Online sequential ensembling of predictive fuzzy systems

Edwin Lughofer¹ · Mahardhika Pratama²

Received: 17 July 2020 / Accepted: 2 August 2021 / Published online: 25 August 2021
© The Author(s) 2021

Abstract

Evolving fuzzy systems (EFS) have enjoyed a wide attraction in the community to handle learning from data streams in an incremental, single-pass and transparent manner. The main concentration so far lied in the development of approaches for single EFS models, basically used for prediction purposes. Forgetting mechanisms have been used to increase their flexibility, especially for the purpose to adapt quickly to changing situations such as drifting data distributions. These require forgetting factors steering the degree of timely out-weighting older learned concepts, whose adequate setting in advance or in adaptive fashion is not an easy and not a fully resolved task. In this paper, we propose a new concept of learning fuzzy systems from data streams, which we call *online sequential ensembling of fuzzy systems (OS-FS)*. It is able to model the recent dependencies in streams on a chunk-wise basis: for each new incoming chunk, a new fuzzy model is trained from scratch and added to the ensemble (of fuzzy systems trained before). This induces (i) maximal flexibility in terms of being able to apply variable chunk sizes according to the actual system delay in receiving target values and (ii) fast reaction possibilities in the case of arising drifts. The latter are realized with specific prediction techniques on new data chunks based on the sequential ensemble members trained so far over time. We propose four different prediction variants including various weighting concepts in order to put higher weights on the members with higher inference certainty during the amalgamation of predictions of single members to a final prediction. In this sense, older members, which keep in mind knowledge about past states, may get dynamically reactivated in the case of cyclic drifts, which induce dynamic changes in the process behavior which are re-occurring from time to time later. Furthermore, we integrate a concept for properly resolving possible contradictions among members with similar inference certainties. The reaction onto drifts is thus autonomously handled on demand and on the fly during the prediction stage (and not during model adaptation/evolution stage as conventionally done in single EFS models), which yields enormous flexibility. Finally, in order to cope with large-scale and (theoretically) infinite data streams within a reasonable amount of prediction time, we demonstrate two concepts for pruning past ensemble members, one based on atypical high error trends of single members and one based on the non-diversity of ensemble members. The results based on two data streams showed significantly improved performance compared to single EFS models in terms of a better convergence of the accumulated chunk-wise ahead prediction error trends, especially in the case of regular and cyclic drifts. Moreover, the more advanced prediction schemes could significantly outperform standard averaging over all members' outputs. Furthermore, resolving contradictory outputs among members helped to improve the performance of the sequential ensemble further. Results on a wider range of data streams from different application scenarios showed (i) improved error trend lines over single EFS models, as well as over related AI methods OS-ELM and MLPs neural networks retrained on data chunks, and (ii) slightly worse trend lines than on-line bagged EFS (as specific EFS ensembles), but with around 100 times faster processing times (achieving low processing times way below requiring milli-seconds for single samples updates).

Keywords Online data streams · Online sequential ensembles · Predictive fuzzy systems · Robust chunk-wise training · Uncertainty-based weighted predictions · Autonomous and flexible drift handling · Resolving contradictory outputs · Dynamic pruning of ensemble members

✉ Edwin Lughofer
edwin.lughofer@jku.at

Extended author information available on the last page of the article

1 Introduction

Due to the increasing complexity, dynamics and non-stationarity of current industrial installations (Lughofer and Sayed-Mouchaweh 2019; Angelov et al. 2010, 2012), the static design of fuzzy systems, e.g. realized through coding of expert knowledge (Siler and Buckley 2005) or through conventional batch learning techniques from pre-collected off-line data (Nelles 2001; Babuska 1998), have become a bottleneck in terms of permanent adaptability and flexibility to react properly on changing situations. Often, they become outdated soon during the on-line process, which in turn leads to inaccurate and wrong predictions, affecting the performance of the whole system. Therefore, *evolving (neuro)-fuzzy systems (E(N)FS)* have emerged as prominent methodology during the last decade in order to properly address the demands in (fast) online modeling and stream mining processes with changing characteristics. This has been achieved within a wider range of real-world applications, e.g., predictive maintenance, fault detection and prognosis, time-series forecasting, (bio-)medical applications, user behavior identification, to name a few, see also Lughofer (2016) for a longer list. This is because E(N)FS possess the ability (i) to learn their parameters and structure from stream samples in an incremental, single-pass manner achieving fast stream processing capabilities, (ii) to adapt quickly to regular system fluctuations and to new operating conditions by expanding their knowledge on the fly, and (ii) to properly react on drifting (changing) system situations (Khamassi et al. 2017) and non-stationary environments by forgetting mechanisms. This is typically achieved in a fully autonomous manner (Angelov 2011, 2012).

1.1 Related state-of-the-art and motivation

Recent surveys on E(N)FS can be found in Skrjanc et al. (2019) and Lughofer (2016), comprising a comparison of 50+ most widely used approaches and their embedded update engines. All of these approaches (with a few exceptions, see below) are operating on a single model level, i.e. one fuzzy system is defined [internally employing a particular structure and rule base architecture, for which various possibilities have been suggested (Skrjanc et al. 2019; Lughofer 2016)], its parameters recursively updated and its structures evolved over time based on stream samples. A single model indeed may have its charm regarding a fast update performance, but may have some flexibility problems in terms of abrupt, gradual and/or cyclic drifts. Indeed, forgetting mechanisms as proposed in Lughofer and Angelov (2011), Pratama et al. (2017), Dovzan et al.

(2015) may help to increase flexibility. This is because these mechanisms over-weigh new samples compared to older ones and include the weight information in the model update process, typically in the recursive learning of consequent parameters, but sometimes also in the incremental and evolving clustering process (Lughofer and Angelov 2011), in order to adapt more strongly to the new situations and to more forget older ones. However, (i) the ideal forgetting factor varies from data stream to data stream case and (ii) the dynamic adaptation of the forgetting factor according to some criteria about the current model nature and performance is by far not an easy and still not a fully resolved task (Shaker and Lughofer 2014)—finally it is an additional parameter, which needs some tuning effort. Furthermore, in the case of drifts, single EFS models usually need some time to adapt to the new, drifted state. This is simply because their heavy structure containing several rules reflecting older states needs sufficient amount of samples of the drifting state, such that this becomes significant in the model inference. Sometimes new rules can be indeed quickly evolved based on a couple of new samples, but often drifts appear in a (slower) gradual manner (Khamassi et al. 2017), which do not induce the evolution of new rules but affect older rules to be blown up and becoming inaccurate (Pratama et al. 2017). Incremental rule splitting, as proposed in Lughofer et al. (2018), helps to overcome this effect, but works in an a posteriori manner, i.e. it needs significant time that it is activated and the drift properly compensated. Furthermore, in the case of target concept drifts with an unaffected input space (i.e. only the functional trend changes), the rules in a single EFS model contribute equally to the final model output (through normalized membership degrees on the input space) before and during the drift; thus, no dynamic out-weighing of older (inappropriate) rules is achieved during the prediction stage.

Ensembles of models can typically increase the robustness in performance in noisy cases, as is known from the (classical) batch machine learning community (Zhou 2012; Brazdil et al. 2009). Furthermore, ensembles are potential candidates to properly deal with drifting data distributions in a natural way. This is especially achieved by evolving new ensemble members upon drift alarms (Sidhu and Bathia 2015), which then can be over-weighed in the final predictions compared to older members. Older members which stay in the ensemble can be quickly reactivated on-the-fly when cyclic drifts occur—these older members are already significant in terms of having been updated on several samples before and thus typically produce reliable predictions (with high certainty) already at the beginning of their reactivation. So, combining the idea of ensembling with EFS, leading to a kind of *online ensembles of EFS*, is a beneficial methodological direction

to increase the robustness and flexibility of current single EFS models. Despite of this, *online ensembles of EFS* have been loosely addressed so far in the community: (i) in Pratama et al. (2018), which is a boosting approach along subspace selection (ensemble members operate in different sub-spaces) and which is designed for classification problems [using fuzzy classifiers with particular properties as ensemble members, which are evolved by the pClass approach (Pratama et al. 2015)]; (ii) the same consideration (classification purpose) goes to the approach recently proposed in Iglesias et al. (2020), which employs bootstrapping (Efron and Tibshirani 1993) for creating diversity among ensemble members and a dynamic selection scheme of the best performing sub-groups within an ensemble; the members are trained by the ALMMo-0 approach proposed in Angelov and Gu (2017) to obtain multi-model classifiers; (iii) in Leite and Skrjanc (2019) the models are essentially different from each other due to a different set of meta-parameters (constraints on the eOGS objectives); the meta-parameters steer the complexity of the models—in this sense, this approach can be seen as a kind of extended boosting approach with the possibility to even include less and more complex ensemble members [in conventional boosting, only weak learners are considered (Collins et al. 2002; Polikar 2006)]; (iv) in Lughofer et al. (2021) an on-line bagging approach of EFS for stream regression problems has been recently demonstrated which performs a specific on-line sampling scheme to increase robustness of the overall ensembles due to convergence to the classical batch sampling approach in bagging and integrates on-the-fly pruning and evolution concepts for base members.

All of these approaches fall into the category of *fully-spanned ensembles* in a sense that all (or at least a significant subset of) ensemble members are permanently updated synchronously with new samples. This may indeed increase stability with respect to catastrophic forgetting issues (French 1999), but on the other hand may lose significant computational power [especially when the ensemble gets huge due to a dynamic creation of ensemble members over time (Pratama et al. 2018; Lughofer et al. 2021)]. Furthermore, as in the case of single EFS models, drift handling is conducted during the model/ensemble update stage (and not during the prediction stage). Moreover, they have been designed and especially evaluated on the assumption to have the target (reference) values permanently (non-delayed) available for each single sample measured in the input space. The same is the case for most of the single EFS modeling approaches (Skrjanc et al. 2019). However, in some applications the target values may be only available with some delay [typically, due to system latency (Marrs et al. 2012; Das et al. 2020) or because of cost reduction reasons due to titration processes etc. (Cernuda et al. 2014)].

