Evolving Systems (2015) 6:269–292
DOI 10.1007/s12530-015-9132-6

ORIGINAL PAPER

Generalized smart evolving fuzzy systems

Edwin Lughofer[1] · Carlos Cernuda[1] · Stefan Kindermann[2] · Mahardhika Pratama[3]

Received: 2 May 2014 / Accepted: 2 March 2015 / Published online: 14 March 2015

Abstract In this paper, we propose a new methodology for learning evolving fuzzy systems (EFS) from data streams in terms of on-line regression/system identification problems. It comes with enhanced *dynamic complexity reduction* steps, acting on model components and on the input structure and by employing *generalized fuzzy rules* in arbitrarily rotated position. It is thus termed as *Gen-Smart-EFS* (*GS-EFS*), short for *generalized smart evolving fuzzy systems*. Equipped with a new *projection concept* for high-dimensional kernels onto one-dimensional fuzzy sets, our approach is able to provide equivalent conventional TS fuzzy systems with axis-parallel rules, thus maintaining interpretability when inferring new query samples. The on-line complexity reduction on rule level integrates a new merging concept based on a combined *adjacency–homogeneity relation* between two clusters (rules). On input structure level, complexity reduction is motivated by a combined statistical-geometric concept and acts in a smooth and soft manner by incrementally adapting feature weights: features may get smoothly *out-weighted over time* (→*soft on-line dimension reduction*) but also may become *reactivated* at a later stage. Out-weighted features will contribute little to the rule evolution criterion, which prevents the generation of unnecessary rules and reduces over-fitting due to curse of dimensionality. The criterion relies on a newly developed *re-scaled Mahalanobis distance measure* for assuring monotonicity between feature weights and distance values. *Gen-Smart-EFS* will be evaluated based on high-dimensional real-world data (streaming) sets and compared with other well-known (evolving) fuzzy systems approaches. The results show improved accuracy with lower rule base complexity as well as smaller rule length when using *Gen-Smart-EFS*.

Keywords Data stream regression · Generalized evolving fuzzy systems (GS-EFS) · Rule merging · Adjacency–homogeneity relation · Soft and smooth on-line dimension reduction · Re-scaled Mahalanobis distance measure

✉ Edwin Lughofer
edwin.lughofer@jku.at

1 Department of Knowledge-Based Mathematical Systems, Johannes Kepler University Linz, Linz, Austria

2 Industrial Mathematics Institute, Johannes Kepler University Linz, Linz, Austria

3 School of Engineering and Information Technology, University of New South Wales, Canberra, Australia

1 Introduction

1.1 Motivation

Due to the increasing complexity and permanent growth of data acquisition sites [e.g. installed through multi-sensor networks (Cohen et al. 2008)], in today's industrial systems there is an increasing demand of fast modeling algorithms from on-line data streams (Gama 2010), which are flexible in the sense that they can be adapted to the actual system situation. Application examples of such dynamic systems are on-line adaptive surface inspection scenarios (Eitzinger et al. 2010; Lughofer et al. 2009; Sannen et al. 2008), stock-market forecasting (Maciel et al. 2012; Leite et al. 2012), any kind of evolving smart sensors for substituting cost-intensive hardware (Macias-Hernandez and Angelov 2010; Angelov and Kordon 2010), timely changing prediction of premise prices (Lughofer et al. 2011), on-line fault detection and condition monitoring systems (Chen et al. 2014; Costa et al. 2015; Lemos et al. 2013), tracking of objects in video streams (Angelov et al. 2011) or evolving modeling of user's behaviors (Iglesias et al. 2010). In particular, changing operating conditions, environmental influences and new unexplored system states may trigger a quite dynamic behavior, causing previously trained models to become inefficient or even inaccurate (Sayed-Mouchaweh and Lughofer 2012). Thus, the field of "evolving intelligent systems (EIS)[1]" (Angelov et al. 2010) or, in a wider machine learning sense, the field of "learning in dynamic environments (LDE)" (Sayed-Mouchaweh and Lughofer 2012) enjoyed a large attraction during the last years. Both fields support learning topologies which operate in single-pass manner and are able to update models and surrogate statistics on-the-fly and on demand. While EIS focus mainly on adaptive evolving models within the field of soft computing, LDE goes a step further and also joins incremental machine learning and data mining techniques. The update in these approaches concerns parameter adaptation and structural changes (e.g. rule, neuron, leaf evolution and pruning).

Evolving fuzzy systems (EFS) (Lughofer 2011) as a sub-field of EIS are helpful whenever interpretable models should be provided to users and operators—at least, they allow some sort of interpretation, especially when equipped with several concepts for assuring linguistic criteria as deeply examined in Lughofer (2013). This is opposed to other types of evolving models such as incremental radial basis functions networks (Huang et al. 2004), recurrent neural networks (Rubio 2010; Lin et al. 2013) or incremental support vector machines (Shilton et al. 2005; Diehl and Cauwenberghs 2003). Furthermore, when equipped with Takagi–Sugeno fuzzy systems architecture, EFS achieve universal approximation capability (Castro and Delgado 1996).

1.2 State of the art

Almost all EFS approaches in literature use the conventional flat TS fuzzy systems with axis-parallel rules (Takagi and Sugeno 1985) [see Lughofer (2011) for a comprehensive survey and further approaches since the last 2 years as published in Hametner and Jakubek (2013), Tung et al. (2013), Lin et al. (2013) and Soleimani et al. (2010) etc.]; some other recent approaches apply kernels in the consequents, which are trained with SVMs (Komijani et al. 2012; Cheng et al. 2011); the first method which employs generalized rules was published in Lemos et al. (2011), later extended for the purpose of data stream mining in Leite et al. (2012); there, however, no projection concept was conducted to obtain interpretable (axis-parallel) rules. There, learning is done with the usage of the methodology based on participatory learning (Yager 1990) and its evolving version (Lima et al. 2010): rule merging is triggered by a compatibility measure between two clusters, seeking whether the center of one cluster is close to the center of the other in terms of the Mahalanobis distance. An extended merging approach for generalized fuzzy rules is presented in Zdsar et al. (2014) (eFuMo), where also the ratio of the Mahalanobis distances in both directions (center A to rule B and vice versa) is taken into account in the merge criterion.

In these approaches, no projection operation is performed to allow interpretability of the evolved rules and the learning is done on the full dimensional space. In the recent published approach *GENEFIS* (Pratama et al. 2014), a projection concept is integrated based on the core span of each cluster according to each dimension. However, longer rule spreads along principal components are not taken into account and may lead to too pessimistic spreads. Furthermore, rule pruning is based on expected statistical contributions of rules in the future rather relying on the current necessary number according to the requested actual non-linearity degree among different local regions of the feature space. Finally, the on-line reduction of features in case of high-dimensional data streams has been handled in EFS for classification problems in Lughofer (2011) [employing single model and multi-model one-versus-rest architectures (Angelov et al. 2008)], and for regression problems under the scope of *eTS+* methodology (Angelov 2010) and *GENEFIS* (Pratama et al. 2014). The two latter ones operate on a crisp basis, i.e. they discard features on-line on demand due to their expected low future contributions. However, none of these foresee any smooth selection in the sense that features may get slowly out-weighted over time (but still contributing to model outputs to some degree), nor the possibility of reactivation/re-inclusion of some features at a later stage in the stream learning phase.

1.3 Our approach: the basic concept

Our approach, termed as *generalized smart evolving fuzzy systems* (*Gen-Smart-EFS*), builds upon the *FLEXFIS* learning engine (Lughofer 2008) and focusses on significant extensions (the impacts from practical point of view mentioned in braces):

1. For generalized fuzzy rules in order to be able to model local correlations more effectively (improving accuracy) (its definition in Sect. 2).
2. For providing more compact rule bases, reducing as much as unnecessary complexity (improving transparency).

[1] http://en.wikipedia.org/wiki/Evolving_intelligent_system.

3. For providing reduced rule lengths (improving interpretability) and weakening curse of dimensionality effect in incremental and soft(=smooth) manner (improving stability and accuracy).

The latter two aspects, which guide the evolved fuzzy models to some *smartness* degree, can be easily used for other types of EFS learning engines, as they offer generic concepts how to tackle rule redundancy, homogeneity and length as well as curse of dimensionality in an on-line way. These sort of complexity reduction steps are a necessary pre-requisite for assuring interpretability of EFS, especially meeting several important criteria, see Lughofer (2013). The first aspect is necessary to be able to model local correlations between input features, implying a more compact, more precise model (see Sect. 2).

Instead of performing axis parallel updates of the spreads of local data regions (=clusters) with the usage of recursive variance formula [as in Lughofer (2008)], the update concerns an incremental sample-wise update of a full occupied inverse covariance matrix, which defines the shape of the ellipsoidal clusters in arbitrary position, thus being able to model local correlations in a more compact form (see Sect. 2) than conventional axis-parallel rules. The rule evolution versus update criterion is steered by a statistical motivated tolerance region radius for multivariate normal distributions (Sect. 3.1). The induced "characteristic spread" can be seen as equivalent to the (core) contours of the ellipsoidal rules (Sun and Wang 2011). The consequent parameters are updated by *generalized recursive fuzzily weighted least squares* (RFWLS), where the weights are the normalized membership degree of the generalized rules (Sect. 3.3).

A minimal required model complexity is sought after each incremental learning cycle based on *novel geometric criteria* giving rise which degree of non-linearity is actually requested among the local regions (Sect. 3.2). Thus, rules are merged (1) which become significantly overlapping [a generalization of the concepts as in Lughofer et al. (2011) for generalized rules], and (2) which becoming slightly overlapping or touching (lying nearby each other), while (a) forming homogenous joint regions and (b) showing similar tendencies in their consequents. The latter occurrence have been not handled so far in state-of-the-art approaches

Furthermore, we are offering a *new design for an incremental reduction of the feature space*. There, we introduce feature weights which are pointing to the importance levels of each feature for explaining the (local) relation in form of a regression problem (Sect. 4). The reduction automatically has two effects: (1) reducing the curse of dimensionality in a dynamic and smooth way, i.e. features may be outweighted at a certain point of time, but may be reactivated at a later stage of the on-line data stream modeling process in an incremental manner; and (2) reducing the rule length and thus improving transparency of the evolved fuzzy systems as features with low weights can be discarded from the rules' antecedents when showing the rules to users/operators. The first point overcomes the deficiency of a crisp incremental feature selection approach as conducted in Pratama et al. (2014), which discards unimportant features due their statistical influences in a strict manner (no re-activation or a re-inclusion at a later stage possible). The extraction of feature weights are conducted by a combined statistical-geometric concept, motivated from Pratama et al. (2014), but acting in both, local (weights per rule) and global manner (weight for the whole model) (Sect. 4.1). Furthermore, we propose a novel concept how to properly integrate the weights into the Mahalanobis distance calculation (Sect. 4.2), which for incoming new samples decides whether or not a new rule should be evolved. Input features which are seen as unimportant, thus receiving low weights, should also effect the Mahalanobis distance little, omitting the evolution of unnecessary rules. This leads to a *re-scaled Mahalanobis distance measure* which assures some sort of monotonicity: successively decreasing feature weights trigger successively lower Mahalanobis distance values.

Such a re-scaled measure assuring monotonicity has been to our best knowledge not handled so far in literature and could be useful in other machine learning/data mining applications as well whenever features are not equally weighted, thus conventional Mahalanobis distance not directly applicable. A further practical usage is that such on-line rule reduction and smartness assurance techniques guarantee that the rule base is not growing forever and provide more compact rule bases. The dynamic feature weighting approach prevents a bad model performance due to over-fitting as it reduces curse of dimensionality, mostly due to the suppression of evolving new rules in case when distances among unimportant features get large. Furthermore, rule lengths are decreased, increasing transparency of rules and offering another interpretability aspect in feature level: operators/experts get a glance which system variables/features are important at which point of time during stream learning (see also the results section for two concrete examples). Finally, the practical usage is also given by the direct applicability of all the on-line complexity reduction steps (on rule and input level) to *all EFS approaches using TS type architecture with Gaussian kernels*, as these are designed completely independently from any concrete learning/adaptation algorithm for rules and fuzzy sets.

The final algorithm (termed as *Gen-Smart-EFS*) will be presented in Sect. 5 and will be evaluated based on several high-dimensional data sets from the UCI repository and on a ten-dimensional dynamic non-linear system identification problem (Sect. 6.2). It will be compared with other renowned evolving fuzzy modeling methods (Lughofer

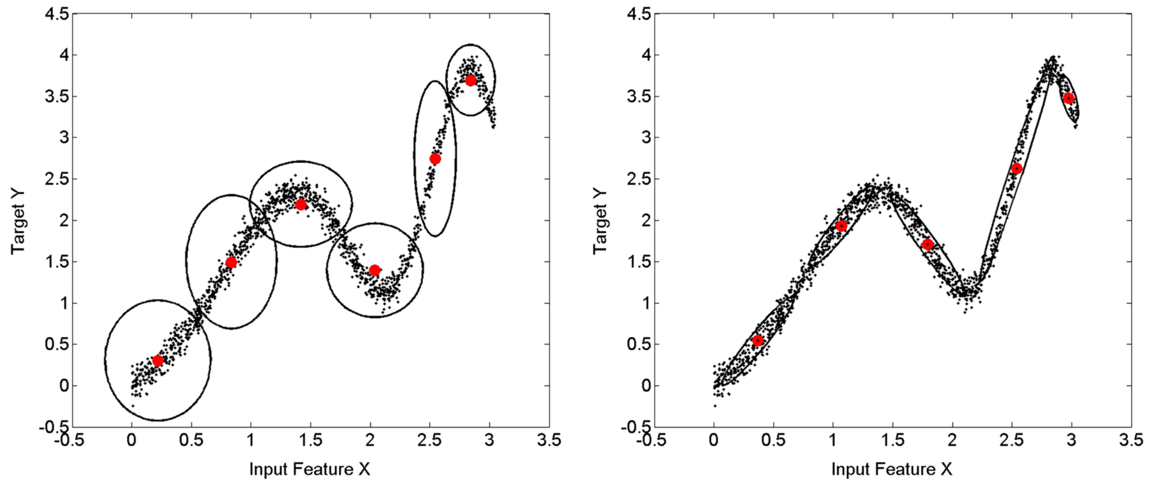

Fig. 1 *Left* conventional axis parallel rules (represented by *ellipsoids*) achieve an inaccurate representation of the local trends (correlations) of a non-linear approximation problem (defined by noisy data samples); *right* generalized rules (by *rotation*) achieve a much more accurate representation

2011) [including conventional *FLEXFIS* (Lughofer 2008), *eMG* (also employing generalized fuzzy rules) (Lemos et al. 2011), *eTS* (Angelov and Filev 2004), online sequential learning of extreme learning machines (*OS-ELM*) (Liang 2006), *FAOS-PFNN* (Wang et al. 2009) and others] as well as with classical fuzzy modeling variants in order to get a glance how close the adaptive variant can converge to batch solutions. *Gen-Smart-EFS* can achieve the models with lowest error while providing the most compact rule base with shorter rules in almost all cases. The computation time suffers only little and may be improved in some cases when significant rule reduction can be achieved. The comparison will also be based on a high-dimensional dynamic real-world scenario, where on-line measurements are recorded at a cold rolling mills process for supervising the resistance value. In this case, earlier results have been reported in literature with the usage of conventional evolving TS fuzzy model architecture and without using enhanced pruning and feature weighting concepts. These turn out to be worse regarding model accuracy and complexity than those ones achieved with the new *Gen-Smart-EFS approach*.

