**Noname manuscript No.**
(will be inserted by the editor)

# On-line Elimination of Local Redundancies in Evolving Fuzzy Systems

**Edwin Lughofer · Jean-Luc Bouchot · Ammar Shaker**

Received: 04.02.2011, Revised: 20.04.2011

**Abstract** In this paper, we examine approaches for reducing the complexity of evolving fuzzy systems (EFS) by eliminating local redundancies during training, evolving the models on on-line data streams. Thus, the complexity reduction steps should support fast incremental single-pass processing steps. In evolving fuzzy systems, such reduction steps are important due to several reasons: 1.) originally distinct rules representing distinct local regions in the input/output data space may move together over time and get significantly over-lapping as data samples are filling up the gaps in-between these, 2.) two or several fuzzy sets in the fuzzy partitions may become redundant because of projecting high-dimensional clusters onto the single axes, 3.) they can be also seen as a first step towards a better readability and interpretability of fuzzy systems, as unnecessary information is discarded and the models are made more transparent. One technique is performing a new rule merging approach directly in the product cluster space using a novel concept for calculating the similarity degree between an updated rule and the remaining ones. Inconsistent rules elicited by comparing the similarity of two redundant rule antecedent parts with the similarity of their consequents are specifically handled in the merging procedure. The second one is operating directly in the fuzzy partition space, where redundant fuzzy sets are merged based on their joint $\alpha$-cut levels. Redundancy is measured by a novel kernel-based similarity measure. The complexity reduction approaches are evaluated based on high-dimensional noisy real-world measurements and an artificially generated data stream containing 1.2 million samples. Based on this empirical comparison, it will be shown that the novel techniques are 1.) fast enough in order to cope with on-line demands and 2.) produce fuzzy systems with less structural components while at the same time achieving accuracies similar to EFS not integrating any reduction steps.

**Keywords**

Edwin Lughofer (corresponding author) and Jean-Luc Bouchot are with the Department of Knowledge-based Mathematical Systems, Johannes Kepler University of Linz, Austria, email: edwin.lughofer@jku.at
Ammar Shaker is with the Department of Mathematics and Computer Science, Philipps-Universität Marburg, Germany

evolving fuzzy systems, redundancy elimination, similarity, complexity reduction, on-line rule and fuzzy set merging

## 1 Introduction

### 1.1 Motivation

Nowadays, in industrial systems, there is an increasing demand of automatic model updates as new upcoming operating conditions, system behaviors [19], new types of classes [38] or even drift occurrences [31] may arise during on-line processes. These situations should be included into the models in order to guarantee robust and process-save operations [26]. As re-training phases based on all samples seen so far usually do not terminate in real-time, incremental learning is the engine for most on-line modeling scenarios, usually conducted in a single-pass manner. During incremental learning phase, the models should update their parameters, evolve their structure, extend their definition space in order to account for new operating conditions, systems states/behaviors, range extensions of various measurement variables etc. The concept of evolving data-driven techniques in connection with the aspect of modeling uncertainties and imprecise situations in a possibilistic manner (measurements, samples are often affected by noise) lead to the development of *evolving fuzzy systems* approaches during the last decade [30]. Examples are the pioneering *DENFIS* (*Dynamic Evolving Neural Fuzzy-Inference System*) approach [20] based on neuro-fuzzy system architecture and a recursive clustering method and *eTS* (*evolving Takagi-Sugeno fuzzy systems*) recursively updating the structure of the model based on the potentials of new incoming data and introducing several enhanced concepts such rules ages, utility function and zone of influence, summarized in the extended *eTS+* approach [2]. Further approaches include *ePL* (*evolving Participatory Learning*) [24] extending the *eTS* approach by using Yager's participatory learning concept [52] or *SAFIS* (*Sequential Adaptive Fuzzy Inference Systems*) [42]. The latter uses the concept of 'influence' of a fuzzy rule to add and remove rules during learning, whereas the influence of a fuzzy rule is defined as its contribution to the system output in the statistical sense.

During the incremental learning process, the situation may happen that redundancies either on rule level or on fuzzy set level arise. For instance, consider rules which may move together when originally distinct data clouds, captured in different clusters (rules), are filled up with new incoming data; in this case, the rules may even get significantly overlapping: an example will be shown in Section 3.1. Whenever an on-line projection of evolved/updated clusters from the high-dimensional space onto the single axes is carried out (as is the case for most of the EFS approaches mentioned above) or movements of fuzzy sets take place according to changes in the local sample distributions, the fuzzy sets may get significantly over-lapping in single fuzzy partitions: an example for such an occurrence will be given in Section 3.2. In this sense, we see it as a big challenge to detect and remove such upcoming redundancies automatically and in an incremental on-line manner. This also can be seen as a step for reducing unnecessary complexity and furthermore as a step towards more transparency of the evolved fuzzy models. It is an essential issue, as fuzzy systems are claimed to provide better interpretability than other data-driven model architectures, which therefore often serves a central reason why this architecture is chosen in modeling applications. A pre-requisite of a good interpretability is a nice transparency which comprehends non-redundant information and consistency in the rule base.

## 1.2 Related Work and Our Approach

In the field of batch off-line trained fuzzy systems, there exist several methods which perform complexity reduction steps by removing redundancies. Most of these are based on similarity measures applied onto pairs of fuzzy sets and/or rules. For instance, in [45], extended in [44], and also in [6] the Jaccard index is exploited for measuring the similarity degree. Similar fuzzy sets are merged in a post-processing manner after the complete learning phase using a convex combination of parameters in fuzzy sets with finite support. This assures that the support of the merged fuzzy sets is the same as the joined one of the old fuzzy sets. In [14] two fuzzy sets are merged which have small widths and close modal values. In [11], Gaussian membership functions are approximated by trapezoidal ones and a fast similarity measure is proposed based on this function type. In [34], rule pruning is conducted by deleting partial premises from the rules (according to features which are not really important) within a hill climbing procedure. Orthogonal least squares learning in batch mode as performed in [12] [51] also contributes to a pruning of rules as rules with an important contribution w.r.t. explaining the variance of the output are selected. In [9] [18], redundancies are prevented in an a priori manner by using constrained-based optimization algorithms which do not allow an overlap of fuzzy sets at membership degrees larger than 0.5. Specific semantic constraints for membership function optimization are used in [37]. Some of these approaches and beyond are summarized in [10]. All these optimization and post-processing procedures require significant computation time and therefore are not applicable in an incremental learning context. Furthermore, incremental learning solutions to the constrained-based optimization problems were not investigated so far. In a machine learning context, rule pruning was studied under the umbrella of inducing decision trees from data [40] [8]: there, as decision trees are providing a hierarchical structure, rule pruning is conducted on the basis of premise parts, i.e. step-wise reducing the sizes of the premises of the rules by eliminating conditions (including a subset of features) which are not really important. A widely used criterion whether branches (premise parts) in trees can be pruned is an internal cross-validation step [48], which is per se a time consuming off-line method.

In evolving fuzzy systems, the main focus so far was placed on precise modeling tasks: the aim is to evolve fuzzy systems in a data streaming context as accurately as possible. Only little attention was paid to complexity reduction, especially in the context to remove local redundancies which may come up during the incremental learning process. The approach in [41] uses a geometric similarity measure for detecting redundant fuzzy sets and a weighted average for merging the parameters of redundant fuzzy sets and of rule consequents. Removing obsolete rules, i.e. rules with extraordinary low support by past samples (but not necessarily redundant ones), was presented in [3]; this is extended in [2] for automatically eliminating unnecessary rules by using the concepts of rule age, which is getting higher than usual in case when rules are outdated, and rule utility, which is getting lower when rules are not used. In [43] [42], rule pruning is implicitly integrated in the incremental learning algorithm: rules are pruned whenever the statistical influence of a fuzzy rule measured in terms of its relative membership activation level (relative to the activation levels of all rules so far) is getting lower than a pre-defined threshold. In [24], redundant rules are detected by calculating the sum of the absolute deviations between the normalized coordinates of two rule (cluster) centers. Another approach for removing redundant fuzzy sets and fuzzy sets with close model values is demonstrated in [32], where Gaussian fuzzy sets are merged by reducing the approximate merging of two Gaussian kernels to the exact merging of two of their $\alpha$-cuts. In this paper, we build upon this approach and on the approach in [41] and extend these significantly by the following aspects:

1. Redundancy of rules is measured in the multi-dimensional feature space according to a similarity criterion defined for the single dimensions and aggregated over these. This is achieved by exploiting virtual projections onto fuzzy set level and using the membership degrees of intersection points between two membership functions in 1-dimensional spaces. This does not suffer from curse of dimensionality and is directly comparable and summable among all dimensions. The aggregation is conducted by using the minimum t-norm, thus a high overall redundancy degree is only achieved when in all dimensions the similarity (overlap) is significant. This yields a stronger condition than when using a (distance-based) similarity directly in a high-dimensional feature space, as the latter may allow significant distances in one, two, three single dimensions, which would contribute to the overall distance in a high-dimensional space only very slightly. Furthermore, our measure is in accordance to the viewpoint of a fuzzy system: antecedent of two rules are already inspected as different when two parts differ significantly, i.e. the two fuzzy sets contained in the two parts do have very little overlap.
2. The merging process of two redundant multi-dimensional rules, represented as ellipsoidal clusters in the feature space, is performed by new criteria for the cluster merging procedure, which are appropriate for EFS. These take the significance of clusters into account and using an expanded form of the (recursive) variance update formula for achieving the range of influence of the new (merged) cluster (rule).
3. Introducing a novel kernel-based similarity measure for Gaussian fuzzy sets, out-performing the Jaccard index significantly in terms of computation power. This will be verified in the empirical tests.
4. A new concept for a consistent treatment of the consequent functions of two rules whose antecedents are similar, i.e. redundant: merging by a weighted average of the linear parameters is only allowed in case of non-contradictory rules, otherwise the less significant rule is deleted. We defined the concept of contradictory rules as special case when the consequent parts are more dissimilar than the antecedent parts; hereby, a new similarity measure of consequents in case of TS fuzzy systems is introduced.

The paper is organized as follows: the next section deals with basic concepts in evolving fuzzy systems (EFS) including a small description of the *FLEXFIS* approach, as this will be used as incremental/evolving learning engine for the purpose to evaluate all redundancy detection and elimination approaches demonstrated in this paper. Section 3 contains two parts: the first part is dedicated to detecting and eliminating upcoming redundancies in the high-dimensional feature space, the second part dedicated to do the same on fuzzy partition level. Section 4 contains an evaluation of the novel approaches based on two high-dimensional noisy real world data sets (dynamic data set for predicting NOx emission and static data for predicting house prices) and on an artificial data streams containing 1 million samples.

## 2 Evolving Fuzzy Systems for Regression Problems

### 2.1 Basic Aspects

Nowadays, the most commonly used model architecture in evolving fuzzy systems approaches such as *eTS*, *ePL*, *DENFIS* or *SOFNN*, is the Takagi-Sugeno fuzzy systems architecture [49] in connection with Gaussian membership functions and product t-norm operator:

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

with the normalized membership functions

$$\Psi_i(\mathbf{x}) = \frac{e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{ij})^2}{\sigma_{ij}^2}}}{\sum_{k=1}^{C} e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j-c_{kj})^2}{\sigma_{kj}^2}}} \tag{2}$$

and consequent functions

$$l_i = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + ... + w_{ip}x_p \tag{3}$$

The symbol $x_j$ denotes the $j$-th input variable (static or dynamically time-delayed), $c_{ij}$ the center and $\sigma_{ij}$ the width of the Gaussian fuzzy set in the $j$-th premise part of the $i$-th rule. As conjunction operator, the product t-norm is applied [21]. The Takagi-Sugeno fuzzy model is able to provide accurate estimates, according to the universal approximation capabilities [50] and is therefore often used for precise modeling tasks. Their consequent parts are containing hyper-planes which are not fully transparent in a linguistic sense. On the other hand, their antecedent parts can be inspected as linguistic rules, as linguistic terms can be assigned to the membership functions. Thus, they provide a reasonable tradeoff between accuracy and interpretability [10] and hence are often used as common model architecture for data-driven modeling tasks [5].

In this paper, we want to reduce the *unnecessary* complexity of the evolving fuzzy models by eliminating redundancies, which can be seen as a step towards an enhanced readability. Most of these techniques are applicable to all of the evolving fuzzy systems approaches mentioned in the state of the art section above. Nevertheless, in the next section we provide a small introduction on one concrete EFS approach, the so-called *FLEXFIS* approach (*FLEXible Fuzzy Inference Systems*), as being used as evaluation engine for the novel redundancy elimination techniques in the evaluation section.

Also, we want to point out that in most of the EFS approaches, equivalent Mamdani systems, offering some more linguistic interpretability, can be extracted from the evolved cluster/rule models by projecting the high-dimensional clusters/rules also onto the output/target variable, instead of applying the recursive (weighted) least squares estimator for consequent parameter adaptation. In the results section, we will show the accuracy loss when doing so (compared to evolved TS fuzzy systems), also in connection with the complexity reduction techniques discussed in this paper.