1.2 Content of this paper—our approach

In this paper, we propose the concept of *on-line sequential ensembles* of predictive fuzzy systems, termed as *OS-FS*. The key aspect of such ensembles is that a new (small) fuzzy system is trained from scratch solely based on a recent data chunk and added to the ensemble of previously learned models. In this sense, 'small' members make up the whole sequential ensemble. No older models are updated and also no evolving concept is necessarily requested to train the (small) fuzzy systems on single chunks [thus, it is in fact an on-line ensemble of fuzzy systems (FS) and not of evolving fuzzy systems (EFS)]. The newly generated models always automatically represent the more recent trends of the stream which can be nicely exploited for handling drifts properly (through the specific prediction variants demonstrated in Sect. 2.2), whereas older ones can still be reactivated on demand (e.g., in the case of cyclic drifts where older states are revisited). This is achieved dynamically and parameter-free on the fly during the prediction stage (see Sect. 2.2).

Our sequential ensembles are built over time (in an incremental data stream-based fashion) and thus differ to other sequential ensembles in literature, which are built over iterations of a batch data set—e.g., sequential bagged ensembles, where latter bags may depend on (the production of) previously generated bags, as discussed in Seni and Elder (2010), Rayana et al. (2016). A timely-based sequential approach has been proposed in Ding et al. (2017) for neural networks, where hidden correlation and target matrices (based on which new predictions are made) are adjoined over data chunks by simple addition. This is possible due to the specific inference scheme, but the approach does not embed real ensemble members in form of distinct models—the hidden structure of a neural network is successively expanded to again form one bigger network with a single prediction output. Another approach was recently introduced in Al-Mahasneh et al. (2019) for general regression neural networks, where the single members are trained by a specific modified version of back-propagation algorithm; the outputs of the members are aggregated by simply averaging the predictions from all members—we will empirically show the bottlenecks of standard averaging especially in the case of drifts in the results section.

For training the single ensemble members (fuzzy systems), we propose a robust variant of Gen-Smart-EFS (Lughofer et al. 2015) with two regularized options for learning consequent parameters in order to handle noise in the data properly (Sect. 2.1): fuzzily weighted elastic net (a convex combination of ridge regression and Lasso) and total weighted least squares which is able to properly handle noise in both, the inputs and the output; the latter, applied in a weighted form, demonstrates a new, alternative concept to the widely-used weighted least squares technique.

One key issue of our approach lies in the way how the over-all predictions for new data chunks are produced based on the single predictions of the ensemble members—e.g., emphasizing more recent members over older members typically leads to faster reactions on newly arising drifts. In this way, the reaction onto drifts is dynamically achieved on demand during the prediction stage (and not during the model adaptation stage as in single EFS models), which yields enormous flexibility and is parameter-free. We propose five different variants of aggregating the single predictions of the ensemble members to an overall prediction (see Sect. 2.2):

- Using the predictions of the most recently trained ensemble member (on the latest chunk) as over-all predictions for the samples in the new chunk → high flexibility in terms of reacting on drifts, but low stability due to forgetting all older members completely.
- Average of predictions over all ensemble members—classical, standard case as having been used by many other ensemble techniques such as bagging, random forests etc., see Seni and Elder (2010).
- Weighted average of predictions over all ensemble members with weights assigned by exponential forgetting of older members: this achieves higher flexibility for properly reacting on any timely based changes of relations, but does not forget older learned relations completely (thus avoiding possible catastrophic forgetting). On the other hand, it assumes permanent changes (older members are always down-weighted) and thus may be too flexible in the case of steady phases; furthermore, cyclic drifts are not respected.
- Weighted average of predictions over all ensemble members with weights equal to the model's coverage degrees of new samples to be predicted (lower weights for lower coverage degrees): the coverage degree serves as a kind of (output) certainty degree for new predictions, see also Lughofer and Pratama (2018)—typically, in the case of (input space) drifts, the coverage of those models which are representing older states before the drift decreases and thus these models are automatically down-weighted in the prediction of new (drifted) chunks. The reaction delay is expected to be short, as the coverage degree is an unsupervised measure, thus, it can act on new chunks without requiring the (delayed) target values; furthermore, older members are autonomously reactivated during the predictions in the case of cyclic drifts (see details below). The coverage degree can be easily calculated with fuzzy systems architecture based on rule fulfillment degrees with a clear geometric interpretation in the feature space—which serves as one motivation for its usage, see also Sect. 2.2.4.

- Weighted average of predictions over all ensemble members with weights achieved by recent error trends of ensemble members (lower weights for higher errors): these trends are incrementally updated once having received the target values of new chunks; in the case of drifts, the errors of older members (generated before the drift) will increase and thus their contributions to the predictions will decrease. The reaction delay is expected to be longer than in the previous case, as the error trend is a supervised measure, thus, target values for a new chunk need to be available first before the drift can become 'visible'; on the other hand, it is able to properly react on (regular and cyclic) drifts in the target concept as well.

Another key issue of our approach is the avoidance of massive ensembles which may deteriorate the performance speed of the ensemble significantly for making new predictions. Therefore, we propose a pruning of older members due to two concepts: (i) based on a statistical process control analysis of their recent error trends, where members with atypically high errors compared to other members become candidates for pruning (Sect. 2.4), and (ii) based on the diversity of ensemble members in terms of their prediction behavior (Sect. 2.5). Finally, another aspect in our approach is a proper handling of possible contradictory outputs among the ensemble members (Sect. 2.3). We present a new concept how to decrease the effect of contradictions.

We empirically compared the five prediction variants based on two data streams: one is an artificial one based on Friedman's multi-variate spline regression data (Friedman 1991), where we artificially built in one abrupt and one gradual drift (in the target concept) with a back-change to the old state (thus, mimicking a cyclic drift). The second one stems from a real-world application scenario, where a dynamic model for predicting the NO_x content should be established over time based on different driving profiles, which led to input space drifts; two new test campaigns were included towards the end of the stream inducing new operating conditions.

The results (Sect. 4) (i) show significantly improved performance in terms of decreased chunk-wise-ahead predicted error trends compared to single EFS model in both cases (see figures in Sect. 4) and for the embedded drifts and (ii) offer several interesting insights into the behavior of the sequential ensembles and their prediction variants. For instance, in the case of input space drifts and new operating conditions, the more advanced prediction schemes, which include uncertainty weighing concepts, could significantly outperform standard averaging over all members' outputs, whereas in the case of target concepts drifts the MAE-based certainty concept could significantly outperform the unsupervised coverage-based variant. Pruning had mostly no effect on the predictive performance, while it could significantly

Fig. 1 Visualization of the sequential ensemble scheme: all members sequentially trained on data chunks form the whole ensemble; the target values of all samples in a new chunk are predicted first by the ensemble ($\hat{y}$) (for which we demonstrate 5 variants in Sect. 2.2), and later, once the real target values (y) are available, a new fuzzy system can be trained and added to the ensemble (right most part with dashed arrows); due to the pruning of members (which become undesired/useless over time), the ensemble size can be dynamically reduced (as indicated by a crossed-out fuzzy system), for which two concepts will be proposed in Sects. 2.4 and 2.5

reduce the complexity of the ensemble and its computational demands (Table 1). Interestingly, the diversity-based pruning reduced the ensemble desirably to a couple of members in the case when the data stream remained pretty stable and thus models were just chunk-wise replicated (and hence actually not really needed)—thus, unnecessarily growing ensembles in stable/steady stream phases could be prevented by the pruning approach. Furthermore, resolving contradictory outputs among members helped to improve the performance of the sequential ensembles further.

Results on a wider range of data streams from different application scenarios show improved error trend lines over single EFS models, as well as over related AI methods OS-ELM and MLPs neural networks sequentially retrained on data chunks, and slightly worse trend lines than on-line bagged EFS (as specific fully-spanned EFS ensembles updating all its member permanently), but with around 100 times faster processing times (achieving low processing times way below requiring milli-seconds for single samples updates).

2 Online sequential ensembles of predictive fuzzy systems

2.1 Chunk-wise training of ensemble members

The training of the sequential ensembles over time is pretty straightforward: for each new incoming data chunk $X_T = (\mathbf{x}_T, \mathbf{y}_T)$ with size N (containing N samples), where N depends on the application scenario, as typically determined by the labelling delay, a new fuzzy system F_T is learned from scratch solely on the new chunk. This also means that (i) any fuzzy systems identification procedure (which is

reasonably fast enough to cope with on-line demands) can be applied for the chunk-wise training cycles (as no information from older samples is used during learning of each F_T), and (ii) the approach works in a single-pass mode (as is the case for most of the standard EFS approaches (Skrjanc et al. 2019; Lughofer 2016)). The learned fuzzy systems $F_t, t = 1, \dots, T$ are collected to form a joint ensemble $F_{ens} = F_1 \cup F_2 \cup \dots \cup F_T$, which is denoted as *sequential ensemble*, as ensembles are built sequentially over time. It memorizes knowledge from older members, while (permanently) acquiring new knowledge by training new members on recent data chunks. The conjunction of all members makes up the whole (enriched) ensemble knowledge, which is then used for making dynamic predictions on new samples. Figure 1 visually depicts this ensembling scheme.

Obviously, the flexibility of such an ensemble regarding dynamic system changes and drifts is high, because the recent ensemble members always represent the most recent states. This can be exploited to over-weigh them in the final predictions to react on changes/drifts properly, see Sect. 2.2. This also abandons a more complex forgetting strategy for out-weighting parameters and internal model structures, as is used in single EFS models during incremental updates in order to react on drifts properly (Skrjanc et al. 2019)—see, e.g., Dovzan et al. (2015), Zdsar et al. (2014), Pratama et al. (2017), Komijani et al. (2012): (i) setting a fixed forgetting factor is often not the ideal choice (due to possible variations in drift occurrences and intensities); (ii) how to adapt the value of the forgetting factor in an ideal manner is still not a fully resolved task—only some (loose) heuristics have been proposed in Shaker and Lughofer (2014) based on the drift intensity or in Lughofer and Angelov (2011) based on rule ages.