2 Generalized TS-fuzzy systems and projection concept

Due to the universal approximation capabilities (Castro and Delgado 1996) and the ability to present a reliable trade-off between accuracy and interpretability (Lughofer 2013), Takagi–Sugeno fuzzy systems (Takagi and Sugeno 1985) enjoy a wide field of application in several real-world modeling problems (Pedrycz and Gomide 2007). It employs conventional rules defined by:

$$\text{IF}\quad x_1\quad \text{IS}\quad \mu_1\quad \text{AND}\quad x_2\quad \text{IS}\quad \mu_2\quad \text{AND}\cdots \\ \text{AND}\quad x_p\quad \text{IS}\quad \mu_p\quad \text{THEN}\quad y=f \tag{1}$$

where f represents a polynomial function, in particular a hyperplane $f = w_0 + x_1w_1 + \cdots + x_pw_p$ and $\mu_.$ the fuzzy sets represented by linguistic terms. It has the deficiency not being able to model general local correlations between input and output variables appropriately, as the t-norm operator used for the AND connections always triggers axis-parallel rule shapes (Klement et al. 2000). Thus, conventional rules may represent inexact approximations of the real local trends and finally causing information loss (Abonyi et al. 2002). An example for visualizing this problematic nature is provided in Fig. 1: in the left image, axis-parallel rules (represented by ellipsoids) are used for modeling the partial tendencies of the regression curves which are not following the input axis direction, but are rotated to some degree; obviously, the volume of the rules are artificially blown-up and the rules do not represent the real characteristics of the local tendencies $\rightarrow$ *information loss*. In the right image, non axis-parallel rules using general multivariate Gaussians are applied for a better representation (rotated ellipsoids).

To avoid such information loss, we are aiming for generalized fuzzy rules, which are defined in Lemos et al. (2011) as

$$\text{IF}\quad \mathbf{x}\quad \text{IS (about)}\quad \Psi\quad \text{THEN}\quad y=f \tag{2}$$

where Ψ denotes a high-dimensional kernel function. Thereby, $\mathbf{x}$ plays the role of a high-dimensional input vector whose degree of assignment to a rule is steered by Ψ. When aiming for a coverage of the input space and smooth approximation surfaces, a widely used and conventional choice for Ψ [also applied in Lemos et al. (2011) and Leite et al. (2012)] is the multivariate Gaussian distribution:

$$\Psi(\mathbf{x}) = exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c})^T\Sigma^{-1}(\mathbf{x}-\mathbf{c})\right) \tag{3}$$

with $\mathbf{c}$ the center and Σ^{-1} the inverse covariance matrix. Apart from the property of infinite support, which, on the one hand, is valuable for a well-defined coverage of the input space (no undefined states may appear), but, on the other hand, triggers a more difficult input–output interpretation (as all rules are firing to a certain degree) (Lughofer 2013), it is also known in the neural network literature that Gaussian radial basis functions are a nice option to characterize local properties (Lemos et al. 2011; Lippmann 1991); especially, someone may inspect the inner core part, i.e. all samples fulfilling $(\mathbf{x}-\mathbf{c})^T\Sigma^{-1}(\mathbf{x}-\mathbf{c}) \leq 1$, as the characteristic contour/spread of the rule.

The fuzzy inference is a linear combination of multivariate Gaussian distributions in the form:

$$\hat{y} = \frac{\sum_{i=1}^{C} f_i * exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_i)^T\Sigma_i^{-1}(\mathbf{x}-\mathbf{c}_i)\right)}{\sum_{i=1}^{C} exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_i)^T\Sigma_i^{-1}(\mathbf{x}-\mathbf{c}_i)\right)} = \sum_{i=1}^{C} f_i\Phi_i(\mathbf{x}) \tag{4}$$

with C the number of rules, f_i the consequent hyper-plane of the ith rule and Φ_i the normalized membership degrees, summing up to 1 for each query sample.

In order to maintain linguistic readability and furthermore interpretability of the evolved TS fuzzy models for users/operators, we provide a projection concept to form the fuzzy sets and the antecedent parts of the classical rules. Our concept is an extension of the approach used in Pratama et al. (2014) by taking into account the degree of rotation and the spread of the ellipsoid in order to obtain a more representative width of the projected fuzzy set. The idea is to use the angle between the eigenvectors and the main axes as multiplication factors of the principal axes lengths: the higher the cosine of the angle, the closer the eigenvector is to the axis (e.g. to x_1); thus only a sightly rotated representation takes place and the principal axes length of the ellipsoid along x_1 can be (almost) directly used as the spread of the corresponding (on x_1) projected Gaussian fuzzy set. An illustration of this strategy is given in Fig. 2, with r the Mahalanobis distance radius (defining the contour of the ellipsoids, usually set to 1), λ_1 the eigenvalue of the first eigenvector a_1 of the inverse covariance matrix Σ^{-1}, and e_i the vector of the ith axis, set to $e_i = (0, 0, \ldots, 1, \ldots, 0)$ with the 1 occurring at the ith position.

From this illustration, it becomes quite clear that the spread σ_i of the projected fuzzy set is set according to:

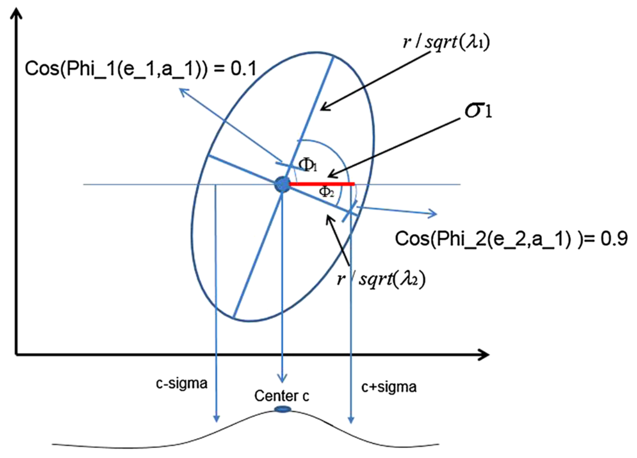

Fig. 2 Illustration of the projection concept, the width σ_1 of the projected fuzzy set highlighted in *thick dotted line*

$$\sigma_i = max_{j=1,\ldots,p}\left(\frac{r}{\sqrt{\lambda_j}}|cos(\Phi(e_i, a_j))|\right) \tag{5}$$

whereas the center of the fuzzy set in the ith dimension is equal to the ith coordinate of the rule center, and $\Phi(e_i, a_j)$ denoting the angle between principal component direction (eigenvector a_j) and the ith axis. The angle between two vectors can be measured by

$$\Phi(e_i, a_j) = \arccos\left(\frac{e_i * a_j}{\|e_i\|\|a_j\|}\right) \tag{6}$$

with $*$ the scalar product. Thus, (5) simplifies to

$$\sigma_i = max_{j=1,\ldots,p}\left(\frac{r}{\sqrt{\lambda_j}}\frac{|e_i. * a_j|}{\|e_i\|\|a_j\|}\right). \tag{7}$$

The maximum in (7) is necessary in order to obtain the maximal characteristic spread of the ellipsoid along all (principal components) directions, subject to the ith axis. This enhanced methodology provides a much better representation of the actual spread along the one-dimensional axes than when using $\sigma_i = \frac{r}{\sqrt{\Sigma_{ii}}}$ (which is the distance of the center to the axis-parallel cutting point) in case of thin long ellipsoids [as applied in Pratama et al. (2014)]. This is underlined in the example of Fig. 3, where the conventional projection onto the x-axis leads to inexact representation of the rule span (fuzzy set much too thin), whereas our concept respects the principal component length as rule span by including it in the projection formula (5).

Inferencing for obtaining predictions on new query samples is best achieved by the classical (projected) inference. The latter is performed in order to maintain input–output interpretation capability (Lughofer 2013), especially when

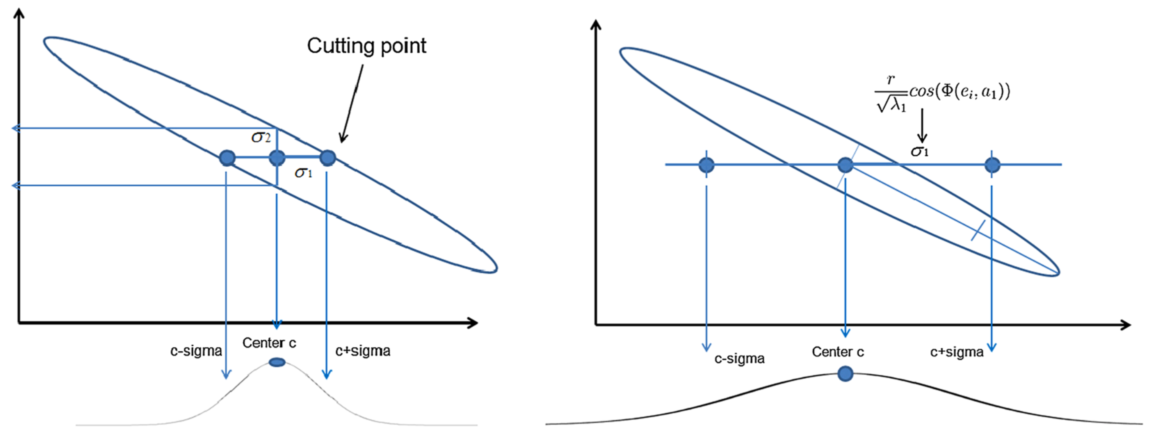

Fig. 3 *Left* conventional projection concept according to cutting point with ellipsoidal contour of the rule leads to too thin fuzzy set on x-axis; *right* our enhanced projection concept respects the maximal span of the rule along a principal component direction → more accurate projection

trying to explain the predictions, why certain outputs have been made (Heidl et al. 2013).

3 Incremental learning concepts (rule evolution, parameter adaptation, on-line complexity reduction)

3.1 Rule extraction with generalized evolving vector quantization

The rule learning policy, including rule evolution and modification in incremental learning steps, is performed in the high-dimensional product clustering space (input + output). Clustering is a concept which is frequently used in data-driven design of fuzzy systems [no matter whether for the batch (Pedrycz and Gomide 2007; Babuska 1998) or for the incremental case (Lughofer 2011)], as it nicely performs a local partitioning of the feature space, usually according to some density-based or distance-based criteria. Rules can be directly associated with the clusters. Here, we use an extended version of evolving vector quantization (*eVQ*) (Lughofer 2008), as is used in the conventional *FLEXFIS* learning engine (Lughofer 2008) for axis-parallel rules in the TS fuzzy systems.

In particular, recall the original *eVQ* as published in Lughofer (2008):

1. For each new incoming sample $\mathbf{x}$, it elicits the cluster whose center coordinates are closest to it; this is denoted as the *winning cluster* (*rule*).
2. It checks whether the new incoming sample matches the current cluster partition: the *distance* of the new point to the winning cluster is compared against a distance threshold (vigilance) $vigi = fac * \frac{\sqrt{p}}{\sqrt{2}}$, with p the dimensionality of the feature space and *fac* a multiplication constant in]0, 1].
3. If it exceeds this threshold, a new cluster is evolved by setting its center to the sample $\mathbf{c} = \mathbf{x}$ and initializing its spread vector $\sigma = \epsilon$.
4. It it does not exceed the threshold, the center and spread of the winning cluster are updated. The update of the center is performed by minimizing the expected squared quantization error [as done in conventional vector quantization (Gray 1984)]:

$$E = \int \|\mathbf{x} - \mathbf{c}\| p(\mathbf{x}) d\mathbf{x} \tag{8}$$

with $p(\mathbf{x})$ a continuous probability density function, whose approximation scheme can be derived as follows (Kohonen 1995):

$$\mathbf{c}_i(N+1) = \mathbf{c}_i(N) + \eta_{win}(\mathbf{x} - \mathbf{c}_i(N)). \tag{9}$$

Due to some nice convergence properties in evolving clustering (Lughofer 2008) and in EFS design when using *eVQ* as rule learning engine (Lughofer 2008), the learning rate was chosen to decrease with the number of samples by $\eta_i = 1/k_i$, with k_i the number of samples belonging to cluster i so far, i.e. the number of samples for which c_i was the winning cluster. In this sense, each cluster has its own learning rate and thus its own flexibility/stability for incremental movements according to its current support.

The distance calculations in above itemization points are performed by the Euclidean distance, thus triggering prototype-based clusters with ellipsoidal shape in main position. Hence, the update of the spread can be performed for each input dimension separately as conducted by the recursive variance formula including rank-1 modification (Qin et al. 2000).

The extension in this paper concerns the distance measure used, which is the Mahalanobis distance (Mahalanobis 1936) required to trigger (ellipsoidal) clusters = rules in arbitrary rotated position as defined in (3). This changes the formal description of a cluster i, as it is not defined by $(\mathbf{c}_i, \sigma_i, vigi)$ any longer, but by $(\mathbf{c}_i, \Sigma_i^{-1}, r_i)$. Instead of a global vigilance parameter, the Mahalanobis distance radius, which is specific for each rule, will decide whether a new cluster should be evolved or not, i.e. the rule evolution criterion becomes:

$$min_{i=1,\ldots,C} \sqrt{(\mathbf{x}-\mathbf{c})^T \Sigma^{-1} (\mathbf{x}-\mathbf{c})} > r_i \tag{10}$$

with

$$r_i = fac * p^{1/\sqrt{2}} * \frac{1.0}{(1-1/(k_i+1))^m} \tag{11}$$

with p the dimensionality of the input feature space and fac an a priori defined parameter, steering the tradeoff between stability (update of an old cluster) and plasticity (evolution of a new cluster). This is the only sensitive parameter in our method and will be varied during evaluation.

The second product term in (11) compensates for the curse of dimensionality (as increasing with p), based on the same considerations as done in *eVQ*, see Lughofer (2008). The difference to the approach used there is that here we are taking the $\sqrt{2}$-root of the dimensionality, instead of the conventional square-root. We then automatically obtain a close approximation of critical distribution values according to χ^2 statistics with p df, which serves as statistical tolerance region for multivariate normal distributions (Krishnamoorthy and Mathew 2009; Tabata and Kudo 2010). The reason why we use the approximation and not the real critical distribution value is the much lower computational cost. The closeness of the approximation can be visualized in Fig. 4, where for different values of p (along the x-axis) the corresponding tolerance region values (along the y-axis) are plotted.

