This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

Incremental Rule Splitting in Generalized Evolving Fuzzy Systems for Autonomous Drift Compensation

Edwin Lughofer, Mahardhika Pratama, Igor Skrjanc

***Abstract*—Gradual drifts in data streams are usually hard to detect and often do not necessarily trigger the evolution of new fuzzy rules during model adaptation steps in order to represent the new, drifted data distribution(s) appropriately in the fuzzy model. Over time, they thus lead to oversized rules with untypically large local errors (typically also worsening the global model error), as representing joint local data distributions before and after a drift happened likewise. We therefore propose an *incremental rule splitting concept* for *generalized fuzzy rules* in order to autonomously compensate these negative effects of gradual drifts. Our splitting condition is based 1.) on the local error of rules measured in terms of a weighted contribution to the whole model error and 2.) on the size of the rules measured in terms of the volume of the associated clusters. We use the concept of statistical process control in order to omit an extra threshold parameter in our splitting condition. The splitting technique relies on the eigen-decomposition of the rule covariance matrix to adequately manipulate the largest eigenvector and eigenvalues in order to retrieve the new centers and contours of the two split rules. Furthermore, we guarantee sufficient flexibility in adapting the shapes and consequents of the split rules to the new drifted situation in the stream by integrating a specific *dynamic and smooth forgetting* concept of older samples, which formed the original (non-split) rules. *Robustness* against outliers is guaranteed by the realization of a *two-layer model building process*, where one layer represents the cluster partition and the other layer the rule partition: only clusters becoming significant over time are accepted as rules in the fuzzy model. The splitting concepts are integrated in the generalized smart evolving learning engine for fuzzy systems (termed as *Gen-Smart-EFS*) and successfully tested on two real-world application scenarios, engine test benches and rolling mills, the latter including a real-occurring gradual drift (whose position in the data is known). Results show clearly improved error trend lines over time when splitting is applied, compared to the case when it is not applied: reduction of the mean absolute model error by about one third (rolling mills) and about one half (engine test benches).**

keywords: data stream modeling, incremental rule splitting, generalized evolving fuzzy systems, drift compensation, smooth forgetting, robustness against outliers, two-layer model building

I. Introduction

A. Motivation and State-of-the-Art

Evolving intelligent systems [1] are nowadays applied in many industrial fields of applications such as on-line quality control, on-line system identification and modeling for evolving smart sensors (eSensors), human-machine interaction tasks, chemometric modeling, financial stock market predictions, video processing etc., to address the needs of nowadays complex processes including fast data streams [2], huge data bases [3] or dynamically changing environments [4]. Therefore, they are equipped with methodologies, which are able to handle time-varying system dynamics [5], upcoming new operation modes and systems behaviors [6] [7], process drift and shift occurrences [8] [9] or new parameter settings in the products and environments [10], adequately and autonomously [11] and often in an on-line and real-time manner [12]. Thereby, they contain and make use of two core functionalities, 1.) recursive adaptation of model parameters and 2.) dynamic knowledge expansion and shrinkage on the fly by component evolution and merging. Recursive adaptation of model parameters is usually able to compensate smaller changes in the process and especially performs refinements of originally estimated parameters from small data sets, thus increasing their statistical significance. It can be often conducted in exact manner (e.g., by using recursive least squares, incremental Levenberg-Marquardt, incremental Gauss Newton and spin-offs [13]), which means that the convergence to the real optimal solution of parameters takes place in each incremental adaptation step. Automatic knowledge expansion becomes necessary whenever significant novelty content is reflected in new incoming stream samples, showing feature range expansions or fillings of gaps in the previous data. In this case, new structural model components need to be evolved to appropriately represent the expanded situation and thus to avoid nasty extrapolation effects [14].

Evolving fuzzy systems (EFS) [15], as a sub-topic of evolving intelligent systems, enjoy a wide attraction in the evolving intelligent systems community [1] due to their ability to 1.) evolve structural components on the fly to account for knowledge expansion and 2.) to represent the evolved knowledge in a transparent and finally interpretable way (in form of rules), while still assuring high precision of the models due to their universal approximation capability [16] — they are in fact well-known for providing models with a fruitful balance between precision and interpretation, as, e.g., deeply analyzed in [17] for the standard batch case and in [18] for the real on-line case. Recursive parameter adaptation is typically conducted with the usage of recursive fuzzily weighted least squares (RFWLS) [19] [20] [21] [22] in order to achieve the local learning effect, while rule evolution and merging is addressed based on various rule evolution and merging criteria [23] (depending on the EFS approach used [24]).

A particular challenge in data stream mining and evolving

Edwin Lughofer (corresponding author) is with the Department of Knowledge-Based Mathematical Systems, Johannes Kepler University Linz, Austria, email: edwin.lughofer@jku.at

Mahardhika Pratama is with the School of Computer Science and Engineering, Nanyang Technological University, Singapore, 639798

Igor Skrjanc is with the Faculty of Electrical Engineering, University of Ljubljana, Slovenia

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

modeling from streams arise when some dynamic changes in form of *gradual* drifts or, in general, of *gradual* growth of structural components (rules) appear over time. The graduality thereby is small enough such that it does not trigger the evolution of new rules (through the violation of the rule evolution criterion/criteria), but it is also large enough that it affects the rule update significantly. This means that a rule may grow larger and larger over time based on new incoming samples and thus may get huge in size and/or may cover heterogeneous data clouds internally (as occurred before and after the drift) — as a result of continuous parameter adaptation. This may cause significant downtrends in local errors of the model components (rules) and thus may deteriorate model's generalization performance, because a rule suffers from an explosion of generality but loses its specificity — see also Section III for a more detailed problem description.

Forgetting mechanisms (as, e.g., proposed in [25] [26]) for outdating older rule positions may indeed prevent the rules to grow in size and local error too much, but they typically induce an additional parameter (determining the degree of forgetting termed as forgetting factor) [27]: setting it adequately in advance or even adaptively according to the stream (drift) characteristics is challenging [9] and still not a fully resolved problem. Furthermore, slight gradual drifts are usually hard to recognize and to quantify in its intensity in order to react upon with an appropriate forgetting degree [8]; and a permanent forgetting in case of no drift usually leads to a deterioration of model performance as analyzed in [9].

An automatic splitting of such growing rules causing high model errors into smaller ones would be thus the most promising option to solve this problem, however has been only loosely handled so far in current EFS approaches — see [24] for a recent survey, where 32 methods are listed and compared: only three of them, namely AHLTNM [28], eFuMo [29] and rClass [30], are equipped with splitting methodologies. AHLTNM relies on hyper-rectangular clusters which are split along the original axes, thus it does not have the flexibility for addressing local (growing) data clouds and extracted rules with arbitrary orientations; similar considerations hold for eFuMo, which indeed performs rule splits according to the local error, but an additional parameter is required for checking significance and furthermore it does not take into account a possible over-fitting occurrence in case of (many) small rules. rClass employs the concept of rule size in terms of measuring its volume for inducing a split, but it does not take into account whether the rule is still appropriate for resolving the current relation between inputs and targets (class labels in this case), because of the absence of system error in the rule splitting context. Furthermore, it was developed for classification problems and does not address the regression context.

In an on-line setting, an automatic splitting requires specific criteria regarding *when-to-split* rules and specific functionality regarding *how-to-split* rules, which have to be ideally conducted in *incremental* and *single-pass* manner (this is opposed to off-line rule splitting concepts where multiple iterations over the whole data set can be carried out [31]). The incremental property is necessary to meet on-line and real-time demands in stream processing, the single-pass functionality avoids ring-buffers containing past samples for each rule to be kept in the memory.

