", as they have been supported by a different number of samples. This should be respected in their impact on the divergence degree and finally in the D_{KL}. Thus, we intend to deduce a weighted Kullback–Leibler divergence, with weights w_1 and w_2

$$w_1 = \frac{k_i(new)}{k_i(new) + k_i(old)} \qquad w_2 = \frac{k_i(old)}{k_i(new) + k_i(old)} \tag{17}$$

where $k_i(new)$ denotes the support of the local distribution G_1 ($R_i(new)$), and $k_i(old)$ denotes the support of the nearest rule G_2 (R_i) (as incremented through stream learning); clearly $w_1 + w_2 = 1$.

In general, the Kullback–Leibler divergence can be written as:

$$D_{KL}(G_1||G_2) = \int G_1(x) log\left(\frac{G_1(x)}{G_2(x)}\right) dx \tag{18}$$

By integrating the weights, we obtain the weighted Kullback–Leibler divergence as

$$D_{KLW}(G_1||G_2) = \int w_1 G_1(x) \log\left(\frac{w_1 G_1(x)}{w_2 G_2(x)}\right) dx$$
$$= w_1 \int G_1(x)\left(\log\left(\frac{w_1}{w_2}\right) + \log\left(\frac{G_1(x)}{G_2(x)}\right)\right) dx$$
$$= w_1\left(\log\left(\frac{w_1}{w_2}\right)\int G_1(x)dx + \int G_1(x)\log\left(\frac{G_1(x)}{G_2(x)}\right)dx\right) \quad (19)$$

$$= w_1\left(\log\left(\frac{w_1}{w_2}\right) + \int G_1(x)\log\left(\frac{G_1(x)}{G_2(x)}\right)dx\right) \quad (20)$$
$$= w_1\left(\log\left(\frac{w_1}{w_2}\right) + D_{KL}(G_1||G_2)\right)$$

as $\int G_1(x)dx = 1$. Thus, the higher the support of the new sample cloud G_1 gets (i.e. the higher w_1 gets), the larger the value of the divergence measure becomes, as the logarithm is a monotonically increasing function. This is a desired property as new distributions with a higher significance should more likely point to a drift when diverging.

In case of applying weighted D_{KL}, the divergence is tendentially not normalized and can reach high values (the same as for extended Page-Hinkley test as described in Sect. 3), thus we suggest to apply the same procedure as for Page-Hinkley test according to (1), *drift_accum* and the algorithm proposed in Fig. 4 for each rule separately to elicit directly the adaptive local forgetting factors $\lambda_{i,t}, i = 1, ..., C$ for the C rules according to (5). This avoids the usage of rule firing elicitation for each test block according to Sect. 5, which gives only an approximative guess for the drift intensity in each local part. The difference is just to apply the weighted DKL values for each rule according to (19) instead of the RMSE values according to (4) as x in (1) and as x_t in Fig. 4 and (5) as well as (10).

Algorithm 1 summarizes the necessary steps for obtaining a local change in rules based on D_{KLW}, assuming an input fuzzy model with C rules.

Algorithm 1 Local Drift Quantification

Input: new data block $b_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, rule-based model F, help variables $U_{t-1}(i)$, $L_{t-1}(i)$, $MinU_{t-1}(i)$, $MaxL_{t-1}(i)$ for each rule $i = 1, ..., C$ in F.
Output: drift indicators $drift\_indicator_i$ and drift quantities $drift\_accum_i$ for all rules $i = 1, ..., C$ in F.

1. $S_i = \{\}$ for all $i = 1, ..., C$.
2. For each sample $k = 1, ..., N$ in b_t:
   (a) Calculate (14).
   (b) Expand $S_{i*} = S_{i*} \cup \{\mathbf{x}_k\}$.
3. For $i = 1, ..., C$:
   (a) **IF** $|S_i| < thr$ (we set $thr = 10$ in all cases)
   (b) **THEN** $D_{KLW}(i)(t) = D_{KLW}(i)(t-1)$
   (c) **ELSE** Calculate (15); calculate $D_{KLW}(i)(t)$ using (19) with $G_1 = (\mathbf{c}_i(new), \sigma_i(new))$ and $G_2 = (\mathbf{c}_i(old), \sigma_i(old))$ the rule component
   (d) Perform Algorithm in Figure 4 with $x_t = D_{KLW}(i)(t)$ using help variables $U_{t-1}(i)$, $L_{t-1}(i)$, $MinU_{t-1}(i)$, $MaxL_{t-1}(i)$ for rule i to obtain $drift\_indicator_i$ and $drift\_accum_i$.
4. Return $drift\_indicator_i$ and $drift\_accum_i$ for all rules $i = 1, ..., C$ in F.

The output values of Algorithm 1 can then be used in (5) for calculating the current forgetting factor values $\lambda_{i,t}$ for all rules $i = 1, ..., C$.

The output values of Algorithm 1 can then be used in (5) for calculating the current forgetting factor values $\lambda_{i,t}$ for all rules $i = 1, ..., C$.