Additionally, we include a third term in (11), which compensates for the uncertainty in clusters with low support/significance (density), increasing the radius in this case and keeping it on the original level for compact, dense clusters. . m steers the degree of this density-based impact and is set to 4 as default value.

3.1.1 Rule evolution and antecedent update

If condition (10) holds, a new cluster is evolved as follows:

$$\mathbf{c}_{C+1} = \mathbf{x} \quad \Sigma_{C+1}^{-1} = \frac{\sum_{i=1}^{C} \Sigma_i^{-1}}{C} \quad \Sigma_1^{-1} = diag\left(\frac{frac}{\mathbf{range}^2}\right) \tag{12}$$

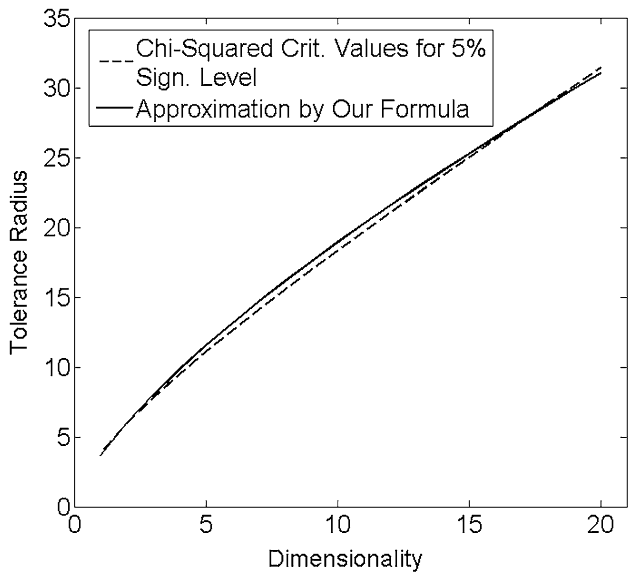

Fig. 4 Increasing $\chi_p^2(\alpha)$ critical values for $\alpha = 5\,\%$ significance level when increasing the dimensionality (*df*): the original in *dotted line*, our approximation in *solid line*

with $\frac{1}{frac}$ denoting a small fraction of the squared input ranges.

If condition (10) does not hold, the components of $\mathbf{c}_{win}$ are updated: updating the center $\mathbf{c}_{win}$ with the new sample $\mathbf{x}$ can be done in the same manner as in conventional *eVQ* by using (9). The difference lies in the update of the spread as it is characterized by the inverse covariance matrix. A possibility is to update the covariance matrix in recursive (almost) exact manner, as done in Pang (2005), Bouchachia and Mittermeir (2006) and Lughofer (2011), however this requires a matrix inversion step after each incremental learning cycle, which sometimes can get unstable (Lughofer 2011) and thus requires time-intensive regularization (Bauer and Lukas 2011). Thus, we are opting for a direct update of the inverse covariance matrix through the investigations about infinite sum expansion of an inverse matrix (Backer and Scheunders 2001): setting $\alpha = \frac{1}{k_{win}+1}$ with k_{win} the number of samples belonging to cluster c_{win} so far we obtain (we neglect here the index win for the purpose of transparency):

$$\begin{aligned}\Sigma^{-1}(k+1) &= ((1-\alpha)\Sigma(k) + \alpha(\mathbf{x}-\mathbf{c})(\mathbf{x}-\mathbf{c})^T)^{-1} \\ &= \left(I + \frac{\alpha}{1-\alpha}\Sigma^{-1}(k)(\mathbf{x}-\mathbf{c})(\mathbf{x}-\mathbf{c})^T\right)^{-1} \frac{\Sigma^{-1}(k)}{1-\alpha}.\end{aligned} \tag{13}$$

By expanding the inverse and setting $\gamma = \frac{\alpha}{1-\alpha}$, we obtain

$$\Big(I - \gamma\Sigma^{-1}(k)(v.v^T) + \gamma^2\Sigma^{-1}(k)(v.v^T) \\ \Sigma^{-1}(k)(v.v^T) - \gamma^3 \ldots\Big).\frac{\Sigma^{-1}(k)}{1-\alpha} \tag{14}$$

By rearranging and substituting $\lambda = v^T \Sigma^{-1}(k) v$, we obtain

$$= \frac{\Sigma^{-1}(k)}{1-\alpha} - \frac{\gamma}{1-\alpha}\Sigma^{-1}(k)(v.v^T)\Sigma^{-1}(k)(1 - \gamma\lambda + \gamma^2\lambda^2 - \cdots)$$
$$= \frac{\Sigma^{-1}(k)}{1-\alpha} - \frac{\gamma}{1-\alpha}\frac{\Sigma^{-1}(k)(v.v^T)\Sigma^{-1}(k)}{1+\gamma\lambda}. \quad (15)$$

After rearranging the infinite sum in the braces, the following update formula (from k to $k+1$) is obtained:

$$\Sigma^{-1}(k+1) = \frac{\Sigma^{-1}(k)}{1-\alpha} - \frac{\alpha}{1-\alpha}\frac{(\Sigma^{-1}(k)(\mathbf{x}-\mathbf{c}))(\Sigma^{-1}(k)(\mathbf{x}-\mathbf{c}))^T}{1+\alpha((\mathbf{x}-\mathbf{c})^T\Sigma^{-1}(k)(\mathbf{x}-\mathbf{c}))}. \quad (16)$$

3.2 On-line merging of unnecessary generalized rules

In this section, we are aiming for EFS with minimal possible complexity, i.e. that complexity which is really necessary to model the required (local) non-linearity. Thus, we are employing geometric-based criteria, involving adjacent rule antecedents and consequents to reflect their local modeling characteristics. It is important to note that these geometric-based criteria can be independently used from the learning engine, thus can be used in connection with all EFS approaches using TS type architecture with Gaussian kernels. This is a different motivation opposed to statistical-based criteria based on rule significance/contribution levels [as performed in Pratama et al. (2014) for generalized rules in evolving context before], which are more aiming to track the (expected) *usage* of rules in the future: rules which are expected to be not addressed within the inference for predictions can be eliminated. The difference to the merging process applied in Lemos et al. (2011) is that there a compatibility measure is used which measures the degree of overlap in terms of a simple Euclidean distance between cluster centers. In this paper, we will go some steps further and propose criteria for measuring the degree of the adjacency–homogeneity relation between two nearby lying rules, also respecting their range of influence.

3.2.1 Problem formulation and merging criteria

During incremental data stream mining, it may happen that clusters (rules), which originally seem to be disjoint and necessary for resolving the particular non-linearity properly, are moving together over their life-span. In extreme cases, they may become significantly overlapping, thus reflecting redundant rules [for a detailed analysis, see Lughofer et al. (2011)]. From the viewpoint of model representation, such rules can always be merged as representing the same local region, thus the model quality in terms of accuracy does usually not suffer. From the view point of *distinguishability*, which serves as one important aspect in (evolving) fuzzy systems (Lughofer 2013), a merge of overlapping rules is even mandatory to assure transparency, readability and unambiguous rules.

Overlap degree (*criterion #1*) In this paper, we are going beyond the approach demonstrated in Lughofer et al. (2011), which employs a merging strategy for redundant axis-parallel fuzzy rules and sets, by extending it to generalized rules and by inspecting whether nearby lying, touching or even slightly overlapping rules could also be merged. Finally, we want to maintain minimal complexity required to resolve non-linearities contained in the actual learning/regression problem. As dealing with multivariate Gaussian rules, we apply the Bhattacharyya distance (Bhattacharyya 1943; Djouadi et al. 1990) for calculating the overlap degree between the updated (the winning) cluster *win* and the other clusters, $k = \{1, \dots, C\}\backslash\{win\}$:

$$olap(win,k) = \frac{1}{8}(\mathbf{c}_{win} - \mathbf{c}_k)^T\Sigma^{-1}(\mathbf{c}_{win} - \mathbf{c}_k) + \frac{1}{2}\ln\left(\frac{\det\Sigma^{-1}}{\sqrt{\det\Sigma_{win}^{-1}\det\Sigma_k^{-1}}}\right) \quad (17)$$

with $\Sigma^{-1} = (\Sigma_{win}^{-1} + \Sigma_k^{-1})/2$—note that, due to the arbitrary ellipsoidal shape of our clusters, their contours in the p-dimensional space (according to r) are equivalent to the characteristic (supporting) spread of multi-variate Gaussian distributions (Sun and Wang 2011); thus, Bhattacharyya distance is in fact valid to be adopted one-to-one to our generalized fuzzy rules. The distance delivers exactly 0 if two ellipsoids are touching, >0 when they are overlapping and <0 when they are disjoint. Thus, a feasible threshold for cluster merging candidates is 0 resp. a high value below 0, allowing only those clusters to be merged which are at least touching each other.

Certainly, when $olap(win,k)$ is high for any k, there is a redundant, highly overlapping situation and the two rules can be merged no matter how the consequent vectors are belonging to each other. In case when the consequents are dissimilar, the two rules indicate an inconsistency in the rule base (e.g. caused by high noise levels in the output), which will be handled within a specific strategy, see Sect. 3.2.2 below.

Continuation of the approximation trend (*criterion #2*) Whenever $olap(win,k)$ is around 0 for any k, the two rules are merging candidates, but the functional trend of the nearby lying rules have to observed first in order to decide whether a merge should be performed or not. This is essential as a different trend indicates a necessary non-linearity contained in the functional relation/approximation between inputs and outputs—see Fig. 5 for an example. In this case, obviously

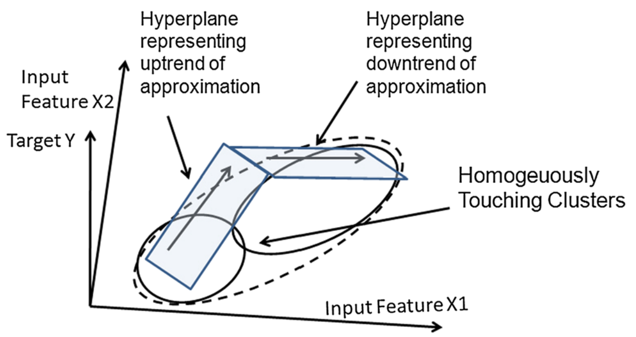
Hyperplane representing uptrend of approximation
Hyperplane representing downtrend of approximation
Input Feature X2
Target Y
Homogeuously Touching Clusters
Input Feature X1

Fig. 5 The corresponding two hyper-planes (consequents) of two touching rules (shown as *ellipsoids*) indicating a necessary non-linearity in the approximation surface; merging this would end up in one rule achieving an undesired constant behavior in that part

two rules cannot be merged to one, as this would not be able to sufficiently resolve the non-linearity any longer.

Thus, we propose an additional similarity criterion based on the degree of deviation in the hyper-planes' gradient information, i.e. in the consequent parameter vectors of the two rules without the intercept. We suggest a criterion based on the dihedral angle of the two hyper-planes they span, which is defined by:

$$\phi = \arccos\left(\frac{\mathbf{a}^T\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}\right) \tag{18}$$

$a = (w_{win;1}\, w_{win;2} \ldots w_{win;p}\ -1)^T$ and $b = (-w_{k;1}\ -w_{k;2} \cdots -w_{k;p}\ +1)^T$ the normal vectors of the two planes corresponding to rules *win* and *k*, showing into the opposite direction with respect to target y (-1 and $+1$ in the last coordinate). If this is close to 180∘ (π), the hyper-planes obviously represent the same trend of the approximation curve, therefore the criterion should be high (close to 1), as the rules can be merged. If it is 90∘ ($\pi/2$) or below a change in the direction of the approximation functions takes place, thus the criterion should be equal or lower than 0.5. Hence, the similarity criterion becomes

$$S_{cons}(\mathbf{w}_{win}, \mathbf{w}_k) = \frac{\phi}{\pi}. \tag{19}$$

We note that range normalization is important in order to obtain comparable impacts of variables in the consequents. This is because their influence on the target is not scale-invariant, and thus the vectors describing the directions of the consequents are affected by the scale. The example shown in Fig. 6 provides a clearer picture of this aspect, as showing a regression surface, given (and estimated) by $y = x_1 + x_2$, where the influence of feature x_1 on the target is obviously much more intense than feature x_2 (due to its range [0, 10] versus [0, 1]). This also means that a direction

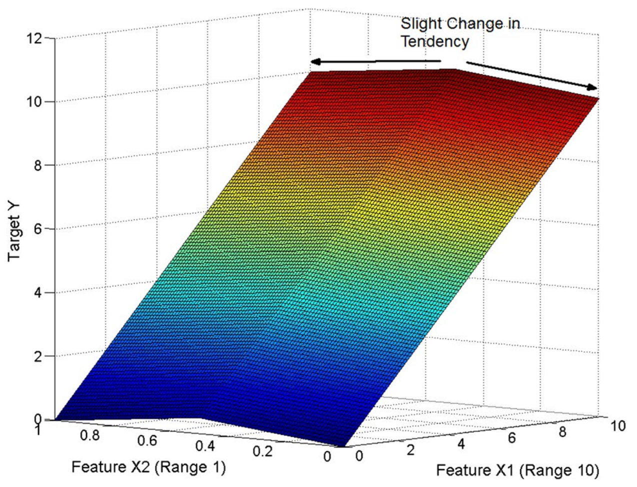

Fig. 6 A change in the sign of the low influencing variable (Feature X2) from 1 to -1 (at around $x_2 = 0.5$) still does not cause a significant change in the tendency of the regression surface, however wrongly triggers a low value of (19) when no range normalization is performed

change in x_2 (from 0.5 on) leading to $y = x_1 - x_2 + 1$ effects the output tendency little, thus should result in a high similarity. However, the similarity in (19) gets low when using the two vectors $(1, 1, -1)$ and $(-1, 1, 1)$ without range normalization. Alternatively, one may multiply the normal vector entries with the corresponding ranges of the features before calculating (18)—in the example above, this would lead to an overweight of feature one in the scalar product, achieving a value close to -1, thus an angle close to π, hence $S_{cons} \approx 1$.