B. Our Approach

We propose a rule splitting methodology for *generalized evolving fuzzy systems* operating in incremental single-pass manner, where rules can appear in arbitrary rotated position in the multi-dimensional space and thus can form more compact and accurate rule bases than conventional axis-parallel rules (as examined in [32]). It comprises the following functionality:

- A *splitting condition* which checks when to best split rules in order to reduce the error of the current evolved fuzzy models. It is based on two criteria: the local error of rules and the size of the rules. The local error measures the precision of a rule in its predictions for samples which are lying nearby the local region the rule represent. If it is untypically high compared to the other rules, an increased non-linearity of the learning problem in this local region might be assumed, which cannot be appropriately modeled by a single rule. However an increased local error may be also the case when over-fitting — a high variance error — takes place, especially due to too many small rules (an effect which is manifested in the concept of *bias-variance error decomposition* [33]). Thus, the size of the rule plays an important role in order to split only those rules (and associated clusters) with high errors which are also big enough.
- A *splitting algorithm*, which shapes two new rules out of the current rule to be split. It performs a split along the main principal component direction of the ellipsoidal rule, i.e. that direction which has the largest eigenvalue. This ellipsoidal main axis is therefore split into two halves based on the eigendecomposition of its covariance matrix (which defines the shape of the rule). The centers of the two split rules are adequately placed towards the two mid points between the original center and the two focal points. The consequent parameters of the two split rules are directly assigned to the consequent parameters of the original rule (as starting point).
- For guaranteeing *sufficient flexibility of split rules*, i.e., to evolve their shapes and consequent orientations quickly with new data samples (after a drift occurred), we propose the inclusion of a forgetting mechanism on older samples (forming the original rule), whose intensity is steered by an adaptively changing forgetting factor.
- For guaranteeing *higher stability against outliers*, a rule base procrastination strategy is proposed and embedded during learning. It induces some software technical challenges in form of a two-layer model evolution stage: 1) a cluster-layer where each sample is respected equally and incrementally included in the partitioning and 2) a rule-layer where only *significant* clusters are associated with rules and thus finally used in the rule base when producing predictions for new query points. Splitting operates solely on the rule layer, thus only rules having at least *double significance* level are split candidates.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

- Update of all statistical help measures to perform a successful splitting in incremental manner. This includes the update formula of the local error for all rules and of the rule size.

Our splitting concept will be embedded in the generalized smart evolving fuzzy systems approach termed as *Gen-Smart-EFS* [32] and evaluated on two real-world data sets (in one a real occurring drift is known). We compare the effect of incremental splitting on the accumulated one-step-ahead prediction error over time with the case when no splitting is carried out (as in original Gen-Smart-EFS). This will be done by inspecting the rule evolution trend lines over time, to compare how many and when rules are evolved by classical rule evolution and how many and when by splitting.

The results showed significantly increased performance for engine test bench data, with a moderate increase of the number of rules due to splitting. In case of on-line data recorded from rolling mills, a real-occurring gradual drift could be better compensated when splitting was used: it lead to an error reduction of about one third compared to the case without splitting, and this with a moderate number of three splits.

II. Generalized Evolving Fuzzy Systems

A. Model Architecture

Our new concepts for *incremental rule splitting* (see Section III below) build upon the *generalized* smart evolving fuzzy systems learning approach (Gen-Smart-EFS, GS-EFS) as published in [32], which employs the *generalized* version of TS fuzzy systems, introduced in [34], and which showed better predictive performance compared to related state-of-the-art methods. It induces more compact rule bases with similar or even less model errors than conventional TS fuzzy systems, as turned out during past studies [35] [29]. This is basically because of its ability to model piecewise local correlations between variables in a more compact and accurate way.

In the generalized case, the rules are defined by:

$$\text{IF } \vec{x} \text{ IS (about) } \mu_i$$
$$\text{THEN } l_i(\vec{x}) = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + ... + w_{ip}x_p \quad (1)$$

where l_i the hyper-plane defining the consequent of the ith rule, μ_i denotes a high-dimensional kernel function, which, in accordance to the basis function networks spirit, is given by the multivariate Gaussian distribution:

$$\mu_i(\vec{x}) = \exp(-\frac{1}{2}(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1}(\vec{x} - \vec{c}_i)) \quad (2)$$

with $\vec{c}_i$ the center and Σ_i^{-1} the inverse covariance matrix of the ith rule, allowing any possible rotation and spread of the rule. This sort of rule premise puts forward a scale-invariant case and retains inter-correlations among input attributes by allowing arbitrarily rotated positions of the rules.

The output of a (generalized) TS fuzzy system consisting of C rules is a weighted linear combination of the outputs produced by the individual rules (through the l_i's), thus:

$$\hat{f}(\vec{x}) = \hat{y} = \sum_{i=1}^{C} \Psi_i(\vec{x}) \cdot l_i(\vec{x}) \quad \Psi_i(\vec{x}) = \frac{\mu_i(\vec{x})}{\sum_{j=1}^{C} \mu_j(\vec{x})} \quad , \quad (3)$$

with $\mu_i(\vec{x})$ the rule firing degree obtained through (2). Thus, in case of generalized TS fuzzy systems, a single rule can be fully represented as a triplet of variables $(\vec{c}_i, \Sigma_i^{-1}, \vec{w}_i)$.

B. The Core Algorithm for Incremental Model Updates and Evolution (Gen-Smart-EFS)

In order to be able to fully explain the splitting concepts, we first provide a compact summary of the basic algorithmic steps in *Gen-Smart-EFS* in Algorithm 1 below, initially the number of rules is set to $C = 0$ to maintain incremental learning from scratch.

Algorithm 1: **Generalized Evolving Fuzzy Systems — Core Engine**

1) Load a new sample $\vec{x}$; if it is the first one, Goto Step 6 (there, ignoring the if-part);
2) Elicit the winning rule, i.e. the rule closest to the current sample, which is then denoted as $\vec{c}_{win}$; for the distance calculation, standard Mahalanobis distance is used (as on the right hand side in (4) below).
3) Check whether the following criterion is met (the *rule evolution criterion*):

$$min_{i=1,...,C} \sqrt{(\vec{x} - \vec{c}_i)^T \Sigma_i^{-1} (\vec{x} - \vec{c}_i)} > r_i$$
$$r_i = \kappa p^{1/\sqrt{2}} \frac{1.0}{(1 - 1/(k_i + 1))^m} \quad (4)$$

with p the dimensionality of the input feature space and κ an a priori defined parameter, steering the tradeoff between stability (update of an old cluster) and plasticity (evolution of a new cluster) as multiplication factor for the prediction interval; this is the only sensitive parameter can be optimized during an initial batch modeling phase (based on first X samples from a stream). k_i denotes the support of the ith cluster (rule) and is elicited by the number of samples falling into this cluster. The criterion is based on the statistical concept of the *prediction interval* [36] serving as statistical tolerance region [37], whose boundary can be elicited by the χ^2 quantile with p degrees of freedom and at a default significance level of $\alpha = 0.05$. The last term on the right hand side is for the purpose to increase rule significance at the beginning when the rule is supported by a small amount of samples, i.e. when k_i small; the weighting exponent m is set to a default value of 4.
4) If (4) is not met, the centre of the winning rule is updated by

$$\vec{c}_{win}(k_{win} + 1) = \vec{c}_{win}(k_{win}) + \eta_{win}(\vec{x} - \vec{c}_{win}(k_{win})) \quad (5)$$

and its inverse covariance matrix by (the index win neglected due to transparency reasons):

$$\Sigma^{-1}(k+1) = \frac{\Sigma^{-1}(k)}{1-\alpha} - \frac{\alpha}{1-\alpha} \frac{(\Sigma^{-1}(k)(\vec{x} - \vec{c}))(\Sigma^{-1}(k)(\vec{x} - \vec{c}))^T}{1 + \alpha((\vec{x} - \vec{c})^T \Sigma^{-1}(k)(\vec{x} - \vec{c}))} \quad (6)$$

with $\alpha = \frac{1}{k_{win}+1}$. The former stems from the idea in vector quantification [38] by minimizing the expected squared quantization error; the learning gain

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

$\eta_{win} = \frac{1}{k_{win}}$ is thereby set in a way that it fulfills the Robbins-Monroe conditions. The latter is a recursive exact update, analytically derived with the usage of the Neumann series, see [39] for full details.

5) If (4) is not met, it is checked whether the updated winning rule has become significantly overlapping with any other rule, and if so, whether there exists sufficient homogeneity among the two overlapping rules in terms of their consequent functions (indicating a nearly linearly continuing trend among the two regions the two rules represents). Sufficient overlapping is checked by applying the Bhattacharyya distance [40] on the rule centers and inverse covariance matrices. Homogeneity in the output space is checked through the dihedral angle between the two hyper-planes the consequent functions of the overlapping rules span up. If both are high (threshold motivated in [32]), the merging takes places by
   a) weighted averaging of rule centers,
   b) weighted adaptation of the inverse covariance matrix of the rule with higher support based on recursive variance concept plus an additional range expansion term for better coverage,
   c) amalgamation of consequent function using participatory learning concept [41].
6) If (4) is met, a new rule is evolved as covering a new region in the feature space (i.e. having sufficient *novelty content*) by setting its center $\vec{c}_{C+1}$ to the coordinates of $\vec{x}$ and initialize its inverse covariance matrix Σ_{win}^{-1} by setting it to a diagonal matrix with entries 1 divided by a small fraction, i.e. 1/100, of the variable ranges (= initial rule spreads); increase the number of rules $C = C + 1$.
7) Perform recursive fuzzily weighted least squares (RFWLS) including forgetting factor λ (default $\lambda = 1$ which means no forgetting), to update the consequent parameters $\vec{w}$ of all C rules. This is the standard consequent update scheme in most of the current EFS approaches and can be inspected in detail in the handbook survey [24] and in many other papers about EFS.
8) Goto Step 1.