## 2.2 The (Conventional) FLEXFIS Approach as Learning Engine

The *FLEXFIS* approach applies three stages in each incremental learning step for training TS fuzzy systems from data streams:

– In the first stage, a new sample is sent through an incremental and evolving clustering algorithm called the *eVQ* approach (*evolving Vector Quantization*) [27]: a new sample is checked whether it fits to the already obtained cluster partition and if this is so, the nearest cluster is moved towards the current sample (plasticity) by a fraction which is monotonically decreasing with the number of samples forming this cluster, i.e. for which the cluster was the nearest one in the past. If it does not fit into the cluster partition, a new rule is evolved (stability). A vigilance parameter measuring the compatibility with the current cluster structure is responsible for controlling the stability-plasticity dilemma (evolving new rules versus updating existing ones) [1].

– In the second stage, a new sample is sent through a recursive fuzzily weighted least squares estimator (RFWLS) used in a local learning context for each rule separately, where the weights to the single rules are obtained by the membership degrees to the corresponding rules. The concrete formulas are given by:

$$\hat{\mathbf{w}}_i(k+1) = \hat{\mathbf{w}}_i(k) + \gamma(k)(y(k+1) - \mathbf{r}^T(k+1)\hat{\mathbf{w}}_i(k)) \tag{4}$$

$$\gamma(k) = \frac{P_i(k)\mathbf{r}(k+1)}{\frac{1}{\Psi_i(\mathbf{x}(k+1))} + \mathbf{r}^T(k+1)P_i(k)\mathbf{r}(k+1)} \tag{5}$$

$$P_i(k+1) = (I - \gamma(k)\mathbf{r}^T(k+1))P_i(k) \tag{6}$$

with $\Psi_i(\mathbf{x}(k+1))$ the normalized membership function value for the $(k+1)$th data sample, $P_i(k)$ the weighted inverse Hessian matrix and $\mathbf{r}(k+1) = [1 \;\; x_1(k+1) \;\; x_2(k+1) \;\; \ldots \;\; x_p(k+1)]^T$ the regressor values of the $(k+1)$th data point, which is the same for all $i$ rules. Therefore, the linear parameters in the previous $k$th step, $\hat{\mathbf{w}}_i(k)$ are updated to a new vector stored into $\hat{\mathbf{w}}_i(k+1)$. This is done based on the correction vector $\gamma$ controlling the amount of correction in dependency of the activation level of the $i$th rule, which is applied to the one-step ahead prediction error (represented by the braces in the second term in (4)). Note that this is a recursive approach meaning that the parameters converge to the real batch solutions in each incremental learning step. This is because the optimization function is a hyper-parabola and (4) to (6) perform one Newton step as long as there is no structural change during the incremental update. In fact, in case a new rule is evolved in the previous step, no structural change takes place, but a new recursive weighted least squares estimator is opened up, which has the effect that it is not 'disturbing' the convergence of the others.

– When an old cluster is updated, a correction vector and a correction matrix is added to the parameters and inverse Hessian matrix in order to balance out the non-optimal situation of the RFWLS estimator [28], as a structural change took place.

The conventional basic *FLEXFIS* algorithm can be briefly summarized by the pseudo code demonstrated in Algorithm 1, see [28] for a more detailed formulation.

**Algorithm 1 FLEXFIS+ (FLEXible Fuzzy Inference Systems from Data Streams)**

```
1a Perform an initial training in batch mode on pre-collected training samples or first on-line samples and obtain the optimal value for the vigilance parameter $\rho$ (e.g. in a grid search scenario coupled with 10-fold cross-validation [48]); estimate the (initial) ranges of all variables.
1b Alternatively, in case when no off-line recordings are available, incremental learning from scratch can be carried out, by estimating the ranges of all variables from some first on-line data stream samples.
2 For each new incoming data sample $\mathbf{x}$ do the following steps
3 Normalize $\mathbf{x}$ to $[0, 1]$ and the cluster centers according to the current feature ranges.
4 If $\|\mathbf{x} - \mathbf{c}_{win}\| \geq \rho$ with $\mathbf{c}_{win}$ the center coordinates of the nearest cluster, then evolve a new rule by increasing the number of rules $C = C + 1$ and setting its center to the current data sample $\mathbf{c}_C = \mathbf{x}$ and its range of influence in each direction by $\boldsymbol{\sigma}_C = 0$.
5 Else Update the center of the nearest cluster $\mathbf{c}_{win}$ by moving it towards the current sample and update its range of influence by recursive variance formula [39], see also (10).
6 Transfer the clusters back to original feature space, according to the ranges of the features. This is for the purpose to obtain better interpretability of the fuzzy systems, as original ranges of system variables are usually better understood by operators and experts.
7 Project modified/evolved cluster to the axes in order to update/evolve the fuzzy set partition in each dimension: the centers of the fuzzy sets are associated with the corresponding center coordinates of the clusters, the widths of the fuzzy sets are set to $\max(\boldsymbol{\sigma}_., \epsilon)$ with $\epsilon$ a small positive constant, in order to avoid numerical instabilities. Each cluster is associated with one rule where the single projected fuzzy sets are conjuncted by a product t-norm forming its antecedent part.
8 Add correction vectors to the linear consequent parameters and correction matrices to the inverse Hessian matrices estimated in the previous step.
9 Perform recursive fuzzily weighted least squares using (4) to (6) for all $C$ rules: updating the consequent of all rules instead of only the nearby lying one(s) increases significance of the parameters faster.
10 Update the ranges of all features.
```

Steps 1a) and 1b) can be seen as two possible alternatives at the initial phase of training: 1a.) requires some initial off-line recordings and is often preferred by experts or operators working at the system; this is because an initial model can be evaluated and verified upon its correctness (in form of accuracy tests, structural examinations, comprehension of the relations/dependencies inside the model etc.), also to get an impression whether the whole problem can be modeled at all (at least with a sufficient performance). A 10-fold cross-validation (CV) is used, which splits the data into ten equal folds and performs a model training for each combination of nine folds and use the remaining one to elicit the prediction error. The later is averaged over all folds and yields a good estimation of the expected prediction error on new samples, see also the funded analysis provided in [16]. This is repeated for different settings of the vigilance and that one achieving minimal CV error is used as optimal parameter setting for further updates. From the minimal CV error, someone may get a glance whether a model with reasonable accuracy is possible at all. On the other hand, Step 1b) provides more automatization power as the evolving fuzzy system can be used in a kind of plug-and-play manner, as being built up from scratch. This, however, prevents a tuning of training parameters. In the evaluation section, we will use this alternative with a default setting of the vigilance parameter and underline the usefulness of the presented rule and fuzzy set merging approach in this learning context.

Some enhancements for *FLEXFIS* algorithm in particular and for EFS approaches in general were proposed in [29], especially for ensuring more robustness and higher process safety.

## 3 On-Line Complexity Reduction in EFS based on Redundancy Criteria

This section deals with two approaches for reducing the complexity of evolving Takagi-Sugeno fuzzy systems as defined in (1) during incremental on-line adaptation based on local redundancy criteria. Hereby, we assume that rules can be directly associated with ellipsoidal clusters in the feature space. The first variant is directly applied in the cluster space and merges clusters which become strongly overlapping as samples are filling up the original gaps in-between these, enforcing strong movements of the two clusters and an effect which we call cluster/rule coalescence. The second variant acts on the fuzzy partition space and performs a fuzzy set merging process for each dimension separately. In this way, two or more rules may become redundant as well whenever all antecedent parts are getting the same due to the fuzzy set merging process. Both variants 1.) are able to cope with sample-wise single-pass modes, i.e. one new data sample from a data stream is sent into the update process of the models and immediately discarded, afterwards; and 2.) do not rely on the concrete incremental learning engine, but can be applied to any evolving fuzzy systems approach.

### 3.1 Rule Merging in the Feature Space

#### *3.1.1 Problematic Nature*

The rule coalescence nature gets evident when thinking of two local regions represented by two rules in a two-dimensional feature space, which seem to be originally distinct, but are moving together with more samples loaded. In the extreme case, they may get significantly overlapping over time when more and more samples

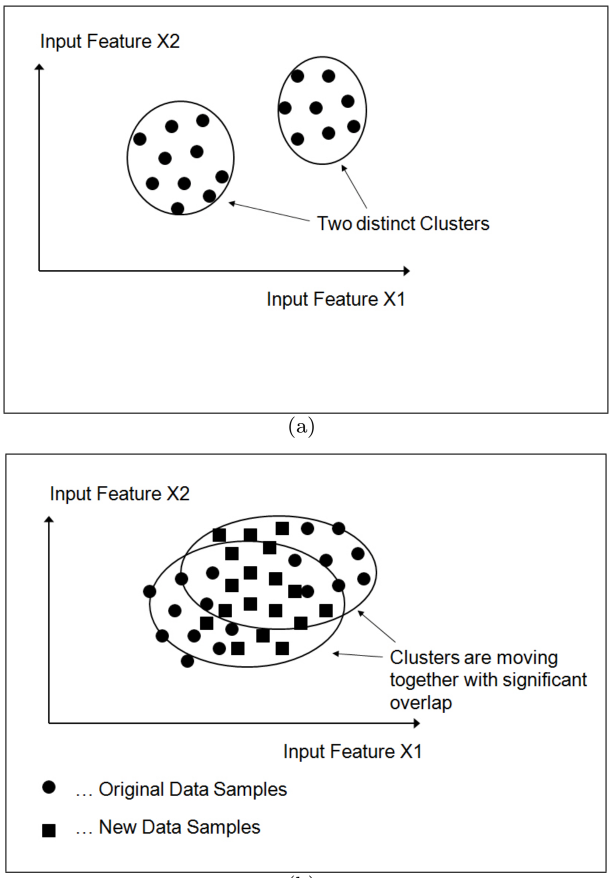


**Fig. 1** (a): two distinct clusters (rules) in the two-dimensional feature space, correctly extracted as indicated by ellipsoids; (b): significantly overlapping clusters (rules) as they are moving together based on new incoming samples filling up the gap in-between these

are passed through the incremental learning engine. An example of such an occurrence is visualized in Figure 1: obviously, two distinct clusters are present in the first bunch of the data set (circles in (a) indicate data samples, ellipsoids indicate clusters), while at a later stage the gap between these two clusters are filled up with new incoming samples, marked with rectangles — as shown in (b). Such an effect simply happens because of the nature of data streams and the incremental learning engine for updating/evolving the fuzzy systems applied in these: the engine receives only small snapshots of the data, in some cases only single samples, at one point of time and usually is not able to look into the future how the data stream may behave or develop. Hence, no a priori prevention of such effects as shown in Figure 1 is possible.

### *3.1.2 Calculating Similarity of Rules*

In this sense, we propose to apply a post-processing step after each incremental learning step, where we check whether the latest rule updates lead to significantly overlapping clusters. Hereby, we assume that the rule update steps in the cluster space are performed in a sample-wise fashion, which is supported by all conventional evolving fuzzy systems approaches. After the update of a cluster with a new sample, e.g. by moving its center towards this sample, by extending its range of influence or by resetting its center, we propose to calculate the similarity of this cluster to all the remaining ones — note that the most similar one is not necessarily the nearest one. Hereby, instead of calculating a time-intensive overlap degree of two clusters with ellipsoidal shape in the high-dimensional feature space using complex mathematical formulas, for instance see [25] [23], we simulate this by inspecting the overlap degree of the two rules dimension-wise and calculating an amalgamated value as similarity degree. The dimension-wise calculation is also justified by the fact that the antecedents of fuzzy rules are always represented by a conjunction of single dimension-wise membership functions. Therefore, our approach is in accordance of the view point of a fuzzy rule base: two rules are only similar (same) if all their antecedent parts are similar (same). In particular, we calculate the intersection points of the two Gaussians used as fuzzy sets in (2), which are belonging to the same dimension $j$ in the antecedent parts of a rule pair, i.e. for the modified rule $i$ and any other rule $k$ these are $e^{-\frac{1}{2}\frac{(x_j-c_{ij})^2}{\sigma_{ij}^2}}$ and $e^{-\frac{1}{2}\frac{(x_j-c_{kj})^2}{\sigma_{kj}^2}}$. In general, there are two intersection points whose $x$-coordinates are obtained by (proof is left to the reader):

$$\begin{aligned} inter_x(1) &= -\frac{c_{kj}\sigma_{ij}^2 - c_{ij}\sigma_{kj}^2}{\sigma_{kj}^2 - \sigma_{ij}^2} + \sqrt{(\frac{c_{kj}\sigma_{ij}^2 - c_{ij}\sigma_{kj}^2}{\sigma_{kj}^2 - \sigma_{ij}^2})^2 - \frac{c_{ij}^2\sigma_{kj}^2 - c_{kj}^2\sigma_{ij}^2}{\sigma_{kj}^2 - \sigma_{ij}^2}} \\ inter_x(2) &= -\frac{c_{kj}\sigma_{ij}^2 - c_{ij}\sigma_{kj}^2}{\sigma_{kj}^2 - \sigma_{ij}^2} - \sqrt{(\frac{c_{kj}\sigma_{ij}^2 - c_{ij}\sigma_{kj}^2}{\sigma_{kj}^2 - \sigma_{ij}^2})^2 - \frac{c_{ij}^2\sigma_{kj}^2 - c_{kj}^2\sigma_{ij}^2}{\sigma_{kj}^2 - \sigma_{ij}^2}} \end{aligned} \quad (7)$$

The maximal membership degree of the two Gaussian membership functions in the intersection coordinates is then used as overlap degree of the corresponding rules' antecedent parts in the $j$th dimension:

$$overlap_{ik}(j) = \max(inter_x(1), inter_x(2)) \quad (8)$$

Note that in the case when there is one Gaussian fuzzy set fully embedded by another one, there is only one intersection point whose membership degree is 1 $\rightarrow$ overlap degree is also 1. Figure 2 presents different occasions of overlaps between two Gaussian membership functions, the intersection points indicated as such by big dark dots; also note the intersection points at far right or left in the figures with very low membership values, which are ignored by using maximum operation in (8).