In case of a low angle (18) but high deviation between the intercepts, the two nearby lying planes are (nearly) parallel but with a significant gap inbetween. This usually points to a modeling case of a step-wise function, and thus the rule should not be merged. Hence, we may extend the similarity expressed by (19) to:

$$S_{cons}(\mathbf{w}_{win}, \mathbf{w}_k) = min\left(\frac{\phi}{\pi}, 1 - \frac{\phi}{\pi}\frac{|w_{win;0} - w_{k;0}|}{range(y)}\right) \tag{20}$$

Homogeneity of adjacent rules (*criterion #3*) Finally, we also demand that the two nearby lying/touching rules form a homogeneous shape and direction when joined together. This is important as otherwise a merged rule may reflect a too inaccurate representation of the two local data clouds. An illustration example is shown in Fig. 7, the right image representing a situation where a merge is not suggested. Thus, in order to restrict the merging action to rules triggering homogenous "smart" joint regions, we examine the blow-up effect of the rules when virtually merged together, i.e. we check whether

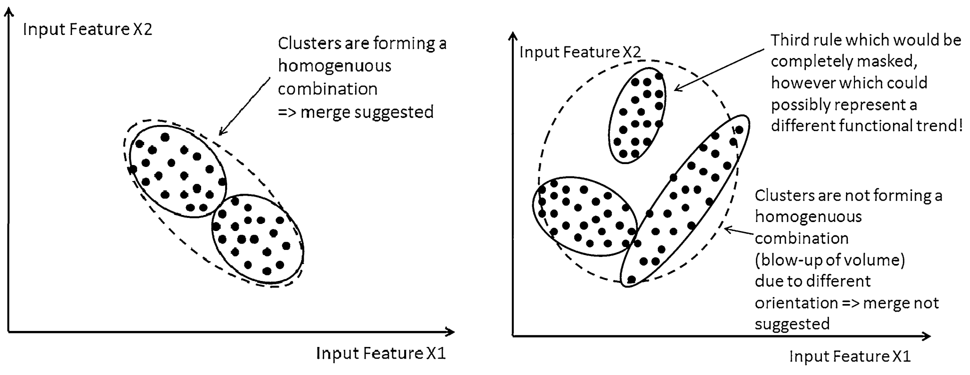
Input Feature X2
Clusters are forming a homogenuous combination => merge suggested
Input Feature X1
Input Feature X2
Third rule which would be completely masked, however which could possibly represent a different functional trend!
Clusters are not forming a homogenuous combination (blow-up of volume) due to different orientation => merge not suggested
Input Feature X1

Fig. 7 *Left* two rules (*solid ellipsoids*) which are touching each other and are homogeneous in the sense that the (volume, orientation of the) merged rule (*dotted ellipsoid*) is in conformity with the original two rules; *right*: two rules (*solid ellipsoids*) which are touching each other and are not homogeneous → inaccurate (too wide) representation of merged rule due to an artificial blow-up, then also covering another rule with a possible different functional local trend

$$V_{merged} \leq p(V_{win} + V_k) \tag{21}$$

where V_k denotes the volume of rule k in the high-dimensional space (Jimenez and Landgrebe 1998):

$$V_k = \frac{2 * \prod_{j=1}^{p} \left(r_k / \lambda_{kj} \right) * \pi^{p/2}}{\Gamma(p/2)} \tag{22}$$

with λ_{kj} the jth eigenvalue corresponding to the jth eigenvector. If (21) holds, the blow-up effect is restricted.

Finally, the joined merging condition is defined by:

$$(Eq.\,(17) \geq thr) | (Eq.\,(17) \geq 0 \wedge Eq.\,(21) \wedge Eq.\,(19) \geq 0.5) \tag{23}$$

with *thr* usually set to 0.8, according to the considerations and experience in Lughofer et al. (2011). The symbol “ ” denotes the conventional “OR” operation, and the symbol “$\wedge$” the conventional “AND”. Hence, in case of a significant overlap, merging is always triggered, whereas in case of touching rules [i.e. $Eq.\,(17) \geq 0$] merging is only triggered when the two clusters form a homogeneous region and their output (consequent) tendency is similar.

3.2.2 *Rule merging policy*

Whenever two rules fulfill condition (23), the merging should be performed in a single-pass, ideally fast, manner, i.e. without using any prior data. Merging of rules antecedents is conducted (1) by using a weighted average of centers, where the weights are defined by the support of the rules, assuring that the new center lies in-between the two centers and is closer to the more supported rule; and (2) by merging directly the inverse covariance matrices also employing a weighted average, but correcting it with a rank-1 modification for achieving more stability:

$$c_j^{new} = \frac{c_j^{win} k_{win} + c_j^k k_k}{k_{win} + k_k} \quad k_{new} = k_{win} + k_k$$
$$\Sigma_{md}^{-1} = \frac{k_{win} \Sigma_{win}^{-1} + k_k \Sigma_k^{-1} + \frac{k_{win} k_k}{k_{win} + k_k} diag\left(\frac{1}{((\mathbf{c}^{new} - \mathbf{c}^i)^T (\mathbf{c}^{new} - \mathbf{c}^i))} \right) * I}{k_{win} + k_k} \tag{24}$$

with $(\mathbf{c}^{new} - \mathbf{c}^i)$ a row vector with $i = argmin(k_{win}, k_k)$, i.e. the index of the less supported rule and *diag* the vector of diagonal entries; the division in the numerator (1/...) (rank-1 modification) is done component-wise for each matrix element. The support of the merged rule is simply the sum of the support of the original rules.

The merging strategy of the rule consequent functions depends on the inconsistency level, that is the level to which the consequents are more dissimilar than the antecedents. It follows the idea of Yager's participatory learning concept (Yager 1990), and is conducted by Lughofer et al. (2011):

$$\mathbf{w}_{new} = \mathbf{w}_j + \alpha \cdot Cons(j,k) \cdot (\mathbf{w}_k - \mathbf{w}_j), \tag{25}$$

where $\alpha = k_k/(k_j + k_k)$ represents the *basic learning rate* and $Cons(j,k)$ the *compatibility measure* between the two rules within the participatory learning context. Here j denotes the index of the more supported rule, i.e. $k_j > k_k$ If (17) is smaller or equal to 0, the consistency degree $Cons(j,k)$ is always set to 1. If it is above 0, there is an overlap. In this case, the consistency degree is measured by a continuous smooth function [leaned on Jin (2000), but modified in a way such that to achieve high consistency degrees in the case of dissimilar antecedents]:

$$Cons(j,k) = e^{-\frac{\left(\frac{S_{rule}(j,k)}{S_{cons}(j,k)} - 1 \right)^2}{\left(\frac{1}{S_{rule}} \right)^7}} \tag{26}$$

with $S_{rule}(j,k) = olap(j,k)$ as in (17) and $S_{cons}(j,k)$ as in (19). Another interpretation of (26) is that the higher the rule overlap, the higher the similarity of the consequents has to be in order to achieve a high consistency degree, which is in-line the classical consistency degree of fuzzy rule bases (Casillas et al. 2003).

Due to (25), the consequent vector of the merged rule is more influenced by the more supported rule when the consistency degree is lower, thus increasing the belief in the more supported rule. In the crisp (boolean) case, *Cons* may be become either 0 or 1, depending whether the rule similarity is higher than the consequent similarity. In case when consistency degree is equal to 1, Eq. (25) becomes either a weighted averaging; in case when it is 0, the vector of the merged rule is set to that one of the more supported rule.. The same merging strategy as in (25) is conducted for the inverse Hessian matrices P_{win} and P_k, which are required as help information for a recursive stable consequent adaptation, see subsequent section.

3.3 Recursive consequent learning

The consequent learning follows the spirit of local learning, employing the RFWLS estimator [as used in most of the conventional EFS approaches (Lughofer 2011)]:

$$\hat{\mathbf{w}}_{\mathbf{i}}(k+1) = \hat{\mathbf{w}}_{\mathbf{i}}(k) + \gamma(k)(y(k+1) - \mathbf{r}^T(k+1)\hat{\mathbf{w}}_{\mathbf{i}}(k)) \tag{27}$$

$$\gamma(k) = \frac{P_i(k)\mathbf{r}(k+1)}{\frac{\lambda_i}{\Psi_i(\mathbf{x}(k+1))} + \mathbf{r}^T(k+1)P_i(k)\mathbf{r}(k+1)} \tag{28}$$

$$P_i(k+1) = \frac{1}{\lambda_i}(I - \gamma(k)\mathbf{r}^T(k+1))P_i(k) \tag{29}$$

with $P_i(k) = (R_i(k)^T Q_i(k) R_i(k))^{-1}$ the inverse weighted Hessian matrix, $\mathbf{r}(k+1) = [1\, x_1(k+1)\, x_2(k+1) \ldots x_p(k+1)]^T$ the regressor values of the $k+1$th data sample, and λ_i a forgetting factor, usually (as not denoted otherwise) set to 1 for all rules (no forgetting). Local learning for each rule separately has the favorable properties (1) of providing more flexibility in terms of automatic rule inclusion and deletion than a global estimator and (2) of being faster and more stable than global learning, as analyzed in detail in Angelov et al. (2008) and Lughofer (2011) (Chapter 2). The most important point of local learning, however, is that it induces hyper-planes which are snuggling along the real trend of the approximation surface (Lughofer 2013), which will be exploited in the subsequent section for feature weighting purposes.

Moreover, whenever a new rule is evolved, then the consequent parameters $\mathbf{w_{C+1}}$ and inverse Hessian matrix of the new rule are set to those ones of the nearest rule, i.e.

$$\mathbf{w}_{C+1} = \mathbf{w}_{nearest} \quad P_{C+1} = P_{nearest} \tag{30}$$

with *nearest* the index of the nearest rule in terms of Mahalanobis distance, thus assuring a continuation of the local trend of the nearest rule.

4 On-line feature weighting for evolving smart regression

Feature weighting within the environment of generalized EFS is motivated by mainly two aspects:

- Measuring the importance levels/degrees of features in different parts of the features space (modeled by different rules), addressing another important interpretability criterion, namely the input/output interpretation which features really fired for the current prediction at hand (Lughofer 2013); even more importantly, the transparency and compactness of the rule base can be increased, whenever feature weights get low: the corresponding features can be eliminated from both, the antecedent and consequent parts of the rules. This reduces the length of AND-connections and improves comprehensibility for users and operators. This, together with the rule pruning concepts explained in the preliminary sections, finally provides the smartness aspect in generalized EFS.
- Reducing the curse of dimensionality effect in case of medium- to high-dimensional problems in a kind of soft and smooth manner: *softness* here means that features can be down-weighted, but not completely discarded (as still having a low weight and impact); *smoothness* addresses slight continuous changes of feature weights, thus down-weighted features may also become significantly reactivated at a later stage of the stream learning process. This abandons both, a crisp selection and the forever elimination of features [as done in Pratama et al. (2014)].

4.1 Monitoring feature importance levels

In regression problems, it is well known that the impact of a feature onto the target concept, i.e. how a change in a feature affects the output in which way, can be tracked by observing the gradient of the feature (Efendic and Re 2006). If the gradient is high, a small change in the feature effects the output already drastically. Usually, in case of non-linear approximation functions, the gradient changes in each point, thus an overall gradient of a feature is time-intensive to compute. TS fuzzy models, however, offer the possibility to track piece-wise local linear predictors over the feature space. This is because each rule is equipped with a hyper-plane showing the tendency in the neighborhood of the rule. Especially, when using the local learning option (per rule) as recursively given by (27)–(29), the hyper-planes are snuggling along the real trend of the surface, as examined in Lughofer (2013).

In this sense, we can directly use the consequent parameter vector $\mathbf{w}_i$ for the local gradients of all features around the ith rule. Thus, in case when using normalized data, the local impact λ_{ij} of feature j in the local region approximated by rule i is given by:

$$\lambda_{ij}(loc) = \frac{w_{ij}}{\sum_{j=1}^{p} w_{ij}}. \tag{31}$$

Normalization through the sum is done to obtain relative weights between [0, 1], which are comparable and interpretable among the features (1 = fully important, 0 = fully unimportant). When data is not normalized, we have to multiple the gradients with the feature ranges to obtain the real impact, as already motivated in Sect. 3.2.1 (Fig. 6). Hence, the local impact λ_{ij} of feature j is then given by:

$$\lambda_{ij}(loc) = \frac{w_{ij} * range(j)}{\sum_{j=1}^{p} .w_{ij} * range(j)} \tag{32}$$

In order to elicit global feature weights, an obvious approach would be to sum up all local feature weights and normalize it by the number of rules in the currently evolved fuzzy system. However, then we assume that all rules are equally supported by past data samples respectively possess the same statistical contribution in the system. In order to respect rule importance levels, we apply the concept introduced in Rong et al. (2006) and adopt it to the case of generalized rules. There, the expected statistical contribution of rule i to the final model output when the number of stream samples N goes to infinity is considered. It is given by:

$$Cont_i = \lim_{N\to\infty} |w_{i0}| \frac{\sum_{k=1}^{N} \Psi_i(\mathbf{x}_k)/N}{\sum_{j=1}^{C}\sum_{k=1}^{N} \Psi_j(\mathbf{x}_k)/N}. \tag{33}$$

We are only interested in the input contribution, thus neglect $|w_{i0}|$ in (33). The central challenge is to calculate $E_i = \lim_{N\to\infty} \sum_{k=1}^{N} \Psi_i(\mathbf{x}_k)/N$, which can be in our case of generalized fuzzy rules achieved through:

$$E_i = \int_X exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_i)^T \Sigma_i^{-1}(\mathbf{x}-\mathbf{c}_i)\right)p(x)dx \tag{34}$$

with $p(x) = \prod_{j=1}^{p} p_j(x_j)$ and $p_j(x_j)$ the density distribution function of the jth feature. When assuming uniform sampling distributions (samples come equally distributed over their ranges), we obtain $p(x) = 1/S(X)$ with $S(X)$ the size of the range X. Hence, we obtain:

$$E_i = \int_X exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_i)^T \Sigma_i^{-1}(\mathbf{x}-\mathbf{c}_i)\right)\frac{1}{S(X)}dx \tag{35}$$

As the center $\mathbf{c}_i$ does not influence the final value of the integral (area of the rule), but just represents a shift of the multivariate Gaussian distribution, Eq. (35) can be written as:

$$E_i = \frac{1}{S(X)}\int_X exp\left(-\frac{1}{2}(\mathbf{x})^T \Sigma_i^{-1}(\mathbf{x})\right)dx \tag{36}$$

As the integral over the multivariate Gaussian distribution with the origin as center, defined as $\frac{1}{\sqrt{(2\pi)^p * det(\Sigma_i)}} exp(-\frac{1}{2}(\mathbf{x})^T\Sigma_i^{-1}(\mathbf{x}))$, is equal to 1, we obtain:

$$E_i = \frac{det(\Sigma_i)^{1/2}(2\pi)^{p/2}}{S(X)}. \tag{37}$$

Hence, by substituting (37) into (33) the contribution of Rule i on the input side finally becomes:

$$Cont_i(inp) = \frac{\sqrt{det(\Sigma_i)}(2\pi)^{p/2}}{\sum_{j=1}^{C}\sqrt{det\Sigma_j}(2\pi)^{p/2}}. \tag{38}$$

Now the global feature weights are defined by a weighted average of the local weights, where the weights in the average are given by the relative contributions of the rules according to (38):

$$\lambda_j(global) = \frac{\sum_{i=1}^{C} Cont_i(inp)\lambda_{ij}(loc)}{\sum_{i=1}^{C} Cont_i(inp)}. \tag{39}$$

It is easy to see that $\sum_{j=1}^{p}\lambda_j(global) = 1$. Hence, in high-dimensional problems, most of the weights are expected to be low. Thus, for monitoring purposes we suggest to observe the feature weights in relation to the maximal weight (most important feature), i.e. by:

$$\begin{aligned}\lambda_j(global) &= \frac{\lambda_j(global)}{\max_{i=1,\ldots,p}(\lambda_i(global))}\\ \lambda_{ij}(loc) &= \frac{\lambda_{ij}(loc)}{\max_{k=1,\ldots,p}(\lambda_{ik}(loc))}.\end{aligned} \tag{40}$$

This guarantees that the most important feature(s) have still a significant impact in the learning phase (otherwise, nothing will be learned, anymore). Then, those features can be ignored whose weights are below X % of the maximal weight.