7 Forgetting factors integration and overall algorithm

The final open issue is how to integrate the global resp. local forgetting factors [λ_t resp. $\lambda_{i,t}$ elicited by (5) resp. by (10)] into the incremental model adaptation scheme. We therefore concentrate on the evolving Takagi–Sugeno type fuzzy model, as this is the most widely applied model

architecture in the field of evolving fuzzy systems (EFS) (Lughofer 2011) in the context of data stream regression problems. This is due to excellent approximation capabilities (high accuracies) while still offering some valuable interpretable insights into the process (Lughofer 2013). EFS are a learning concept for permanently updating fuzzy models with new incoming data samples, including fast and robust adaptations of model parameters, evolution, merging and pruning of components on demand based on the actual characteristics of the stream model. Using TS fuzzy model architecture offers the application of piece-wise local linear predictors l_i which are connected with multivariate kernels Ψ to model the local data clouds:

$$\hat{f}(\boldsymbol{x}) = \hat{y} = \sum_{i=1}^{C} \Psi_i(\boldsymbol{x}) l_i(\boldsymbol{x}) \tag{21}$$

Usually, the Ψ_i's are normalized by their sum over all rules for each data sample to ensure numerical stability. The transition of one predictor to the other is continuous and smooth whenever the kernels are steady differentiable and possess infinite support, thus no undefined input states occur. Hence, this architecture is perfectly shaped for our smooth and local drift handling intensions. Local drift indication strategies as presented in the previous section can be applied when the TS models are equipped with Gaussian fuzzy sets in connection with product t-norm (Klement et al. 2000) which leads to the multivariate definition as in (13).

In most of the common approaches, for the linear consequent parameter the local learning scheme in a recursive context is applied for minimizing the least squares error between predicted and observed values; this leads to the recursive fuzzily weighted least squares (RFWLS) formulation (Angelov et al. 2008; Angelov and Filev 2004) and has some advantageous properties over global learning, see Angelov et al. (2008). We integrate an exponential forgetting in the optimization function using the forgetting factors as calculated in (10) resp. (5) with $\lambda_{i,t} = \lambda_t$ for all i:

$$J_i = \sum_{t=1}^{N} \lambda_{i,t}^{N-t} \Psi_i(\boldsymbol{x}(t)) e_i^2(t) \longrightarrow \min_{\boldsymbol{w_i}} \tag{22}$$

with $e_i(t) = y(t) - \hat{y}(t)$ the error of the ith rule in sample t. For this optimization problems, the recursive formulation is achieved by Lughofer and Angelov (2011) (here for the ith rule):

$$\hat{\boldsymbol{w}}_{\boldsymbol{i}}(t+1) = \hat{\boldsymbol{w}}_{\boldsymbol{i}}(t) + \gamma(k)(y(t+1) - \boldsymbol{r}^T(t+1)\hat{\boldsymbol{w}}_{\boldsymbol{i}}(t)) \tag{23}$$

$$\gamma(t) = \frac{P_i(t)\boldsymbol{r}(t+1)}{\frac{\lambda_{i,t}}{\Psi_i(\boldsymbol{x}(t+1))} + \boldsymbol{r}^T(t+1)P_i(t)\boldsymbol{r}(t+1)} \tag{24}$$

$$P_i(t+1) = (I - \gamma(t)\boldsymbol{r}^T(t+1))P_i(t)\frac{1}{\lambda_{i,t}} \tag{25}$$

with $\boldsymbol{r}(t+1)$ the regressor for the current data sample. Thus, the forgetting factor $\lambda_{i,t}$ steers the degree of forgetting in the inverse Hessian P_i as well as the consequent parameters $\boldsymbol{w}_i$. Furthermore, the rule fulfillment degree Ψ_i in the denominator of (24) steers the contribution of the rule to the current drift situation: a 'drifted' sample lying far away from the rule i, i.e. the drift is more active in another part of the feature space, will have a low degree Ψ_i, yielding a high denominator in (24) and therefore $\gamma \approx 0$, thus (also a strong) forgetting will influence the update of the consequent parameters in (23) very slightly; on the other hand, a 'drifted' sample will trigger a higher forgetting for those rules the drift is really close to. A similar integration can be done into the generalized version of RFWLS, emphasizing weight decays, as recently introduced in EFS in Pratama et al. (2014).

Regarding the antecedent space, the EFS approaches differ how they perform the update of the fuzzy sets and the non-linear parameters therein. Here, we shortly summarize the concept in connection with the FLEXFIS family (Lughofer 2008, 2012), as this is the method which we are going to apply in the evaluation section [for the integration in case of eTS, another very prominent approach, please refer to Lughofer and Angelov (2011)]. Whenever the rule is updated, it is checked whether its support is higher than a threshold (we used 30 samples in all tests): if yes, the converged rule centers and spreads are re-activated by decreasing the support $n_.$ in the following way (here for the ith rule):

$$n_i = n_i - n_i * min(\lambda\_trans, 0.99) \quad \lambda\_trans = -9.9\lambda_{i,t} + 9.9 \tag{26}$$

This automatically increases the learning gain $\eta_i = 0.5/n_i$, helping out the cluster from its converged position, as affecting a stronger rule movement for the next samples. It is also easy to realize that when $\lambda_{i,t} = 1$, i.e. no forgetting is triggered by our change quantifier, no increase of the learning gain takes place and the cluster remains in the converged position. On the other hand, in case of a low value of $\lambda_{i,t}$, a large percentage of n_i is subtracted from n_i, thus enforcing a strong increase of the learning gain and a strong movement of the cluster. This also affects its spread, as n_i appears as contribution factor of old spread versus new sample

distance in the recursive variance formula (Qin et al. 2000). The concrete values for the slope (−9.9) and intercept (9.9) in the second formula in (26) are explained by the fact that we assume a strongest possible forgetting of $\lambda_{i,t} = 0.9$ [according to the maximum operation in (5) and (10)], and the weakest one with $\lambda_{i,t} = 1.0$ (no forgetting). In the first case, we want to map 0.9 to $\lambda\_trans = 0.99$ in order to almost (but not completely) diminish the current learning gain n_i; in the second case we want to map 1.0 to $\lambda\_trans = 0$ in order to not decrease n_i and to continue learning without re-setting the learning gain. In-between, we are aiming for a pure linear decrease of *lambda_trans*, leading to the straight line (with slope −9.9 and intercept 9.9) in the second formula of (26).