Obviously, this algorithm is able to perform incremental parameter adaptation and rule evolution from scratch. However, in some cases it might be beneficial that an initial model is learnt, e.g., for representing it to an expert or operator before it is going to be installed on-line or for checking the whole 'modelability' of the problem. For such cases, Gen-Smart-EFS has been extended to a *robust batch learning variant* in [42] (termed as *Rob-GenFIS*), where, once the number of rules are learnt, the rule centers and inverse covariance matrices are updated in multiple optimization runs over the whole data set with a specific decreasing learning gain, to assure convergence. The robustness aspect also concerns the estimation of the consequent parameters which is done separately on the whole data set, once the rules antecedents are optimally placed. Therefore, it employs the concept of elastic net regularization [43], which is adopted for learning of the consequent parameters. It incorporates a convex combination of Lasso [44] and ridge regularization term [33], thus its optimization problem in the context of fuzzy systems consequent training is defined as (for the ith rule):

$$J_i = \sum_{k=1}^{N} \Psi_i(\vec{x}(k)) e_i^2(k) + \lambda \sum_{j=1}^{p} (\alpha w_{ij}^2 + (1-\alpha)|w_{ij}|) \longrightarrow \min_{\vec{w_i}} \quad i = 1, ..., C \tag{7}$$

with λ the regularization parameter and α a parameter in $[0, 1]$, steering the degree of influence of the 'Lasso term' $\sum_{j=1}^{p} |w_{ij}|$ versus the 'ridge term' $\sum_{j=1}^{p} w_{ij}^2$, with p the dimensionality of the input feature space. This mechanism is useful in resolving the over-fitting situation due to the curse of dimensionality effect, provided that the regularization parameter is properly chosen. The problem in (7) can be efficiently solved through a quadratic programming approach, termed as LARS-EN, see [45].

III. Incremental Rule Splitting in Generalized EFS

During the evolving modeling process from data streams, the following situations may arise:

- Dynamic changes such as drifts in the process arise which are of slow gradual nature and cause certain rules to grow larger and larger over time; these changes are often not intense enough in order to trigger the rule evolution criterion. In our case, this means that there are no samples during and after the change which are lying significantly out of the boundaries of the prediction interval in order to meet (4); the samples are just gradually expanding the rule ellipsoid — see Figure 1 for an example. In alternative density-based rule evolution approaches, the density of the new drifted data cloud may be significantly lower than those of older clouds such that no new rule is evolved either. Typically, in such cases automatic drift detection methods, as, e.g., discussed in [9], are not able to detect a drift, either. Thus, no appropriate forgetting factor can be set (based on a detected drift intensity), which would allow the rule to adapt to the new drifted situation.
- The parameter, which is most responsible for the trade-off to update old rules versus to evolve new ones, is inadequately set for the current data stream modeling problem (such a parameter is typically present in all EFS approaches! [24]); in our case, this means that κ is set too high such that not enough rules are evolved to represent the new, drifted data distribution adequately $\rightarrow$ this leads to older rules covering a too large area — see Figure 1 for an example.

In order to resolve such unpleasant situations, we suggest an incremental rule splitting strategy, which can act in fully single-pass manner, i.e. based on single loaded samples which can be immediately forgotten after they have been processed. Thus, it does not require any ring-buffers storing samples which formed the rules in the past [39], neither does it require any batch-type re-estimation steps as being used in batch clustering approaches with splitting concepts embedded [46] [47].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

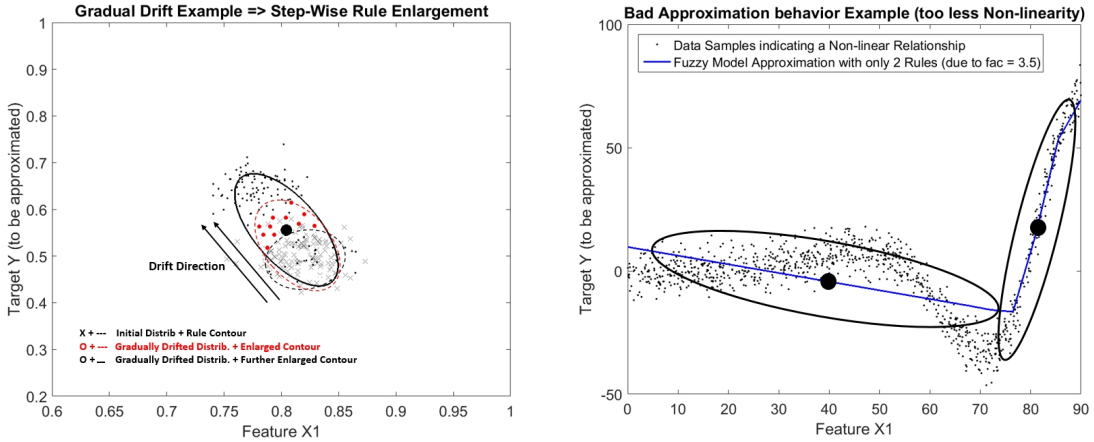

Figure 1. left: rule growing by gradual drifts over time not necessarily triggering the evolution of a new rule; right: bad approximation of a non-linear trend by only 2 rules (due to an inappropriate parametrization of the rule evolution criterion), the larger rule cannot resolve the non-linearity degree adequately.

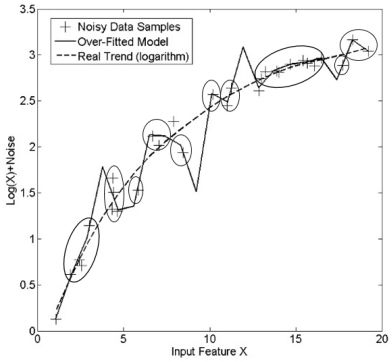

Figure 2. Over-fitting of a pure logarithmic trend between input and target in case of the extraction of too many fuzzy rules → high local errors

A. Splitting Condition

Regarding the issue *when-to-split* rules, we suggest a condition which is based 1.) on the local model error (error per rule) and 2.) on the size of the rule. The first issue is motivated by the fact that rules with high local errors point to regions where the non-linearity is not sufficiently resolved by the current rule base — as shown in the right image in Figure 1. The second issue checks whether the rule is big enough to be a candidate for splitting. The reason for this is that high local errors can also arise in case when there is an over-fitting effect due to too many small rules (where typically most of them cover only very sparse regions). Figure 2 underlines this fact by showing a logarithmic trend in the data (marked by a dashed line), which is modelled by a too complex fuzzy model (there are too many rules indicated by ellipsoids). This over-fits the real trend, thus it induces a high model error. In such a case, a further splitting of a small rule into two rules would make the situation even worse.

In order to avoid additional parameters to be tuned in form of fixed thresholds on large errors and large rule sizes, we employ the concept of statistical process control [48] and trigger a splitting of a rule when the following conditions is met:

$$err_i > \frac{1}{C}\sum_{j=1}^{C} err_j + n * \sqrt{\frac{1}{C}\sum_{j=1}^{C}(err_j - \overline{err})^2} \quad \wedge$$
$$size_i > \frac{1}{C}\sum_{j=1}^{C} size_j + n * \sqrt{\frac{1}{C}\sum_{j=1}^{C}(size_j - \overline{size})^2} \qquad (8)$$

with n a multiplication constant typically set to 1 or 2 and with err_i the error of the ith rule, which is defined by:

$$err_i = \frac{\sum_{k=1}^{N}(y(k) - \bar{y}_i(k))^2 \Psi_i(\vec{x}_k)}{\sum_{k=1}^{N}\Psi_i(\vec{x}_k)} \qquad (9)$$

,i.e. as a weighted sum of sample-wise squared errors. Thus, $\overline{err}$ and $\overline{size}$ denote the mean over all local errors and the mean over all cluster/rule sizes, respectively. $\bar{y}_i(k)$ denotes the prediction of the local hyper-plane on the kth sample, and the weights are given by the normalized rule activation levels:

$$\Psi_i(\vec{x}_k) = \frac{\mu_i(\vec{x}_k)}{\sum_{j=1}^{C}\mu_j(\vec{x}_k)} \qquad (10)$$