The amalgamation over all rule antecedent parts leads to the final overlap degree between the modified rule $i$ and any other rule $k$:

$$overlap_{ik} = Agg_{j=1}^{p} overlap_{ik}(j) \quad (9)$$

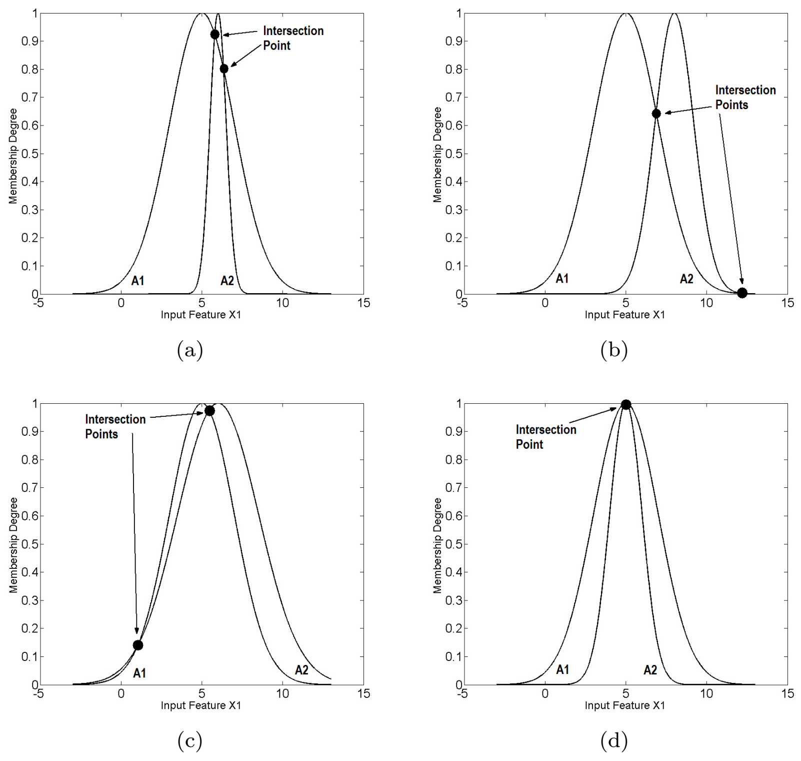


**Fig. 2** (a): two overlapping fuzzy sets, where one is significantly smaller than the other and therefore almost completely covered → high overlapping degree; (b): two overlapping fuzzy sets with similar width and significantly moved centers → medium overlapping degree; (c) two overlapping fuzzy sets with similar width and near centers → very high overlap degree; (d) one fuzzy set completely embedded in another one but having smaller width → only one intersection point with maximal membership value of 1

where *Agg* denotes an aggregation operator. A feasible choice is a t-norm [21], as a strong non-overlap along one single dimension is sufficient that the clusters do not overlap at all. For instance, consider a three dimensional clustering example as shown in Figure 3: (a) shows the projection of the two three-dimensional clusters onto the y-axis, where a strong overlap of these two clusters can be recognized. However, from the three-dimensional point of view as shown in (b), we can realize that the clusters do not overlap at all: the third dimension is responsible for tearing the clusters significantly apart from each other. In this case, using the minimum as strongest t-norm, the minimal overlap degree among all dimensions is very low (using weaker t-norms such as product would even decrease the overlap). On the other hand, if the minimal overlap degree among all dimensions is still high, we can reliably assume that there is a strong overall overlap of clusters.

When fuzzy sets are completely embedded, as shown in Figure 2 (d), or nearly embedded, as shown in Figure 2 (a), along all dimensions delivering a value for (9) of 1 or near 1, there are two different possibilities for this extreme cases:

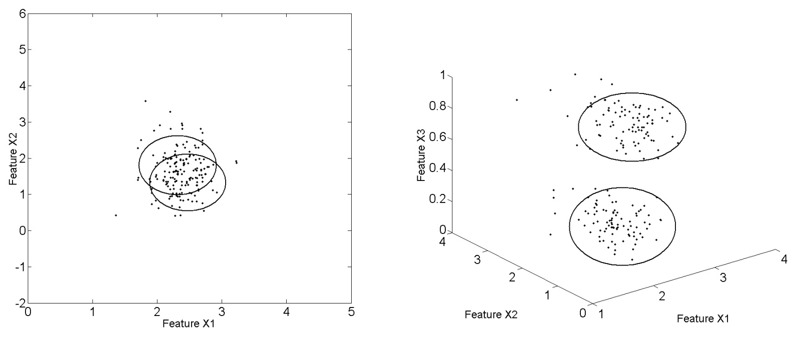


**Fig. 3** (a): two over-lapping clusters in the projected two-dimensional feature space (b): clusters significantly torn apart in the three dimensional feature space, as no overlap along the third dimension (Feature X3) is present

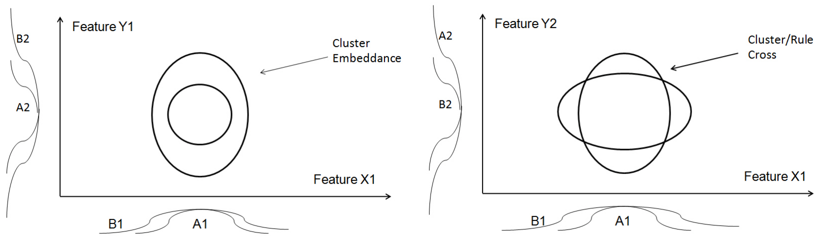


**Fig. 4** (a): one cluster is fully embedded in another cluster $\rightarrow$ overlap degree is equal to 1 as the intersection of the fuzzy sets is at membership degree 1 in both dimensions (b): cluster cross, reverse embedded along both dimensions $\rightarrow$ overlap degree is equal to 1

– One cluster is fully embedded in the other.
– The clusters are forming a cross in the feature space.

The first occasion is shown in Figure 4 (a), the second in Figure 4 (b). Apart from the fact that such cluster/rule occurrences usually do not happen when using a reasonable evolving fuzzy systems approach (at least none of the above referenced approaches would evolve such rules), in both cases the rules should be merged, therefore an overlap value of 1 or near 1 is appropriate. Also, long drawn-out cluster crosses should be resolved, as these are causing unpleasant crossed local hyperplanes and strange shape/behavior in the approximation/regression function (and we are assuming that we are not dealing with switching regression models [17]).

### *3.1.3 Merging Process*

If the overlap calculated by (9) is greater than a threshold $sim_{thr}$ (we used a default value of 0.8 in all our tests), a merging of the two clusters $i$ and $k$ is performed.

A reasonable merging of two clusters centers can be carried out by taking their weighted average, whereas the weights are represented by the significance of the clusters, i.e. the number of samples $k_i$ and $k_k$ supporting the two clusters — note that these values can be simply updated by counting. A reasonable merging of the ranges of influence of the two clusters is achieved by updating the range of influence in each direction of the more significant cluster with the range of influence of the less significant cluster by exploiting the recursive variance formula [39], which is defined for a new incoming sample $\mathbf{x}(N+1)$ by:

$$\sigma^2(new) = \frac{1}{N+1}(N\sigma^2(old) + (N+1)\Delta\overline{X}(N+1)^2 + (\overline{X}(N+1) - \mathbf{x}(N+1))^2) \quad (10)$$

with $\overline{X}(N+1)$ the updated mean value of all the variables over the last $N+1$ samples, $\Delta\overline{X}(N+1)$ the deviation between the mean values before and after its update and $\sigma^2(old)$ the old variances of all variables. In our case, the less significant cluster represents a whole collection of points rather than a single sample with which the update is carried out. This means that in the last term in (10) the sample in the $N+1$th time instant $\mathbf{x}(N+1)$ is substituted by the center of the less significant cluster, as representing the mean of the 'new' data, i.e. the data which formed this cluster; the mean value $\bar{X}(N+1)$ is represented by the cluster center of the more significant cluster. The new number of samples is determined by adding the number of samples falling into both clusters, i.e. by $k_i + k_k$, which substitutes $N+1$ in (10). In order to guarantee a good coverage of the joint data cloud by the merged cluster, a fraction of the variance of samples belonging to the less significant cluster is added, which is determined by the percentage of samples belonging to the less significant cluster with respect to the samples belonging to both clusters. Hence, the cluster merge of two clusters $i$ and $k$ is done in the following way, for each $j = 1, ..., p$ separately and achieving a new cluster/rule indicated by index *new*:

$$\begin{aligned} c_{new,j} &= \frac{c_{i,j}k_i + c_{k,j}k_k}{k_i + k_k} \\ \sigma_{new,j} &= \sqrt{\frac{k_{cl_1}\sigma^2_{cl_1,j}}{k_{cl_1} + k_{cl_2}} + (c_{cl_1,j} - c_{new,j})^2 + \frac{(c_{new,j} - c_{cl_2,j})^2}{k_{cl_1} + k_{cl_2}}} \\ &+ \frac{k_{cl_2}}{k_{cl_1} + k_{cl_2}}\sigma_{cl_2,j} \\ k_{new} &= k_i + k_k \end{aligned} \quad (11)$$

where $cl_1 = \arg\max(k_i, k_k)$ denoting the (index of the) more significant cluster, and consequently $cl_2 = \arg\min(k_i, k_k)$ denoting the (index of the) less significant cluster.

Figure 5 demonstrates two examples where two clusters are merged because of a high and medium overlap degree:

- in (a) the originally extracted clusters based on the first 100 samples are shown in dotted lines; they are quite distinct at this stage, however due to new data, indicated by samples marked as pluses and filling up the gap inbetween the two clusters, the clusters are moving together, finally achieving a significant overlap: similarity is then 0.88. Merging of the two clusters leads to the big cluster, its range of influence, i.e. the 2-$\sigma$ range, shown as thick solid line and its center shown as big dark dot.

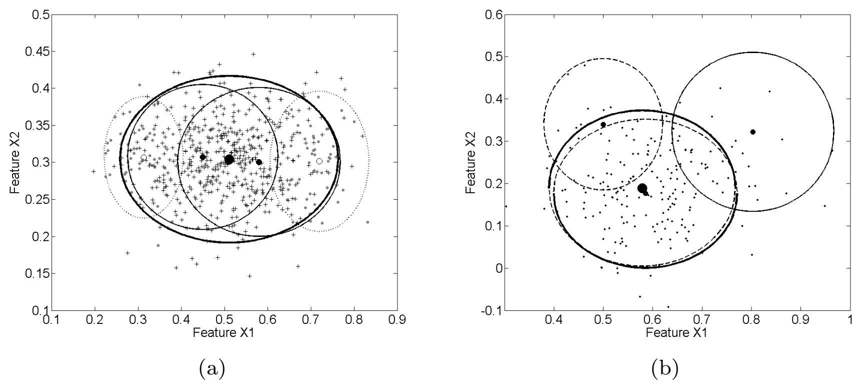


**Fig. 5** (a): strongly overlapping clusters (overlap degree of 0.88) shown as thin solid lines are merged to one bigger cluster (thick solid line), the original cluster before the update phase are shown in dotted lines, which are extracted from the original samples shown as grey dots; (b): overlapping clusters (bottom and upper left) with medium similarity degree of 0.71, the merged cluster is nearly the same as the original bottom cluster as the upper left cluster has very low support, as less than 10% samples support this cluster while the bottom cluster is supported by about 75% of the data

- in (b) three clusters evolved during the incremental process, shown as thin solid lines; the two left most clusters have an overlap degree of 0.71. Merging these two clusters to one, lead to the cluster in thick solid line — in this case, the bottom cluster changed only slightly to the new merged cluster as originally supported by 147 samples, whereas the upper left cluster was only supported by 15 samples, which effects the update of the center and $\sigma$ in (11) marginally.

### *3.1.4 Consequent Merging and Assuring Consistency of the Rule Base*

Merging of two clusters by (11) automatically implies a merging of the corresponding rules' antecedent parts. In order to continue with one merged rule instead of two, it is also necessary to merge the consequent parts of the two strongly overlapping rules. This is achieved by taking the weighted sum of these, where the weights are presented by the significance of the two rules $i$ and $k$ expressed by the support of these:

$$\mathbf{w}_{new} = \frac{\mathbf{w}_i k_i + \mathbf{w}_k k_k}{k_i + k_k} \tag{12}$$

where $\mathbf{w}_i$ and $\mathbf{w}_k$ represent the linear parameter vectors in the consequents of the $i$th and $k$th rule, while $k_i$ and $k_k$ are the supports of the $i$th and the $k$th rule (local area), elicited by the number of data samples belonging to the corresponding clusters. In fact, this means that the merged consequent hyper-plane does not end up exactly in the middle of the other original two hyper-planes, but points more into the direction of the more significant rule. This is somehow intuitive, as the more significant rule is supported by more samples and therefore should have a higher impact in the merging process.