7.1 Overall algorithm

The particular steps of the whole drift quantification and integration approach are summarized in Algorithm 2. It is therefore assumed that a previously trained model F is already available (e.g. evolved from first data blocks in a stream or in a previous batch off-line phase)—note that some first *minsize* blocks are required, too, to set up an initial drift indicator, see the Algorithm in Fig. 4.

Algorithm 2 Drift Quantification and Reaction in (Fuzzy) Rule-Based Model

1. Input: new data block $b_t = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, rule-based model F, help variables U_{t-1}, L_{t-1}, $MinU_{t-1}$, $MaxL_{t-1}$.
2. Output: flexibly updated rule-based model F_{upd}, updated help variables U_t, L_t, $MinU_t$, $MaxL_t$.
3. **Case global quantification (default), Steps 5-7**:
4.
5. Estimate the error $RMSE_{normed}(t)$ as in (4).
6. Apply Algorithm as in Figure 4 to retrieve drift indication $drift\_indicator$ and the drift quantity $drift\_accum$.
7. **IF** $drift\_indicator$= **Up** or $drift\_indicator$= **Down**
   (a) Estimate the forgetting factor$\lambda_{i,t}$ using Eq. (5) for global forgetting case and using Eq. (10) for the local forgetting case.
8.
9. **Case local quantification (extended), Steps 11 and 12**:
10.
11. Perform Algorithm 1 to retrieve drift indications $drift\_indicator_i$ and drift quantities $drift\_accum_i$ for all rules $i = 1, ..., C$.
12. **IF** $drift\_indicator_i$= **Up** or $drift\_indicator_i$= **Down**
    (a) Estimate the forgetting factor $\lambda_{i,t}$ using Eq. (5).
13.
14. Integrate the new forgetting factors into the recursive optimization in (22) (useable for most EFS approaches in literature), continue learning with the next training samples.
15. Integrate the new forgetting factors into (26) for more intense antecedent change (EFS approach dependent), continue learning with the next training samples.
16. $\rightarrow F_{upd}$

8 Evaluation

8.1 Experimental setup

In our experiments, we employ a holdout method for measuring predictive accuracy, also motivated by the holdout

procedure in the MOA framework (Bifet et al. 2010), as is the most prominent strategy in data stream evaluation (mimicking conventional cross-validation). Here, the idea is to interleave the training and the testing phase of a learner as follows: the learner is trained incrementally on a block of M instances and then evaluated (but no longer adapted) on the next N instances, then again trained on the next M and tested on the subsequent N instances, and so forth; as parameters, we use different values for M and N, depending on the size of the used data set.

In Gama et al. (2013), the authors proof that, for large enough hold-outs, the limit of the estimated hold-out error is the Bayes error. Thus, the choice of N aims only at reaching the limit of the loss, namely the Bayes limit. Furthermore, we want to point out that the optimal setting for the other parameters (threshold Λ in Algorithm 4, δ, ρ and α) was elicited in multiple trial-and-error test runs on various data streams. We finally obtained a single setting, which was optimal for all streams, namely $\Lambda = 0.02$, $\delta = 0.005$, $\rho = 1,000$ and $\alpha = 0.999$. Also, *minsize* was set to 10, such that drift quantification could start after the first 10 blocks.

We will apply different fixed forgetting factors [as used in Lughofer and Angelov (2011)], as well as global and local forgetting according to their descriptions in Sect. 4. For comparing incremental learning on data streams without and with our drift handling approach, we applied FLEXFIS as evolving rule-based model approach (Lughofer 2008); this is simply because we have a long-term past experience with this method, thus minimizing efforts for integrating and testing the new drift concepts and assuring their correctness by comparison with the performance of original FLEXFIS. Furthermore, we will use its extended form as demonstrated in Lughofer et al. (2011) for eliminating unnecessary rules in order to obtain models with low complexity and faster performance.

8.2 Characteristics of the data sets and contained drifts

8.2.1 CT slices, year prediction MSD with rotational change

The real data sets are standard benchmarks taken from the UCI repository[2]. In the first experiment, we used the UCI data set about relative location of CT (computed tomography) slices on axial axis. This data set is extracted from 53,500 images taken for 74 different patients (43 males and 31 females). Each CT image is described in terms of 384 features. The target attribute is the relative location of the CT slice on the axial axis of the human body; this is a numeric value in the range $[0, 180]$, where 0 denotes the top of the head and 180 the soles of the feet. Moreover, using a modified variant of the forward selection method as proposed in Cernuda et al. (2011), Groißböck et al. (2004), the number of features was reduced to 9. Learning was started on an initial block of 100 instances, and incremental learning was done using M =1,000 and N = 500.

The second data set used in this section is the YearPredictionMSD data. Here, the task is to predict the year in which a song was released, based on 90 audio features, 12 of which are for timbre average and 78 for timbre covariance. The songs are chosen from the years between 1922 till 2011. The number of features is reduced to 8 features in the same way for the former data set. Besides we set M = 5,000 and N = 1,000.