The weights are important to assure locality of the error, i.e. samples which are far away from rules should have little or no influence on their prediction errors. The normalization in (10) is important, as samples lying close to the border of a local region (modeled by a rule) and far away from all other rules should have a significant impact on the local error of that region where it lies close to — which would not be the case when using the non-normalized rule activation levels (all rule membership degrees would be low). It could be then expected that a gradual, moderate drift starts to arise → splitting required (see motivation above). The local error of the ith rule can be easily updated by

$$err_i(N+1) = \frac{err_i(N)sum_{\Psi_i}(N) + (y(N+1) - \bar{y}_i(N+1))^2\Psi_i(\vec{x}_{N+1})}{sum_{\Psi_i}(N) + \Psi_i(\vec{x}_{N+1})} \qquad (11)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

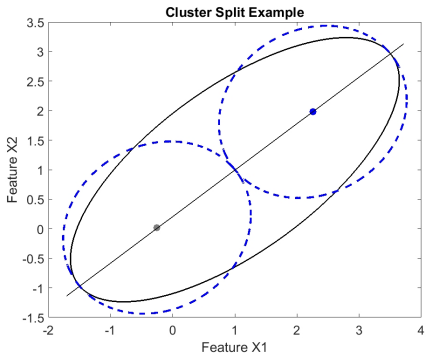

Figure 3. Rule splitting example, the original ellipsoid shown in solid dark (black) line is split into two shown in dotted grey (blue) lines

with $sum_{\Psi_i}(N)$ the sum of the normalized membership values of the ith rule up to the current sample N, updated by $sum_{\Psi_i}(N+1) = sum_{\Psi_i}(N) + \Psi_i(\vec{x}_{N+1})$.

The size of the ith rule given by $size_i$ is determined by the covariance matrix based definition of the volume of the ellipsoid defined by the rule [49], thus calculated as:

$$size_i = \frac{2 * det(\Sigma_i^{-1}) * \pi^{p/2}}{\Gamma(p/2)} \tag{12}$$

with p the input dimensionality, det the determinant and Γ the gamma function.

B. Splitting Algorithm

In each incremental update cycle, only the rule recently updated need to be checked whether it meets the criterion in (8). Thus, the condition in (8) is embedded in Step 4 of Algorithm 1 as Step 4b. The splitting operations are then conducted inside Step 4b along the major principal component direction (having the largest eigenvalue), as this corresponds to the direction where the ellipsoid of the rule has the largest spread and thus the most impact on its bad local error. Splitting is done within a half-half situation, so to split the rule into two equal portions, which are then further updated and thus shaped into the right direction upon receiving new data samples. This is the most intuitive way, achieving touching rules with no overlap, but still covering the input space well, and without requiring extra parameters for steering the overlap degree. Figure 3 shows an example, where the rule shown with a dark solid ellipsoidal line is split into two rules shown as dotted grey lines. The centers are thereby set in the middle between the center and the outer focal points of the first (main) principal component axis [31].

Formally, this can be achieved by:

$$\vec{c}_i(split1) = \vec{c}_i + a_i\frac{\sqrt{\lambda_i}}{2} \quad \vec{c}_i(split2) = \vec{c}_i - a_i\frac{\sqrt{\lambda_i}}{2} \tag{13}$$

where λ_i corresponds to the largest eigenvalue of the covariance matrix of rule i, i.e. $\lambda_i = max(\Lambda)$ and a_i to the corresponding eigenvector, which can both be obtained through classical eigendecomposition of the covariance matrix (which is symmetric, thus the solutions exist):

$$\Sigma_i = A\Lambda A^T \tag{14}$$

where A is a matrix of eigenvectors stored in columns and Λ a diagonal matrix containing all eigenvalues of Σ_i in descending order. Furthermore, in order to split the shape of the rules also into two halves as shown in Figure 3, the largest eigenvalue in Λ is set to a fourth of its value, because taking the square-root of it would then trigger the length of the corresponding ellipsoidal axis reduced to its half. This means, the two new covariance matrices for the split rules are obtained by back-multiplication of the matrices from the eigendecomposition:

$$\Sigma_i(split1) = \Sigma_i(split2) = A\Lambda^* A^T, \quad \Lambda^*_{jj} = \begin{cases} \Lambda_{jj} & j \neq 1 \\ \frac{\Lambda_{jj}}{4} & j = 1 \end{cases} \tag{15}$$

where Λ_{11} is assumed to be the entry for the largest eigenvalue. Please note that the orientation of the principal component having the largest eigenvalue remains the same for both split rules (only its length is halved) at the time of their birth (later they will be further adjusted with new data), so no eigenvector needs to be modified. This guarantees very fast updates.

Algorithm 2 summarizes the functionally necessary splitting operations step-by-step — to be embedded as Step 4b into Algorithm 1, after the winning rule has been updated when (4) is not met. It can be also embedded in other EFS learning engine which employ Gaussian rules.

Algorithm 2: **Incremental Rule Splitting**

Input: new sample $\vec{x}$, current fuzzy rule base containing C rules, local errors of all rules $err_1, ..., err_C$.

1) Update all local errors of all rules by applying (11).
2) Update covariance matrix of winning rule Σ_{win} by $\Sigma_{win}(k+1) = \frac{1}{k+1}(k\Sigma_{win}(k) + \frac{k}{k+1}(\vec{c}_{win} - \vec{x})^T(\vec{c}_{win} - \vec{x}))$ with $k = k_{win}$ the number of past samples supporting the winning rule.
3) **For** all rules $i = 1, ..., C$:
4) Check Condition (8) using (12), where Σ_i^{-1} is given as permanently updated in Algorithm 1 by (6).
5) If Condition (8) is met for Rule i,
   a) Perform the eigendecomposition of the ith rule by using (14).
   b) Receive $\lambda_i = max(\Lambda)$ and its corresponding eigenvector a_i.
   c) Set the centers of the new, split rules $\vec{c}_i(split1)$ and $\vec{c}_i(split2)$ according to (13).
   d) Calculate the covariance matrix of the new split rules $\Sigma_i(split1)$ and $\Sigma_i(split2)$ according to (15).
   e) Set the number of data samples supporting the split rules to the half value of the original rule, i.e. $k_i(split1) = k_i(split1) = k_i/2$.
   f) Set the error of both split rules to the average local error over all rules, i.e. $err_i(split1) = err_i(split2) = \frac{1}{C}\sum_{i=1}^{C} err_i$.
   g) Set the consequent parameters of both split rules to the consequent parameters of the original rule, i.e., $\vec{w}_i(split1) = \vec{w}_i(split2) = \vec{w}_i$
   h) Set the inverse Hessian matrices of both split rules to the inverse Hessian P_i of the original rule (inverse Hessian is required in the RWFLS update).
   i) Overwrite Rule i with the first split rule (and all of its associated variables), $Rule_i \leftarrow Rule_i(split1)$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

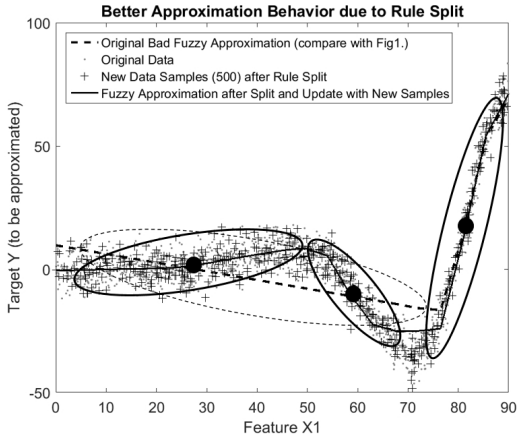

Figure 4. The approximation behavior due to a splitting of the large rule shown in Figure 1 into two smaller ones, which are further updated with the new samples (shown as crosses) to achieve the correct trend in this local region

j) Introduce a new Rule $Rule_{C+1}$ and assign the second split rule (and all of its associated variables) to it, $Rule_{C+1} \leftarrow Rule_i(split2)$.
k) Increase the number of rules, $C = C + 1$.

6) **End For**.

The consequent parameters and the inverse Hessian matrices, which are required in the RWFLS update in order to avoid time-intensive and instable matrix inversions, are indeed directly transferred to both split rules, but will become updated when new samples come in, in order to be turned into the right direction of the locally increased non-linearity. Figure 4 exemplifies this issue on the basis of the bad fuzzy approximation due to the enlarge ruled shown in Figure 1.

C. Increasing Flexibility of Split Rules

Indeed, the current splitting methodology is able to compensate drifts which have affected model performance in a negative way by increasing the flexibility of the fuzzy system in local regions (one rule is split into two). However, if this rule has been supported by many samples before, the likelihood is high that the shapes and consequent parameters of the two split rules are 'frozen' because of still having high support from past samples — e.g., compare with the case shown in the right image in Figure 1: many samples formed the rules, half of them are then still supporting two split rules (according to Step 4e.) in Algorithm 2), as samples are almost equally distributed among the input feature (x-axis). Then, new samples falling into the region will not have enough weight to turn the shapes and hyper-planes of the two rules sufficiently enough to appropriately model the regression trends in the two split regions. This means that indeed the rule size is reduced, but the high local error will remain in the two split regions to a similar extent as before.