On the other hand, two dissimilar consequent functions of strongly over-lapping rules may point to an *inconsistency* in the rule base which weakens the transparency and interpretability of the evolved fuzzy system. In fact, from Boolean logic point of view, it is precarious to merge two rules whose antecedent parts are similar, while having dissimilar consequents. In fact, such rules usually represent inconsistent parts of the rule base as they are somewhat contradicting each other. This concept can be easily extended to fuzzy rule bases by replacing the concept of equivalence between Boolean propositions by the degree of similarity between fuzzy sets. In this sense, two fuzzy rules given by

IF $A$ THEN $C$
IF $B$ THEN $D$

can be seen as contradictory, if the similarity between premises $A$ and $B$ is (significantly) greater than the similarity between the consequents $C$ and $D$. This crisp condition can be extended to a fuzzy one, delivering a degree of contradiction. In a case like this, a simple linear combination of the consequent parts as conducted in (12) may not appear appropriate.

Thus, inspired by Yagers idea of *participatory learning* [52], we propose the following modification of the combination rule (12):

$$\mathbf{w}_{new} = \mathbf{w}_i + \alpha \cdot \rho(\mathbf{w}_i, \mathbf{w}_k) \cdot (\mathbf{w}_k - \mathbf{w}_i), \tag{13}$$

where $\alpha = k_k/(k_i + k_k)$ and $\rho(\mathbf{w}_i, \mathbf{w}_k)$ is a measure of consistency of the two rule consequents. This measure can be defined in different ways, e.g., smoothly by $\rho(\mathbf{w}_i, \mathbf{w}_k) = S_{cons}(y_i, y_k)$ or, more drastically, by

$$\rho(\mathbf{w}_i, \mathbf{w}_k) = \begin{cases} 1 & \text{if } S_{cons}(y_i, y_k) \geq overlap_{ik} \\ 0 & \text{if } S_{cons}(y_i, y_k) < overlap_{ik} \end{cases} .$$

with $S_{cons}$ the similarity degree of the two consequents belonging to rules $i$ and $k$. For $\rho = 0$, indicating an inconsistency in the rule base, we obtain $\mathbf{w}_{new} = \mathbf{w}_i$, i.e., the consequent of the more relevant rule. For $\rho = 1$, on the other hand, (13) reduces to the original combination rule (12).

The remaining question is now how to calculate the similarity between two consequent functions. In Takagi-Sugeno fuzzy systems, rule consequent functions are represented by hyper-planes in the high-dimensional space. When using local learning approach (as conducted in most of the EFS approaches), these hyper-planes are snuggling along the real trend of the approximation curve in the corresponding local region, i.e. the hyper-planes exactly represent the tendency of the data cloud in the local regions where they are defined, achieving a good interpretation capability of TS fuzzy systems [53] [32]. Thus, a reasonable measure for similarity is the angle between two hyper-planes, as it measures the difference of the direction of the consequent functions follow in the high-dimensional space. The angle between two hyper-planes corresponding to the $i$th and the $k$th rule can be measured by calculating the angle between their normal vectors $a = (w_{i1}\ w_{i2}\ ...\ w_{ip}\ \ -1)^T$ and $b = (w_{k1}\ w_{k2}\ ...\ w_{kp}\ \ -1)^T$:

$$\phi = \arccos\left(\left|\frac{\mathbf{a}^T\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right|\right) \tag{14}$$

with $\phi \in [0, \pi]$. The maximal dissimilarity is obtained when the angle between the two normal vectors is $\frac{\pi}{2}$, as the orientation of the vectors does not play a role when using the hyper-planes for prediction purposes. The extreme case, i.e. an angle of $\pi$ or 0 would indicate parallel hyper-planes, whose shift is restricted due to the large overlap of the antecedent parts of both rules. The similarity measure of two hyper-planes $y_i$ and $y_k$ can then be defined as:

$$S_{cons}(y_i, y_k) = \begin{cases} 1 - \frac{2}{\pi} * \phi & \phi \in [0, \frac{\pi}{2}] \\ \frac{2}{\pi} * (\phi - \frac{\pi}{2}) & \phi \in [\frac{\pi}{2}, \pi] \end{cases} \tag{15}$$

In Mamdani fuzzy systems the consequents are containing fuzzy sets, whose similarity can be measured in the same manner as will be described in Section 3.2.2 below. Thus, $S_{cons}$ can be substituted by $S(A, B)$ from (17) or when using Gaussians by $S_{ker}(A, B)$ from (18), where $A$ and $B$ are the consequent fuzzy sets of rule $i$ and $k$. Merging of consequent fuzzy sets can be conducted by using (21).

#### *3.1.5 Integration Concept (into EFS Approach)*

In case when no cluster is updated but a new one evolved, there is obviously no necessity to perform a rule merging process. Hence, the integration of the whole rule merging process into EFS is straightforward by adding the steps shown in Algorithm 2 after the end of each incremental learning cycle, e.g. after Step 10 in Algorithm 1.

**Algorithm 2 Redundancy Elimination in Feature Space**

```
(11) **If** a new cluster was evolved, do nothing.
(12) **Else** perform the following steps:
(13) Check if similarity of moved/updated cluster $i$ with any other cluster $k = 1, ..., C \backslash i$ calculated by (9) is higher than a pre-defined threshold $sim_{thr}$ (default value is 0.8)
(14) **If** yes
    (a) Perform rule merging of cluster $i$ with cluster $k = \arg\max_{k_1=1}^{C} overlap_{ik_1}$ after (11).
    (b) Perform merging of corresponding rule consequent functions by using equation (13).
    (c) Overwrite parameters $(\mathbf{c}_i, \boldsymbol{\sigma}_i)$ of rule $i$ with the parameters of the merged cluster $(\mathbf{c}_{new}, \boldsymbol{\sigma}_{new})$.
    (d) Delete rule $k$ belonging to cluster $k$.
    (e) Decrease number of clusters/rules: $C = C - 1$.
```

For other evolving fuzzy systems approaches these steps can be inserted at a similar place, whenever a movement of rules/cluster took place in the preliminary incremental learning step. This is because the rule merging strategy is independent from the rule/cluster learning engine as long as multidimensional Gaussian functions are used.

### 3.2 Fuzzy Set and Rule Merging in the Partition Space

#### *3.2.1 Problematic Nature*

In the previous section, we proposed a novel rule merging procedure which was acting directly on the complete rule antecedent part. In this section, we want to

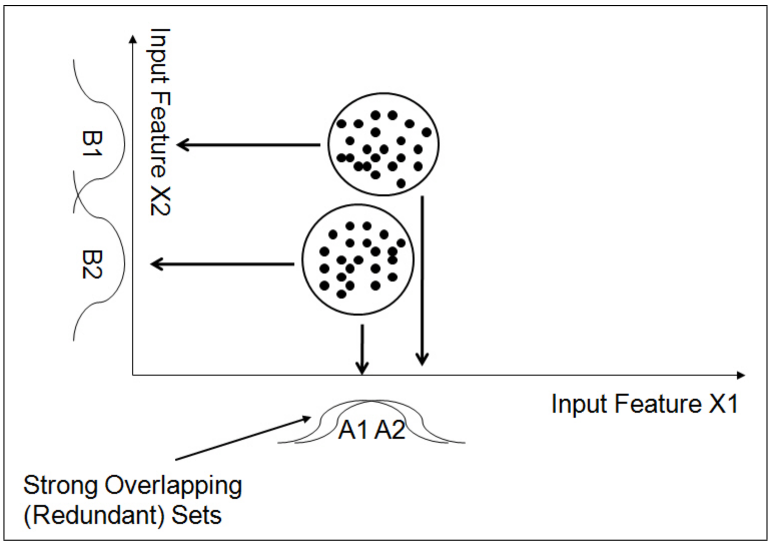


**Fig. 6** Strongly overlapping fuzzy sets in the X dimension because of projection onto the axes

go a step further and demonstrate an approach for merging two or more single fuzzy sets in the fuzzy partitions of the input variables/features. In usual cases, the appearance of redundant fuzzy sets is more frequent than of redundant rules. This fact gets evident, when considering two or more clusters lying one over each other with respect to a single dimension. Due to the projection concept in order to form fuzzy sets along the single axes, these sets may get strongly overlapping as the high-dimensional view is simply discarded. For underlining such an occurrence, a two-dimensional example is shown in Figure 6. Note that such overlapping sets can also become apparent when training fuzzy systems from data in batch mode, see [10] [36]. On-line merging of fuzzy sets is necessary in order to assure distinguishability of the membership functions within the fuzzy partitions in the single dimensions. Distinguishability of the fuzzy sets is one key step towards a better interpretability of the fuzzy models [54]. In fact, a fuzzy set should represent a linguistic term with a clear semantic meaning. To obviate the subjective establishment of this fuzzy sets/linguistic terms association, the fuzzy sets of each input variable should be distinct from each other.

### *3.2.2 Calculating Similarity of Fuzzy Sets*

In this sense, many methods for detecting redundant fuzzy sets and merging of these in a post-processing manner (after the complete fuzzy systems training phase) were proposed during the 90ties, see [44] [45] or [46], most of these using the Jaccard index as fuzzy set-theoretic similarity measure:

$$S(A,B) = \frac{\int (\mu_A \cap \mu_B)(x)\,dx}{\int (\mu_A \cup \mu_B)(x)\,dx}, \tag{16}$$

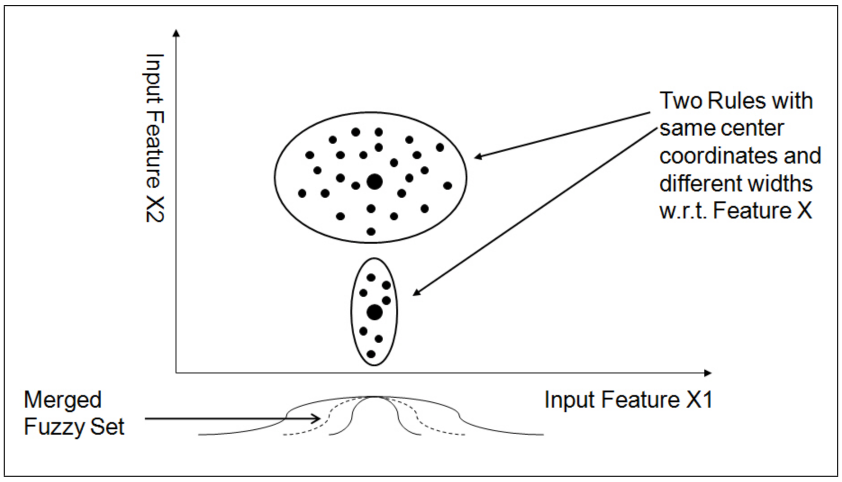

Input Feature X2
Two Rules with same center coordinates and different widths w.r.t. Feature X
Merged Fuzzy Set
Input Feature X1

**Fig. 7** Embedded fuzzy sets (for input feature X) due to projection concept and as one cluster lie over the other with respect to dimension Y; the merged set indicated by dotted line, yielding an inexact representation of the local data clouds w.r.t. dimension X

with $A$ and $B$ two fuzzy sets. In discrete form, the Jaccard index is defined by:

$$S(A,B) = \frac{\sum_{q=1}^{n} \min(\mu_A(x_q), \mu_B(x_q))}{\sum_{q=1}^{n} \max(\mu_A(x_q), \mu_B(x_q))} \tag{17}$$

with $n$ the number of discretization points. The Jaccard index was also applied in [32] for calculating the similarity measure of fuzzy sets in evolving fuzzy systems. This is in principle possible as it requires no samples from the past and hence is applicable within a single-pass on-line learning scenario. However, the Jaccard index has severe deficits regarding computational burden, as depending strongly on the discretization steps in (17), such that it is hardly applicable in on-line settings with real-time requirements. We will verify this in the evaluation section when reporting computation times. The overlap measure as proposed in (8) for rule merging purposes is not applicable for single fuzzy set merging purposes, because the membership degrees of inflection points can be also high in case of very dissimilar sets. For instance, consider the case where a fuzzy set is fully embedded in another one as shown in Figure 2 (d): then the similarity based on the inflection points would be 1, whereas a merging of the two sets is not recommended as the two sets are representing different distributions/widths of data clouds in different local regions — Figure 7 illustrates such an occurrence: a merging of the two fuzzy sets, indicated by the dotted fuzzy set, would yield an imprecise representation of the span of the data cloud in the first dimension and cause a model with weak accuracy. Note that in the high-dimensional case, the situation is different as there an embeddance means that a complete rule is embedded in another, i.e. the fuzzy sets along *all* dimensions are embedded in others: in this case, merging should be triggered, see also the consideration in Section 3.1.2 and Figure 4.

We propose an alternative similarity measure which is applicable for Gaussian membership functions and directly acts on their parameters $c$ and $\sigma$. This similarity measure is defined by:

$$S_{ker}(A,B) = e^{-|c_A - c_B| - |\sigma_A - \sigma_B|} \tag{18}$$

with $c_A$ and $\sigma_A$ the center and width of fuzzy set $A$, and $c_B$ and $\sigma_B$ the center and width of fuzzy set $B$; in order to be scale-invariant, the centers and widths of the Gaussians should be normalized before hand according to the ranges of the variables. We call it kernel-based similarity measure which is leaned on the discrepancy norm [35]. Obviously, the following holds:

$$S_{ker}(A,B) = 1 \Longleftrightarrow |c_A - c_B| + |\sigma_A - \sigma_B| = 0 \Longleftrightarrow c_A = c_B \ \wedge \ \sigma_A = \sigma_B \tag{19}$$

$$S_{ker}(A,B) < \epsilon \Longleftrightarrow |c_A - c_B| > \delta \ \vee \ |\sigma_A - \sigma_B| > \delta \tag{20}$$

The first condition means that only in case of a perfect match, the similarity measure between two fuzzy sets has the maximal degree of 1. The second condition assures that an embedded set as shown in Figure 7 is not merged with a set covering it and having a significantly larger width, which would result in a too inexact representation of the data. This is also the reason why we chose the same weight for differences in centers and spreads: a difference in the specificity between two sets (representing data clouds with different distributions/spreads) should have the same effect as a shift of the centers.