4.2 Integrating feature importance levels (on-line curse of dimensionality reduction)

The second issue concerns the integration of the features weights into the incremental learning engine of EFS. This may have a significant impact on the stability and accuracy of the evolving fuzzy models: especially, in case of high-dimensional feature spaces, an out-weighting of features

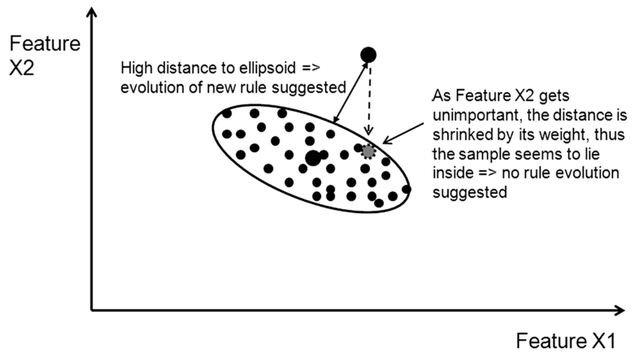
Feature X2
High distance to ellipsoid => evolution of new rule suggested
As Feature X2 gets unimportant, the distance is shrinked by its weight, thus the sample seems to lie inside => no rule evolution suggested
Feature X1

Fig. 8 No rule evolution suggested although sample lies outside the tolerance region of a rule, as Feature X2 is unimportant, thus the distance of a new sample to the *ellipsoid* shrinked

(e.g., in distance calculations, rule evolution criteria etc.) may decrease the curse of dimensionality effect (Hastie et al. 2009) by localizing and densifying. For instance, when calculating the Mahalanobis distance to already existing clusters/rules, unimportant features with low weights should lead to small distances, even for samples which seem to appear outside the region of influence of a rule—an example is provided in Fig. 8. This has the effect that the rule evolution criterion will be triggered less frequent than in the fully weighted high-dimensional feature space, omitting unnecessary complexity and over-fitting due to unimportant features.

The integration into the learning process is motivated by Fig. 8, i.e. the aim is to oppress rule evolution cases whenever unimportant features trigger high values of the Mahalanobis distance, whereas important features are not significantly influencing it. Hence, we are aiming for a "weighted" version of the Mahalanobis distance in (48) in order to reduce the influence of unimportant features. An ad-hoc calculation, leaned on the weighted Euclidean distance (Rao 2013), would integrate the feature weights in the following way:

$$mahal = \sqrt{(\lambda. * (\mathbf{x} - \mathbf{c}_i))\Sigma_i^{-1}(\lambda. * (\mathbf{x} - \mathbf{c}_i))} \quad (41)$$

which could be interpreted as a down-weight of component-wise distances $(\mathbf{x} - \mathbf{c}_i)$. However, in a closer look, it turns out that this induces an undesired rotation of the real ellipsoidal cluster contours by $D_\lambda = diag(\lambda)$. This had the consequence that in various trial-and-error test runs we could observe increasing values of (41) (due to the rotational effect), although some feature weights were nearly dragged down to 0, leading to an undesired non-monotonic behavior.

On the other hand, according to the singular value decomposition, the inverse covariance matrix can be represented as follows:

$$\Sigma_i^{-1} = V * D * V^T \quad (42)$$

with V the matrix containing all eigenvectors as columns and D a diagonal matrix containing the eigenvalues; note that $V^T = V^{-1}$ as V is orthogonal. Now, if we would have the weights for the eigenvectors $\lambda*$ available, we could simply use the re-scaled inverse covariance matrix given by

$$\Sigma_i^{-1}(rescaled) = V * diag(\lambda*) * D * diag(\lambda*) * V^T \quad (43)$$

in the rule evolution criterion (10), yielding the desired effect. However, due to the local geometric interpretation based on the linear hyper-planes (steepness), we are only able to extract the importance levels for the original features (original coordinate space). Thus, we are aiming for a transformation of feature weights λ in the original space to the feature weights $\lambda*$ into the respective rotated space of the corresponding rule. The scoring option, i.e. $\lambda* = V^T * \lambda$, would lead to a similar rotational effect as in (41), sometimes not guaranteeing monotonicity. However, it can easily be verified that (as $V^T = V^{-1}$) (41) can be written as

$$mahal = \sqrt{(\mathbf{x} - \mathbf{c}_i)\Sigma_{i,fweight}^{-1}(\mathbf{x} - \mathbf{c}_i)} \quad (44)$$

with

$$\begin{aligned}\Sigma_{i,fweight}^{-1} &= diag(\lambda)\Sigma_i^{-1}diag(\lambda) \\ &= V[V^T diag(\lambda)V]D[V^T diag(\lambda)V]V^T\end{aligned} \quad (45)$$

according to (42). This can be rephrased into the form

$$\Sigma_{i,fweight}^{-1} = V\Lambda_* D\Lambda_* V^T \quad (46)$$

with $\Lambda_* = V^T * diag(\lambda) * V$. However, this matrix is not diagonal, so this explains why (41) can lack monotonicity. Trying to preserve monotonicity as well as interpretability of feature weights in (41), we thus are led to use a compromise: we choose the same matrix Λ_* as in (46), diagonalized. This leads to an approximation of the scaling in (41) resp. (44):

$$diag(\lambda*) = diag(V^T * diag(\lambda) * V) \quad (47)$$

(47) is then used in (43) to obtain the re-scaled inverse covariance matrix, which is finally used in (48) for the rule evolution criterion (comparison with r_i).

In general, the approximation is an "under-estimator", leading to underestimated re-scaled inverse covariance matrices and further to under-estimated Mahalanobis distances. Thus, we are integrating a scale factor $\delta \in [1.0, 1.5]$ to boost a bit the distance, and then the rule evolution criterion (in both cases, local and global feature weights) becomes:

$$\min_{i=1,\dots,C}\left(\delta * \sqrt{(\mathbf{x}-\mathbf{c}_i)^T\Sigma_i^{-1}(rescaled)(\mathbf{x}-\mathbf{c}_i)}\right) > r_i \tag{48}$$

with r_i as in (11). We used $\delta = 1.35$ in all our experiments.

Integration of feature weights in all other adaptive learning components (i.e. into the update formulas for centers and inverse covariance matrices) is neglected, as it is aimed to achieve the best position of cluster centers and ellipsoidal contours following the real characterization of the data clouds. For instance, not adapting a center in the direction of an unimportant feature may lead to misplacement of centers, i.e. becoming to lie in a part of the region where no samples have occurred at all.

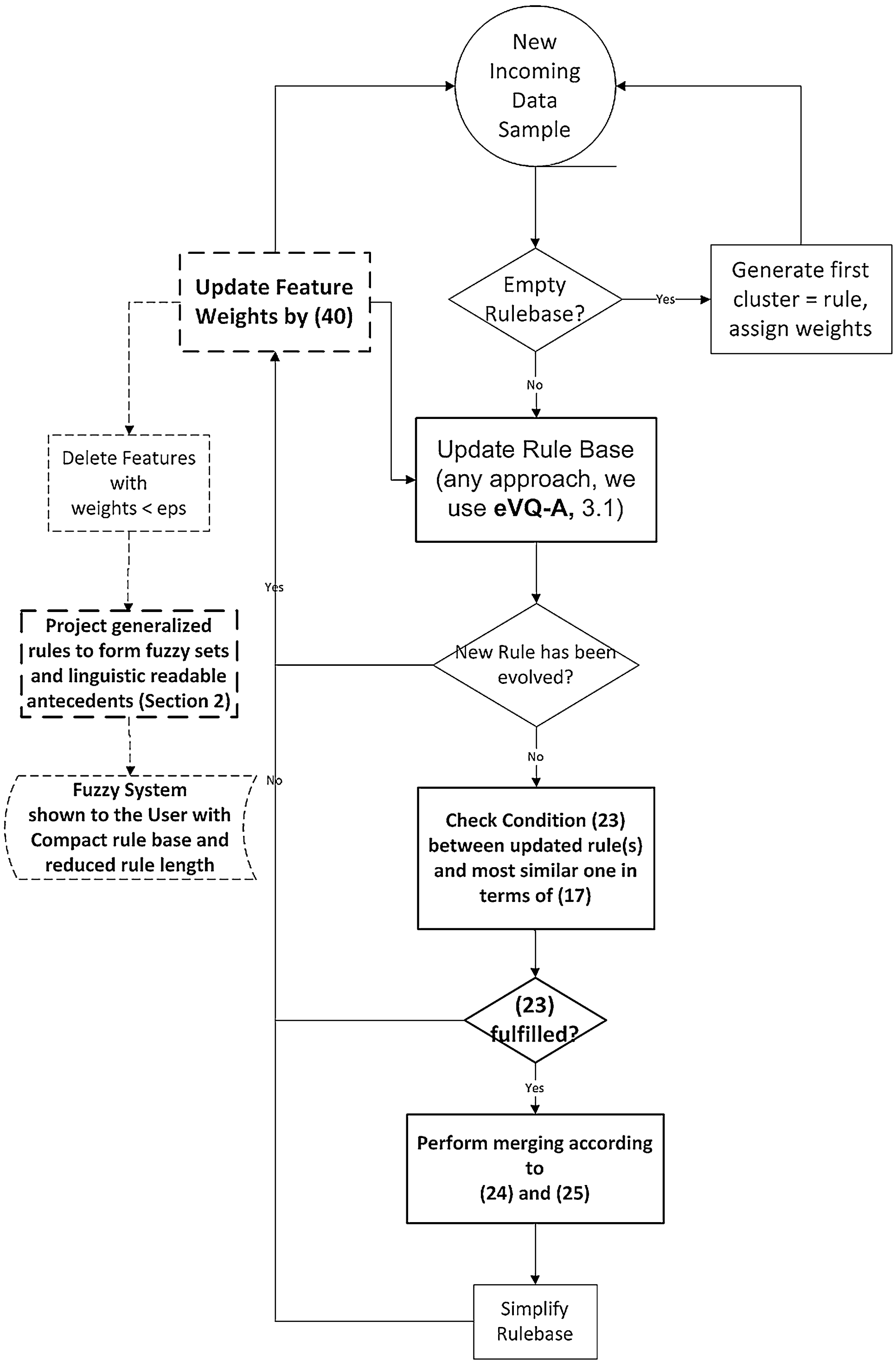
New Incoming Data Sample
Empty Rulebase?
Yes
Generate first cluster = rule, assign weights
No
Update Rule Base (any approach, we use eVQ-A, 3.1)
New Rule has been evolved?
Yes
No
Check Condition (23) between updated rule(s) and most similar one in terms of (17)
(23) fulfilled?
No
Yes
Perform merging according to (24) and (25)
Simplify Rulebase
Update Feature Weights by (40)
Delete Features with weights < eps
Project generalized rules to form fuzzy sets and linguistic readable antecedents (Section 2)
Fuzzy System shown to the User with Compact rule base and reduced rule length

Fig. 9 Flowchart *GS-EFS* (Generalized Smart Evolving Fuzzy Systems), the components highlighted in *bold font* are the new suggested ones in this paper, and can be connected with an arbitrary learning engine for EFS employing generalized rules

Table 1 Some data sets from the UCI repository and engine test bench and their characteristics

	# Training samples	# Input variables	Source	Noise level
Auto-MPG	398	8	UCI	None
Concrete	1030	8	UCI	None
Housing	506	13	UCI	None
NO_X	667	181	Engine test bench	Medium to high

5 Generalized smart EFS: the algorithm

The flowchart of our algorithm termed as *Gen-Smart-EFS* is shown in Fig. 9, where the newly introduced components are shown in bold font and can be connected with any evolving learning engine for EFS employing generalized rules: this is because the complexity reduction techniques fully operate on a geometrical basis based on the tendency, shape and outlook of the current evolved fuzzy rules, and because the incremental feature weighting concept operate on a combined geometric-statistical-oriented concept, where only structural information like the rule representation (Σ-matrix) and the consequent parameters need to be available. Simplifying the rule-base (bottom component in Fig. 9) is a software-technical issue, by making the corresponding virtual memory space free for the merged rule.

6 Evaluation

6.1 Experimental setup

The evaluation of the new methodologies will be divided into three parts:

- Three data sets from the UCI repository[2] (clean, noise-free): auto-mpg, housing and concrete; one data set from an engine test bench (own application project), where measurement data were collected on-line and has been affected by some white noise and disturbances: the task is to predict the NO_X content based on other synchronously recorded measurement channels and their time lags ($\rightarrow$ virtual sensor). The data sets are summarized in Table 1. For these four data sets, the same evaluation scheme as conducted in Lughofer and Kindermann (2010) is performed, where original *FLEXFIS* (Lughofer 2008) was compared with other batch off-line data-driven fuzzy systems extraction methods (as hard benchmark). The evaluation scheme follows a cross-validation procedure coupled with best parameter grid search to find the optimal setting (that one achieving the lowest CV-error) of the most sensitive parameters in each method. In case of all incremental/adaptive modeling variants, the models are sample-wise single-pass evolved on each combination of folds (and tested versus the remaining fold).
- High-dimensional non-linear system identification problem as tested on several EFS approaches in Lemos et al. (2011). Here, the task is to identify the following problem:

$$y(t) = \frac{\sum_{i=1}^{m} y(t-i)}{1+\sum_{i=1}^{m} y(t-1)^2} + u(t-1) \tag{49}$$

where $u(t) = sin(2\pi k/20)$ and $y(t) = 0$ for $j = 1, \ldots, m$, m set to 10. The purpose is to predict the output variable y from past inputs and own lagged outputs, thus:

$$\hat{y}(t) = f(y(t-1), y(t-2), \ldots, y(t-10), u(t-1)) \tag{50}$$

The first 3000 samples were created for data stream learning, additional 300 samples were generated for evaluation the model on a separate test data set. As error measure the root mean squared error has been used for the comparison with the approaches in Lemos et al. (2011) and with other adaptive, evolving variants such as OS-ELM (Liang 2006) and FAOS-PFNN (Wang et al. 2009).
- Real-world application: automatic prediction of resistance value in rolling mill processes. In particular, the task is to identify a prediction model for the resistance value of a steel plate in a rolling mill. 11 measurement variables are recorded per second additionally to the resistance value, which are time delayed up to a maximal lag of 10. In this sense, we are including prediction horizons of maximal 10 s. In sum, two data sets with 6503 resp. 6652 samples at two different points of time (2 months inbetween) have been recorded and stored in the same order as they appeared on-line. The first data set has been used as training data stream based on which the models are evolved, the other data set as independent test set. Original results when using conventional EFS (in particular the *FLEXFIS* approach) without any generalization, pruning and feature selection option have been already reported in Lughofer and Angelov (2011) and serve as benchmark for *Gen-Smart-EFS* (*GS-EFS*) method.