For the concrete learning of a fuzzy system on a current data chunk X_T , any (batch or evolving) fuzzy systems training approach with reasonable computational complexity could be used, i.e., which is able to terminate the training before the next chunk arrives. Batch fuzzy systems belong to the conventional training techniques [such as ANFIS (Jang 1993), genfis2 (Chiu 1994),...], allowing (optimization) iterations over a batch data set multiple times to achieve optimal solutions of parameters and structures. Thus, they are typically slower than EFS methods (see their description in the introduction section), which are operating in single-pass incremental manner without re-iterations. We use a modified version of Gen-Smart-EFS (Lughofer et al. 2015), which, as a particular EFS method, is a single-pass learning algorithm allowing very fast processing of the samples (as it has a linear complexity with the number of samples in a chunk), and this with the achievement of an autonomous elicitation of the adequate number of rules to represent the implicit learning problem properly by local granules in form of fuzzy rules. Therefore, it integrates generalized fuzzy rules (with arbitrarily rotated ellipsoidal contours), as proposed in Lemos et al. (2011) to model local correlations between variables properly. One local correlation (model) can be seen as a local piece-wise part of a global (in general) non-linear regression trend. A single pass over the current chunk is performed in order to elicit the adequate number of rules for modeling the input-target relation. Note that this number can vary from chunk to chunk according to changes in the degree of non-linearity in the relation over time. So, due to a re-learning based on chunks, maximal flexibility can be achieved to dynamically reflect these changes well. The single-pass functionality for antecedent learning includes (i) a statistical-based rule evolution criterion which decides when to evolve a new rule (based on tolerance regions of prediction intervals in the product space), and (ii) incremental, recursive updates of rule centers and inverse covariance matrices (approaching batch estimations well), see Lughofer et al. (2015) for derivations based on the Neumann series trick on matrices.

2.1.1 Robust consequent estimation with elastic net regularization

After the single-pass rule antecedent learning, the consequent parameters $\mathbf{w}_i$, $i = 1, \dots, C$ of the TS fuzzy system, with C the number of rules, are estimated, which appear as linear parameters to define the hyper-planes in the rule consequents (Takagi and Sugeno 1985):

$$f_i(\mathbf{x}, \mathbf{w}_i) = w_{i0} + w_{i1} * x_1 + \dots + w_{ip} * x_p \quad (1)$$

with p the number of inputs and i the index of the i th rule. Therefore, we propose a fuzzily weighted elastic net

approach, embedding a convex combination of ridge regression and Lasso, in order to assure an implicit regularization effect in the case of noise in the data and to shrink as many parameters as possible towards 0. The latter is typically achieved by the Lasso approach (Tibshirani 1996), which includes the sum of absolute values over all regression coefficients in the objective function. This induces a sharp shrinkage effect towards 0, as explained in detail with figures in Hastie et al. (2009), also in comparison with a softer shrinkage effect achieved by ridge regression. This yields the following objective function for all $i = 1, \dots, C$ rules in order to regularize the conventional weighted least squares (WLS) approach:

$$J_i = \sum_{k=1}^N \Psi_i(\mathbf{x}(k)) e_i^2(k) + \lambda \sum_{j=1}^p (\alpha w_{ij}^2 + (1 - \alpha) |w_{ij}|) \longrightarrow \min_{\mathbf{w}_i} \quad (2)$$

with $e_i^2(k) = (y(k) - f_i(\mathbf{x}, \mathbf{w}_i))^2$ the squared residual between observed target values $y(k)$ and the predicted ones based on (1), $\Psi_i(\mathbf{x}(k))$ the normalized rule membership degree in the k th input sample $\mathbf{x}(k)$, λ the regularization parameter and α a parameter in $[0, 1]$, steering the degree of the influence of the Lasso regularization (third term) versus the classical ridge regression regularization (second term) (it is set to 0.5 per default to achieve an equal tradeoff). The first term denotes the fuzzily weighted least squares functional for achieving local learning for all C rules separately [reducing computational demands while increasing interpretability of the solutions (Angelov et al. 2008)], which almost all of the current E(N)FS approaches apply in their incremental update schemes (Skrjanc et al. 2019). Together with the Lasso regularization (third term), this leads to a kind of local feature selection, as unimportant parameters for explaining the target y are locally shrunk towards 0 (Hastie et al. 2010). The solution representation of the consequent vector $\mathbf{w}_i$ of (2) in a closed analytical form is given by the convex combination of the representations obtained by Lasso and ridge regression, thus by Hastie et al. (2010):

$$\hat{\mathbf{w}}_i = (\mathbf{X}^T \mathbf{Q}_i \mathbf{X} + \lambda_1 I)^{-1} \left(\mathbf{X}^T \mathbf{Q}_i \mathbf{y} - \frac{\lambda_2}{2} \text{sign}(\mathbf{w}_i) \right), \quad (3)$$

with $\mathbf{Q}_i$ the diagonal weight matrix $\mathbf{Q}_i = \text{diag}(\Psi_i(\mathbf{x}(k)))$, $k = 1, \dots, N$, $\lambda_1 = \lambda * \alpha$ and $\lambda_2 = \lambda * (1 - \alpha)$, and $\mathbf{X}$ the $N \times (p + 1)$ regression matrix: $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p, \mathbf{1}]$. This results in a least squares problem with 2^{p+1} inequality constraints, as there are 2^{p+1} possible sign patterns $\in \{-1, 1\}$ for the entries in the consequent parameter vector $\mathbf{w}_i$. This can be solved through a quadratic programming approach, termed as LARS-EN, see Zou and Hastie (2005).

The problem with this form of solution is its high computational complexity [iterative optimization cycles are required (Zou and Hastie 2005) in combination with a CV-based tuning of α], which may make the solution of (3) inapplicable in a fast streaming environment. Therefore, we suggest a slim version of elastic net regularization by setting $\alpha = 1$, which ends up in the pure ridge regression approach, as the third term in (2) vanishes. The solution representation of the consequent vector $\mathbf{w}_i$ can then be provided in a closed analytical form (Hastie et al. 2009), which in the fuzzily weighted (local) form is then given by Lughofer (2011):

$$\hat{\mathbf{w}}_i = (\mathbf{X}^T \mathbf{Q}_i \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{Q}_i \mathbf{y} \quad (4)$$

The final open issue is how to best set the regularization parameter λ . For this, several sophisticated (but also slow) algorithm are possible with pros and cons as extensively discussed in the survey paper (Bauer and Lukas 2011). We use our own approach as proposed in Lughofer and Kindermann (2010) (having achieved robust results in the past outperforming several other related methods), which just requires the calculation of the smallest and highest eigenvalues of the Hessian matrix $\mathbf{X}^T \mathbf{Q}_i \mathbf{X}$: λ is set according to the proportion of this eigenvalue, see Lughofer and Kindermann (2010) for details.

2.1.2 Total least squares estimation for respecting noise in the input

Although elastic net and ridge regression perform regularization to reduce the impact of noise on the solution and thus to stabilize it, they only handle noise in the target adequately, but do not emphasize stable solutions in the case of noise in the inputs.

In order to circumvent this bottleneck and thus to obtain a bias-free parameter estimation in the case of noisy targets and inputs, it is necessary to reconstruct both, targets and inputs. The total least squares optimization problem addresses this aspect by optimizing the following problem [for detailed derivations, see Markovsky and Huffel (2007)]:

$$J_{tot} = \frac{1}{2N} \left(\sum_{k=1}^N \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|^2 + \sum_{i=1}^N (y(k) - \hat{y}(k))^2 \right) \quad (5)$$

From a geometric point of view, with $\|\cdot\|$ denoting the L^2 -norm, the optimization of (5) achieves that the normal Euclidean distances between data samples and the hyper-plane (regression model) is minimized, whereas in standard LS the vertical distance is minimized.

In our case of consequent learning in fuzzy systems, we want to apply again a weighted approach in order to follow the local learning spirit (per rule). Hence, we formulate the weighted total least squares (WTLS) optimization problem as:

$$J_i = \frac{\sum_{k=1}^N \Psi_i(\mathbf{x}(k)) \|\mathbf{x}(k) - \hat{\mathbf{x}}(k)\|^2 + \sum_{i=1}^N \Psi_i(\mathbf{x}(k)) (y(k) - f_i(\mathbf{x}, \mathbf{w}_i))^2}{2N} \quad (6)$$

with $\Psi_i(\mathbf{x}(k))$ again the normalized membership value of the k th sample in the i th rule. According to the derivation in Markovsky and Huffel (2007) for the non-weighted TLS, the following norm needs to be optimized for the weighted version:

$$\|\sqrt{\mathbf{Q}_i}(\mathbf{R} - \hat{\mathbf{R}})\|^2 \quad (7)$$

where $\mathbf{R}$ denotes the regression matrix containing the p input variables (without a column of ones) joined with the target vector $\mathbf{y}$, i.e. $\mathbf{R} = [\mathbf{y} | \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_p]$, and $\hat{\mathbf{R}}$ the affine linear reconstruction of $\mathbf{R}$ (Markovsky and Huffel 2007):

$$\hat{\mathbf{R}} = \mathbf{R} - ((\mathbf{R} - \mathbf{1}\mathbf{p}_i^T)\mathbf{a}_i)^T \quad (8)$$

with $\mathbf{a}_i$ the unit normal vector to the affine hyper-plane of the i th rule and $\mathbf{p}_i$ a point the hyper-plane passes through. Since $\|\mathbf{a}_i\| = 1$, the norm in (7) becomes

$$\|\sqrt{\mathbf{Q}_i}(\mathbf{R} - \hat{\mathbf{R}})\|^2 = \mathbf{a}_i^T (\mathbf{R}^T - \mathbf{p}_i \mathbf{1}^T) \mathbf{Q}_i (\mathbf{R} - \mathbf{1}\mathbf{p}_i^T) \mathbf{a}_i \quad (9)$$

By mean centering the matrix $\mathbf{R}$ with a weighted centroid vector $\boldsymbol{\Psi}_i = \text{diag}(\mathbf{Q}_i)$

$$\mu_{\mathbf{R}_i} = \frac{\mathbf{R}^T \boldsymbol{\Psi}_i}{\mathbf{1}^T \boldsymbol{\Psi}_i}, \quad (10)$$

i.e. the weighted mean of all samples (in each regressor) with weights given by the membership degrees to the i th rule, obtaining a mean centered matrix $\tilde{\mathbf{R}} = \mathbf{R} - \mu_{\mathbf{R}_i}$ by column-wise subtraction, the minimization of (9) yields:

$$\mathbf{a}_i^T (\tilde{\mathbf{R}}^T \mathbf{Q}_i \tilde{\mathbf{R}}) \mathbf{a}_i + \mathbf{1}^T \boldsymbol{\Psi}_i ((\mathbf{p}_i - \mu_{\mathbf{R}_i})^T \mathbf{a}_i)^2 = \min!_{\mathbf{a}_i, \mathbf{p}_i} \quad (11)$$

where the matrix $\tilde{\mathbf{R}}^T \mathbf{Q}_i \tilde{\mathbf{R}}$ denotes the Hessian matrix with the inclusion of the target vector $\mathbf{y}$ as first column and thus is symmetric and positive definite. Minimization of the first term in (11) is achieved when $\mathbf{a}_i$ is associated with the smallest eigenvector of the Hessian matrix [as equivalent to the classical matrix eigenvector decomposition problem—see Davila (1994)], whereas the minimization of the second term is obviously achieved in the case when $\mathbf{p}_i = \mu_{\mathbf{R}_i}$, as only then this term vanishes.