Therefore, we propose a dynamic forgetting strategy, which, immediately after the split has been conducted, embeds a forgetting factor $\lambda \in [0.9, 1.0[$ in the update process of the rule shapes and hyper-planes. The lower its value becomes, the faster older data is forgotten, thus the higher new samples are weighted into the update process, thus the more the two split rules are able to turn their shape adequately. The forgetting factor is initialized with a value of 0.9 (due to good past experience in case of drifts, see, e.g. [9]), which, according to exponential forgetting strategy λ^{N-k} with N denoting the current sample index, is equivalent to the case that only the latest 20 samples receive a weight bigger than $\epsilon = 0.1$

In order to allow convergence of the shapes and hyper-planes of the two split rules, $\lambda_i(split1/2)$ is recursively increased step-wise towards 1.0 over new incoming samples $\vec{x}_{N+1,...}$ by:

$$\begin{aligned}\lambda_i(split1) &= min(\lambda_i(split1) + 0.01 * \Psi_i(\vec{x})(split1), 1.0)\\ \lambda_i(split2) &= min(\lambda_i(split2) + 0.01 * \Psi_i(\vec{x})(split2), 1.0)\end{aligned} \tag{16}$$

with $\Psi_i(\vec{x})(split1)$ the normalized membership degree of the first split rule to the current sample $\vec{x}$, as defined in (3). The justification of this formula is that new samples falling nearby the split rules (thus with high $\Psi_i(\vec{x})$'s) should already have a high significance and influence for 'forming' the new shapes and hyper-planes. Thus, λ can be increased more intensively than in case of samples lying farer away.

The forgetting factors $\lambda_i(split1)$ and $\lambda_i(split2)$ are integrated into the update of the antecedents and consequents of the two split rules in the following way:

- For the antecedent parts, the influence of older samples in the rule centers and inverse covariance matrices of the split rules is decreased to:

$$\begin{aligned}k_i(split1/2) =& k_i(split1/2) - k_i(split1/2)\\ &* min(\lambda_i(split1/2), 0.99)\end{aligned} \tag{17}$$

  This automatically increases the learning gain $\eta_i = \frac{0.5}{k_i(split1/2)}$ for centers and $\alpha = \frac{1}{k_i(split1/2)+1}$ for the inverse covariance matrix updates, and helps out the cluster from its converged position and shapes, as a stronger rule movement for the next samples is affected.
- For the consequent parts, the RWFLS estimator provides a possibility to integrate $\lambda_i(split1/2)$ in a way that convergence to the minimum of the exponentially weighted least squares objective for the two split rules $J_i(split1/2) = \sum_{k=1}^{N} \lambda_i(split1/2)^{N-k} \Psi_i(\vec{x}(t)) e_i^2(k) \longrightarrow \min_{\vec{w}_i}$ with $e_i(k) = y(k) - \hat{y}(k)$, is granted in each update step, see [15]. In this update scheme for consequent parameters, past samples contribute with exponentially decreasing weights.

D. Increased Learning Robustness with Two-Layer-Stage Model Learning

Another important aspect in learning from streams concerns an adequate treatment of outliers by omitting them in the model update process. This assures that models are not 'spoiled' by outliers, although in case of evolving fuzzy systems, due to their localized learning engines, the effect of outliers is expected to be much weaker than in case of global

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

regression models, but still they may contribute negatively to the inference process and thus to the final model output.

We propose a so-called *rule procrastination* strategy which is supported by a two-layer-stage model learning and adaptation. The basic idea is that a new cluster is indeed generated once the evolution criterion in (4) is fulfilled, but it is not associated with a rule, thus neither embedded in the fuzzy rule base, before a significant number of samples confirm the new cluster, i.e., fall into its region of influence and form its center and shape further.

Algorithmically, this means that all the steps in Algorithm 1 are run through on cluster level in the same way as stated (thus having C clusters which are updated by Steps 4 und 5 and new clusters evolved in the same manner as rules in Step 6), but an additional step is introduced between Step 6) and 7) which then assigns only significant clusters $c_i, i = 1, ..., C$ to fuzzy rules $R_j, j = 1, ..., C_r \leq C$:

Step 6b) $j = 1$; For $i = 1, ..., C$: if $k_i \geq S$ then $R_j = c_i$, $j = j + 1$.

with S the minimal number of samples such that a cluster is accepted as rule (in our case set to 3 in all experiments below). The C_r fuzzy rules (instead of C ones) are then used in the inference process in (3). Splitting is then only performed on the rule base containing the C_r rules (substituting C in Algorithm 2), as only their local errors contribute to the overall model error (insignificant clusters are not part of the inference process). An additional criterion then automatically emerges in order to check whether two rules obtained from a split (of Rule i) would be still supported by a significant number of samples, thus the condition $k_i \geq 2 * S$ has to be added to our original splitting condition (8), guaranteeing that both the two split rules are supported by at least S samples.

Software-technically, it is a bit more tricky as clusters and rules have to be handled in parallel and have to be associated with links, i.e. it has to be known at any stage of the stream learning process which cluster is associated with which rule and which cluster to no rule. This leads to a two-layer model building process, whose concept is visualized in Figure 5.

IV. Experimental Setup

A. Two Application Scenarios

a) Real-world application: engine test benches: The first real-world application we are dealing with is the supervision of the behavior of a car engine during simulated driving operations at an engine test bench. These include the engine speed and torque profiles during an MVEG (Motor Vehicle Emissions Group) cycle, a sportive driving profile in a mountain road and two different synthetic profiles. Several sensors at the engine test bench have been installed which measure various important engine characteristics such as pressure of the oil, various temperatures or emission gases (NOx, CO2 etc.). Online data have been recorded containing 22302 measurements and 42 channels (from the sensors) in sum for evaluation purposes. The main task is to build an accurate prediction model for NOx emission, the most important gas according to strict standards in the ISO-norm to meet the requirements for newly built cars. NOx measurements are costly and thus should be minimized during the car development phase. The number of channels could be reduced to 8 most important input channels for NOx approximation, based on a modified variant of forward selection [50], emphasizing the selection for non-linear (fuzzy) models. For further description of the application, please also refer to [51].

b) Real-world application: rolling mills data: This application originates from metal industry, in particular from the tension-leveler part of the cold rolling process within the biggest steel company of Austria. Tension leveling is a process used in the steel industry to remove any shape defects present in coil material [52]. It is usually the final and most sensitive process in the production chain before the cold rolled product is delivered to the customer, thus its automatic supervision is urgently requested; for a detailed description, please refer to [53] [54], where this application has been used for fault detection and isolation research and development purposes. From a set of 240 measurement channels recorded by a large-scale multi-sensor network along the production chain, SysID models were extracted based on approximately 9450 recorded on-line samples to establish causal relations and dependency models, which also have predictive spirits (time lags in the variables were integrated). In one particular relationship (containing 4 input and 1 target channels), a real-occurring (gradual) system drift could be observed, starting at around sample 2500. This data subset is used in this paper for checking whether our splitting approach can dynamically compensate the drift and really brings improvement in the model error trends opposed to classical rule evolution.

B. Evaluation Strategies

The evaluation strategy follows the *interleaved-test-and-then-train* procedure [55], also termed as *accumulated one-step-ahead prediction error/accuracy*. It is based on the idea to measure model performance in one-step ahead cycles, i.e. based on one newly loaded sample only. In particular, the following steps are carried out:

1) Load a new sample (the Nth).
2) Predict its target $\hat{y}$ using the current evolved model M.
3) Compare the prediction $\hat{y}$ with the true target value y and update the performance measure:
$$pm(y, \hat{y})(N) \leftarrow upd(pm(y, \hat{y})(N-1)) \tag{18}$$
4) Update and evolve the model M.
5) Erase sample and goto Step 1.