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

Additionally to the model errors (in form of mean absolute error = MAE and root mean squared error = RMSE), also a statistical preference analysis on the residual vectors will be performed and reported in separate preference tables, see subsequent section. This is achieved with the usage of a non-parametric Wilcoxon pair-wise comparison tests (Hill and Lewicki 2007) coupled with Holm–Bonferroni test in order to reduce the family wise error rate (FWER) (Holm 1979). Model complexities will be reported in terms of the number of extracted components (rules, kernels, neurons), which are comparable over all methods we apply as serving as localizers of the feature space, indicating the degree of granulation and non-linearity of the models (more complex models with similar errors are generally expected to be less robust regarding over-fitting).

6.2 Results

6.2.1 Results on data from UCI repository

Table 2 summarizes the results achieved on the three UCI repository data sets and the virtual NO_X sensor problem for predicting its value from other sensors installed within the engine test bench. The table includes the minimal cross-validation error in terms of mean absolute error between predicted and observed values, together with the standard deviation over the CV-folds (after the ± symbol), the maximal error between predicted and observed over all training data samples and the average model complexity corresponding to the minimal CV error in terms of the average number of rules over the CV-folds. The first parts of the two tables (before the horizontal lines) represent the batch modeling variants for fuzzy system modeling, the second parts correspond to the incremental, evolving variants.

Clearly, it can be recognized that our *Gen-Smart-EFS*, is able to outperform conventional *FLEXFIS* (with no pruning and axis-parallel rules) in three out of four cases (Auto-MPG, Concrete, Housing), as achieving lower CV errors with smaller complexities, especially when using the extended pruning option which is triggered by the full condition in (23). If using the conventional pruning option which is the application of only the first part of (23) (before the OR), the number of rules increases and are even

Table 2 Comparison of fuzzy modelling variants on three data sets from UCI repository and (noisy) NO_X data from an engine test bench

Method	Auto-MPG			Concrete		
	MAE ± STD	Max	# Rules	MAE ± STD	Max	# Rules
genfis2 (Yager and Filev 1994)	2.23 ± 0.85	3.88	6	8.37 ± 1.70	11.52	3
ANFIS (Jang 1993)	2.41 ± 0.84	4.07	16	11.25 ± 9.98	39.13	8
FMCLUST (Babuska 1998)	2.35 ± 0.91	3.99	20	7.75 ± 2.06	12.11	3
SparseFIS (Lughofer and Kindermann 2010)	2.01 ± 0.55	3.29	9	7.73 ± 1.77	21.01	7
SparseFIS uncon. (Lughofer and Kindermann 2010)	2.14 ± 0.78	4.08	22	8.05 ± 1.85	12.52	11
OS-ELM (Liang 2006)	4.45 ± 2.21	8.4	100	10.87 ± 1.23	12.68	46
FAOS-PFNN (Wang et al. 2009)	6.23 ± 1.66	9.8	144.6	7.13 ± 1.3	7.34	25.2
FLEXFIS (Lughofer 2008)	2.17 ± 0.73	3.59	11	7.73 ± 1.97	11.96	8
Gen-Smart-EFS no pruning	2.09 ± 0.62	3.60	5.8	6.04 ± 0.23	6.47	9.9
Gen-Smart-EFS conv. pruning	2.09 ± 0.62	3.60	5.8	5.99 ± 0.41	6.44	9.6
Gen-Smart-EFS ext. pruning	2.09 ± 0.62	3.60	5.8	6.35 ± 0.52	7.14	3.8
Method	**Housing**			**NO_X**		
	MAE ± STD	Max	# Rules	MAE ± STD	Max	# Rules
genfis2 (Yager and Filev 1994)	3.14 ± 1.31	6.53	4	12.97 ± 1.08	15.98	5
ANFIS (Jang 1993)	3.59 ± 1.39	6.56	4	13.72 ± 1.01	14.72	8
FMCLUST (Babuska 1998)	2.84 ± 1.08	5.38	6	13.99 ± 1.05	15.55	6
SparseFIS (Lughofer and Kindermann 2010)	3.02 ± 1.19	6.04	6	11.35 ± 0.97	13.63	3
SparseFIS uncon. (Lughofer and Kindermann 2010)	3.60 ± 1.56	6.86	19	14.01 ± 1.81	16.68	9
OS-ELM (Liang 2006)	3.42 ± 1.60	7.29	17	19.31 ± 1.83	21.54	36
FAOS-PFNN (Wang et al. 2009)	4.05 ± 1.75	7.12	20.3	24.94 ± 3.73	31.59	32.6
FLEXFIS (Lughofer 2008)	2.98 ± 1.27	6.09	6	12.96 ± 0.98	14.31	8
Gen-Smart-EFS no pruning	2.91 ± 1.18	5.57	11	12.91 ± 1.31	15.89	10.1
Gen-Smart-EFS conv. pruning	2.94 ± 1.18	5.63	7.6	13.08 ± 1.15	15.75	8.9
Gen-Smart-EFS ext. pruning	2.94 ± 1.18	5.84	4.9	12.91 ± 1.31	15.89	10.1

higher than in case of conventional *FLEXFIS* for concrete and housing data. Furthermore, the *Gen-Smart-EFS* in all forms (without and with pruning options) seems to outperform other on-line learning methods such as *OS-ELM* and *FAOS-PFNN*, and even compete with the batch modelling methods, which is remarkable as it evolves sample-wise on the CV-based training folds, rather see all the training folds at once.

In order to confirm this first impression from the readings in Table 2, we performed a statistical significance test

Table 3 Statistical preferences of one method (row) over another (column) based on the residual vectors over time for *UCI repository and NO_X emission data sets*; statistical preference (p value of 0.05) is denoted by '+', strong preference (p value of 0.025) is denoted by '++', whereas minus '−' and '−−' indicate the opposite

Set/level	genfis2	ANFIS	FMCLUST	SparseFIS	OS-ELM	FAOS-PFNN	FLEXFIS	GS-EFS ext. prune
Auto-MPG								
genfis2 (Yager and Filev 1994)	0	0	0	0	++	++	0	0
ANFIS (Jang 1993)	0	0	0	−	++	++	0	−
FMCLUST (Babuska 1998)	0	0	0	−	++	++	0	−
SparseFIS (Lughofer and Kindermann 2010)	0	+	+	0	++	++	0	0
OS-ELM (Liang 2006)	−−	−−	−−	−−	0	++	−−	−−
FAOS-PFNN (Wang et al. 2009)	−−	−−	−−	−−	−−	0	−−	−−
FLEXFIS (Lughofer 2008)	0	0	0	0	++	++	0	0
GS-EFS ext. prune	0	+	+	0	++	++	0	0
Concrete								
genfis2 (Yager and Filev 1994)	0	++	0	0	+	−	0	−−
ANFIS (Jang 1993)	−−	0	−−	−−	0	−−	−−	−−
FMCLUST (Babuska 1998)	0	++	0	0	++	0	0	−
SparseFIS (Lughofer and Kindermann 2010)	0	++	0	0	0	++	0	−
OS-ELM (Liang 2006)	−	0	−−	−−	0	−−	−−	−−
FAOS-PFNN (Wang et al. 2009)	+	++	0	0	++	0	0	0
FLEXFIS (Lughofer 2008)	0	++	0	0	++	0	0	−
GS-EFS ext. prune	++	++	+	+	++	0	+	0
Housing								
genfis2 (Yager and Filev 1994)	0	0	0	0	0	+	0	0
ANFIS (Jang 1993)	0	0	−	0	0	0	0	−
FMCLUST (Babuska 1998)	0	+	0	0	0	+	0	0
SparseFIS (Lughofer and Kindermann 2010)	0	0	0	0	0	+	0	0
OS-ELM (Liang 2006)	0	0	−	0	0	0	0	0
FAOS-PFNN (Wang et al. 2009)	−	0	−	−	0	0	−	−
FLEXFIS (Lughofer 2008)	0	0	0	0	0	+	0	0
GS-EFS ext. prune	0	+	0	0	0	+	0	0
NO_X								
genfis2 (Yager and Filev 1994)	0	0	0	−	++	++	0	0
ANFIS (Jang 1993)	0	0	0	−	++	++	0	0
FMCLUST (Babuska 1998)	0	0	0	−	++	++	0	−
SparseFIS (Lughofer and Kindermann 2010)	+	+	+	0	++	++	+	0
OS-ELM (Liang (2006))	−−	−−	−−	−−	0	+	−−	−−
FAOS-PFNN (Wang et al. 2009)	−−	−−	−−	−−	−−	0	−−	−−
FLEXFIS (Lughofer 2008)	0	0	0	-	++	++	0	0
GS-EFS ext. prune	0	0	+	0	++	++	0	0

Zero (0) denotes no statistical preference

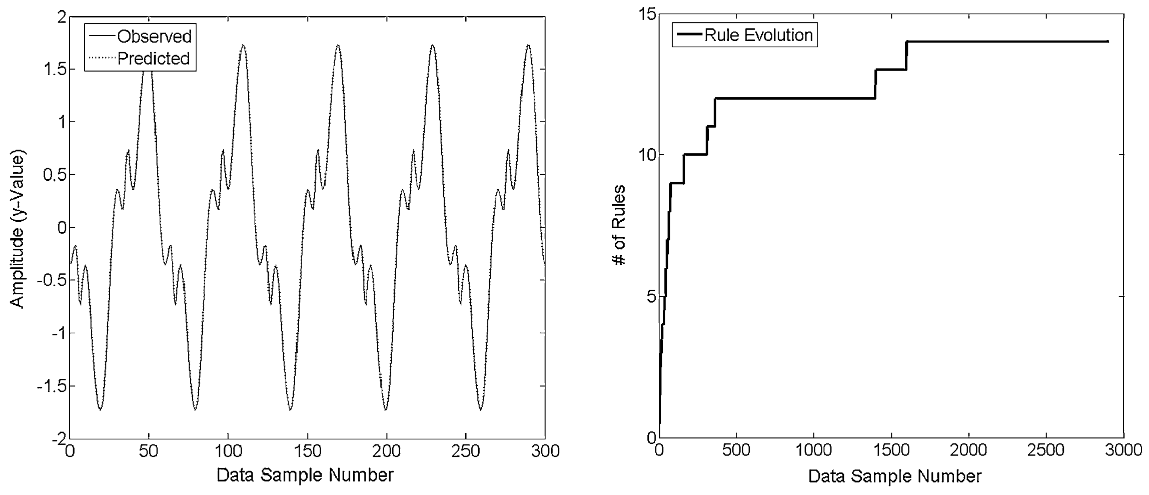

Fig. 10 *Left* observed versus predicted values on the high-dimensional non-linear system identification problem; *right* the rule evolution over time

with the usage of a non-parametric Wilcoxon pair-wise comparison tests (Hill and Lewicki 2007) coupled with Holm–Bonferroni test in order to reduce the FWER (Holm 1979). This is performed on the left out folds used as separate test data sets within the CV procedure. The statistically significant preferences are shown in Table 3, indicated by '+' whenever a method denoted in the row is preferred over the method denoted in the column with a significance value of $p = 0.05$ (5 % probability that hypothesis that two methods are equal is wrongly rejected) and indicated by '++' whenever the preference is more significant with a p value of 0.025. From this statistical analysis, it can be seen that our proposed method *GS-EFS* is clearly the best evolving method for all data sets as reaching scores of 6, 9, 2 and 5 for the four data sets, whereas the second best method (FLEXFIS) achieves scores of 4, 3, 1 and 3 ('++' equals a score of 2, '+' a score of 1, '−−' and '−' the negative of it). In case of concrete data, it is also the best over all batch modeling methods and equal to *SparseFIS* and *FMCLUST* in case of Auto-MPG and Housing, respectively; only in case of NO_X data, *SparseFIS* can outperform our method in terms of the score, however this requires multiple iterations in optimization cycles over the complete data set and thus is much slower than *GS-EFS* (as also been designed for batch modelling problems). Most importantly, *GS-EFS* is never outperformed by any other method for all four data sets (no '−' sign in any of the rows respresenting "GS-EFS ext. prune"), but on the other hand outperforms various other methods on different sets. As it also ends up with the less complex models in three cases and having a similar complexity as FLEXFIS on NO_X data, we can conclude that it is the best choice in terms of "minimal description length" (best fit with lowest possible model complexity) over all data sets.

6.2.2 *Results on dynamic on-line system identification problem*

The results in terms of prediction capabilities on an on-line data stream are shown in Fig. 10, the left image shows the observed versus predicted values on the separate test data set: both are lying over each other. The rule evolution plotted on the right hand side: at the beginning of the stream, many rules are evolved, but after some time a saturation takes place. No rules have been pruned for this data set, thus the rules have been generated distinctively enough by *Gen-Smart-EFS* and obviously did not move together sufficiently over time. Furthermore, Table 4 performs a comparison of two parametrization variants of *Gen-Smart-EFS* with the other evolving fuzzy modeling variants tested on this problem in Lemos et al. (2011) and with widely used OS-ELM (Liang 2006) and FAOS-PFNN (Wang et al. 2009). Clearly, the new method can outperform state-of-the-art EFS approaches in terms of RMSE and also model complexity (8 vs. 13 rules, achieving a slightly lower RMSE than eMG). Interestingly, also to see the effect of the integration of feature weights: in both parametrization cases ($fac = 0.45$ and $fac = 0.6$), 1 resp. 2 rules less are evolved due to the shrinkage of an unimportant direction in the evolution criterion, not increasing the RMSE; in fact, in case of $fac = 0.45$, the RMSE is even slightly decreased, thus lowering curse of dimensionality.

A further comparison is conducted with another well-known incremental learning algorithm, namely with the so-called *OS-ELM* as proposed in Liang (2006). This method opens the possibility to use four different shapes for the neurons such as radial basis functions, sigmoid functions, sine functions or hard-lim functions; we used the best option out of these for each result (row) as shown in Table 4.