As $\mathbf{a}_i = [a_{i0}, a_{i1}, \dots, a_{ip}]$ is the smallest eigenvector of the Hessian matrix and all eigenvectors are orthogonal, it can be realized as the normal vector of the hyper-plane solution spanned by the more significant eigenvectors. Thus, the (implicit) normal vector form of the solution plane is given by:

$$a_{i0} * y + a_{i1} * r_1 + a_{i2} * r_2 + \dots + a_{ip} * r_p = \mathbf{p}_i^T \mathbf{a}_i, \quad (12)$$

with p the number of regressors. This is because $\mathbf{p}_i$ is a point the hyper-plane passes through and thus its scalar product with the normal vector coefficients, i.e. $\mathbf{p}_i^T \mathbf{a}_i$, yields the right hand side of the equational normal vector form. The prediction of the i th local model for a new data sample $\mathbf{x}$ is thus given by:

$$\hat{y} = \frac{\mathbf{p}_i^T \mathbf{a}_i - (a_{i1} * r_1 + a_{i2} * r_2 + \dots + a_{ip} * r_p)}{a_{i0}} \quad (13)$$

hence the consequent parameter vector $\mathbf{w}_i$ of the i th rule is set to

$$\mathbf{w}_i = \left[\frac{\mathbf{p}_i^T \mathbf{a}_i}{a_{i0}}, \frac{-a_{i1}}{a_{i0}}, \dots, \frac{-a_{ip}}{a_{i0}} \right]. \quad (14)$$

Thus, $\frac{\mathbf{p}_i^T \mathbf{a}_i}{a_{i0}}$ belongs to the intercept and the other coefficients to the p regressors. In the case when a_{i0} approaches 0, the prediction and the consequent parameter vector may become unstable, which can be avoided by dividing through a very small value of ϵ .

2.2 Variants of prediction schemes over trained sequential models

Apart from the antecedent and advanced consequent parameter learning techniques described in the previous section, the major methodological meat of our approach lies in the prediction scheme for aggregating the single predictions from the single members $F_1, \dots, F_T$ on new incoming samples $\mathbf{x}_1, \mathbf{x}_2, \dots$ (from a newly arising chunk X_{T+1}) — please note that the real (measured) target values of these samples do not need to be available at prediction stage, so predictions are made so long ahead until new targets are available that make up the entire chunk X_{T+1} . In the prediction schemes, it is decided how to respect different levels of prediction uncertainty of the single members in the final decision and how to address timely varying system changes and drifts properly. These aspects could not be handled during training as each member is trained based on a single chunk only, thus has a local partial view on the whole data stream. We propose five variants for producing an overall prediction from the single ones, which will be described in the subsequent sections and compared against each other in the results section.

2.2.1 Recent member takes all

The first variant is straightforward by using the predictions of the most recent trained member F_T (based on the latest data chunk X_T) as final predictions for new data samples $\mathbf{x}_1, \mathbf{x}_2, \dots$ in a recent (possibly unsupervised) chunk X_{T+1} , thus:

$$\hat{\mathbf{y}}_{T+1} = F_T(\mathbf{x}_{1, \dots, k_{T+1}}) \quad (15)$$

with $\mathbf{x}_i$ the input vector of the i th sample, leading to one prediction entry in $\hat{\mathbf{y}}_{T+1}$, with k_{T+1} samples included in data chunk X_{T+1} (so, k_{T+1} samples ahead are predicted). F_T is the functional response (output prediction) of the fuzzy system trained on the latest data chunk X_T ; it is produced through the conventional TS fuzzy systems inference scheme i.e., a weighted combination of consequent hyper-planes, with weights being the *normalized = relative* rule membership degrees (Takagi and Sugeno 1985), as also used in most of the current EFS approaches for stream regression:

$$\hat{F}_T(\mathbf{x}) = \sum_{i=1}^C \Psi_i(\mathbf{x}) \cdot f_i(\mathbf{x}, \mathbf{w}_i) \quad (16)$$

with $\Psi_i(\mathbf{x})$ the normalized rule membership degrees and $f_i(\mathbf{x}, \mathbf{w}_i)$ the consequent hyper-plane function (including the linear parameters $\mathbf{w}_i$). This strategy allows maximal flexibility to address arising drifts quickly, but may loose stability due to catastrophic forgetting (as no older learnt relations are respected for predicting the new chunk, thus completely forgotten).

2.2.2 Standard average

The second variant is composed by the standard average over all $|F_{ens}|$ members contained in the ensemble F_{ens} :

$$\hat{\mathbf{y}}_{T+1} = \frac{1}{|F_{ens}|} \sum_{i=1}^{|F_{ens}|} F_i(\mathbf{x}_{1, \dots, k_{T+1}}) \quad (17)$$

with F_i the functional response (output prediction) of the fuzzy system trained on the i th chunk. Please note that $|F_{ens}| = T$, with T the number of chunks seen so far, in the case when no pruning is used (see below); otherwise $|F_{ens}|$ can be smaller than T , and thus only the non-pruned member are applied in (17). In this variant, the time aspect is not respected at all, neither is the uncertainty of the single models' outputs. This may make it pretty inflexible to changing, drifting situations (which we will verify in the results section). This variant has been exclusively used by the approaches in the related works (Ding et al. 2017; Al-Mahasneh et al. 2019) before.

Fig. 2 Two dimensional example of drifting samples occurring in a new chunk X_{T+1} ; left: the newest fuzzy system member covers the drifting samples badly; middle: an older fuzzy system member covers the drifting samples better $\rightarrow$ higher coverage degree $\rightarrow$ higher weight

2.2.3 Timely exponential forgetting

The third variant integrates the points of time when the ensemble members have been created: further preceding members successively receive lower weights in the weighted average prediction than more recent ones and this in a smooth exponential way, thus:

$$\hat{y}_{T+1} = \frac{1}{\sum_{i=1}^{|F_{ens}|} \lambda^{T-t(i)}} \sum_{i=1}^{|F_{ens}|} F_i(\mathbf{x}_{1,\dots,k_{T+1}}) * \lambda^{T-t(i)} \quad (18)$$

with λ a forgetting factor (set per default to 0.99 to emphasize slow forgetting), T the index of the latest chunk based on which a member was trained and $t(i)$ the index of the chunk based on which the i th member was trained; when no pruning is applied (see below), $t(i) = i$, otherwise $t(i)$ may be greater than i . The motivation of this approach lies at hand: it can be more flexibly reacted upon drifts, while not suffering from any catastrophic forgetting, because all older members still contribute to the final predictions. On the other hand, it does not respect cyclic drifts, which may lead to a re-occurrence of past states after some time and thus may be already reflected in older members, neither does it explicitly react on regular drifts upon their happening—in the case when no drifts happen, the timely out-weighting may not be necessary or even counter-productive. Therefore, we further propose two additional variants which take into account these issues.

2.2.4 Weighted average based on prediction uncertainty

In this variant, the prediction certainty $cert_i, i = 1, \dots, |F_{ens}|$ of each ensemble member is used as weight in the overall weighted average prediction:

in the prediction; right: cyclic drift, samples are back-drifting to an old state which gets even better covered by Rule 4 in the older member $\rightarrow$ further increase in the prediction weight

$$\hat{y}_{T+1} = \frac{\sum_{i=1}^{|F_{ens}|} F_i(\mathbf{x}_{1,\dots,k_{T+1}}) * cert_i(\mathbf{x}_{1,\dots,k_{T+1}})}{\sum_{i=1}^{|F_{ens}|} cert_i(\mathbf{x}_{1,\dots,k_{T+1}})} \quad (19)$$

where $cert_i$ is calculated based on the rule coverage degree on single samples, which is measured through the ignorance concept proposed in Lughofer and Pratama (2018):

$$cert_i(\mathbf{x}_j) = \max_{k(i)=1,\dots,C(i)} \mu_{k(i)}(\mathbf{x}_j) \quad (20)$$

with $\mu_{k(i)}(\mathbf{x}_j)$ the rule membership fulfillment degree of the $k(i)$ th rule in sample $\mathbf{x}_j$ (with $C(i)$ rules in total included in the i th member). This degree can be calculated either through applying t-norms on the fuzzy set activation levels among the inputs [axis-parallel rules (Pedrycz and Gomide 2007)] or directly in the multi-dimensional space [arbitrarily rotated rules (Lemos et al. 2011)].

The certainty degree in (20) is automatically normalized into $[0, 1]$, where a value near 0 denotes that the sample is hardly covered by any rule, and a value near 1 denotes that it falls together with a rule center (thus, it is fully covered). In this sense, it is possible to take into account drifting phases in a new chunk X_{T+1} (to be predicted) in an *unsupervised manner*, as during these usually the samples are moving away from their original distribution (modelled through the antecedent parts of the rules). Thus, their coverage degrees decrease, which in turn effects the weighted prediction in (19). A two-dimensional example of a typical drift phase and its impact onto the sample coverage of the rules is visualized in Fig. 2.

On the left hand side, a fuzzy system with three rules (shown as ellipsoidal contours) is shown, where new samples from a drifting state are marked by dark dots (with the drifting direction indicated by an arrow): the rule fulfillment degree is low and decreases for further drifting samples, thus

this model receives a low weight in the weighted prediction scheme, whereas the older model F_2 (trained on the second chunk) receives a higher weight as its embedded Rule 4 covers the drifted samples better (middle plot). Furthermore, the certainty measure in (20) is able to handle cyclic drifts, i.e. system states may re-occur after some time which have been already reflected in older members (as being trained into them). This is because such older members then will cover the drifting samples well and thus receive higher weights than newer members. An example of such a situation is provided in the right plot in Fig. 2, where Rule 4 in F_2 becomes revisited, thus its coverage on new samples increases, which in turn leads to a higher impact of F_2 in (19) when using (20), and this fully automatically and autonomously without requiring any setting of forgetting factors or of other parameters or explicit drift detection techniques (Khamassi et al. 2017). Finally, we want to highlight that such a coverage degree can be hardly calculated with other types of machine learning models not having any geometric interpretation of their model components in the feature space. This yields a clear motivation for the usage of the fuzzy systems architecture.