In order to simulate drifts in the above mentioned data sets, we randomly selected two features out of the complete feature space, which we rotated by 90° after a pre-determined fixed point of time, i.e. number of samples passed by (for the slice location data set this number was set to 18,000 and for the YearPredictionMSD set to 66,000). This strategy is in accordance to the methodology proposed and analyzed in Kurlej and Wozniak (2011). Assuming all features normalized to $[-1, 1]$ and to rotate the $X \times Y$-space either in clock-wise or counter clock-wise manner, the following transformations are obtained:

$$(x, y) \xrightarrow{R\left(\frac{\pi}{2}\right)} (-y, x) \quad (x, y) \xrightarrow{R\left(\frac{-\pi}{2}\right)} (y, -x) \tag{27}$$

8.2.2 Wine data set with gradual type change

Another artificial drift type was tested in combination with the wine data set taken from the UCI repository[3] and including samples for red whine and white wine, which are related to variants of the Portuguese "Vinho Verde" wine (Cortez et al. 2009). The task is to predict the wine quality on a scale of [0,10] (graded by experts), which can be viewed either as classification or as regression problem (the latter was able to outperform classification modeling in past experiments, justifying the usage within our regression approach). The data set contains 11 input variables and 4,858 samples of white whine and 1,599 samples of red wine. In order to examine the effect of the wine type onto the prediction of the wine quality, we artificially built in a drift which changes the concept from white wine to red wine gradually. This is achieved when the probability of seeing examples of red wine increases gradually until it becomes the dominant concept. Therefore, we foresee two intensity levels of this gradual change, i.e. setting α to 0.04 indicating a slow drift, and setting α to 0.5 indicating a fast drift—the change concept with the two different α's is shown in Fig. 7 (x-axis denotes the sample number).

[2] http://archive.ics.uci.edu/ml/.

[3] http://archive.ics.uci.edu/ml/.

8.2.3 Rolling mill data with real-occurring drift

The data originates from metal industry, in particular from the tension-leveler part of the cold rolling process within the biggest steel company of Austria. For this purpose, a large-scale multi-sensor networks is installed at the system, which is distributed over a wider range of several meters and contains 240 measurement channels in sum. The database provided by the company has over 1 million samples, based on which data-driven models have been extracted, which have been successfully used for fault detection purposes as underlying fault-free reference models, see Serdio et al. (2014). A real-occurring drift due to a change in the steel type has been identified within a subsystem modeled by 5 input channels and a target channel in a regression setting. In the corresponding data sets containing 9,486 samples, the drift is known to start at Sample # 3005 and to end at Sample # 4261.

8.3 Results

The results on the slice location data set are visualized in Fig. 8. This picture shows the normalized RMSE over t time steps for different variants of the learning procedure: no forgetting (dotted line), fixed forgetting with $\lambda_t = 0.9$ and $\lambda_t = 0.95$ (dashed dotted lines) for all rules and our adaptive forgetting approaches shown by the solid lines (the local one indicated with crosses as line markers)—note that local and global adaptive forgetting perform equally for this data set, thus completely overlaying each other.

Obviously, the fixed forgetting strategies suffer significantly from the inability to adapt the intensity of the forgetting according to the current situation in the data stream. In fact, during the first half of the incremental learning process, they cause instabilities in the model (highlighted by ellipsoids in the figure) due to a too strong forgetting. In the second half of the process they are showing the same picture, no matter whether a drift appears (Fig. 8b or not a).

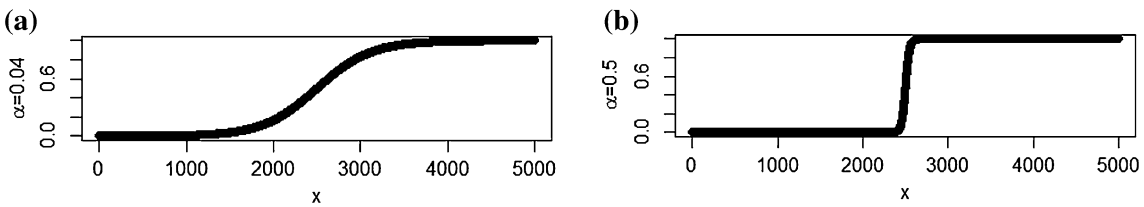

Fig. 7 **a** Concept change with $\alpha = 0.04$; **b** concept change with $\alpha = 0.5$