The different results by parameterizing different numbers of hidden neurons (second column) indicate that OS-ELM needs around five times more structural components (50 in number) to achieve approximately the same model error as *Gen-Smart-EFS*. The accuracy of FAOS-PFNN (Wang et al. 2009), another important and widely used evolving modeling variant, is pretty good and can compete with the best parametrization variants of Gen-Smart-EFS, however requires more structural components (70 versus 12–14). When reducing the complexity by increasing the overlap factor k for the RBF units and increase β towards 1, the error increases: in case of 12 rules, the error is the same as for Gen-Smart-EFS in case of 8/9 rules and higher than for Gen-Smart-EFS in case of the same number of rules.

Table 4 Comparison of RMSE and final number rules on high-dimensional dynamic system identification problems, preferred choices shown in bold font

Method	# of rules/neurons	RMSE
xTS (Angelov and Zhou 2006)	9	0.0331
eMG ($\Sigma_{init} = 2 \times 10^{-1} I_{11}$) (Lemos et al. 2011)	9	0.0288
FLEXFIS (Lughofer 2008)	15	0.0085
eTS (Angelov and Filev 2004)	14	0.0075
eMG ($\Sigma_{init} = 10^{-1} I_{11}$) (Lemos et al. 2011)	13	0.0050
FAOS-PFNN (Wang et al. 2009), k = 1.15	70	0.002
FAOS-PFNN (Wang et al. 2009), k = 2.0	39	0.003
FAOS-PFNN (Wang et al. 2009), k = 4.0	29	0.0035
FAOS-PFNN (Wang et al. 2009), k = 4.0, $\beta = 0.997$	12	0.0042
FAOS-PFNN (Wang et al. 2009), k = 4.0, $\beta = 0.999$	6	0.0073
OS-ELM (Liang 2006)	8	0.0462
OS-ELM (Liang 2006)	12	0.0347
OS-ELM (Liang 2006)	20	0.0229
OS-ELM (Liang 2006)	50	0.004
Gen-Smart-EFS ($fac = 0.6$)	**9**	**0.0042**
Gen-Smart-EFS + feat. weights ($fac = 0.6$)	**8**	**0.0042**
Gen-Smart-EFS ($fac = 0.45$)	**14**	**0.0020**
Gen-Smart-EFS + feat. weights ($fac = 0.45$)	**12**	**0.0018**

In order to manifest and confirm our loose considerations based on the final RMSE values, we performed again a statistical preference test on the residuals (observed vs. predicted) obtained on the independent test set (same test variant as in the preliminary subsection). We selected for each method the best variant, i.e. those variant leading to the smallest RMSE as this corresponds to the residual vectors with lowest entries. The results are presented in Table 5. From this table it gets immediately clear that FAOS-PFNN and the new method *GS-EFS* using smooth feature weights integration can be significantly preferred among the others, as achieving a score of 8 ('++' count as 2 score point, one '+' as 1 score points, for the minus signs the negative), while the third best method (OS-ELM) reaches just a score of 0. When now comparing these two methods with respect to the final model complexity, *GS-EFS* achieves 12 rules while FAOS-PFNN requires 70 rules to achieve the same approximation quality, and this then also (unsurprisingly) with a much lower computation speed as will be shown below in Table 6.

Figure 11 shows the global weights of the 11 input features in the same order as included in the functional definition in (50): the three curves correspond to the weights at the beginning of the learning process (after 500 samples of the stream) as dotted line, in the middle of the learning process (after around 1500 samples) as dashed line and at the end of the learning process, after the stream terminated (3000 samples) as solid line. Obviously, some features which are more important at the beginning turn out to become more and more unimportant over time [e.g. Features #6 and #7 = Features $y(t-6)$ and $y(t-7)$]. Obviously, $y(t-2)$ and $y(t-3)$ are the most important features, pointing to an important output feedback prediction

Table 5 Statistical preferences of one method (row) over another (column) based on the residual vectors over time for *dynamic on-line system identification*; statistical preference (p value of 0.05) is denoted by '+', strong preference (p value of 0.025) is denoted by '++', whereas minus '−' and '−−' indicate the opposite

Set/level	eMG (best)	FLEXFIS	eTS	FAOS-PFNN (best)	OS-ELM (best)	GS-EFS (fac = 0.6)	GS-EFS (fac = 0.45)
eMG (best)	0	+	+	−−	0	0	−−
FLEXFIS	−	0	0	−−	-	−−	−−
eTS	−	0	0	−−	−	−	−−
FAOS-PFNN (best)	++	++	++	0	+	+	0
OS-ELM (best)	0	+	+	−	0	0	−
GS-EFS (fac = 0.6)	0	++	+	−	0	0	−
GS-EFS (fac = 0.45)	++	++	++	0	+	+	0

Zero (0) denotes no statistical preference

Table 6 Comparison of the computation times of the various evolving learning methods, which is measured as the average seconds used to process a single sample (prediction + update) through the learning engine

Method	Seconds for single samples in average
FLEXFIS (Lughofer 2008)	0.0018
eTS (Angelov and Filev 2004)	0.0019
eMG ($\Sigma_{init} = 10^{-1}I_{11}$) (Lemos et al. 2011)	0.024
FAOS-PFNN (Wang et al. 2009)	0.153
OS-ELM (Liang 2006)	0.0011
Gen-Smart-EFS (*fac* = 0.45)	0.0028
Gen-Smart-EFS + feat.weights (*fac* = 0.45)	0.0031

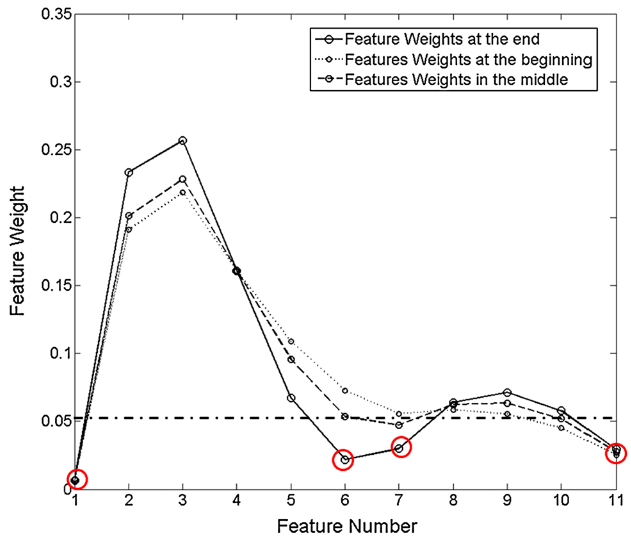

Fig. 11 Feature weights at three different points of time during the incremental learning phase (*beginning*, *middle*, *end*), the *horizontal line* indicates 20 % of the maximum weight below which features may be seen as unimportant and discarded when interpreting the model (those features indicated by an *ellipsis*)

horizon of 2–3. The horizontal line represents a 20 % threshold from the maximal feature weight (0.256) after the whole learning process. Features below this threshold can be neglected [i.e. $y(t-1)$, $y(t-6)$, $y(t-7)$ and $u(t-1)$] when showing the rules to operators.

It is also surprising that the exogenous variable u is never really important for explaining the target, finally suggesting a pure AR (autoregressive) model instead of an ARX model as defined in (50). This provides another (physical) interpretable insight into the learning problem not necessarily expected before hand!

Finally, computation time in sequential real-world dynamic problems is an important issue to be studied, as sometimes the learning engine should terminate in real-time for processing (predicting + updating) new incoming samples, resp. at least to be fast enough to terminate in a reasonable short time frame. A comparison of the speed of various methods for processing a single sample in average is provided in Table 6 (for those parametrization variants achieving the lowest RMSE in Table 4). From this, it becomes clear that all methods (except FAOS-PFNN) can cope with a quite high frequency of sample loadings up to more than 100 Hz, but FAOS-PFNN is still fast enough to approximately handle 10 Hz. It is also interesting to see that the rotation of the ellipsoidal rules inducing covariance matrix estimation and update increases the computation time much more than the integration of feature weights for smooth dimensionality reduction—compare the difference between *FLEXFIS* and *Gen-Smart-EFS* with that one between the last two rows (note that original *FLEXFIS* uses axis-parallel rules with a vector quantization based engine extracting such rules). *OS-ELM is the fastest method*, almost two times faster than the second fastest one (*FLEXFIS*). This is not a big surprise as it only updates parameters, but does not change its structure (neuron evolution/pruning); the number of hidden neurons have to be pre-parameterized a priori. Due to this fixed positioning, it looses significant accuracy when using approximately the same number of neurons as rules in Gen-Smart-EFS, see Table 4.

6.2.3 *Results on a real-world application in rolling mills*

The initial situation was as follows: an analytical model has been installed at the system in which some parameters were estimated through linear regression and should be improved by a non-linear soft computing model, which also possess the possibility to adapt on-line its parameters and structure.

Original results when using conventional EFS (in particular the *FLEXFIS* approach) without any generalization, pruning and feature selection option have been already reported in Lughofer and Angelov (2011) and serve as benchmark for our newly developed method. There, it turned out that conventional EFS can already significantly improve the accuracy of analytical models, namely by around 40.7 %. Now, in this paper we investigate whether these results can be further improved by our new approach *Gen-Smart-EFS*. Table 7 shows the results in terms of the accumulated one-step-ahead MAE on test samples, following the interleaved-test-and-then-train procedure, which is a quite convenient evaluation option on data streaming mining and modeling techniques (Bifet and Kirkby 2011).

The last but one column is dedicated to the extreme deviation values (more than 20 units). Especially, the values "too high" are the most critical ones (upon expert feedback), as these may cause discontinuities in the rolling mill process. Thus, the number of predictions which are too high (last column) on a basis of 6652 samples, serves as the most important value in this table. From rows #4 and

Table 7 Comparison of evolving fuzzy prediction models obtained by conventional *FLEXFIS* and the new approach *Gen-Smart-EFS* with different parametrization options (rule pruning variants, *fac*) and feature weighting switched on and off

Method	MAE	# of rules	# MAEs >20	Comp. time
Analytical	7.84	One model	259	0.0002 (pred. only)
Static fuzzy models	6.76	N/A	176	0.0002 (pred. only)
OS-ELM (*best param*)	6.31	30	175	0.0020
FAOS-PFNN (*best param*)	5.53	24	156	0.0523
FLEXFIS (*conv. EFS*) *no forget*	5.41	N/A	159	0.0023
FLEXFIS (*conv. EFS*)	4.65	N/A	68	0.0023
Gen-Smart-EFS, no pruning fac = 1.55	4.28	24	45	0.0035
Gen-Smart-EFS, conv. pruning, fac = 1.55	4.28	24	45	0.0047
Gen-Smart-EFS, ext. pruning, fac = 1.55	4.28	18	47	0.021
Gen-Smart-EFS, ext. pruning + *feat. weights*, *fac* = 1.55	4.34	12	53	0.017

#5, someone can immediately recognize that forgetting in the consequents (with a slight forgetting factor of 0.99) is essential to improve accuracy significantly (159 versus 68 MAEs higher than 20). Thus in *Gen-Smart-EFS*, we also foresee only the fixed forgetting option. The interpretation of these results should be obvious: our generalized version is able to improve conventional EFS by about 8 % in terms of accuracy and 33.8 % in terms of the number of errors which are significantly too high (the essential measure!); conventional pruning achieves the same results, so no significant overlap appeared during the evolution phase, whereas extended pruning is able to reduce the complexity by 6 rules while achieving a very similar accuracy. All variants of our method are able to cope with the real-time demand required by the operating system, that is, samples are coming in with a frequency of around 1 Hz (1 sample per second) and should be processed within this time frame (prediction + update) in order to avoid ever-growing back-log buffers. Static fuzzy models are indeed faster (only prediction is performed, no update), but cannot cope with dynamic process changes and hence loose significant model performance (50 % higher error and approx. three times more critical errors).

Again, we perform a statistical preference analysis over all methods on the residual vectors (observed vs. predicted values) over time, using a non-parametric Wilcoxon pairwise comparison tests (Hill and Lewicki 2007) coupled with Holm–Bonferroni test in order to reduce the FWER (Holm 1979). The statistically significant preferences are shown in Table 8, indicated by '+' whenever a method denoted in the row is preferred over the method denoted in the column with a significance value of $p = 0.05$ (5 % probability that hypothesis that two methods are equal is wrongly rejected) and indicated by '++' whenever the preference is more significant with a p value of 0.025.

From this table it is easy to realize that the new method has a statistical preference over all other evolving learning methods, especially when equipped with any pruning strategy. In case of no pruning, the new method performs equally to *FLEXFIS*. FAOS-PFNN is superior to OS-ELM and both can be preferred to older static and analytical models, as being implemented at the system before. It is also interesting to see that there is no preference between the four variants of our new method, which is not a surprise as all are achieving an MAE of around 4.3. On the other hand, integrating extending pruning scheme and feature weights (last row and column) leads to the less complex and thus most compact model by far (12 rules versus 18, 24 and 30 for the other evolving methods). In this sense, and also because its computational complexity increase compared to the others is not that dramatic (still being able to cope with the on-line processing demands of the system, which is around 1 Hz), this variant may be suggested to be finally preferred and used in the on-line system.

Figure 12 (left) shows the development of the feature weights for the best parametrization option (last row in Table 7), which leads to a further simplification of the final evolved system (12 versus 18 rules) by not loosing significant accuracy. Finally, only five features turned out to be really important, which after feedback of the operators were quite expected to be among the most important ones, as mostly influencing the rolling mill process. Interestingly, the integration of feature weights can also slightly decrease the computation time required for processing a single sample in average (prediction + update), as can be seen from the last two rows in Table 7—this is due to the dynamic decrease of the number of rules, i.e. the feature weights calculation requires less additional time than the reduced time which is achieved by the smaller complexity.