### *3.2.3 Fuzzy Set Merging Process and Integration Concept*

If $S_{ker}(A,B)$ exceeds a certain threshold (we used a value of 0.8 in all tests), the two sets can be treated as similar (redundant), as $c_A$ is close to $c_B$ and $\sigma_A$ close to $\sigma_B$, and a merging of the two fuzzy sets should be applied in order to reduce the complexity. For high thresholds, a low number of merging steps will be carried out, whereas the loss in model precisions will be negligible. Finally, the value of the thresholds steers the tradeoff between precision and distinguishability and therefore readability. In our approach, two Gaussian fuzzy sets are merged into a new Gaussian kernel with the following parameters:

$$\begin{aligned} \mu_{new} &= (\max(U) + \min(U))/2 \\ \sigma_{new} &= (\max(U) - \min(U))/2 \end{aligned} \tag{21}$$

where $U = \{\mu_A \pm \sigma_A, \mu_B \pm \sigma_B\}$. The idea underlying this definition is to reduce the *approximate* merging of two Gaussian kernels to the *exact* merging of two of their $\alpha$-cuts, for a specific value of $\alpha$. Here, we choose $\alpha = e^{(-1/2)} \approx 0.6$, which is the membership degree of the inflection points $\mu \pm \sigma$ of a Gaussian kernel with parameters $\mu$ and $\sigma$. A concrete merging example is presented in Figure 8.

After each incremental learning step, only those fuzzy sets are checked to be similar to any other ones which are belonging to a rule updated in the last incremental cycle. When a new rule is evolved, then no similarity checks are performed. Opposed to the rule merging method as demonstrated in the preliminary section, integrating the merging process of two fuzzy sets in an EFS approach is not necessarily straightforward, especially not for those approaches which are applying an incremental evolving clustering concept such as *DENFIS*, *ePL*, *eTS*, *FLEXFIS* and

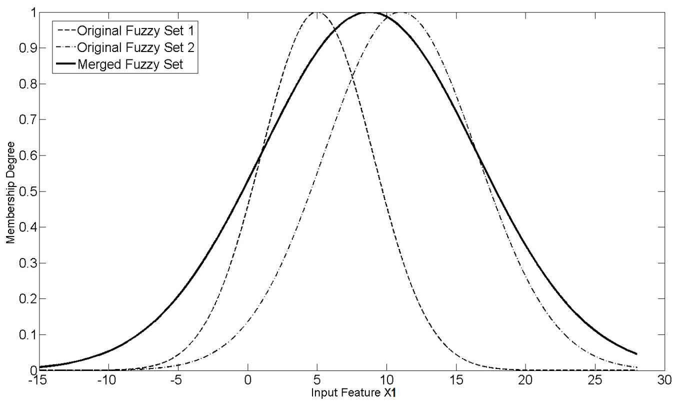


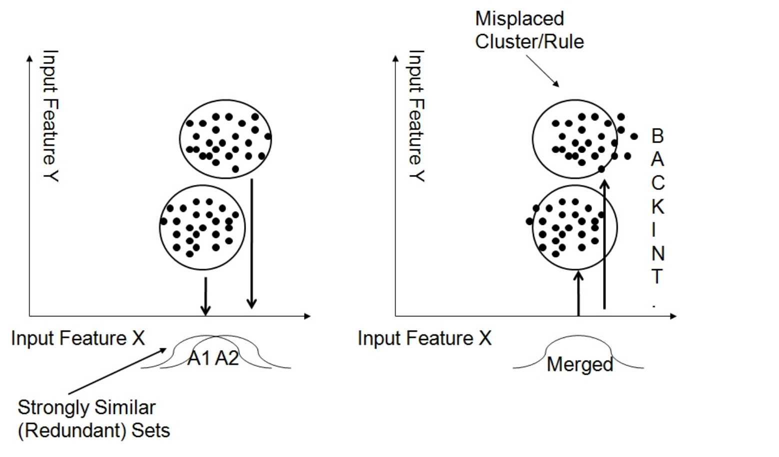
**Fig. 8** Merging of two Gaussian fuzzy sets (dashed and dotted dashed lines) to a new fuzzy set according to (21)

many others. This is because a merged fuzzy set has to be 'back-integrated' to the high-dimensional cluster space and represented appropriately. In fact, this means that the spread of two clusters, with respect to that dimension for which the merge was carried out has to be updated. However, this may cause a blurred representation of the whole cluster partition not representing the real characteristics of the data clouds any longer, finally leading to wrong cluster updates: one cluster may move not sufficiently due to back alignment, whereas the other may move too much away from the real distribution in the ongoing incremental learning context. Figure 9 demonstrates the negative effect of this occasion: cluster misplacement in (a) (upper cluster) and insufficient cluster movement in (b) according to 6 new samples (upper cluster) as well as 'cluster dragging' due to alignment with updated cluster (misplaced lower cluster).

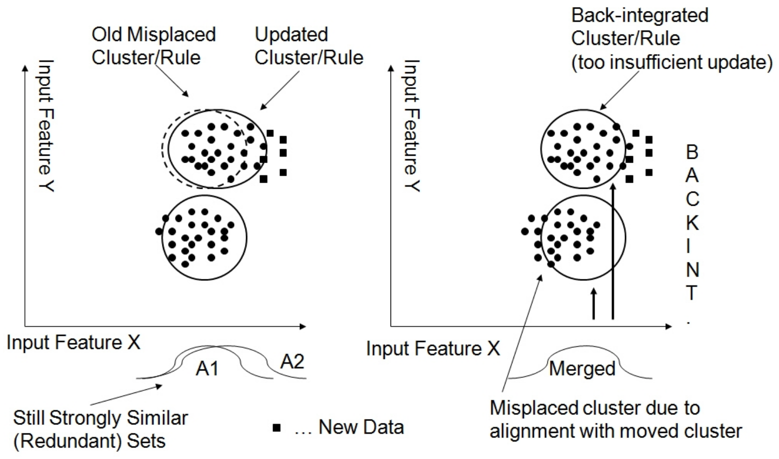
Therefore, we propose a two layer model building scheme, where one evolved fuzzy model contains the original unreduced fuzzy partitions, i.e. represents the full precise partitions as extracted from the data streams (ev. pruned due to the techniques described in the previous section), and the other contains the merged fuzzy sets due to the redundancy criteria and merging process as described above and always conducted on the precise model. The latter is for visualization and eventual manipulation purposes and dedicated to the expert/user of a system. It actually always represents a smaller fuzzy model with less complex partitions, but with an accuracy close to the full precise model. This will be empirically verified in the evaluation section.

## 3.3 Comments on Computational Cost

In case of incremental evolving learning scenarios, the computational cost when updating a fuzzy model with new incoming samples is an important aspect. This is because usually such type of learning is performed during on-line processes, often demanding proper termination of the training / prediction cycles in real-



(a)



(b)

**Fig. 9** (a): Cluster misplacement of upper cluster due to alignment along dimension X, as the merged set is back-integrated; (b): updating upper cluster with new data makes situation worse: also the lower cluster is misplaced according to dragging effect and alignment and the update of the upper cluster is not sufficient as starting from misplaced position

time. For the computational cost of various EFS approaches, please refer to the corresponding publications. Here, we inspect the additional cost which is caused when applying the fuzzy set and rule merging concepts as demonstrated in the previous sections. In case of the rule merging approach directly in the cluster space as discussed in Section 3.1, the following costs arise:

– Step 13 requires $((12+1)*(C-1)+1)p$ floating point operations (squared, square-root, multiplication, addition, subtraction) for calculating the overlap degree between updated cluster and all other clusters ($C-1$ intersection points in all $p$ dimensions).

– Step 14 (a) for merging of two clusters, i.e. the updated one with that one with most significant overlap, requires 18p floating point operations.
– Step 14 (b) for merging two rule consequents requires $3p + 1$ floating point operations.
– Steps 14 (c) and (d) are assignments of parameters and rule deletion and therefore negligible.

In sum, this means that an additional computational complexity of $O(Cp)$, with $C$ the number of clusters and $p$ the dimensionality of the feature space, is caused, i.e. an additional complexity which is a product between the number of rules and dimensionality of the feature space. In case of a reasonable parameter steering the evolution of rules and similarity threshold, we can expect that the number of rules in the feature space is bounded.

For the fuzzy set merging process as discussed in Section 3.2, the following costs arise:

– Calculating the similarity of each fuzzy set corresponding to an updated rule with all other fuzzy sets requires $7(C - 1)p$ floating point operations.
– The merging process of two fuzzy sets itself requires 8 floating point operations.
– The merging of rule consequent parameters in case when rules are getting redundant due to the fuzzy set merging process requires again $4p$ floating point operations.

In sum, this means that the fuzzy set merging process requires a computational complexity of $O(Cp)$. In case of using Jaccard index, the costs for calculating the similarity would rise to $2 * 10 * n * (C - 1)p$ with $n$ the number of grid points in the sum, counting minimum and maximum as two operations and 5 operations for calculating each, $\mu_A$ and $\mu_B$ on one specific grid point, leading to a complexity of $O(Cpn)$; often, $n > p$, as to our best knowledge a minimum of $n = 20$ to 30 is required in order to achieve an accurate calculation of the discrete Jaccard index, thus significantly increasing the complexity compared to kernel-based measure. In Section 4.2, we will see that the additional cost caused by the kernel-based measure will be negligible in multi-dimensional problems, whereas the cost caused by the Jaccard index is significantly slowing down the learning process.

## 4 Evaluation

For the purpose of verification and evaluation of the proposed novel complexity reduction methods, we exploit the *FLEXFIS* learning engine as defined in Algorithm 1.

### 4.1 Experimental Setup

The performance of the new rule and fuzzy set redundancy detection and elimination concepts will be demonstrated in connection with the *FLEXFIS* approach; the integration of the rule merging process in the feature space is achieved by the additional Steps for Algorithm 1 mentioned in Section 3.1.5.

The performance of the novel concepts will be measured in terms of model complexity versus accuracy and compared with numbers from evolved precise models without any redundancy deletion steps integrated (standard *FLEXFIS* approach). We will also demonstrate the evolution of the number of rules in both cases, give examples how improved and not improved fuzzy partitions look like and empirically measure the additional computational cost caused by the redundancy detection and elimination processes. The vigilance parameter used as essential sensitive parameter in *FLEXFIS* as controlling the tradeoff between rule evolution (stability) versus rule update (plasticity) and therefore taking the main responsibility for the stability-plasticity dilemma, is set to the standard value of $0.3\frac{\sqrt{p+1}}{\sqrt{2}}$ with $p$ the dimensionality of the input feature space (as also proposed in [28]) for all incremental learning phases — note that we do not allow manual or CV-based tuning in an off-line phase as building the models from scratch, using Option 1b) in Algorithm 1. The threshold for rule similarity in the feature space as well as for fuzzy set similarity based on kernel measure is set to 0.8 for all data sets and to 0.35 in case of using Jaccard index. For merging the consequents in case of similar rules, we used the same procedure as described in Section 3.1.4, exploiting formula (13) with consistency check.

The following data sets were applied, where each one of these were split into a training and a test set, where the training sets were sent as pseudo-streams into the incremental learning update of the fuzzy systems and the test sets used for evaluation purposes on the accuracy of the achieved models:

– Data from an engine test bench: the purpose was to estimate the NOx content in the gas emission produced by the engine during development and test phase. It is important to keep this content below a certain threshold, otherwise the car cannot be licensed for driving. The expenses for directly measuring the NOx content with hardware sensors is often too expensive and has some severe shortcomings, as sensors may get over-heated showing drift effects etc., see also [33] for a more detailed explanation. Hence, it is an important issue to accurately estimate the NOx content from other measurement channels for which sensor installations have lower costs and are more process-save. This can be achieved with data-driven prediction models called *Soft Sensors* [22], in an on-line adaptation setting with evolving models called *eSensors* [4].
– Data containing the prices of residential premises: this data set includes the five most important input features required for accurately estimating the prices of residential premises; an important goal is to predict the house prices based on past conditions for prospective years. In this sense, the training data set contains values in the period from 1998 to 2004, whereas the test data contains values from the period 2005 to 2006.
– An artificially generated data stream containing in sum 1.2 million samples by applying the synthetic stream generator from the MOA (Massive On-line Analysis) framework [7], also used in [47] for evaluation purposes. The task is to generate a binary classification problem, taking a random hyperplane in d-dimensional Euclidean space as a decision boundary. In which the classification problem is to predict position of points in relation to a hyperplane, for instance the point $x = (x_0, \dots, x_d)$ in $d$ dimensional space lays on this hyperplane if it satisfies the equation

$$\sum_{i=1}^{d} w_i x_i = w_0.$$

**Table 1** Applied data sets and their characteristics

| | # Training | # Test | # Input Var. | Source |
|---|---|---|---|---|
| NOX | 664 | 159 | 17 | Engine Test Bench |
| Premises | 2902 | 1371 | 5 | Historic Data Base of Sales |
| Hyperplane | 1 mio. | 200K | 4 | Synthetic Data incl. Drift |

A hyperplane separates the space into two classes of points, points for which $\sum_{i=1}^{d} w_i x_i > w_0$ are labeled as positives, while points satisfying $\sum_{i=1}^{d} w_i x_i < w_0$ are considered as negatives. This hyper-plane data set can be also applied in a regression context when assigning positive labels to 1 and negative to 0. Predicting the labels of new samples is conducted by producing outputs of the evolved fuzzy regression model: if these are greater than 0.5, they belong to the positive class, otherwise to the negative class. Furthermore, a concept drift was simulated by the ConceptDriftStream procedure in MOA and integrated into the hyper-plane data. The idea underlying this procedure is to mix two pure distributions in a probabilistic way, smoothly varying the corresponding probability degrees. In the beginning, examples are taken from the first pure stream with probability 1, and this probability is decreased in favor of the second stream in the course of time.