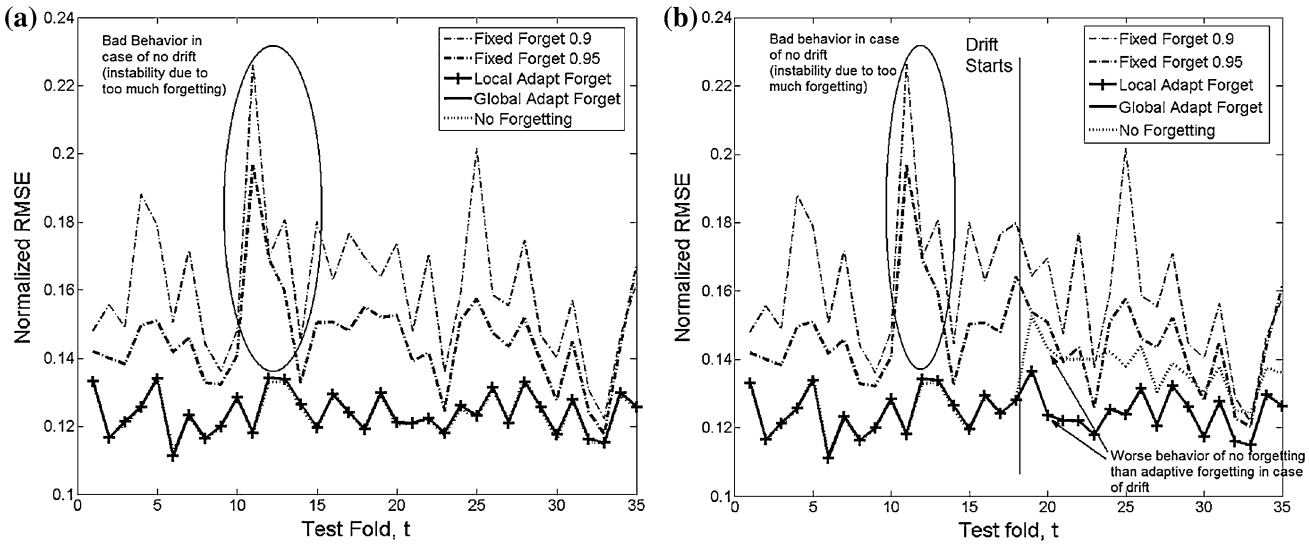

Fig. 8 **a** The evolution of RMSE over the number of test folds for slice localization data set without including any drifts: note the much worse performance of a fixed forgetting (0.9 and 0.95). **b** The evolution of RMSE for the same data set when including a drift at position $t = 18$: when including no forgetting in the learning, the model looses significant accuracy compared to our proposed adaptive forgetting schemes

Interestingly to see that conventional adaptation of the model with no forgetting performs equally well as our adaptive strategies during the non-drift phase, but significantly deteriorates when a drift appears (from $t = 18$ on, indicated by a vertical line in Fig. 8b).

The results on the YearPredictionMSD data are visualized in Fig. 9. The fixed forgetting variants show a similar behavior as in case of slice location data set (Fig. 9a): again a fixed forgetting of either 0.9 or 0.95 worsens the stability of the models, no matter whether a drift appears or not; it is interesting to see that in case of a drift both options are increasing the error drastically, hence over-shooting the intensity of the drift, which can be better modeled by our local and global adaptive approach: again a decrease in accuracy can be observed immediately after the drift started at around $t = 41$, but the peaks are much less intense than in case of fixed forgetting or no forgetting. The latter is not even able to model the drifted situation properly throughout the remaining lifetime of the stream. It is also worth to mention that local adaptive forgetting outperforms global adaptive forgetting after the drift started (the positions marked in Fig. 9b); here, only a slight increase of the

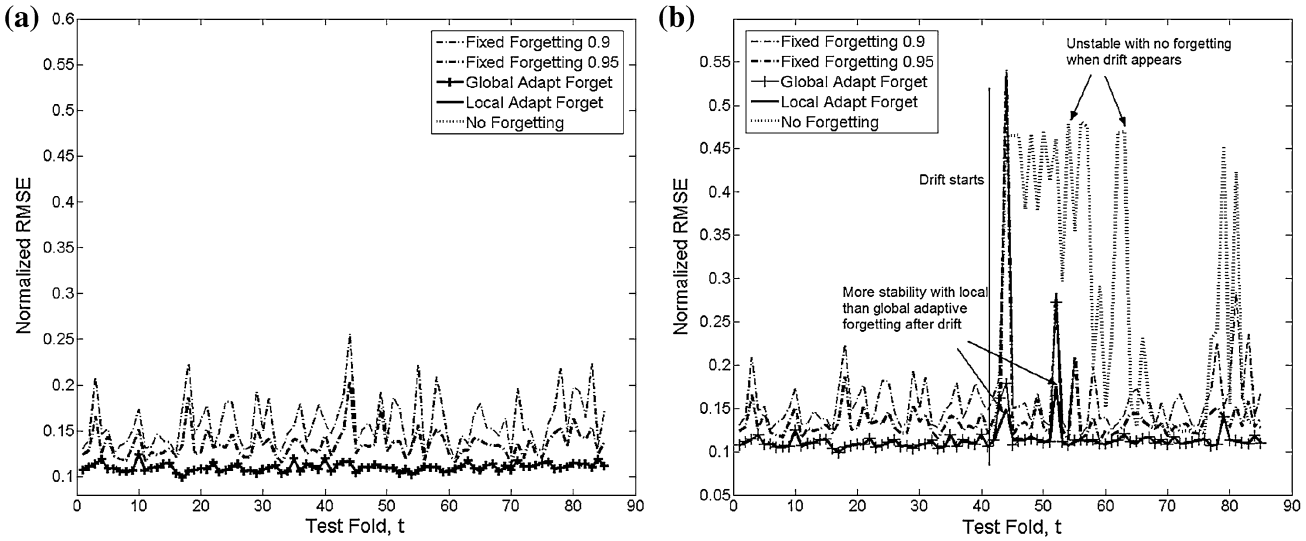

Fig. 9 **a** The evolution of RMSE over the number of test folds for YearPredictionMSD data without including any drifts: note the much worse performance of a fixed forgetting (0.9 and 0.95); **b** the evolution of RMSE for the same data set when including a drift at position $t = 41$: when including no forgetting in the learning, the predictive quality of the model significantly deteriorates for the remaining lifetime of the stream; local adaptive forgetting can outperform global one especially around data blocks 43, 44 as well 51, 52