Table 8 Statistical preferences of one method (row) over another (column) based on the residual vectors over time for *rolling mill data*; statistical preference (p value of 0.05) is denoted by '+', strong preference (p value of 0.025) is denoted by '++', whereas minus '−' and '−−' indicate the opposite

Set/level	Analytical	Static fuzzy	OS-ELM	FAOS-PFNN	FLEXFIS conv.	FLEXFIS forget	GS-EFS no prune	GS-EFS conv. prune	GS-EFS ext. prune	GS-EFS weights + ext. pr.
Analytical	0	−	−−	−−	−−	−−	−−	−−	−−	−−
Static Fuzzy	+	0	0	−−	−−	−−	−−	−−	−−	−−
OS-ELM	++	0	0	−−	−−	−−	−−	−−	−−	−−
FAOS-PFNN	++	++	++	0	0	−	−	−	−	−
FLEXFIS conv.	++	++	++	0	0	−	−	−	−	−
FLEXFIS forget	++	++	++	+	+	0	0	−	−	−
GS-EFS no prune	++	++	++	+	+	0	0	0	0	0
GS-EFS conv. prune	++	++	++	+	+	+	0	0	0	0
GS-EFS ext. prune	++	++	++	+	+	+	0	0	0	0
GS-EFS weights + ext. pr.	++	++	++	+	+	+	0	0	0	0

Zero (0) denotes no statistical preference

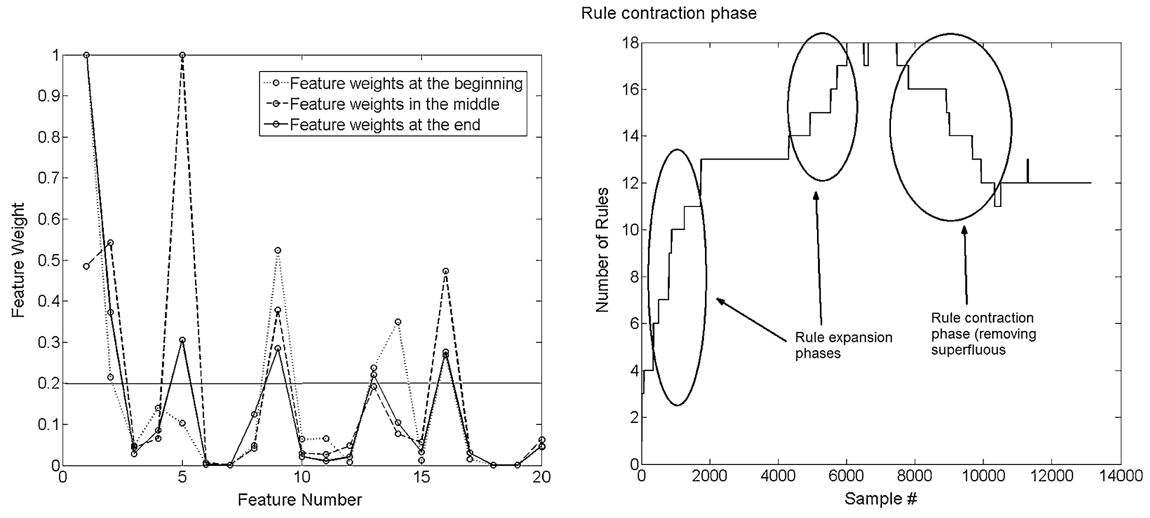

Fig. 12 *Left* feature weights development from the start of the learning phase (*dotted line*), through the middle (*dashed line*) towards the end of the learning phase (*solid line*)—please note that only five features finally appear as important, as indicated by the *horizontal line*; *right* the rule evolution over time including expansion phase and then contraction phase, as some rules turn out to be superfluous, also due to the feature weights integration during learning (better "real" winning clusters are elicited, thus moved), finally ending up with 12 rules instead of 18 (achieved w/o feature weights)

7 Conclusion

This paper presents a new approach for EFS which can cope with correlation-based local relations by introducing generalized rules with arbitrary rotated positions. This leads to a more reliable, accurate representation of local approximation behavior, finally inducing a significant improvement of model accuracy in most cases. Furthermore, the learning engine includes enhanced methodologies in the direction of rule merging and pruning by respecting the approximation trends and homogeneity of nearby lying (close or touching) rules. This leads to more compact rule bases with an accuracy which is very similar to the full spanned, not simplified rule bases. Results can be even further improved by introducing feature weights pointing to the importance of features in terms of a mixture of expected statistical contributions in the rule contours and gradients in the hyper-planes (rule consequents). In particular, some unnecessary rule evolution steps (as arising due to high distances along unimportant features) can be suppressed. Thus, in some cases even the size of the rule base can be further reduced and the model error decreased due to a soft dimension reduction step.

All these new concepts may be of great practical importance whenever a fast modeling method is required within an on-line learning framework, guaranteeing that the size of the rule bases are bounded and also increasing the interpretability of the fuzzy systems in several aspects, namely in the number of rules, in the compactness and consistency of rules and in the rule lengths (features with low weights can be excluded when showing the rules to an expert). Furthermore, in real-world data stream mining problems often the input dimensionality is huge (e.g. multi-sensor networks recording on-line measurements) such that a smooth and dynamic on-line dimensionality reduction is indispensable to guarantee models with a solid performance.

Acknowledgments The first author acknowledges the support of the Austrian COMET-K2 programme of the Linz Center of Mechatronics (LCM), funded by the Austrian federal government and the federal state of Upper Austria. This publication reflects only the authors' views.

References

Abonyi J, Babuska R, Szeifert F (2002) Modified Gath-Geva fuzzy clustering for identification of Takagi–Sugeno fuzzy models. IEEE Trans Syst Man Cybern Part B 32(5):612–621

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

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

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

Angelov P, Kordon A (2010) Evolving inferential sensors in the chemical process industry. In: Angelov P, Filev D, Kasabov N (eds) Evolving intelligent systems—methodology and applications. Wiley, New York, pp 313–336

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

Angelov P, Sadeghi-Tehran P, Ramezani R (2011) An approach to automatic real-time novelty detection, object identification, and tracking in video streams based on recursive density estimation and evolving Takagi–Sugeno fuzzy systems. Int J Intell Syst 26(3):189–205

Angelov P, Zhou XW (2006) Evolving fuzzy systems from data streams in real-time. In: 2006 international symposium on evolving fuzzy systems (EFS'06), Ambleside, pp 29–35 (2006)

Babuska R (1998) Fuzzy modeling for control. Kluwer Academic Publishers, Norwell

Backer SD, Scheunders P (2001) Texture segmentation by frequency-sensitive elliptical competitive learning. Image Vis Comput 19(9–10):639–648

Bauer F, Lukas M (2011) Comparing parameter choice methods for regularization of ill-posed problems. Math Comput Simul 81(9):1795–1841

Bhattacharyya A (1943) On a measure of divergence between two statistical populations defined by their probability distributions. Bull Calcutta Math Soc 35:99–109

Bifet A, Kirkby R (2011) Data stream mining—a practical approach. Tech. rep., Department of Computer Sciences, University of Waikato, Japan

Bouchachia A, Mittermeir R (2006) Towards incremental fuzzy classifiers. Soft Comput 11(2):193–207

Casillas J, Cordon O, Herrera F, Magdalena L (2003) Interpretability issues in fuzzy modeling. Springer, Berlin

Castro J, Delgado M (1996) Fuzzy systems with defuzzification are universal approximators. IEEE Trans Syst Man Cybern Part B Cybern 26(1):149–152

Chen H, Tino P, Yao X, Rodan A (2014) Learning in the model space for fault diagnosis. IEEE Trans Neural Netw Learn Syst 25(1):124–136

Cheng W, Juang C (2011) An incremental support vector machine-trained TS-type fuzzy system for online classification problems. Fuzzy Sets Syst 163(1):24–44

Cohen L, Avrahami-Bakish G, Last M, Kandel A, Kipersztok O (2008) Real-time data mining of non-stationary data streams from sensor networks. Inf Fusion 9(3):344–353

Costa B, Angelov P, Guedes L (2015) Fully unsupervised fault detection and identification based on recursive density estimation and self-evolving cloud-based classifier. Neurocomputing 150(A):289–303

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

Djouadi A, Snorrason O, Garber F (1990) The quality of training-sample estimates of the Bhattacharyya coefficient. IEEE Trans Pattern Anal Mach Intell 12(1):92–97

Efendic H, Re LD (2006) Automatic iterative fault diagnosis approach for complex systems. WSEAS Trans Syst 5(2):360–367

Eitzinger C, Heidl W, Lughofer E, Raiser S, Smith J, Tahir M, Sannen D, van Brussel H (2010) Assessment of the influence of adaptive components in trainable surface inspection systems. Mach Vis Appl 21(5):613–626

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

Gray R (1984) Vector quantization. IEEE ASSP Mag 1(2):4–29

Hametner C, Jakubek S (2013) Local model network identification for online engine modelling. Inf Sci 220:210–225

Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference and prediction, 2nd edn. Springer, New York

Heidl W, Thumfart S, Lughofer E, Eitzinger C, Klement E (2013) Machine learning based analysis of gender differences in visual inspection decision making. Inf Sci 224:62–76

Hill T, Lewicki P (2007) Statistics: methods and applications. StatSoft, Tulsa

Holm S (1979) A simple sequentially rejective multiple test procedure. Scand J Stat 6:65–70

Huang G, Saratchandran P, Sundararajan N (2004) An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. IEEE Trans Syst Man Cybern Part B Cybern 34(6):2284–2292

Iglesias J, Angelov P, Ledezma A, Sanchis A (2010) Evolving classification of agent's behaviors: a general approach. Evol Syst 1(3):161–172

Jang JS (1993) ANFIS: adaptive-network-based fuzzy inference systems. IEEE Trans Syst Man Cybern 23(3):665–685

Jimenez L, Landgrebe D (1998) Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. IEEE Trans Syst Man Cybern Part C Rev Appl 28(1):39–54

Jin Y (2000) Fuzzy modelling of high dimensional systems: complexity reduction and interpretability improvement. IEEE Trans Fuzzy Syst 8(2):212–221

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

Kohonen T (1995) Self-organizing maps, 2nd edn. Springer, Berlin

Komijani M, Lucas C, Araabi B, Kalhor A (2012) Introducing evolving Takagi–Sugeno method based on local least squares support vector machine models. Evol Syst 3(2):81–93

Krishnamoorthy K, Mathew T (2009) Statistical tolerance regions: theory, applications, and computation. Wiley, Hoboken

Leite D, Ballini R, Costa P, Gomide F (2012) Evolving fuzzy granular modeling from nonstationary fuzzy data streams. Evol Syst 3(2):65–79

Leite D, Costa P, Gomide F (2012) Interval approach for evolving granular system modeling. In: Sayed-Mouchaweh M, Lughofer E (eds) Learning in non-stationary environments: methods and applications. Springer, New York, pp 271–300

Lemos A, Caminhas W, Gomide F (2011) Multivariable Gaussian evolving fuzzy modeling system. IEEE Trans Fuzzy Syst 19(1):91–104

Lemos A, Caminhas W, Gomide F (2013) Adaptive fault detection and diagnosis using an evolving fuzzy classifier. Inf Sci 220:64–85

Liang N, Huang G, Saratchandran P, Sundararajan N (2006) A fast and accurate online sequential learning algorithm for feedforward networks. IEEE Trans Neural Netw 17:1411–1423

Lima E, Hell M, Ballini R, Gomide F (2010) Evolving fuzzy modeling using participatory learning. In: Angelov P, Filev D, Kasabov N (eds) Evolving intelligent systems: methodology and applications. Wiley, New York, pp 67–86

Lin Y, Chang JY, Lin CT (2013) Identification and prediction of dynamic systems using an interactively recurrent self-evolving fuzzy neural network. IEEE Trans Neural Netw Learn Syst 24(2):310–321

Lippmann R (1991) A critical overview of neural network pattern classifiers. In: Proceedings of the IEEE workshop neural networks and signal processing, pp 266–275

Lughofer E (2008) Extensions of vector quantization for incremental clustering. Pattern Recognit 41(3):995–1011

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

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

Lughofer E (2011) On-line incremental feature weighting in evolving fuzzy classifiers. Fuzzy Sets Syst 163(1):1–23

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

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

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

Lughofer E, Kindermann S (2010) SparseFIS: data-driven learning of fuzzy systems with sparsity constraints. IEEE Trans Fuzzy Syst 18(2):396–411

Lughofer E, Smith JE, Caleb-Solly P, Tahir M, Eitzinger C, Sannen D, Nuttin M (2009) Human–machine interaction issues in quality control based on on-line image classification. IEEE Trans Syst Man Cybern Part A Syst Hum 39(5):960–971

Lughofer E, Trawinski B, Trawinski K, Kempa O, Lasota T (2011) On employing fuzzy modeling algorithms for the valuation of residential premises. Inf Sci 181(23):5123–5142

Macias-Hernandez J, Angelov P (2010) Applications of evolving intelligent systems to the oil and gas industry. In: Angelov P, Filev D, Kasabov N (eds) Evolving intelligent systems: methodology and applications. Wiley, New York, pp 401–421

Maciel L, Lemos A, Gomide F, Ballini R (2012) Evolving fuzzy systems for pricing fixed income options. Evol Syst 3(1):5–18

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

Pang S, Ozawa S, Kasabov N (2005) Incremental linear discriminant analysis for classification of data streams. IEEE Trans Syst Men Cybern Part B Cybern 35(5):905–914

Pedrycz W, Gomide F (2007) Fuzzy systems engineering: toward human-centric computing. Wiley, Hoboken

Pratama M, Anavatti S, Lughofer E (2014) GENEFIS: towards an effective localist network. IEEE Trans Fuzzy Syst 22(3):547–562

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

Rao R (2013) A novel weighted euclidean distance-based approach. In: Decision making in manufacturing environment using graph theory and fuzzy multiple attribute decision making methods. Springer Series in Manufacturing, New York, pp 159–191

Rong HJ, Sundararajan N, Huang GB, Saratchandran P (2006) Sequential adaptive fuzzy inference system (SAFIS) for nonlinear system identification and prediction. Fuzzy Sets Syst 157(9):1260–1275

Rubio J (2010) Stability analysis for an on-line evolving neuro-fuzzy recurrent network. In: Angelov P, Filev D, Kasabov N (eds) Evolving intelligent systems: methodology and applications. Wiley, New York, pp 173–199

Sannen D, Nuttin M, Smith J, Tahir M, Lughofer E, Eitzinger C (2008) An interactive self-adaptive on-line image classification framework. In: Gasteratos A, Vincze M, Tsotsos J (eds) Proceedings of ICVS 2008, LNCS, vol 5008. Springer, Santorini Island, pp 173–180

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

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

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

Sun H, Wang S (2011) Measuring the component overlapping in the Gaussian mixture model. Data Min Knowl Discov 23:479–502

Tabata K, Kudo MSM (2010) Data compression by volume prototypes for streaming data. Pattern Recognit 43(9):3162–3176

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

Tung S, Quek C, Guan C (2013) eT2FIS: an evolving type-2 neural fuzzy inference system. Inf Sci 220:124–148

Wang N, Er M, Meng X (2009) A fast and accurate online self-organizing scheme for parsimonious fuzzy neural networks. Neurocomputing 72(16–18):3818–3829

Yager R, Filev D (1994) Approximate clustering via the mountain method. IEEE Trans Syst Man Cybern 24(8):1279–1284

Yager RR (1990) A model of participatory learning. IEEE Trans Syst Man Cybern 20(5):1229–1234

Zdsar A, Dovzan D, Skrjanc I (2014) Self-tuning of 2 DOF control based on evolving fuzzy model. Appl Soft Comput 19:403–418