2.2.5 Weighted average based on prediction error trend

The previous variant may handle drifts in an unsupervised fashion, thus may have an immediate effect on drifting samples included in a new data chunk. On the other hand, the certainty measure represented by the coverage degree does not respect how well the predictions of the ensemble members on the recent (and also older) chunks has actually been—they could be still good in the case of drifted states. In other words, the coverage degree may indicate correctly a drifted state within one or more ensemble members, but the models' predictions of these members may be unaffected. This may be especially the case when the extrapolation behavior of a member is robust—in the context of fuzzy systems, typically a less complex member (with lower number of rules/non-linearity degree) has a better extrapolation behavior than a more complex one, see several examinations in Nelles (2001), Lughofer (2011), Pedrycz and Gomide (2007). Furthermore, another bottleneck is that the coverage-based uncertainty degree as an unsupervised criterion is not able to properly handle sole target concept drifts, which are typical in form of a change in the functional trend while the data distribution in the input space remains (nearly) unchanged → input samples are still covered well by older members and thus the drift is not handled in the weighted prediction.

Therefore, to explicitly respect the (recent) prediction behavior of the single members, and thus to address target concept drifts properly, we propose a supervised certainty criterion to be integrated into the weighted prediction in (19). This is based on the accumulated error trends (measured in terms of mean absolute errors = MAEs) of the ensemble members $i = 1, \dots, F_{ens}$ over time, which for the N th target value upon its availability is updated by:

$$MAE_i(M) = \frac{(M-1)MAE_i(M-1) + |y(N) - \hat{y}_i(N)|}{M} \quad (21)$$

with $M = N - t(i) * k_T$, N the number of samples seen so far over all chunks for which the target values $y(1, \dots, N)$ have been available, T the number of chunks seen so far, k_T their chunk-size, $\hat{y}_i(N)$ the predicted target of the N th sample by the i th member, $t(i)$ the index of the chunk based on which the i th member was trained. Initially, $MAE_i(0) = 0$, thus the MAE of a newly generated member is set to 0. Note that the i th member has been trained on the $t(i)$ th chunk and used for predictions from the $t(i) + 1$ st chunk on, thus its MAE was updated/calculated based on $M = N - t(i) * k_T$ samples so far; generally $t(i) = i$, when no pruning took place, see below.

Now, for new data samples forming the most recent chunk X_{T+1} and having the MAEs from the members $F_1, \dots, F_{|F_{ens}|}$ available, the certainty weights for the weighted prediction in (19) are calculated by:

$$cert_i = \frac{MAE_i(N) - \max_{j=1, \dots, |F_{ens}|} (MAE_j(N))}{\min_{j=1, \dots, |F_{ens}|} (MAE_j(N)) - \max_{j=1, \dots, |F_{ens}|} (MAE_j(N))} \quad (22)$$

This yields that the best member with the smallest MAE receives a weight of 1 and the worst member with the highest MAE a weight of 0—in-between a linear interpolation is made. According to this weighing concept, the members with higher errors on the recent chunks (reflected in $MAE(N)$) are down-weighted, as they have either been outdated (e.g., due to drifts, now really affecting the members performance, not just the coverage) or been trained before on the least representative samples (e.g., with higher noise levels etc.) than other members. In this sense, undesired members can be automatically out-masked in current predictions, but may be reactivated at a latter stage. Finally, in order to respect input space and target concept drifts likewise, we combine the MAE-based certainty level with the coverage-based one as discussed in the previous section simply by multiplication, to form a new prediction variant (Variant 5 in the algorithm below).

2.3 Reducing the effect of contradicting predictions

It may happen that two or more members in the sequential ensemble have a high certainty in their predictions (due to a good coverage or a low MAE), but produce significantly different prediction values, i.e. widely spread ones within the range of the target—for instance, one ensemble member predicts a value of 0.1, and a second one a value of 0.9 for a $[0, 1]$ -normalized target, and both with a certainty close to 1. Thus, none of the predicted values will be down-weighted, which ends up in a final predicted value close to 0.5 when conducting the weighted average. This is somewhat undesired, because it is an average of predictions coming from predictors that are expected to act well in specific (implicit) conditions: in this sense, the average will produce something arbitrary in-between.

In order to resolve this undesired effect, we integrate an additional weight factor for the single ensemble members, which depends on the weighted standard deviation of the predictions w_{std} , with weights being the uncertainty degrees of the single ensemble members in their predictions (here for a single sample $\mathbf{x}(k)$), as defined by:

$$w_{fac} = 1 - \frac{w_{std}(F(\mathbf{x}(k)))}{\max(w_{std})} \quad (23)$$

$$w_{std}(F(\mathbf{x}_k)) = \sqrt{\frac{\sum_{i=1}^{|F_{ens}|} cert_i * (F_i(\mathbf{x}(k)) - \bar{F}_w)^2}{\sum_{i=1}^{|F_{ens}|} cert_i}}$$

where $\max(w_{std})$ is the maximal variance a vector of predictions can have: assuming to have $[0, 1]$ -normalized target values, $\max(w_{std}) \approx 0.5$, which is the case when half of the predictions have the minimal value of 0 and the other half the maximal value of 1 and all of the ensemble members have equal certainty in their predictions. $\bar{F}_w$ denotes the weighted mean over all predictions with weights being the certainty degrees.

Equation (23) results in a low weight factor $w_{fac} \in [0, 1]$ when the weighted standard deviation w_{std} is high, which characterizes contradicting predictions. Hence, this weight factor is integrated in a form of decreasing exponential weights (similarly as the forgetting factor used in the timely-based prediction scheme, see above), where the decrease here is not conducted over time but over the certainty weights $cert_i$. In particular, this leads to the following weight factors for the $i = 1, \dots, |F_{ens}|$ ensemble members:

$$w_{fac}(i) = (w_{fac})^{i*} \quad i* = \text{rank}(F_i \in (Cert, >)) - 1$$

$$(Cert, >) = \{(cert_i > cert_j) \vee (cert_i = cert_j \wedge i > j) \mid i, j \in \{1, \dots, |F_{ens}|\}, i \neq j\} \quad (24)$$

where $\text{rank}(F_i \in (Cert, >))$ denotes the rank of the i th ensemble member in the ordered (descending) list of certainty degrees $(Cert, >)$, i.e. the F_i with highest certainty degree receives a rank of 1 and thus an exponential factor of 0 for its final weight factor $w_{fac}(i)$ ($\rightarrow$ yielding a full weight of 1), the F_i with second highest certainty degree a rank of 2 and thus an exponential factor of 1 for its final weight factor $w_{fac}(i)$ and so on. The definition also clarifies the case when two members have the same certainty degrees in their predictions: then, the more recent member (with higher index) appears first in the ordered list and thus receives a higher weight factor $w_{fac}(i)$. The weight factor $w_{fac}(i)$ is thus integrated into the weighted average (19) after re-assigning the certainty degree with:

$$cert_i = cert_i * w_{fac}(i) \quad (25)$$

This means that the member with the highest certainty degree stays with its original certainty degree (due to the exponent of $i* = 0$) when producing the weighted average, whereas the other members successively receive lower weights (in exponential decrease) due to their certainty rank, where the degree of decrease is steered by the standard deviation (and thus to some extent by the contradicting degree) of the original predictions: a high standard deviation leads to a low value of w_{fac} which in turn leads to a faster decrease of member weights over the ranked list due to the exponent $i*$. This means that the member with highest certainty still receives a weight of 1 while the influence of the others becomes much lower; in this sense, contradicting predictions are resolved by some sort of elitism. On the other hand, when the standard deviation (and thus the contradicting degree) is low, w_{fac} is close to 1 and the exponential weights $w_{fac}(i)$ achieved through (24) are just moderately decreasing over the ensemble members. Thus, (25) has then a negligible effect onto re-setting the certainty degrees and thus also for the weighted averaging of the predictions.

2.4 Eliminating undesired members

In the case of large (possibly infinite) data streams, the fuzzy ensemble may get so huge (containing thousands

of members) such that a significant slow-down in predicting new samples could become apparent, especially when applying the more complex schemes with certainty weights.

Therefore, we propose a pruning concept which eliminates undesired members. The idea is to check whether some ensemble members have much higher errors than others, thus whether they contribute badly to the final output prediction. Assuming that the accumulated MAEs [as calculated by (21)] over the ensemble members are (approximately) normally distributed, we employ the concept of statistical process control (Gama 2010) to check for atypical high errors. Therefore, the following condition is applied:

$$\exists_{i=1, \dots, |F_{ens}|} MAE_i(N) > \mu(MAE(N)) + n\sigma(MAE(N)) \quad (26)$$

with

$$\begin{aligned} \mu(MAE(N)) &= \frac{1}{|F_{ens}|} \sum_{i=1}^{|F_{ens}|} MAE_i(N) \\ \sigma(MAE(N)) &= \sqrt{\frac{1}{|F_{ens}|} \sum_{i=1}^{|F_{ens}|} (MAE_i(N) - \mu(MAE(N)))^2} \end{aligned} \quad (27)$$

and n a factor. The condition in (26) checks whether any ensemble member has a higher error than the mean error plus n standard deviations over the other ensemble members; and if so, this member is pruned: $F_{ens} = F_{ens} \setminus F_i$ and its MAE MAE_i erased. Per default, n can be set to 2 in order to emphasize a 2σ confidence interval (covering 96% of the samples).

This can be done for each new incoming data sample without running into danger that a too fine 'checking resolution' may lead to too optimistic prunings, because each MAE_i already contains a longer error trend over the stream and is updated smoothly with N becoming larger. Furthermore, a variable factor of n may help to omit too early prunings of members, especially in the case when the variance (and standard deviation) is close to 0. Thus, according to the considerations in Ashfahani et al. (2020), n can be adapted by

$$n = 1.3e^{-\sigma(MAE(N))} + 0.7 \quad (28)$$

This means that higher n 's close to 2 are achieved for low σ 's, which makes the pruning becoming less active whenever the variance is low.