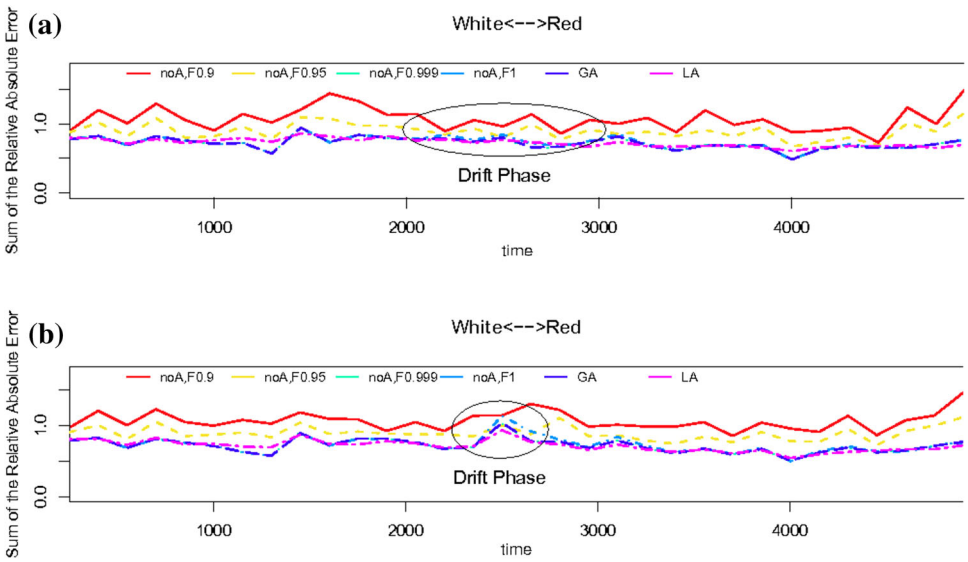

Fig. 10 **a** The error trends in case of a slow white wine to red wine change (drift) between Sample 2,000 and 3,000); **b** the same according to a fast change (drift) around Sample 2,500; *solid* and *dotted line* corresponds to fixed forgetting, *dashed* and *dashed dotted lines* belong to global and local adaptive forgetting (yielding lowest error in the drift phase around Sample 2,500 in **b**)

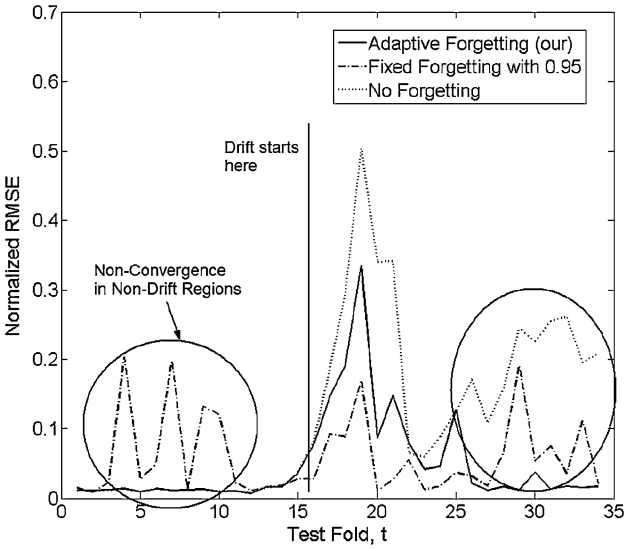

Fig. 11 Error trends on the rolling mill data including a real drift in the target concept

model error can be achieved, thus a drift can almost be prevented.

Results on the wine data set are following a similar pattern as the results on CT slices and year prediction. It is interesting to see that, when the drift is fast enough (Fig. 10b), there is a clear performance gain when using our adaptive methods, where local variant can again outperform slightly the global one (see the error trends within the indicated ellipsis in Fig. 10b). In case of a moderate drift changing slowly the data distribution from white wine to red wine, the resume is that conventional learning without the integration of forgetting achieves a similar model quality as all forgetting variants. This is not a big surprise as step-wise slight adaptation in model parameters and structures are then sufficient and still fast enough to track such a moderate, slow change. As in the previous two data sets, a fixed forgetting produces error trends which are usually above the error trends of our adaptive forgetting variants, also when no drift is present. This is simply due to the unnecessary forgetting effect, which decreases stability in usual life-long learning environments.

Finally, results on rolling mill data are visualized in Fig. 11. For this data set, global and local adaptive forgetting perform very similar, which is not a big surprise as the target concept was affected over the whole model definition space with the same magnitude (thus global model error already achieves an early trigger); thus we neglected the curve of local adaptive forgetting. Interestingly, what happens here is that a fixed forgetting factor of 0.95 can outperform our adaptive forgetting strategy in the drift region, i.e. its induced forgetting strength complies better with the drift intensity. However, (1) it was hand-tuned to achieve this result in multiple test runs, which is not possible in real on-line single-pass stream cases and (2) it overshoots in case when there is no drift at the beginning and at the end of the stream (indicated by ellipsoids). No forgetting does the job well at the beginning of the stream, but once a drift is introduced it causes a higher model error (almost double than adaptive forgetting) and afterwards is not able to converge to the drifted state (model error stays at a high level). Please note that fixed forgetting with 0.9 and 0.99 produced exactly the same results, indicating a quite low sensitivity with respect to the forgetting factor setting.

Finally, we performed a sensitivity analysis of the two most sensitive parameters: we therefore varied $\Lambda = 0.02 \pm 30\,\%$ and $\delta = 0.005 \pm 30\,\%$ and obtained exactly the same results as shown in figures above. When increasing the range of variation to around 40 %, still there are only marginal differences in the range of 1–2 % in terms of RMSE error trends over the whole streams.