In our case, it is a perfect measure to track the prediction error development over time in an accumulated fashion. We will show this development by using one-dimensional plots, were the x-axis will represent the time line (= sample number) and the y-axis the accumulated error up to the corresponding sample/time steps. pm denotes a general updateable performance measure, for which we use the percentage mean absolute error, which is updated by:

$$MAE(N) = \frac{(N-1)MAE(N-1) + |y(N) - \hat{y}(N)|}{N * range(y)} \tag{19}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

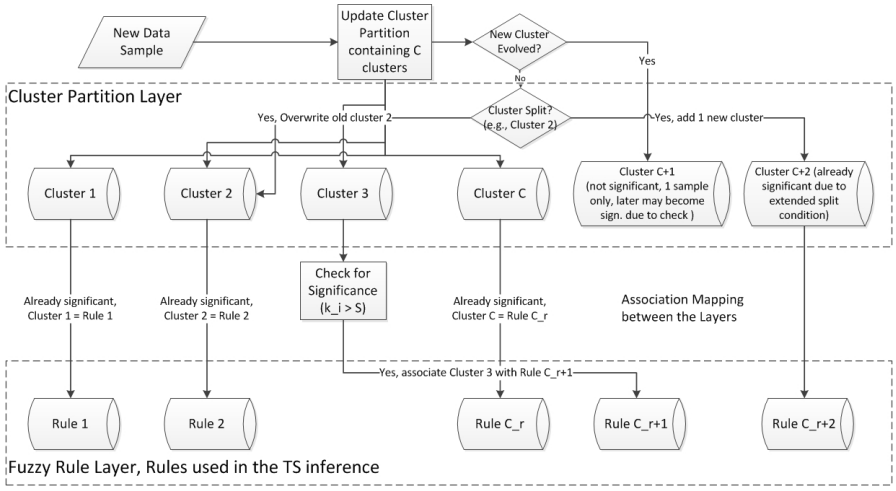

Figure 5. Two-layer model building process during incremental learning of generalized fuzzy systems, embedding rule evolution and rule splitting functionality; please mind the two layers and the association mapping in-between them: newly evolved clusters are not directly associated with rules, first they have to become significant; split clusters are already significant due to the extended split condition ((8) $\wedge k_i \geq 2 * S$), thus one cluster obtained from the split overwrites the original cluster (and its associated rule) and the other extends the cluster partition and the rule base by an additional cluster and rule; an updated cluster may be checked whether it has become significant (here for Cluster #3) and if so, a new rule is added as well (here the $C_r + 1$th) and associated with this cluster; clusters which are already linked with rules (e.g., Cluster 1 and C) do not need to be checked whether they became significant.

with $range(y) = max(y) - min(y)$, and MAE(0) = 0; $y(N)$ denotes the real measured target and $\hat{y}(N)$ the predicted target in the current, the Nth, sample. We will compare the error trend lines over time with and without splitting included.

Additionally, we will examine the structural change of the fuzzy rule base over time in terms of a.) classical rule evolution (as in original Gen-Smart-EFS) and b.) rule splitting according to our approach. In all cases, we perform an initial model training on the first 500 samples in order to optimize the parameter κ (responsible for rule update versus rule evolution), such that we receive a fair comparison among the runs — otherwise, we could set κ in advance to a large value in order to favor rule splitting, because then only very few (too large) rules will be evolved, and then splitting will obviously always improve the model.

V. Results

A. Engine Test Benches

The error trend lines for the engine test bench data are visualized in Figure 6. It can be easily recognized that a moderate splitting using $n = 1$ in (8) leads to a slight improvement of the error trend line, compared to the case when no splitting is performed. This is achieved with a slight increase of the number of rules, as can be seen in the left image in Figure 7, where the lower solid line shows the evolution indicator: this is set to 1 in case of classical rule evolution, and set to 2 when splitting is performed, whereas the dashed line shows the evolution of the number of rules. Thus, a split is only carried out three times, two times around sample 1200, one time around sample 7200. But, obviously the first two splits bring in significant reduction in error trend line, which is rising to a much lower value than in case of no splitting, compare with Figure 6. The third split, however, do not have a real effect (neither positively nor negatively), as both trend lines continue in parallel in the same manner.

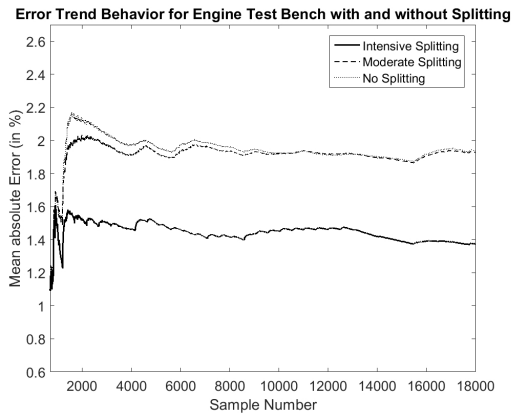

Figure 6. Error trend lines over time for **engine test bench data** when no splitting is applied (dotted line) and when splitting in two variants is applied: one time a moderate splitting in case when setting $n = 1$ in the splitting condition (8) (dashed line) and one time a more intense splitting in case when setting $n = 2$.

Whenever a more intense splitting is used due to a setting of $n = 2$ in (8), the error trend line can be drastically improved as shown by the solid line in Figure 6. This is mainly because of the compensation of the raising error at the beginning of the learning process (around Sample 1000-1700). As can be seen from the right image in Figure 7, a lot of splitting operations are performed during this time frame, which are obviously necessary to account for a changing drivers profile in that region and thus to compensate an inappropriate setting of κ (out of the first 500 samples). During the middle phase, some sporadic splitting actions take place which have no effect on the trend line. During the end phase, again much more splitting operations are triggered, which finally can omit the (slightly) increasing error curve trend line (from around 1.8 to around 1.95 — as can be observed in Figure 6) in case when no

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TFUZZ.2017.2753727, IEEE Transactions on Fuzzy Systems

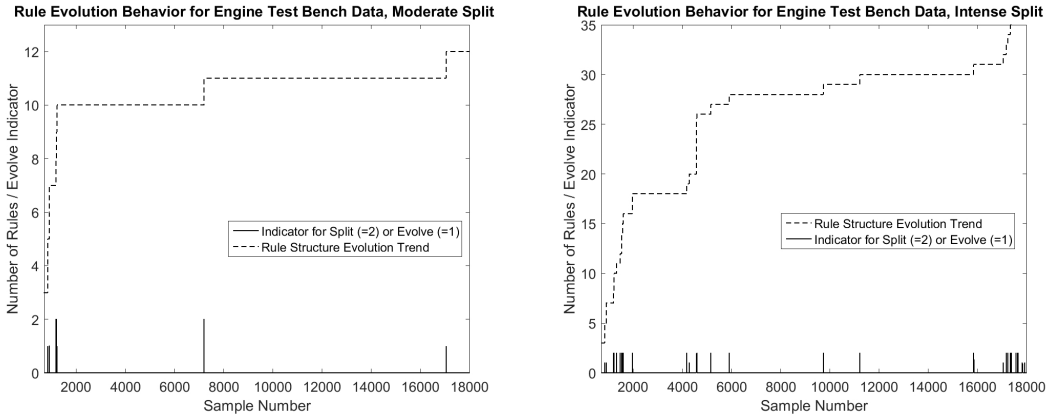

Figure 7. Left: Rule evolution trend line over time (dashed line) for a moderate splitting during the modeling on **engine test benches** and the indicator whether classical evolution or splitting was conducted in solid line; right: the same with a more intense splitting ($n = 2$) — please mind the different scales in the y-axis

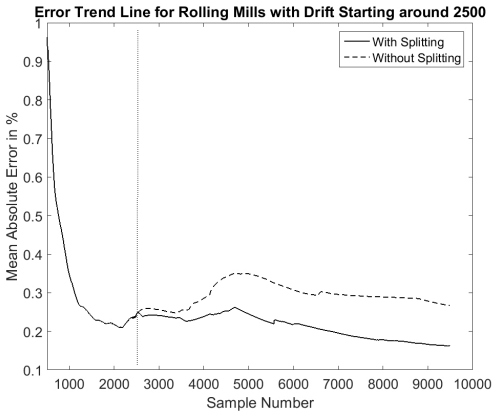

Figure 8. Error trend lines over time for **rolling mills data** when no splitting is applied (dashed line) and when splitting is applied in case when setting $n = 1$ in the splitting condition (8); the start of the drift is indicated by the vertical dotted line

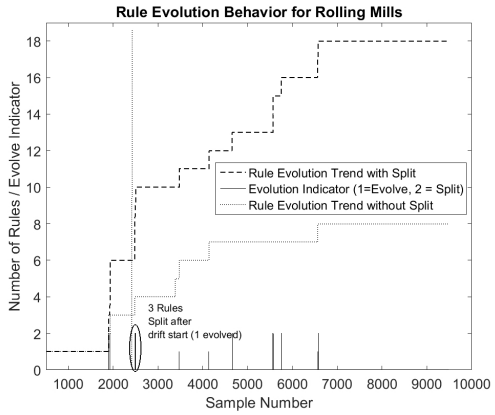

Figure 9. Rule evolution trend line over time for a moderate splitting with $n = 1$ during the modeling on **rolling mills data** and the indicator whether classical evolution or splitting was conducted in solid line

splitting is carried out.