2.5 Eliminating non-diverse members

Another possibility to decrease the size of the ensemble is to check for diversity of the members within the ensemble. Diversity of model ensembles is understood as the condition of having or being composed of differing members with either different structures or, even more significant in the case of (on-line) regression/classification problems, with different prediction behaviors (Kuncheva 2004). It is usually an essential criterion in order to assure high predictive performance of the whole ensemble, see Kuncheva and Whitaker (2003), Sannen et al. (2010) [also known from the concept of boosting ensemble members (Polikar et al. 2001)], as members with very similar behaviors would just end up in a pure confirmation in single prediction values. In our case of performing a weighted averaging of predictions, non-diversity among the single predictions of the members would not provide any further contribution to the predictive output and thus to the accuracy of the ensemble (even worse, it may hinder a good performance). This means that whenever there are K members with a very similar prediction behavior over time, $K - 1$ of them can be pruned as the remaining member still represent well their specific behavior. In the case when K is large, this may also avoid the undesired overwhelming of the diverse predictions of (a few) other members and the production of (permanent) monotonous outputs.

In our case of stream modeling, the prediction behavior would be an obvious measure to check for non-diverse members to be pruned. This also means that this pruning criterion becomes an unsupervised one as not requiring any target values; it can be thus immediately applied for each new incoming sample. The idea is to collect N predictions on N past samples from each member, termed as $\hat{\mathbf{y}}_i = [\hat{y}_i(1), \dots, \hat{y}_i(N)]$ for the i th member, and perform a hypothesis testing on the correlation coefficient of the predictions between members pairs whether they are (very) similar or not. According to the theory of statistics (Harrel 2001), it is well-known that the empirical correlation coefficient $r(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)$ between $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{y}}_j$, given by:

$$r(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j) = \frac{\sum_{k=1}^N (\hat{y}_i(k) - \bar{\hat{y}}_i)(\hat{y}_j(k) - \bar{\hat{y}}_j)}{\sqrt{\sum_{k=1}^N (\hat{y}_i(k) - \bar{\hat{y}}_i)^2} \sqrt{\sum_{k=1}^N (\hat{y}_j(k) - \bar{\hat{y}}_j)^2}} \quad (29)$$

with $\bar{\hat{y}}_i$ the mean value of $\hat{y}_i$, can be tested against the null-hypothesis that it shows no significant linear correlation through the t-test, whose test statistic is given by Ando (2010):

$$t(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j) = \frac{r(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j) \sqrt{N-2}}{\sqrt{1-r^2(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)}} \quad (30)$$

Thus, the null-hypothesis can be safely rejected, whenever the probability that $P_t(x > t(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j))$ is smaller than a significance level α (usually set to 0.05 per default), i.e.

$$P_t(x > t(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)) < \alpha. \quad (31)$$

The probability $P_t(x \leq t(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)) = 1 - P_t(x > t(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j))$ thereby follows the student distribution whose cumulative distribution function can be calculated as

$$P_t(x \leq t(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)) = \int_{-\infty}^{t(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)} \frac{\Gamma\left(\frac{v+1}{2}\right)}{\Gamma\left(\frac{v}{2}\right)} \frac{1}{\sqrt{v\pi}} \frac{1}{\left(1 + \frac{t^2}{v}\right)^{(v+1)/2}} dt \quad (32)$$

with $v = N - 1$ degrees of freedom and $\Gamma(v) = \int_0^\infty t^{v-1} e^{-t} dt$.

The update of the correlation coefficient between the predictions of two ensemble members (in order to avoid time-intensive re-calculations) can be established by dividing the formula (29) into three parts of variance-like calculations, namely $cov(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j) = \sum_{k=1}^N (\hat{\mathbf{y}}_i(k) - \bar{\hat{\mathbf{y}}}_i)(\hat{\mathbf{y}}_j(k) - \bar{\hat{\mathbf{y}}}_j)$, $\sigma(\hat{\mathbf{y}}_i) = \sqrt{\sum_{k=1}^N (\hat{\mathbf{y}}_i(k) - \bar{\hat{\mathbf{y}}}_i)^2}$ and $\sigma(\hat{\mathbf{y}}_j) = \sqrt{\sum_{k=1}^N (\hat{\mathbf{y}}_j(k) - \bar{\hat{\mathbf{y}}}_j)^2}$, i.e. one co-variance and two standard deviations with the difference not to divide the sums through N . Each part can then be sample-wise updated by recursive variance and covariance formula as provided in Lughofer (2011) (Chapter 2), Qin et al. (2000):

$$\sigma^2(\hat{\mathbf{y}}_i)(N) = \frac{1}{N} ((N-1)\sigma^2(\hat{\mathbf{y}}_i) + N\Delta\bar{\hat{\mathbf{y}}}_i(N)^2 + (\bar{\hat{\mathbf{y}}}_i(N) - \hat{\mathbf{y}}_i(N))^2) \quad (33)$$

with $\hat{\mathbf{y}}_i(N)$ the latest (N th) prediction of the i th member (on the N th sample) and $\Delta\bar{\hat{\mathbf{y}}}_i(N) = \bar{\hat{\mathbf{y}}}_i(N) - \bar{\hat{\mathbf{y}}}_i(N-1)$ the difference between two consecutive means of $\hat{\mathbf{y}}_i$ [which can be straightforwardly updated by the incremental mean (Tschumitschew and Klawonn 2010)]. The latter is used as a rank-one modification for more stable convergence (Qin et al. 2000).

$$cov(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j) = \frac{1}{N} ((N-1)cov(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j) + N\Delta\bar{\hat{\mathbf{y}}}_i(N)\Delta\bar{\hat{\mathbf{y}}}_j(N) + (\bar{\hat{\mathbf{y}}}_i(N) - \hat{\mathbf{y}}_i(N))(\bar{\hat{\mathbf{y}}}_j(N) - \hat{\mathbf{y}}_j(N))) \quad (34)$$

We aim for updating the correlation coefficient between all member pairs (half of $|F_{ens}|$ over 2' in sum) and testing the null-hypothesis according to (30) between all member pairs i, j . Those for which the null-hypothesis is rejected are stored in a specific set of significant correlation pairs:

$$Corr_set = \{(i, j) | i < j \wedge P_t(x > t(\hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j)) < \alpha\} \quad (35)$$

(the condition $i < j$ in (35) is used because the correlation is commutative). Based on this set, pruning can be conducted by sorting the members according to the highest number of significant correlations they have to others and then by pruning successively the correlated variables by respecting which ones have been pruned before or not.

As an example, consider the following correlation set in the case of six members:

$$Corr_set = \{(1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (4, 5), (5, 6)\}$$

Then, member 1 has the highest number of significant correlations with any other members (3 in sum), so members 3, 4 and 5 are pruned and member 1 is kept in the ensemble. Then, as member 2 was not pruned so far and has the second highest number of significant correlations, it is taken in the next pruning round and members 3 and 4 are pruned (i.e., in this case confirmed to be pruned), whereas member 2 is kept in the ensemble. Then, as members 4 and 5 have been already pruned, member 6 is kept in the ensemble $\rightarrow$ leaving members 1, 2 and 6 in the ensemble. Furthermore, we always exclude the two newest members from pruning as for them not enough predictions have been seen so far (avoiding statistical insignificance). This means that at least three members are always present in the ensemble, which abandons a(n undesired) shrinkage of the ensemble to a single model in the case of (longer) steady process phases.

Finally, a pseudo-code of our sequential training procedure is presented in Algorithm 1, working on a single sample-wise basis for prediction (multi-sample ahead) and on a chunk-wise basis for updating MAEs and training of new members; the 'mod' statement in the if-condition denotes the modulo operation.

3 Experimental setup

Algorithm 1 On-line Sequential Ensembling of Fuzzy Systems (OS-FS)

Input: ensemble F_{ens} containing members $F_1, \dots, F_{|F_{ens}|}$, their MAEs $MAE_1, \dots, MAE_{|F_{ens}|}$, new stream sample $\mathbf{x}(N) \in X_{T+1}$ (T chunks seen so far), upper right correlation matrix $\mathbf{R} = [r(\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_j)]$; chunk size k_T (the only variable parameter).

Output: updated F_{ens} and MAEs of single members (if chunk X_{T+1} is ready), predicted value $\hat{y}(N)$.

Prediction Stage:

Case Var1: Perform prediction by (15) on $\mathbf{x}(N) \rightarrow \hat{y}(N)$.

Case Var2: Perform prediction by (17) on $\mathbf{x}(N) \rightarrow \hat{y}(N)$.

Case Var3: Perform prediction by (18) on $\mathbf{x}(N) \rightarrow \hat{y}(N)$.

Case Var4a: Perform prediction by (19) on $\mathbf{x}(N) \rightarrow \hat{y}(N)$ using (20) (or (20)).

Case Var4b: Perform prediction by (19) on $\mathbf{x}(N) \rightarrow \hat{y}(N)$ using (20) and re-assigning $cert_i$ by (25).

Case Var5a: Perform prediction by (19) on $\mathbf{x}(N) \rightarrow \hat{y}(N)$ using (22) multiplied with (20).

Case Var5b: Perform prediction by (19) on $\mathbf{x}(N) \rightarrow \hat{y}(N)$ using (22) multiplied with (20) and re-assigning $cert_i$ by (25).

Update Stage:

if $\text{mod}(N, k_T) \neq 0 \rightarrow$ chunk is not full then

$\hat{y}_{T+1} = \hat{y}_{T+1} \cup \hat{y}(N)$.

$X_{T+1} = X_{T+1} \cup \mathbf{x}(N)$.

for $i = 1$ to $|F_{ens}|$ do

$\hat{\mathbf{x}}_i \neq \hat{\mathbf{y}}_i \hat{y}_i(N)$ (store predictions of single members).

Update $\sigma^2(\hat{\mathbf{x}}_i)$ using (33).

end for

$Corr_set = \{\}$.

for $i = 1$ to $|F_{ens}| - 2$ do

for $j = i + 1$ to $|F_{ens}| - 2$ do

Update correlation coefficient $r(\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_j)$ by updating $cov(\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_j)$ with (34) and then calculating (29) with $\sigma^2(\hat{\mathbf{x}}_i)$ and $\sigma^2(\hat{\mathbf{y}}_j)$ and $cov(\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_j)$ store $r(\hat{\mathbf{x}}_i, \hat{\mathbf{y}}_j)$ into $\mathbf{R}$.