In case of the real-world problem data sets, the error on a separate test set, as containing the recordings at the end of the on-line process, denotes the worst case scenario regarding expected predictive performance of the evolved models. In case of the synthetic hyper-plane data, we conducted an analysis on a periodic hold-out test scenario: each odd block of data is used for incremental model updates, each even block for eliciting the accuracy of the fuzzy model evolved so far from the stream. The block size was set to 5000 samples for training and 1000 samples for test cycles. A characteristics of the data for all sets is summarized in Table 1.

## 4.2 Results

### *4.2.1 Results on NOx Data*

For the high-dimensional NOx data set, in order to gain more interpretability, the most important variables were pre-selected by engine experts which they saw as sufficient to build reliable models, namely
Te = Engine output torque
P2offset = Pressure in cylinder #2
N = Rotation speed
Alpha = Accelerator pedal
Tgas = Exhaust temperature
reducing the input dimensionality from 17 to 5 and the AND connections in the fuzzy systems from 16 to 4. Time delays of these inputs up to 10 time steps (10 seconds) in the past were considered. A time delay extraction method based on a modified variant of forward selection [15] was applied in order to select the most influential time delay for each of the pre-selected input channels. This resulted in the following input channels for the evolving fuzzy systems training: Te(k-5),

**Table 2** Performance of EFS without and with pruning of redundant rules and fuzzy sets on NOx data

| Method | MAE TS / MAM | # of Rules | # of Fuzzy Sets | Comp. Time in sec. |
|---|---|---|---|---|
| EFS conv. | 13.15 / 20.95 | 14 | 70 | 1.68 |
| EFS + ERR | 13.08 / 22.45 | 6 | 30 | 1.89 |
| EFS + ERF | 13.01 / 22.46 | 14 | 17 | 1.92 / 26.21 |
| EFS + ERR + ERF | 13.16 / 23.36 | 6 | 13 | 1.88 / 3.48 |

P2offset(k-5), N(k-4), Alpha(k-6), Tgas(k-9) in order to predict the NOx content at the current (the $k$th) time instance.

The results of the evolving fuzzy systems (EFS) training procedure are shown in Table 2, where the various lines compare 1.) conventional EFS, 2.) EFS with elimination of redundant rules (ERR) included, 3.) EFS with elimination of redundant fuzzy sets (ERF) included, 4.) combining rule and fuzzy set merging. As accuracy measure the mean absolute error (MAE) was applied, which is nearly redundant to RMSE and correlation coefficient, showing the same relative tendency in the quality of the approximation. From this table, we can realize that even though the complexity of the evolved fuzzy models are significantly reduced by eliminating redundant rules (compare Column #3 among Rows #2 and #3), the errors of the models stayed almost on the same level (compare Column #2 among Rows #2 and #3), in fact are even slightly lower than when not applying any rule merging process. In this sense, as a nice side effect, over-fitting of the models may be also decreased. The same is the case when performing fuzzy set merging process (compare Column #2 among Rows #2 and #4). Combining rule and fuzzy set redundancy deletion leads to the lowest complexity, i.e. 6 rules with 13 fuzzy sets, i.e. 2-3 in average for each of the 5 inputs, not loosing significant ground in terms of MAE over the precise evolved fuzzy model. The values after the slashes in Table 2, second column, show the normalized mean absolute errors obtained when using the equivalent Mamdani fuzzy system for prediction. The equivalent Mamdani is simply obtain by projecting the evolved clusters from the product space also to the output/target variable, instead of estimating hyper-planes as piece-wise linear consequent models. A Mamdani fuzzy system offers a better linguistic interpretability, as also the consequents are consisting of membership functions and their associated linguistic terms, however suffers of accuracy by an error increase of about 50%.

Figure 10 shows the effect of rule merging during the incremental learning phase. During the first half of the training set, no merging is requested: obviously, here the rules which are evolved are really needed for representing and approximating different regions in the feature space. At the beginning of the second half, there is a severe drop in the number of rules as some of the former evolved rules are becoming significantly overlapping due to coalescence of their centers. After this phase, new rules get again evolved as necessary in so far unexplored regions and others merged etc. Finally, the learning is more vital and flexible than when not using any rule redundancy deletion approach (compare dotted versus solid line): necessary rules are evolved and unnecessary ones are pruned. Also note that the concept of avoiding a merge of two contradictory rules could even improve the error of the final obtained fuzzy models slightly, in sum about 3%-5%.

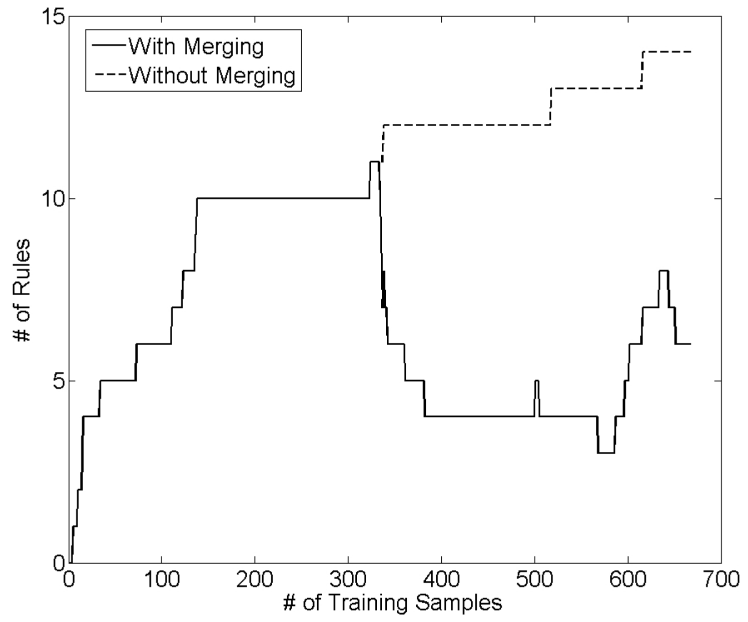
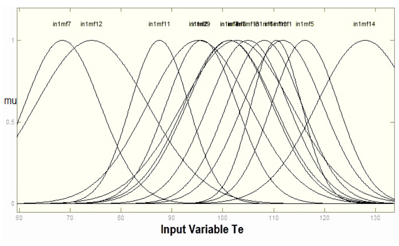


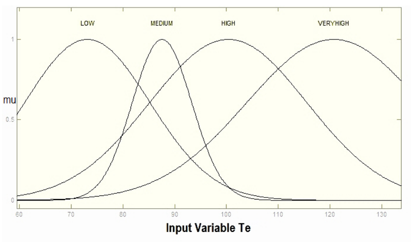
**Fig. 10** (Evolution progress on the number of rules for NOx data set when using conventional EFS (without redundancy deletion) as shown in dotted line versus evolution progress on the number of rules when using EFS with coupled rule merging in case of strongly overlapping rules (solid line)

The fuzzy partitions for all input variables are shown in Figure 11 (captured from MATLAB's fuzzy logic toolbox), whereas the left side shows all the partitions obtained when not integrating any fuzzy set redundancy elimination technique and the right side the improved partitions. Clearly, the improvement is significant, as the partitions on the left hand side are hardly readable, while the partitions on the right hand side contain clearly separable fuzzy sets, some having a nearly ideal overlap at the membership degree of 0.5. This clear separability serves as basis for assigning linguistic labels to the sets, providing a set of 6 linguistically readable rules premises, with 5 antecedent parts for the 5 inputs.

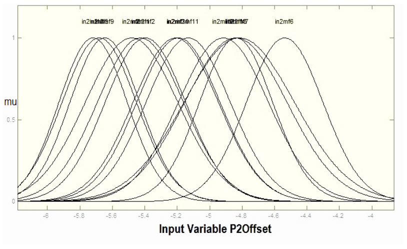
The computation burden for doing additional redundancy checking and elimination steps is listed in the last column of Table 2. While conventional *EFS* takes 1.68 seconds for learning the fuzzy models from the 664 training samples, therefore, the update time for one single sample is below one millisecond, additional rule and fuzzy set merging steps increase this value by only about 0.2 seconds, and can therefore be seen as negligible. The tiny increase is basically because in each incremental learning step only the modified rule is checked for becoming redundant. The entries after the slashes in the last column of Table 2 are the required computation times when applying the Jaccard index as similarity measured instead of kernel-based similarity after (18). A severe drop in computation speed and hence on-line performance of *EFS* can be observed in this case, whereas similar fuzzy



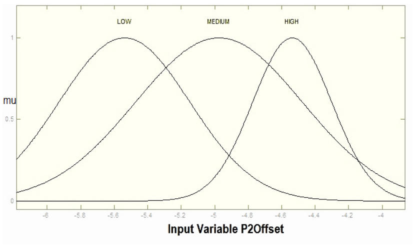
(a)



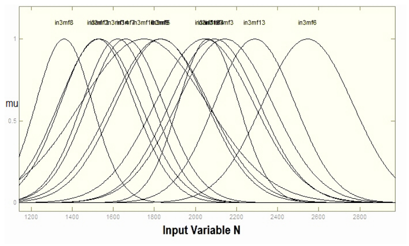
(b)



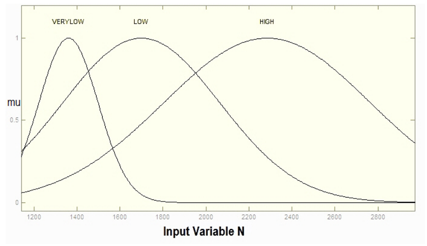
(c)



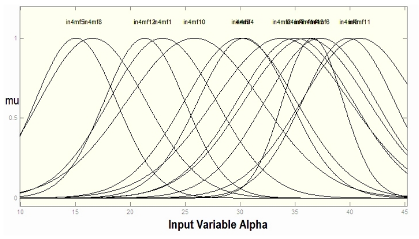
(d)



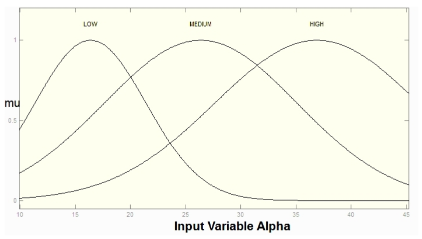
(e)



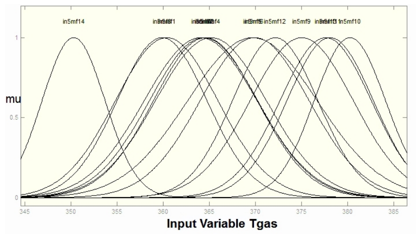
(f)



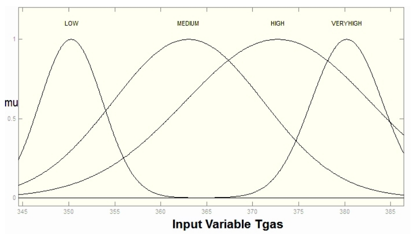
(g)



(h)



(i)



(j)

**Fig. 11** Left side: final fuzzy partitions obtained for (from top to bottom) Te, P2offset, N, Alpha and Tgas when using full precise EFS; right side: the final partitions obtained for the same variables when detecting fuzzy set redundancies and eliminating these in connection with EFS

**Table 3** Performance of EFS without and with pruning of redundant rules and fuzzy sets on residential premise data

| Method | MAE norm | # of Rules | # of Fuzzy Sets | Comp. Time in sec. |
|---|---|---|---|---|
| EFS conv. | 0.1728 | 18 | 90 | 9.75 |
| EFS + ERR | 0.1635 | 4 | 20 | 7.91 |
| EFS + ERF | 0.1677 | 16 | 16 | 9.27 / 62.71 |
| EFS + ERR + ERF | 0.1579 | 4 | 12 | 8.10 / 8.53 |

sets as shown in Figure 11 (left side) are obtained. This means that especially in case of a high number of rules (last but one row), the usage of Jaccard index may fail in fast on-line modeling processes. The errors obtained when using Jaccard index with the default threshold of 0.35 are very similar to the obtained when using kernel-based similarity measure: 13.22 versus 13.01 in case of EFS+ERF and 13.14 versus 13.16 in case of EFS+ERR+ERF.