B. Rolling Mills (with real occurring drift included)

The error trend lines for the rolling mills data are visualized in Figure 8. It can be easily recognized the significant impact of the splitting process on the error trend lines, as this improves much its level by about one third after the start of the drift at around sample #2500 (indicated by the dotted vertical line). When inspecting the rule evolution trend line in Figure 9 after the drift happened, it can be realized, by comparing the dashed with the dotted line (representing rule evolution trend with and without splits), that three rules are generated due to splitting whereas only one rule is born due to the classical evolution: leaving this one rule bearing along leads to the worse error trend line in Figure 8. Obviously, the drift is not intense enough (as also confirmed by experts which observed a moderate gradual drift) in order to trigger more classical rule evolutions. Instead available rules are growing, which then model the data (and the associated regression context) before and after the drift within a 'wide-spread region' with low accuracy (note that the size of the rule also has to be big enough such that a split is triggered). Latter additional rules are split at around sample #5500 and #6500, being responsible that the error trend line decrease better towards the end of the stream than when applying no splits.

VI. Conclusion and Outlook

We proposed a rule splitting concept for generalized evolving fuzzy models in case of streaming regression problems, which is capable to be performed in fully on-line and single-pass manner. It directly acts on the multi-dimensional rule kernel functions employing the eigenvalue decomposition of the inverse covariance matrix, and thus does not require any back-projections and axis-parallel splits. By employing two splitting conditions, it assures that no over-fitted situations are further split (which would even increase the over-fitting and the local error of rules). Its ability to adequately compensate gradual drifts which cause an untypical growth of rules leading to high model errors has been underlined by several experiments on streaming data recorded at two different application scenarios: rule splitting lead to a higher flexibility of evolving

fuzzy models to account for changes (drifts) and thus could finally reduce the error trend lines over time significantly.

Future work includes (i) the embedding of sample significance in the statistical process control equation (8) for deciding when to split rules for the purpose to dynamically change the parameter n and (ii) the integration of the 'smartness' aspect (dynamic dimensionality reduction) in the whole evolving learning process.

ACKNOWLEDGEMENTS

The first author acknowledges the Austrian research funding association (FFG) within the scope of the 'IKT of the future' programme, project 'Generating process feedback from heterogeneous data sources in quality control (mvControl)' (contract # 849962).

REFERENCES

[1] P. Angelov, D. Filev, and N. Kasabov, *Evolving Intelligent Systems — Methodology and Applications*. New York: John Wiley & Sons, 2010.

[2] J. Gama, *Knowledge Discovery from Data Streams*. Boca Raton, Florida: Chapman & Hall/CRC, 2010.

[3] J. Dean, *Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners*. Hoboken, New Jersey: John Wiley and Sons, 2012.

[4] M. Sayed-Mouchaweh and E. Lughofer, *Learning in Non-Stationary Environments: Methods and Applications*. New York: Springer, 2012.

[5] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, New Jersey: Prentice Hall PTR, Prentic Hall Inc., 1999.

[6] A. Lemos, W. Caminhas, and F. Gomide, "Adaptive fault detection and diagnosis using an evolving fuzzy classifier," *Information Sciences*, vol. 220, pp. 64–85, 2013.

[7] P. Angelov, P. Sadeghi-Tehran, and R. Ramezani, "An approach to automatic real-time novelty detection, object identification, and tracking in video streams based on recursive density estimation and evolving takagisugeno fuzzy systems," *International Journal of Intelligent Systems*, vol. 26, no. 3, pp. 189–205, 2011.

[8] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghedira, "Discussion and review on evolving data streams and concept drift adapting," *Evolving Systems*, vol. on-line and in press, pp. doi: 10.1007/s12 530–016–9168–2, 2016.

[9] A. Shaker and E. Lughofer, "Self-adaptive and local strategies for a smooth treatment of drifts in data streams," *Evolving Systems*, vol. 5, no. 4, pp. 239–257, 2014.

[10] P. Zhao, S. Hoi, J. Wang, and B. Li, "Online transfer learning," *Artificial Intelligence*, vol. 216, pp. 76–102, 2014.

[11] P. Angelov, *Autonomous Learning Systems: From Data Streams to Knowledge in Real-time*. New York: John Wiley & Sons, 2012.

[12] E. Lughofer and M. Pratama, "On-line active learning in data stream regression using uncertainty sampling based on evolving generalized fuzzy models," *IEEE Transactions on Fuzzy Systems*, vol. on-line and in press, no. DOI: 10.1109/TFUZZ.2017.2654504, 2017.

[13] L. Ngia and J. Sjöberg, "Efficient training of neural nets for nonlinear adaptive filtering using a recursive Levenberg-Marquardt algorithm," *IEEE Trans. Signal Processing*, vol. 48, no. 7, pp. 1915–1926, 2000.

[14] E. Lughofer, "Towards robust evolving fuzzy systems," in *Evolving Intelligent Systems: Methodology and Applications*, P. Angelov, D. Filev, and N. Kasabov, Eds. New York: John Wiley & Sons, March 2010, pp. 87–126.

[15] ——, *Evolving Fuzzy Systems — Methodologies, Advanced Concepts and Applications*. Berlin Heidelberg: Springer, 2011.

[16] J. Castro and M. Delgado, "Fuzzy systems with defuzzification are universal approximators," *IEEE Transactions on Systems, Man and Cybernetics, part B: Cybernetics*, vol. 26, no. 1, pp. 149–152, 1996.

[17] M. Gacto, R. Alcala, and F. Herrera, "Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures," *Information Sciences*, vol. 181, no. 20, pp. 4340–4360, 2011.

[18] E. Lughofer, "On-line assurance of interpretability criteria in evolving fuzzy systems — achievements, new concepts and open issues," *Information Sciences*, vol. 251, pp. 22–46, 2013.

[19] J. Rubio, "Least square neural network model of the crude oil blending process," *Neural Networks*, vol. 78, pp. 88–96, 2016.

[20] ——, "Adaptive least square control in discrete time of robotic arms," *Soft Computing*, vol. 19, no. 12, pp. 3665–3676, 2015.

[21] Q. Liu, J. Yin, V. Leung, J. Zhai, Z. Cai, and J. Lin, "Applying a new localized generalization error model to design neural networks trained with extreme learning machine," *Neural Computing and Applications*, vol. 27, no. 1, pp. 59–66, 2016.

[22] J. Viola and L. Angel, "Identification, control and robustness analysis of a robotic system using fractional control," *IEEE Latin America Transactions*, vol. 19, no. 12, pp. 3665–3676, 2015.

[23] G. Leng, X.-J. Zeng, and J. Keane, "An improved approach of self-organising fuzzy neural network based on similarity measures," *Evolving Systems*, vol. 3, no. 1, pp. 19–30, 2012.

[24] E. Lughofer, "Evolving fuzzy systems — fundamentals, reliability, interpretability and useability," in *Handbook of Computational Intelligence*, P. Angelov, Ed. New York: World Scientific, 2016, pp. 67–135.

[25] D. Dovzan and I. Skrjanc, "Recursive clustering based on a gustafson-kessel algorithm," *Evolving Systems*, vol. 2, no. 1, pp. 15–24, 2011.

[26] W. Wang and J. Vrbanek, "An evolving fuzzy predictor for industrial applications," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1439–1449, 2008.

[27] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Applied Soft Computing*, vol. 11, no. 2, pp. 2057–2068, 2011.

[28] A. Kalhor, B. Araabi, and C. Lucas, "An online predictor model as adaptive habitually linear and transiently nonlinear model," *Evolving Systems*, vol. 1, no. 1, pp. 29–41, 2010.

[29] D. Dovzan, V. Logar, and I. Skrjanc, "Implementation of an evolving fuzzy model (eFuMo) in a monitoring system for a waste-water treatment process," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 5, pp. 1761–1776, 2015.

[30] M. Pratama, S. Anavatti, and J. Lu, "Recurrent classifier based on an incremental meta-cognitive scaffolding algorithm," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2048–2066, 2015.

[31] B. Hartmann, O. Banfer, O. Nelles, A. Sodja, L. Teslic, and I. Skrjanc, "Supervised hierarchical clustering in fuzzy model identification," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 6, pp. 1163–1176, 2011.

[32] E. Lughofer, C. Cernuda, S. Kindermann, and M. Pratama, "Generalized smart evolving fuzzy systems," *Evolving Systems*, vol. 6, no. 4, pp. 269–292, 2015.