Check whether condition (31) holds with $\alpha = 0.05$; if so, $Corr_set = Corr_set \cup \{(i, j)\}$.

end for

end for

Elicit members to be pruned F_{pruned} according to the $Corr_set$ as described in Section 2.5; $F_{ens} = F_{ens} \setminus F_{pruned}$.

else

Retrieve real target values on latest chunk: $y_{T+1} = \{y(N - k_T + 1), \dots, y(N)\}$.

for $i = 1$ to $|F_{ens}|$ do

Update MAE_i using (21) for all k_T samples in the recent chunk (X_{T+1}, y_{T+1}) .

end for

for $i = 1$ to $|F_{ens}|$ do

Check whether (26) holds using (28) for setting n ; if so, prune F_i , i.e. $F_{ens} = F_{ens} \setminus F_i$, and its MAE MAE_i .

end for

Train a new member F_{T+1} on (X_{T+1}, y_{T+1}) .

$F_{ens} = F_{ens} \cup F_{T+1}$.

$X_{T+1} = \{\}, y_{T+1} = \{\}$.

$T = T + 1$.

end if

$N = N + 1$.

Fig. 3 Error trend lines (accumulated MAEs) for the four prediction variants in the sequential ensembles and for the classical single EFS models, shown in different line markers (and colors); the start and end points of the two drifts (abrupt and gradual cyclic) are marked by vertical markers; left: for all original prediction variants and single

EFS model, right: for the two certainty-based weighted variants (variants 4a.) and (5a.) in (Algorithm 1) in comparison with integrating the contradiction resolving concept (CR) into them (variants 4b.) and (5b.) in (Algorithm 1)

3.1 Streaming data sets

Artificial data set with (known) drifts we used the multi-variate data set from the paper of Friedman (1991), which has been also applied before for drift analysis [see, e.g., Ikonovska et al. (2009)] and which is based on the functional trend given by:

$$y = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + x_5 + N(0, 1) \quad (36)$$

with $N(0, 1)$ the standard normal distribution. We generated 60,000 samples from it by randomly drawing samples from the $[0, 1]^5$ -normalized input space and calculated the y value with $N(0, 1)$ -noise according to (36). We included an abrupt (target concept) drift starting at Sample #15,000 and ranging to Sample #25,000 according to the following changed functional trend:

$$y = 10 \sin(\pi x_4 x_5) + 20(x_2 - 0.5)^2 + 10x_1 + x_3 + N(0, 1) \quad (37)$$

We further included a gradual (target concept) drift starting at Sample # 35,000 and ranging to Sample #45,000, where we sampled again from concept (37), but this time with permanently increasing probability over windows with a length of 100 samples, starting with probability 0.01 (thus, in the mean expectation, one sample from the first window ranging from 35000 to 35100 comes from the function in (37), and the remaining 99 from that one in (36)) and ending with probability 1 (thus, increasing the probability over each window by 0.01). Please note that, as the functional

trend according to (37) is revisited, the gradual drift is a *cyclic* one.

Real-world application—engine test benches another data stream we are dealing with is recorded from 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 the pressure of the oil, various temperatures or emission gases (NOx, CO₂ etc.) (Lughofer et al. 2011). Online data have been recorded containing 22,302 measurements in total and 42 channels, which could be reduced to 9 most important ones for the purpose of NOx modelling, according to expert-based and data-driven pre-selection, see Lughofer et al. (2011). The main task is thus to build an accurate prediction model for NOx emission online in an adaptive manner, as time-intensive (batch) re-training should be prevented when new profiles arise. Due to the varying profiles, input variable shifts in form of range extensions and distributions changes appear. Towards the end of the stream (around Sample # 15000) two completely new test campaigns arise in the data stream. In this sense, this data set served as interesting benchmark for our OS-FS algorithm, especially in comparison with single EFS model.

Several data streams from various real-world applications furthermore, we studied the performance of our OS-FS method with related SoA works based on several data streams for regression problems randomly selected from

the UCI repository—please refer to Sect. 4.3 and Table 2 for an overview of the characteristics of the data sets.

3.2 Evaluation strategy

The evaluation is conducted by performing a direct one-to-one comparison between on-line sequential training of the fuzzy systems (OS-FS) ensemble and single EFS models. The latter are trained by the learning engine Gen-Smart-EFS (Lughofer et al. 2015) for incrementally updating single fuzzy systems in a recursive and evolving procedure. This engine has been also used in our sequential training procedure for establishing the rules and their antecedents, see Sect. 2.1, such that a fair direct comparison between the performance of our ensembles and of the native single EFS models could be obtained. Furthermore, we used online bagged EFS (OB-EFS) (Lughofer et al. 2021) as related EFS ensemble technique for performance comparison, as well as two AI methods based on extreme learning machines and MLP neural networks.

In our data-driver, we loaded chunks of data sequentially into the memory. For each incoming chunk, first the predictions on *all* samples contained in this chunk are made, which are then compared with the real measured target values (available in all our supervised data streams described in the previous section). Then, the MAEs of all members are updated according to (21), error-based pruning conducted and a new member is trained based on this recent chunk and added to the ensemble—as stated in Algorithm 1. The same data-driver (embedding the same data chunk processing, the same error trends accumulations etc.) has been used for single EFS models and related SoA works to achieve a fair comparison.

In the results section, we show the error trend lines over time as achieved by single EFS models, related SoA methods and by our new OS-FS scheme in the same plots—this serves for the purpose of a direct comparison whether our new approach brings some improvement, i.e. whether it can show a better decreasing error behavior or lower error trends in major parts of the streams. For the data streams from the UCI repository, we show a summary measure of the error trends termed as AUC (area under the error curve). Computational aspects such as the time needed for processing a single sample (through prediction + model training/update) will also play an important role in order to realize on-line and real-time capabilities of the various methods. We used a default chunk size of 500 samples, but also varied the chunk size in the artificial and engine test bench data streams to see the sensitivity in this parameter. We also used the pruning options discussed in Sects. 2.4 and 2.5 to realize their effect (i) onto the predictive performance, (ii) on the ensemble complexity and (iii) onto the computation speed. In all

experiments, we used as default setting for the significance level that two predictions from two members are correlated a value of 0.05 and a fully adaptive n for error-based pruning according to (28).

4 Results

4.1 Artificial data set with (known) drifts

The left plot in Fig. 3 shows the error trend lines (in terms of accumulated MAEs) for the four prediction variants (as discussed in Sect. 2.2) in comparison with the single EFS model (SoA) over time in different lines styles and colors, whereas the vertical markers denote the beginning and end points of the included drifts as indicated in the text boxes.

It is easy to realize that the single EFS model and the predictions schemes based on native averaging of ensemble members' outputs and also weighted averaging using coverage degree-based weights perform very similarly over the whole stream: showing a significant sudden increase in the case of the abrupt drift and a more moderate one when the gradual cyclic drift starts (whereas showing robust and converging (steady and decreasing MAE) behavior in-between and towards the end of the stream). The similarity between native averaging and weighted averaging with coverage degree-based weights lies at hand, as no input space drift happens in the stream, especially the input space is well covered with samples from the beginning on (first couple of chunks), as the stream samples have been uniformly generated based on $[0, 1]$ equal distribution. Hence, the coverage degrees of all ensemble members for samples from each new chunk can always be expected as similar → weights are similar as well in the weighted average which thus approximates the standard average. The similarity of results between averaging ensemble members and single EFS model is most possibly because within the (strong) single EFS model (with many rules evolved, especially during the drift phases) the output of the rules are averaged (with weights) and in the ensemble the outputs of the members (with a low number of rules each, 2–3 here) are averaged. Finally, this means that in the context of fuzzy systems, as containing partial local predictors, the standard averaging of weak models' outputs (containing a few rules each) typically may turn out to become similar to the averaging of outputs' from a higher number of rules as embedded in a single model. Thus, the standard averaging over sequential ensembles as suggested in Al-Mahasneh et al. (2019) (there for neural networks as base members) cannot really contribute to a performance improvement over single models in the case of fuzzy systems.

Fig. 4 Left: error trend lines (accumulated MAEs) for the MAE-based weighted prediction, when switching on/off intensive and moderate member pruning, shown in different line markers (and colors); the start and end points of the two drifts (abrupt and gradual)

It is interesting to see that the weighted predictions based on recent MAE trends of the ensemble members could (i) significantly reduce the overall accumulated MAE during the abrupt drift, (ii) result in a decreasing error trend during the gradual cyclic drift much earlier (already after the half of the drift phase) and (iii) produce lower error trends during the stable phases (Samples 0–15,000 and 45,000–60,000) than the aforementioned methods. The reason for the first improvement is due to the dynamic out-weighting effect of older members through (22) used in (19) as producing significantly higher MAEs on new drifting phases than newer ones (already trained during the drift phases): thus, only the newer ones actually fire significantly in the weighted over-all predictions. This cannot be achieved with standard unweighted averaging, neither with a single EFS model,

are marked by vertical markers; right: the evolution of the ensemble members for the two pruning cases, compared with no pruning; upper: without contradiction resolving, lower: with contradiction resolving and diversity-based pruning variants

as therein the rules always blindly fire in a similar manner over all chunks, because the input space is covered in a similar manner from the beginning ($[0, 1]^5$ -uniformly distributed samples) and the drift happens only in the target concept (change of the functional mapping, see Sect. 3.1). The second improvement is basically because older members already trained on the drifted state before (which is revisited in a gradual manner from sample 35,000 on to form a cyclic drift) are more actively firing in the overall predictions, as covering the new samples better than newer ones. The lower error trends during the stable phases are simply because members, which turn out to become undesired on new chunks (because they were, e.g., trained on 'unlucky' chunks containing more noise than others or more weakly distributed input samples), induce higher MAEs and are thus

Fig. 5 Error trend lines (accumulated MAEs) in the case of MAE-based weighted predictions when using different chunk sizes to train the sequential ensemble members (= sensitivity analysis on the (only)

parameter chunk size); left: using conventional pruning and no CR, right: using CR plus diversity-based pruning (best variant)

Fig. 6 Error trend lines (accumulated MAEs) in the case of engine test benches for the four prediction variants in the sequential ensembles and for the classical single EFS models, shown in different line markers (and colors); left: for all original prediction variants and the

single EFS model, right: for the two certainty-based weighted variants (variants 4a.) and (5a.) in (Algorithm 1) in comparison with integrating the contradiction resolving concept (CR) into them (variants 4b.) and (5b.) in (Algorithm 1)

down-weighted according to (22)—a strong robust and fully autonomous aspect of the sequential ensemble. This cannot be tackled easily with single EFS models, as all rules evolved in the past are always equally used for predicting new samples.