### *4.2.2 Results on Residential Premise Data*

Opposed to the NOx data set, the data set of residential premises denotes a static data set, i.e. no inclusion of time delays of the input variables for a-head-type predictions of future value prices of the residential premises is required. This is also because the characteristics of one premise contained in one single data sample already represents the full price of the premise. The demand for an incremental modeling task is requested because the data base containing past premise prices is regularly updated with new ones which should be automatically included in the model without using time-intensive re-training phases. In this sense, it is a challenge to build up models in incremental evolving manner and to study their performance on a separate test data set. For doing so, the test data set contained the prices of two consecutive years 2005-2006 following the years included in the training set, i.e. 1998-2004. The results when evolving the fuzzy models with and without merging criteria are reported in Table 3. Here, the impact of eliminating local redundancies in feature spaces and fuzzy partitions on the predictive performance of the final achieved model is even higher: the normalized mean absolute error (MAE) can be even decreased from 0.1728 to 0.1635 and finally to 0.1579, i.e. by about 10% while synchronously also decreasing the complexity of the models: from 18 to 4 rules, from 90 to 12 fuzzy sets in sum, i.e. from 18 to 2.5 fuzzy sets on average in each input dimension. This means that the rule and fuzzy set redundancy approach is not only able to achieve more transparent models, but also to decrease the over-fitting effect caused by these, and this with negligible additional computational cost — compare computation times in the last column of Table 3. In fact, the computation time can be even decreased as after merging of some redundant rules, a lower number of rules needs to be updated in the fuzzy models. Hence, in this case, the additional cost caused by the rule and fuzzy set merging process, is compensated by a reduced cost in the model updates. Again Jaccard index requires significantly higher time in the merging process (compare 4th row, last column the values before and after the slash), whereas the errors obtained when using Jaccard index are very similar to the obtained when using

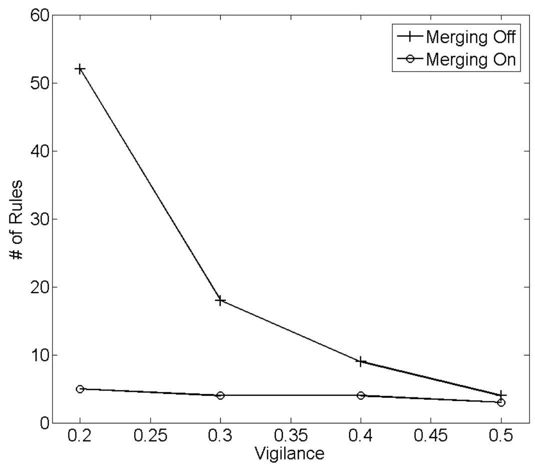
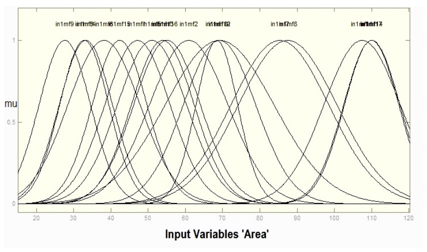


**Fig. 12** The effect of different settings of the vigilance parameter on the number of extracted rules in case when rule merging is switched off (line with crosses) and on (line with circles)

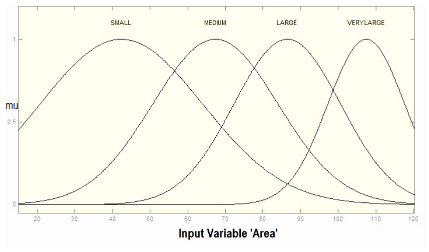
kernel-based similarity measure: 0.1751 versus 0.1677 in case of EFS+ERF and 0.1592 versus 0.1579 in case of EFS+ERR+ERF.

An interesting point could be investigated, namely that an increased vigilance parameter of $0.5\frac{\sqrt{p+1}}{\sqrt{2}}$ (standard value: $0.3\frac{\sqrt{p+1}}{\sqrt{2}}$) used in conventional EFS without pruning could also achieve 4 rules, but results in an increased normalized MAE of 0.1774. This finally means that not even a manually tuned vigilance (which is usually not possible in a real incremental learning setting where data is received sample per sample) can result in better performance than a more flexible automatic rule merging-and-evolution process. It also means that a systematic decrease of over-fitting is not simply a matter of increasing the distance threshold for reducing the number of rule evolutions, but in fact more a matter of eliminating unnecessary complexity arising during the update process as mentioned at the beginning of Section 3.1. Another interesting behavior was observed when decreasing the vigilance parameter to a value of $0.2\frac{\sqrt{p+1}}{\sqrt{2}}$: in this case, the number of rules exploded from 18 to 52 in case of conventional EFS, whereas in case of included rule merging process the number rule only increase from 4 to 5, which is a remarkable stability regarding the vigilance parameter. This is further underlined that also an increase of the vigilance to $0.4\frac{\sqrt{p+1}}{\sqrt{2}}$ resulted in 4 rules. Table 4 and Figure 12 summarize the effect of different settings of the vigilance parameter on the number of evolved rules, the default setting $0.3\frac{\sqrt{p+1}}{\sqrt{2}}$ highlighted in bold font.

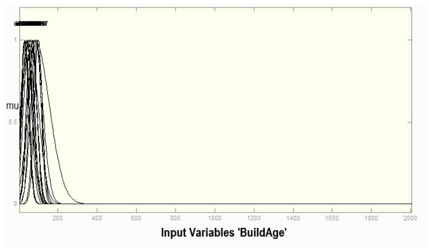
The final achieved fuzzy sets when using conventional EFS and using EFS with redundancy elimination are compared in Figure 13 (left side versus right side): shown are the partitions for all the input variables, namely 'Area', 'BuildAge', 'Storeys', 'Rooms' and 'Floor', whereas the order of the features matters, i.e. 'Area' was selected as the most important, 'BuildAge' as the second most important features and so on... A clear improvement regarding separability of the fuzzy sets



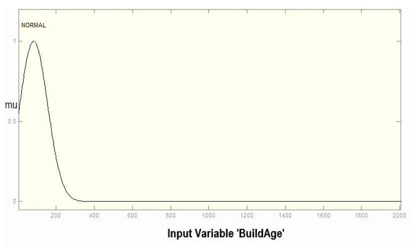
(a)



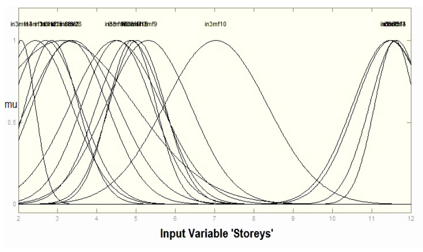
(b)



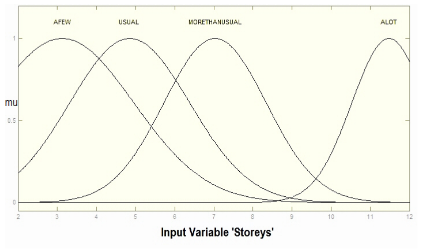
(c)



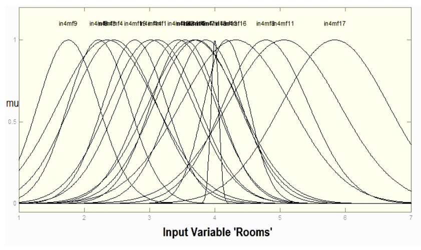
(d)



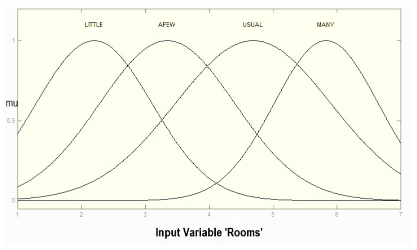
(e)



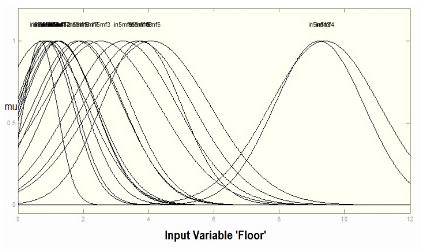
(f)



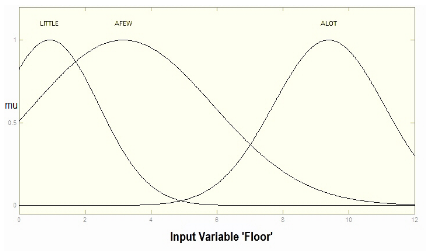
(g)



(h)



(i)



(j)

**Fig. 13** Left side: final fuzzy partitions obtained for (from top to bottom) Te, P2offset, and N when using full precise EFS; right side: the final partitions obtained for the same variables when detecting fuzzy set redundancies and eliminating of these in connection with EFS

**Table 4** Effect of different vigilance parameter on the number of evolved rules Second column: no rule merging process is performed, third column: rule merging is integrated, note the less sensitivity in this case with respect to the # of rules (achieving fuzzy models with almost same complexity for all settings)

| Vigilance | # of Rules conv. EFS / MAE | # of Rules EFS with ERR / MAE |
|---|---|---|
| $0.2\frac{\sqrt{p+1}}{\sqrt{2}}$ | 52 / 0.2041 | 5 / 0.1714 |
| $\mathbf{0.3\frac{\sqrt{p+1}}{\sqrt{2}}}$ | **18 / 0.1728** | **4 / 0.1635** |
| $0.4\frac{\sqrt{p+1}}{\sqrt{2}}$ | 9 / 0.1737 | 4 / 0.1660 |
| $0.5\frac{\sqrt{p+1}}{\sqrt{2}}$ | 4 / 0.1774 | 3 / 0.1721 |

could be achieved, also opening the possibility to assign a linguistic term to each fuzzy set. Also interesting to see is that for the input feature 'BuildAge', (correctly) no fuzzy sets in the upper range area were evolved as ignoring the outliers in the data by the rule-base procrastination approach integrated in *FLEXFIS*, see [28] — this ensures a more stable fuzzy model.

### *4.2.3 Results on Hyper-Plane Data*

For the synthetic data set including 1.2 million samples and a drift after approximately 450K samples, we conducted a periodic hold out test in order to observe the accuracies achieved by the evolved models with and without rule merging/pruning over time. Each odd block of data is used for further updating and evolving the models, each even block of data for eliciting the accuracies in terms of classification labels (assigned from the regression output). The block size was set to 5000 samples for training and 1000 samples for test cycles. This scenario is usually conducted when intending to inspect the impact of a drift in a stream onto the model accuracy: as the periodic test sets are completely independent and the accuracy is not accumulated (as done in an interleaved-test-and-then-train scenario), drifts can be recognized easily by a (sudden) downtrend of the accuracy trend line. In such a case, a forgetting mechanism can be triggered including a gradual outdating of older samples over time as proposed in [31]. We applied a forgetting factor of 0.999, i.e. a slow forgetting as also the drift was moderate. For comparison purposes, we also applied Hoeffding trees [13] as widely-used incremental classifiers from the MOA framework.

Figure 14 shows the accuracy trend lines achieved by 1.) no merging and no forgetting, 2.) no merging and forgetting, 3.) merging and no forgetting and 4.) merging and forgetting. Clearly, the impact of forgetting is very clear as the downtrend starting at around 450000 samples can be faster compensated than when not using any forgetting mechanism, no matter whether rule merging is switched on or off. Also, it is interesting to see that rule merging does not really weaken the accuracy over time significantly and EFS can out-perform Hoeffding trees significantly, especially before the drift phase and also when using forgetting during the drift. The average classification accuracy over all periodic held out test samples was 90.24% without rule merging and no forgetting, 92.6% without rule merging and forgetting, 89.03% with rule merging and no forgetting and 92.08% with rule merging and forgetting.

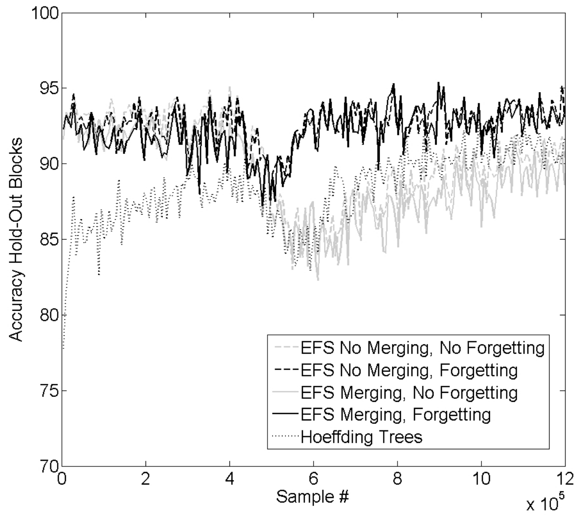


**Fig. 14** Accuracy trend lines of evolving fuzzy systems applied onto hyper-plane data: without rule merging and no forgetting (grey dashed line), with rule merging and no forgetting (grey solid line), without rule merging and forgetting (dark dashed line) and with rule merging and forgetting (dark solid line); the dotted line represents the accuracy trend of Hoeffding trees, available in the MOA framework and which has no forgetting integrated.

Furthermore, we were also interested in the number of evolved structural components during the incremental learning phase: number of rules in case of EFS and number of nodes (tree size) in case of Hoeffding trees — Figure 15 compares the results. EFS without pruning permanently increases the number of rules until a kind of saturation is achieved, ending up with 70 rules. EFS with pruning shows a more dynamic lively behavior, the average number of rules over the whole lifetime was 7, Hoeffding trees permanently increase linearly the number of rules up to 747. Du to the heavy data stream of 1 million training samples used, the significantly decreased size of EFS when using merging option (7 versus 65-70) pays off in computation time: EFS with merging takes only 15.9 minutes, while EFS without merging takes 97.4 minutes on a conventional PC with Windows 7 and MATLAB 2010 installed.