9 Conclusion

The paper demonstrates a new methodology for handling global and local drifts in adaptive and smooth manner. Thus, a new extended version of the Page-Hinkley test is designed, which is able to quantify the change of model errors in terms of its magnitude and duration. This opens the possibility of designing a forgetting strategy which is adaptively changing its forgetting factor, according to the intensity of the drift. Furthermore, our method supports the option to react on drifts locally, which may out-perform global drift reaction subject to the fact that drifts may appear in local regions of the feature space (as could be verified for YearPredictionMSD data and wine data set, where almost drift prevention could be achieved). Smoothness is also guaranteed by using local model components with infinite support, thus always ensuring an overlap between neighboring regions, affected with different intensity. The results on four data streams also show that conventional life-long learning without forgetting is not a feasible option in case when drifts appear, the same is the case when using fixed forgetting values throughout the whole learning process. Finally, our approach can be in large parts easily adopted to other model architectures which implement local model components (especially Sects. 3, 4 and 6).

Future works include the investigation of appropriate drift handling strategies for case-based reasoning systems, where models on demand are re-created from data block to data block based on selected past exemplars. In this context, the drift quantification based on global indicators (such as RMSE) will remain the same, but the integration

and localization of the forgetting factors will be completely different: this can be achieved either in form of sample weights stored in the selected history buffer or by specific modification steps of the history buffer.

Acknowledgments This work was funded by the German Research Foundation (DFG) and the Austrian Science Fund (FWF, contract number I328-N23). The second author also acknowledges the support of the Austrian COMET-K2 programme of the Linz Center of Mechatronics (LCM), funded by the Austrian federal government and the federal state of Upper Austria. This publication reflects only the authors' views.

References

Angelov P (2010) Evolving Takagi–Sugeno fuzzy systems from streaming data, eTS+. In: Angelov P, Filev D, Kasabov N (eds) Evolving intelligent systems: methodology and applications. Wiley, New York, pp 21–50

Angelov P, Filev D (2004) An approach to online identification of Takagi–Sugeno fuzzy models. IEEE Trans Syst Man Cybernet Part B: Cybernet 34(1):484–498

Angelov P, Filev D, Kasabov N (2010) Evolving intelligent systems—methodology and applications. Wiley, New York

Angelov P, Lughofer E, Zhou X (2008) Evolving fuzzy classifiers using different model architectures. Fuzzy Sets Syst 159(23):3160–3182

Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: massive online analysis. J Mach Learn Res 11:1601–1604

Bouchachia A (2011) Evolving clustering: an asset for evolving systems. IEEE SMC Newsl 36. http://www.my-smc.org/news/back/2011_09/main_article3.html

Bouchachia A, Vanaret C (2011) Incremental learning based on growing gaussian mixture models. In: Proceedings of 10th International Conference on machine learning and applications (ICMLA 2011), p to appear. Honululu, Haweii

Cernuda C, Lughofer E, Maerzinger W, Kasberger J (2011) NIR-based quantification of process parameters in polyetheracrylat (PEA) production using flexible non-linear fuzzy systems. Chemom Intell Lab Syst 109(1):22–33

Cortez P, Cerdeira A, Almeida F, Matos T, Reis J (2009) Modeling wine preferences by data mining from physicochemical properties. Decis Support Syst 47(4):547–553

Delany SJ, Cunningham P, Tsymbal A, Coyle L (2005) A case-based technique for tracking concept drift in spam filtering. Knowl Based Syst 18(4–5):187–195

Diehl C, Cauwenberghs G (2003) SVM incremental learning, adaptation and optimization. In: Proceedings of the International Joint Conference on neural networks, vol 4. Boston, pp 2685–2690

Dovzan D, Skrjanc I (2011) Recursive clustering based on a Gustafson–Kessel algorithm. Evol Syst 2(1):15–24

French RM (1999) Catastrophic forgetting in connectionist networks. Trends Cogn Sci 3(4):128–135

Gama J (2010) Knowledge discovery from data streams. Chapman & Hall/CRC, Boca Raton

Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. In: Lecture notes in computer science, vol 3171. Springer, Berlin Heidelberg, , pp 286–295

Gama J, Rodrigues P, Sebastiao R (2009) Evaluating algorithms that learn from data streams. In: SAC '09 Proceedings of the 2009 ACM symposium on applied computing. ACM, New York, pp 1496–1500

Gama J, Sebastião R, Rodrigues P (2013) On evaluating stream learning algorithms. Mach Learn 90(3):317–346

Groißböck W, Lughofer E, Klement E (2004) A comparison of variable selection methods with the main focus on orthogonalization. In: Lopéz-Díaz M, Gil M, Grzegorzewski P, Hryniewicz O, Lawry J (eds) Soft methodology and random information systems, advances in soft computing. Springer, Berlin, Heidelberg, New York, pp 479–486

Hamker F (2001) RBF learning in a non-stationary environment: the stability–plasticity dilemma. In: Howlett R, Jain L (eds) Radial basis function networks 1: recent developments in theory and applications. Physica Verlag, Heidelberg, New York, pp 219–251

Hisada M, Ozawa S, Zhang K, Kasabov N (2010) Incremental linear discriminant analysis for evolving feature spaces in multitask pattern recognition problems. Evol Syst 1(1):17–27