[33] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference and Prediction - Second Edition*. New York Berlin Heidelberg: Springer, 2009.

[34] A. Lemos, W. Caminhas, and F. Gomide, "Multivariable gaussian evolving fuzzy modeling system," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 91–104, 2011.

[35] M. Pratama, S. Anavatti, P. Angelov, and E. Lughofer, "PANFIS: A novel incremental learning machine," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 55–68, 2014.

[36] K. Krishnamoorthy and T. Mathew, *Statistical Tolerance Regions: Theory, Applications, and Computation*. Hoboken, New Jersey: John Wiley & Sons, 2009.

[37] K. Tabata and M. S. M. Kudo, "Data compression by volume prototypes for streaming data," *Pattern Recognition*, vol. 43, no. 9, pp. 3162–3176, 2010.

[38] R. Gray, "Vector quantization," *IEEE ASSP Magazine*, vol. 1, no. 2, pp. 4–29, 1984.

[39] E. Lughofer and M. Sayed-Mouchaweh, "Autonomous data stream clustering implementing incremental split-and-merge techniques — towards a plug-and-play approach," *Information Sciences*, vol. 204, pp. 54–79, 2015.

[40] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bulletin of the Calcutta Mathematical Society*, vol. 35, pp. 99–109, 1943.

[41] R. R. Yager, "A model of participatory learning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 5, pp. 1229–1234, 1990.

[42] E. Lughofer, G. Kronberger, M. Kommenda, S. Saminger-Platz, A. Promberger, F. Nickel, S. Winkler, and M. Affenzeller, "Robust fuzzy modeling and symbolic regression for establishing accurate and interpretable prediction models in supervising tribological systems," in *Proceedings of the IJCCI 2016 Conference*, Porto, Portugal, 2016, pp. 51–63.

[43] T. Hastie, R. Tibshirani, and J. Friedman, "Regularized paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, 2010.

[44] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society*, vol. 58 B, no. 1, pp. 267–288, 1996.
[45] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society, Series B*, pp. 301–320, 2005.
[46] I. Skrjanc, "Evolving fuzzy-model-based design of experiments with supervised hierarchical clustering," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 4, pp. 861–871, 2015.
[47] L. Teslic, B. Hartmann, O. Nelles, and I. Skrjanc, "Nonlinear system identification by gustafsonkessel fuzzy clustering and supervised local model network learning for the drug absorption spectra process," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 1941–1951, 2011.
[48] A. Shaker and E. Hüllermeier, "IBLStreams: a system for instance-based classification and regression on data streams," *Evolving Systems*, vol. 3, pp. 239–249, 2012.
[49] L. Jimenez and D. Landgrebe, "Supervised classification in high-dimensional space: Geometrical, statistical, and asymptotical properties of multivariate data," *IEEE Transactions on Systems, Man and Cybernetics, part C: Reviews and Applications*, vol. 28, no. 1, pp. 39–54, 1998.
[50] C. Cernuda, E. Lughofer, W. Maerzinger, and J. Kasberger, "NIR-based quantification of process parameters in polyetheracrylat (PEA) production using flexible non-linear fuzzy systems," *Chemometrics and Intelligent Laboratory Systems*, vol. 109, no. 1, pp. 22–33, 2011.
[51] E. Lughofer, V. Macian, C. Guardiola, and E. Klement, "Identifying static and dynamic prediction models for nox emissions with evolving fuzzy systems," *Applied Soft Computing*, vol. 11, no. 2, pp. 2487–2500, 2011.
[52] J. Morris, S. Hardy, and J. Thomas, "Some fundamental considerations for the control of residual flatness in tension leveling," *Journal of Materials Processing Technology*, vol. 120, pp. 385–396, 2002.
[53] F. Serdio, E. Lughofer, A.-C. Zavoianu, K. Pichler, M. Pichler, T. Buchegger, and H. Efendic, "Improved fault detection employing hybrid memetic fuzzy modeling and adaptive filters," *Applied Soft Computing*, vol. 51, pp. 60–82, 2017.
[54] F. Serdio, E. Lughofer, K. Pichler, M. Pichler, T. Buchegger, and H. Efendic, "Fuzzy fault isolation using gradient information and quality criteria from system identification models," *Information Sciences*, vol. 316, pp. 18–39, 2015.
[55] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive online analysis," *Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

Edwin Lughofer received his PhD-degree from the Johannes Kepler University Linz (JKU) in 2005. He is currently Key Researcher with the Fuzzy Logic Laboratorium Linz / Department of Knowledge-Based Mathematical Systems (JKU) in the Softwarepark Hagenberg, see www.flll.jku.at/staff/edwin/. He has participated in several basic and applied research projects on European and national level, with a specific focus on topics of Industry 4.0 and FoF (Factories of the Future). He has published around 180 publications in the fields of evolving fuzzy systems, machine learning and vision, data stream mining, chemometrics, active learning, classification and clustering, fault detection and diagnosis, quality control, predictive maintenance, including around 70 journal papers in SCI-expanded impact journals, a monograph on Evolving Fuzzy Systems (Springer) and an edited book on Learning in Non-stationary Environments (Springer). In sum, his publications received 3240 references achieving an h-index of 33. He is associate editor of the international journals IEEE Transactions on Fuzzy Systems, Evolving Systems, Information Fusion, Soft Computing and Complex and Intelligent Systems, the general chair of the IEEE Conference on EAIS 2014 in Linz, the publication chair of IEEE EAIS 2015, 2016, 2017 and 2018, and the Area chair of the FUZZ-IEEE 2015 conference in Istanbul. He co-organized around 12 special issues in international journals and more than 20 special sessions in international conferences. In 2006 he received the best paper award at the International Symposium on Evolving Fuzzy Systems, in 2013 the best paper award at the IFAC conference in Manufacturing Modeling, Management and Control (800 participants) and in 2016 the best paper award at the IEEE Intelligent Systems Conference.

Mahardhika Pratama received his PhD degree from the University of New South Wales, Australia in 2014. He completed his PhD in 2.5 years with a special approval of the UNSW higher degree committee due to his outstanding PhD research achievement. He is currently assistant professor at School of Computer Science and Engineering, Nanyang Technological University. Before joining NTU. He worked as a lecturer at the Department of Computer Science and IT, La Trobe University from 2015 till 2017. Prior to joining La Trobe University, he was with the Centre of Quantum Computation and Intelligent System, University of Technology, Sydney as a postdoctoral research fellow of Australian Research Council Discovery Project. Dr. Pratama received various competitive research awards in the past 5 years, namely the Institution of Engineers, Singapore (IES) Prestigious Engineering Achievement Award in 2011, the UNSW high impact publication award in 2013 and 2014. He recently has been appointed as Indonesias government world-class professor. Dr. Pratama has produced over 64 high-quality papers in journals and conferences and edited one book, and has been invited to deliver keynote speeches in international conferences. Dr. Pratama has led five special sessions and two special issues in prestigious conferences and journals. He currently serves as an editor in-chief of International Journal of Business Intelligence and Data Mining and a consultant at Lifebytes, Australia. Dr. Pratama is a member of IEEE, IEEE Computational Intelligent Society (CIS) and IEEE System, Man and Cybernetic Society (SMCS), and Indonesian Soft Computing Society (ISC-INA). His research interests involve machine learning, computational intelligent, evolutionary computation, fuzzy logic, neural network and evolving adaptive systems.

Igor Skrjanc is a Full Professor at the Faculty of Electrical Engineering, University of Ljubljana, Slovenia. His main research areas are adaptive, predictive, neuro-fuzzy and fuzzy adaptive control systems. His current research interests include also the field of autonomous mobile systems in sense of localization, direct visual control and trajectory tracking control. His publications include more than 80 papers with SCI factor; 8 chapters in international books; two scientific monographies Predictive approaches to control of complex systems published by Springer (as co-author) and Wheeled Mobile Robotics, From Fundamentals Towards Autonomous Systems published by Elsevier; 153 conference contributions (as author and co-author); 6 university books. Conducting 24 international and domestic projects, he has obtained 4 patents and several awards: the best diploma work in the field of Automation, Bedjani award (1988), the Vodovnik award, for outstanding research results in the field of intelligent control (2007), the Humboldt research award for long term stay and research at University of Siegen (2008), Chair of Excellence research grant from University Carlos 3 of Madrid (2017), JSPS grant for research at University of Kobe (2016), 1st place at the competition organized by IEEE Computational Society, Learning from the data in the frame of IEEE World Congress on Computational Intelligence, Brisbane, Australia (2012), the most important Slovenian research award for his work in the area of computational intelligence in control Zois award (2013).