## 5 Conclusion and Outlook

In this paper, we demonstrated new methodologies and concepts for detecting and eliminating redundancies in evolving fuzzy systems which may come up during the incremental learning process. In fact, the common and natural denominator for all EFS approaches is that they always see the samples as current snapshots of the data and it is impossible for them to look into the future which characteristics in the data will evolve and which original gaps in the feature space will be filled up forcing significant overlap of originally disjoint data clouds. The redundancies may arise on rule level in the high-dimensional features space or on fuzzy set

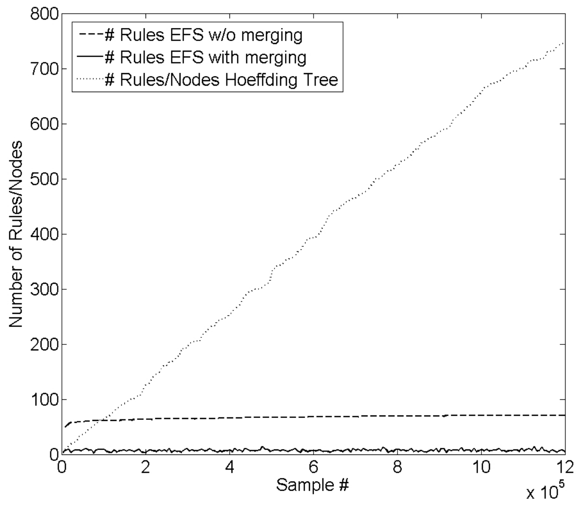


**Fig. 15** Development of the number of rules over time for EFS without merging (dashed line), with merging (solid line) and Hoeffding trees (dotted line) — note the linear increase of the latter and the logarithmic saturation of EFS without merging at around 70 rules; EFS with merging achieves 7 rules in average

level in the fuzzy partitions of the single features. Detecting redundancies are carried out with the help of similarity and overlap measures based on kernel-based similarity and intersection points. Eliminating redundancies can be achieved by specific merging procedures not requiring any past data and also taking into account consistencies within the rule base. The new techniques were applied to real-world application examples, i.e. prediction of NOx emission and prices of residential premises, and also to a huge synthetic data stream including 1.2 million samples and a concept drift. They underline the applicability of the new methods, as reducing the complexities of the fuzzy models significantly by not worsening their predictive accuracy. In case of the synthetic data stream, the computation time suffered even significantly when not applying any merging option as a full precise model with a high number of rules had to be updated throughout the learning phase. The concepts are applicable for all EFS approaches as long as Gaussian fuzzy sets are used. In this sense, the concepts can be finally seen as an important aspect in various existing EFS approaches not only for dynamic model complexity reduction steps, but also as an attempt for an automatic compensation of inappropriate setting of learning parameters at the start of an incremental learning process. Future work will include the generalization of all the similarity and merging concepts to arbitrary fuzzy sets and the application of all the concepts demonstrated in this paper to evolving fuzzy classifiers. For the former issue, first ideas are investigated by using an enhanced, general cluster inclusion measure.

## Acknowledgements

This work was funded by the Austrian and German fund for promoting scientific research (FWF and DFG, contract number I328-N23, acronym IREFS). It reflects only the authors' views. The authors also acknowledge Eyke Hüllermeier for providing valuable comments and suggestions on the manuscript.

## References

1. Abraham, W., Robins, A.: Memory retention the synaptic stability versus plasticity dilemma. Trends in Neurosciences **28**(2), 73–78 (2005)
2. Angelov, P.: Evolving takagi-sugeno fuzzy systems from streaming data, eTS+. In: P. Angelov, D. Filev, N. Kasabov (eds.) Evolving Intelligent Systems: Methodology and Applications, pp. 21–50. John Wiley & Sons, New York (2010)
3. Angelov, P., Filev, D.: Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models. In: Proceedings of FUZZ-IEEE 2005, pp. 1068–1073. Reno, Nevada, U.S.A. (2005)
4. Angelov, P., Kordon, A.: Evolving inferential sensors in the chemical process industry. In: P. Angelov, D. Filev, N. Kasabov (eds.) Evolving Intelligent Systems: Methodology and Applications, pp. 313–336. John Wiley & Sons, New York (2010)
5. Babuska, R.: Fuzzy Modeling for Control. Kluwer Academic Publishers, Norwell, Massachusetts (1998)
6. Baturone, I., Moreno-Velo, F., Gersnoviez, A.: A cad approach to simplify fuzzy system descriptions. In: Proceedings of the 2006 IEEE International Conference on Fuzzy Systems, pp. 2392–2399. Vancouver (2006)
7. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive online analysis. Journal of Machine Learning Research **11**, 1601–1604 (2010)
8. Breiman, L., Friedman, J., Stone, C., Olshen, R.: Classification and Regression Trees. Chapman and Hall, Boca Raton (1993)
9. Burger, M., Haslinger, J., Bodenhofer, U., Engl, H.W.: Regularized data-driven construction of fuzzy controllers. Journal of Inverse Ill-Posed Problems **10**(4), 319–344 (2002)
10. Casillas, J., Cordon, O., Herrera, F., Magdalena, L.: Interpretability Issues in Fuzzy Modeling. Springer Verlag, Berlin Heidelberg (2003)
11. Chen, M., Linkens, D.: Rule-base self-generation and simplification for data-driven fuzzy models. Fuzzy Sets and Systems **142**(2), 243–265 (2004)
12. Destercke, S., Guillaume, S., Charnomordic, B.: Building an interpretable fuzzy rule base from data using orthogonal least squares—application to a depollution problem. Fuzzy Sets and Systems **158**(18), 2078–2094 (2007)
13. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 71–80. Boston, MA (2000)
14. Espinosa, J., Vandewalle, J.: Constructing fuzzy models with linguistic intergrity from numerical data - AFRELI algorithm. IEEE Transactions on Fuzzy Systems **8**(5), 591–600 (2000)
15. Groißböck, W., Lughofer, E., Klement, E.: A comparison of variable selection methods with the main focus on orthogonalization. In: M. Lopéz-Díaz, M. Gil, P. Grzegorzewski, O. Hryniewicz, J. Lawry (eds.) Soft Methodology and Random Information Systems, Advances in Soft Computing, pp. 479–486. Springer, Berlin, Heidelberg, New York (2004)
16. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction - Second Edition. Springer, New York Berlin Heidelberg (2009)
17. Hathaway, R., Bezdek, J.: Switching regression models and fuzzy clustering. IEEE Transactions on Fuzzy Systems **1**(3), 195–204 (1993)
18. Jimenez, F., Gomez-Skarmeta, A.F., Sanchez, G., Roubos, H., Babuska, R.: Accurate, transparent and compact fuzzy models by multi-objective evolutionary algorithms. In: J. Casillas, O. Cordón, F. Herrera, L. Magdalena (eds.) Interpretability Issues in Fuzzy Modeling, *Studies in Fuzziness and Soft Computing*, vol. 128, pp. 431–451. Springer, Berlin (2003)

19. Kasabov, N.: Evolving Connectionist Systems: The Knowledge Engineering Approach - Second Edition. Springer Verlag, London (2007)
20. Kasabov, N.K., Song, Q.: DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Transactions on Fuzzy Systems **10**(2), 144–154 (2002)
21. Klement, E., Mesiar, R., Pap, E.: Triangular Norms. Kluwer Academic Publishers, Dordrecht Norwell New York London (2000)
22. Kordon, A., Smits, G., Kalos, A., Jordaan, E.: Robust soft sensor development using genetic programming. In: R. Leardi (ed.) Nature-Inspired Methods in Chemometrics, pp. 69–108 (2003)
23. Kurzhanskiy, A.A., Varaiya, P.: Ellipsoidal toolbox. Tech. Rep. UCB/EECS-2006-46, EECS Department, University of California, Berkeley (2006). URL http://code.google.com/p/ellipsoids
24. Lima, E., Hell, M., Ballini, R., Gomide, F.: Evolving fuzzy modeling using participatory learning. In: P. Angelov, D. Filev, N. Kasabov (eds.) Evolving Intelligent Systems: Methodology and Applications, pp. 67–86. John Wiley & Sons, New York (2010)
25. L.Ros, A.Sabater, F.Thomas: An ellipsoidal calculus based on propagation and fusion. IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics **32**(4), 430–442 (2002)
26. Lughofer, E.: Process safety enhancements for data-driven evolving fuzzy models. In: Proceedings of 2nd Symposium on Evolving Fuzzy Systems, pp. 42–48. Lake District, UK (2006)
27. Lughofer, E.: Extensions of vector quantization for incremental clustering. Pattern Recognition **41**(3), 995–1011 (2008)
28. Lughofer, E.: FLEXFIS: A robust incremental learning approach for evolving TS fuzzy models. IEEE Transactions on Fuzzy Systems **16**(6), 1393–1410 (2008)
29. Lughofer, E.: Towards robust evolving fuzzy systems. In: P. Angelov, D. Filev, N. Kasabov (eds.) Evolving Intelligent Systems: Methodology and Applications, pp. 87–126. John Wiley & Sons, New York (2010)
30. Lughofer, E.: Evolving Fuzzy Systems — Methodologies, Advanced Concepts and Applications. Springer, Berlin Heidelberg (2011). ISBN: 978-3-642-18086-6
31. Lughofer, E., Angelov, P.: Handling drifts and shifts in on-line data streams with evolving fuzzy systems. Applied Soft Computing **11**(2), 2057–2068 (2011)
32. Lughofer, E., Hüllermeier, E., Klement, E.: Improving the interpretability of data-driven evolving fuzzy systems. In: Proceedings of EUSFLAT 2005, pp. 28–33. Barcelona, Spain (2005)
33. Lughofer, E., Macian, V., Guardiola, C., Klement, E.: Data-driven design of takagi-sugeno fuzzy systems for predicting NOx emissions. In: E. Hüllermeier, R. Kruse, F. Hoffmann (eds.) Proc. of the 13th International Conference on Information Processing and Management of Uncertainty, IPMU 2010, Part II (Applications), *CCIS*, vol. 81, pp. 1–10. Springer, Dortmund, Germany (2010)
34. Mikut, R., Mäkel, J., Gröll, L.: Interpretability issues in data-based learning of fuzzy systems. Fuzzy Sets and Systems **150**(2), 179–197 (2005)
35. Moser, B.: A similarity measure for images and volumetric data based on Hermann Weyl's discrepancy. IEEE Transactions on Pattern Analysis and Machine Intelligence (2010). DOI: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2009.50
36. Nelles, O.: Nonlinear System Identification. Springer Verlag Berlin, Germany (2001)
37. Oliveira, J.V.D.: Semantic constraints for membership function optimization. IEEE Transactions on Systems, Man and Cybernetics, part A: Systems and Humans **29**(1), 128–138 (1999)
38. Pang, S., Ozawa, S., Kasabov, N.: Incremental linear discriminant analysis for classification of data streams. IEEE Transactions on Systems, Men and Cybernetics - part B: Cybernetics **35**(5), 905–914 (2005)
39. Qin, S., Li, W., Yue, H.: Recursive PCA for adaptive process monitoring. Journal of Process Control **10**, 471–486 (2000)
40. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Francisco, CA (1993)
41. Ramos, J., Dourado, A.: Pruning for interpretability of large spanned eTS. In: Proceedings of the 2006 International Symposium on Evolving Fuzzy Systems, pp. 55–60. Ambleside, UK (2006)

42. Rong, H.J., Sundararajan, N., Huang, G.B., Saratchandran, P.: Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. Fuzzy Sets and Systems **157**(9), 1260–1275 (2006)
43. Rong, H.J., Sundararajan, N., Huang, G.B., Zhao, G.S.: Extended sequential adaptive fuzzy inference system for classification problems. Evolving Systems **in press**, DOI: 10.1007/s12,530–010–9023–9 (2011)
44. Setnes, M.: Simplification and reduction of fuzzy rules. In: J. Casillas, O. Cordón, F. Herrera, L. Magdalena (eds.) Interpretability Issues in Fuzzy Modeling, *Studies in Fuzziness and Soft Computing*, vol. 128, pp. 278–302. Springer, Berlin (2003)
45. Setnes, M., Babuska, R., Kaymak, U., Lemke, H.: Similarity measures in fuzzy rule base simplification. IEEE Transactions on Systems, Men and Cybernetics - part B: Cybernetics **28**(3), 376–386 (1998)
46. Setnes, M., Babuska, R., Verbruggen, H.: Complexity reduction in fuzzy modeling. Mathematics and Computers in Simulation **46**(5-6), 509–518 (1998)
47. Shaker, A., Senge, R., Hüllermeier, E.: Evolving fuzzy pattern trees for binary classification on data streams. Information Sciences **to appear** (2011)
48. Stone, M.: Cross-validatory choice and assessment of statistical predictions. Journal of the Royal Statistical Society **36**, 111–147 (1974)
49. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man and Cybernetics **15**(1), 116–132 (1985)
50. Wang, L.: Fuzzy systems are universal approximators. In: Proc. 1st IEEE Conf. Fuzzy Systems, pp. 1163–1169. San Diego, CA (1992)
51. Wang, L., Mendel, J.: Fuzzy basis functions, universal approximation and orthogonal least-squares learning. IEEE Transactions on Neural Networks **3**(5), 807–814 (1992)
52. Yager, R.R.: A model of participatory learning. IEEE Trans. on Systems, Man and Cybernetics **20**, 1229–1234 (1990)
53. Yen, J., Wang, L., Gillespie, C.: Improving the interpretability of TSK fuzzy models by combining global learning and local learning. IEEE Trans. on Fuzzy Systems **6**(4), 530–537 (1998)
54. Zhou, S., Gan, J.: Low-level interpretability and high-level interpretability: a unified view of data-driven interpretable fuzzy systems modelling. Fuzzy Seta and Systems **159**(23), 3091–3131 (2008)