Ikonomovska E, Gama J, Sebastiao R, Gjorgjevik D (2009) Regression trees from data streams with drift detection. In: v. Lecture Notes in Computer Science (ed) Discovery science. Springer, Berlin, Heidelberg, pp 121–135

Kalhor A, Araabi B, Lucas C (2010) An online predictor model as adaptive habitually linear and transiently nonlinear model. Evol Syst 1(1):29–41

Kasabov N (2007) Evolving connectionist systems: the knowledge engineering approach, 2nd edn. Springer, London

Klement E, Mesiar R, Pap E (2000) Triangular norms. Kluwer Academic Publishers, Dordrecht, Norwell, New York, London

Klinkenberg R (2004) Learning drifting concepts: example selection vs. example weighting. Intell Data Anal 8(3):281–300

Kullback S, Leibler R (1951) On information and sufficiency. Ann Math Stat 22(1):79–86

Kurlej B, Wozniak M (2011) Learning curve in concept drift while using active learning paradigm. In: Bouchachia A (ed) ICAIS 2011, LNAI 6943. Springer, Berlin, Heidelberg, pp 98–106

Lindstrom P, Namee B, Delany S (2013) Drift detection using uncertainty distribution divergence. Evol Syst 4(1):13–25

Lughofer E (2005) Aspects of incremental rule consequent learning. Technical report FLLL-TR-0502. Fuzzy logic laboratorium Linz-Hagenberg, A-4232 Hagenberg, Austria

Lughofer E (2008) FLEXFIS: a robust incremental learning approach for evolving TS fuzzy models. IEEE Trans Fuzzy Syst 16(6):1393–1410

Lughofer E (2011) Evolving fuzzy systems—methodologies. Advanced concepts and applications. Springer, Berlin, Heidelberg

Lughofer E (2012) Flexible evolving fuzzy inference systems from data streams (FLEXFIS++). In: Sayed-Mouchaweh M, Lughofer E (eds) Learning in non-stationary environments: methods and applications. Springer, New York, pp 205–246

Lughofer E (2012) Single-pass active learning with conflict and ignorance. Evol Syst 3(4):251–271

Lughofer E (2013) On-line assurance of interpretability criteria in evolving fuzzy systems—achievements, new concepts and open issues. Inf Sci 251:22–46

Lughofer E, Angelov P (2011) Handling drifts and shifts in on-line data streams with evolving fuzzy systems. Appl Soft Comput 11(2):2057–2068

Lughofer E, Bouchot JL, Shaker A (2011) On-line elimination of local redundancies in evolving fuzzy systems. Evol Syst 2(3):165–187

Mahalanobis PC (1936) On the generalised distance in statistics. Proc Natl Inst Sci India 2(1):49–55

Moe-Helgesen OM, Stranden H (2005) Catastophic forgetting in neural networks. Technical report, Norwegian University of Science and Technology, Trondheim

Mouss H, Mouss D, Mouss N, Sefouhi L (2004) Test of Page-Hinkley, an approach for fault detection in an agro-alimentary production system. Proc Asian Control Conf 2:815–818

Pratama M, Anavatti S, Lughofer E (2014) GENFIS: towards and effective localist network. IEEE Trans Fuzzy Syst. doi:10.1109/TFUZZ.2013.2264938

Qin S, Li W, Yue H (2000) Recursive PCA for adaptive process monitoring. J Process Control 10(5):471–486

Ramamurthy S, Bhatnagar R (2007) Tracking recurrent concept drift in streaming data using ensemble classifiers. In: Proceedings of the Sixth International Conference on machine learning and applications (ICMLA). Cincinnati, Ohio, pp 404–409

Raquel Sebastião Margarida M, Silva JG, Mendonça T (2011) Contributions to an advisory system for changes detection in depth of anesthesia signals. In: LEMEDS11: Proceedings of the Learning from medical data streams. Bled, Slovenia

Sayed-Mouchaweh M, Lughofer E (2012) Learning in non-stationary environments: methods and applications. Springer, New York

Sebastiao R, Silva M, Rabico R, Gama J, Mendonca T (2013) Real-time algorithm for changes detection in depth of anesthesia signals. Evol Syst 4(1):3–12

Serdio F, Lughofer E, Pichler K, Buchegger T, Efendic H (2014) Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills. Inf Sci 259:304–320

Shaker A, Hüllermeier E (2012) Instance-based classification and regression on data streams. In: Lughofer E, Sayed-Mouchaweh M (eds) Learning in non-stationary environments: methods and applications. Springer, New York, pp 185–201

Shaker A, Senge R, Hüllermeier E (2013) Evolving fuzzy patterns trees for binary classification on data streams. Inf Sci 220:34–45

Shilton A, Palaniswami M, Ralph D, Tsoi AC (2005) Incremental training of support vector machines. IEEE Trans Neural Netw 16(1):114–131

Soleimani H, Lucas K, Araabi B (2010) Recursive Gath–Geva clustering as a basis for evolving neuro-fuzzy modeling. Evol Syst 1(1):59–71

Song M, Wang H (2005) Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In: Priddy KL (ed) Intelligent computing: theory and applications III, Proceedings of the SPIE, vol 5803. pp 174–183

Takagi T, Sugeno M (1985) Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Syst Man Cybernet 15(1):116–132

Tsymbal A (2004) The problem of concept drift: definitions and related work. Technical report TCD-CS-2004-15, Department of Computer Science, Trinity College Dublin, Ireland

Utgoff P, Berkman NC, Clouse JA (1997) Decision tree induction based on efficient tree restructuring. Mach Learn 29(1):5–44

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