A similar performance is achieved for the timely-based weighting approach with exponential forgetting over past ensemble members, which can even further decrease the MAE trends of MAE-based weighted predictions at the start of the abrupt drift phase. This is the case because out-weighting older trained members takes permanently place without requiring an increase of the MAE trends of the older

members (which typically takes some chunks to be actually reflected). During the stable phases, however, it performs a bit weaker than MAE-based weighted predictions as undesired members (showing higher errors than others) are not necessarily down-weighted (as this depends on the time when they were generated).

Moreover, the right plot in Fig. 3 compares the two uncertainty-based weighted variants in native form as also shown in the left plot (variants 4a.) and (5a.) in (Algorithm 1) with their extended forms (variants 4b.) and (5b.) in (Algorithm 1) by integrating the contradiction resolving concept discussed in Sect. 2.3. Obviously, in the case of variant 5b.)

Fig. 7 Left: error trend lines (accumulated MAEs) for the MAE-based weighted prediction, when switching on/off intensive and moderate member pruning, shown in different line markers (and colors); the start and end points of the two drifts (abrupt and gradual)

are marked by vertical markers; right: the evolution of the ensemble members for the two pruning cases, compared with no pruning; upper: without contradiction resolving, lower: with contradiction resolving and diversity-based pruning variants

Table 1 Computational performance of the various sequential ensembling variants as well as of the single EFS model regarding the processing of single samples (through prediction and adaptation) in average (unit: milliseconds)

Methods	Engine Test Bench	Artificial Data Set
Single EFS model	0.59	0.42
Conventional Averaging	0.076	0.55
Weighted Av. with Coverage	0.094	0.56
Weighted Av. with Coverage + CR	0.143	1.0
Weighted Av. with Timely Weights	0.076	0.54
Weighted Av. with MAE	0.089	0.59
Weighted Av. with MAE + CR	0.143	1.1
Weighted Av. with MAE + CR + Moderate Pruning	0.098	0.86
Weighted Av. with MAE + CR + Intensive Pruning	0.049	0.21
Weighted Av. with MAE + CR + Corr-Based Pruning	0.044	0.062

(MAE-based uncertainty weights), the resolution of contradictions can improve the error trend lines over the whole stream (during non-drift and drift phases), please compare dark blue with light blue trend lines, whereas in the case of (variant 4b.) contradiction resolving performs similar during the drift phases and the first and last stable phase, but weaker in the case of the back-normal phase (cyclic drift) between Sample 15,000 and 25,000.

Furthermore, we inspected the impact of the pruning effect of ensemble members on the accuracy trends as well as computation times needed for processing a single sample through update and prediction. The error trend lines are shown in the left plot in Fig. 4, whereas in the right plot the number of ensemble members are drawn (both, in comparison with no pruning).

It is easy to realize that intensive pruning [using $n = 1$ in (26)] can even lower the error trends for almost the whole stream. Compared to a much lower number of ensembles (see right plot), leading to significant lower sample processing times by around two thirds (see Table 1 below), this is a remarkable issue and we can conclude that it is really pruned what is not needed or even counterproductive for the final output predictions. It is also interesting to see that during the stable, non-drift phases (0–15,000 and 45,000–60,000) the ensemble grows, whereas during the drift phases the ensemble shrinks. This effect can be simply explained due to the statistical process control for finding members with atypically high MAE trends, which get more apparent during drift phases because a lot of older members do not fit the new drifted distributions and thus produce atypically high prediction errors on the new samples; while during stable phases a lot of older members still produce reliable predictions (thus they do not exceed control limits for high MAEs). However, during stable phases, a lot of redundancies among the prediction behavior of members can be expected, which provided us the motivation to also include the correlation-based diversity measure as proposed in Sect. 2.5. The results on this stream when applying this measure are shown in the lower plots of Fig. 4, here in combination with the MAE-based uncertainty weights together with contradiction resolving (as turning out to be the best variant, compare with Fig. 3). It is easy to realize that the error trend line is not much affected (compare grey dotted plot including diversity-based pruning with the others), while leaving only a few members in the ensemble, actually needed for diversity: during the stable phase from 0 to 15,000, three members are constantly in the ensemble, thus newly generated members during this phase (on incoming new chunks) are immediately back-pruned as they are correctly recognized as non-diverse to older members (they are obviously non-diverse because the data distribution does not change, so each chunk of 500 samples contains the same concept to be learned). During the abrupt drift, an additional member is

obviously needed (and again sufficient to describe the new drifted phase whose concept does not change further), which is the case only for a shorter time period during the gradual, cyclic drift phase.

Finally, we studied the effect of the chunk-size (the only parameter in our method and which may be sometimes provided by the user due to the known delay for retrieving target values within the application), used for training the single ensemble members, on the error trend lines. This result is shown in Fig. 5, where the left plot visualizes the case when not applying CR and conventional, error-based pruning and the right plot the case when also applying CR plus diversity-based pruning (best variant, compare with Fig. 3).

Obviously the effect between a chunk size of 250, 500 or even 1000 is not that big during the stable and the gradual drift phases; only in the case of the abrupt drift, the error rises significantly higher with larger chunk sizes, due to the delay of drift compensation: more samples need to be collected before a new member is trained during the drift phase.

4.2 Engine test benches

The left plot in Fig. 6 shows the error trend lines for the four prediction variants (as discussed in Sect. 2.2) over time in different lines styles and colors.

It is easy to realize that here standard averaging and the single EFS model perform worse than the more advanced weighted prediction variants (especially at the beginning of the stream). This is basically because input variable shifts in form of range extensions and distributions changes appear pretty regularly, and these are even more intense at the beginning of the stream (as latter a 'saturation' of covering the input space takes place). Performing an average among past sequential members does not take into account this situation at all, as the outputs from all past members are seen as equally important—thus, although all rules from older members may fire very weakly for new (drifted) chunks, they are reflected fully in the final prediction, due to the normalized (= relative) inference conducted within each member, see (16). Single EFS model can improve this situation, because the outputs produced are weighted with normalized rule fulfillment degrees across **all** rules evolved so far across all chunks (and for shifted situations older rules typically have a lower fulfillment degree than newer ones, thus are outweighed in comparison to the newer ones within the normalized inference)—please note that this improved situation could not be achieved for the target concept drifts in the previous stream results, because there no input shifts took place, thus the unsupervised rule fulfillment degrees in older and newer members turned out to be unaffected by the drift.

Furthermore, it is interesting to see that in this case the integration of coverage degrees as prediction weights can

Table 2 Characteristics of the data streams used for comparison with related works

Dataset	# of samples	# of features	Comment
Airfoil	1503	5	Airfoils at various wind tunnel speeds and angles of attack
Appliances Energy	19,735	25	Appliances energy prediction in buildings
Facebook Post Count	40,949	53	Predicting how many comments facebook posts will receive.
Metro Traffic	48,204	6	Traffic volume prediction for westbound I-94.
Naval Plants	11,934	16	Condition-based maintenance of naval propulsion plants.
Protein Properties	45730	9	Predicting size of the residue of protein tertiary structure.
Seoul Bike	8760	13	Predicting the demand of bikes for rental.
Sgemm Product	241,600	17	Predicting the computation time for large-scale matrix multiplication.
SML2010	4137	23	Indoor temperature prediction due to outdoor weather characteristics.
Worker's Productivity	1197	14	Productivity prediction of garment employees.

Table 3 Performance comparison of new OS-FS variant with online-bagged EFS (OB-EFS), single EFS model and ML models (OS-ELM, MLPs)

Data sets	Single EFS		OB-EFS		OS-FS (new)		MLPs		OS-ELM	
	AUC	Time	AUC	Time/#	AUC	Time/#	AUC	Time	AUC	Time
Airfoil	248	5.8	150	135/36	217	0.9/6	342	2.83	269	0.9
Appliances Energy	9066	19	1476	2442/20	2642	15/26	2187	68	2272	10
Facebook Post Count	1541	126	1024	1329/20	953	82/25	1523	120	1549	10
Metro Traffic	11709	50	11858	2898/15	11641	130/37	13363	101	11891	25
Naval Plants (Maintenance)	2472	82	1005	881/10	1572	6/13	1587	67	3018	7
Protein Properties	9523	197	9284	2323	9212	118/29	10102	115	11816	19
Seoul Bike	779	11	657	446/21	763	5/23	878	20	943	5.5
Sgemm Product	5415	705	4039	5196/15	4069	259/60	4388	127	5153	79
SML2010	383	18	263	261/28	310	1.5/11	462	4.6	552	2
Worker's Productivity	86	2	101	48/10	85	0.6/5	96	2	119	1

The values in bold font denote the lowest AUCs and computation times among the methods for each data set

significantly improve the standard average, because input shifts affect the coverage degrees significantly (as stream samples drift away from rules representing older distributions)—so, only the best covering members are impacting the final prediction. Timely based weights with exponential forgetting achieve a similar error trend as coverage degree (and a little bit more rising towards the end of the stream), although this required the tuning of the forgetting factor in (18), whereas the coverage degree does not require any parameters. MAE weights combined with coverage weights can further decrease the error trend line a bit, so the MAEs of members over recent chunks again play a role for improving the performance, although this role is less significant than in the case when a target concept drift happens (see previous subsection). This is intuitively clear as in the latter case a supervised criterion is needed to properly handle the drift, whereas in the case of input space drifts an unsupervised criterion can already handle a drift sufficiently well. Towards the end of the stream (beginning from around Sample # 15,000), where two new test campaigns appear, all

methods perform pretty stable, i.e. they can expand well to these new situations.

The right plot in Fig. 6 shows how the integration of contradiction resolving concepts as discussed in Sect. 2.3 affects the error trend lines: this is done for the MAE-based and coverage-based uncertainty weights and the trend lines indicate that there is hardly any difference to the case when not applying CR at all.

Furthermore, we inspected the impact of the pruning effect of ensemble members on the accuracy trends as well as computation times needed for processing a single sample through update and prediction. The error trend lines are shown in the left plot in Fig. 7, whereas in the right plot the number of ensemble members are drawn (both, in comparison with no pruning).

It is easy to realize that both pruning options perform nearly the same as when not using any pruning, where intensive pruning can greatly reduce the ensemble members and thus the performance speed. This is the case for

Fig. 8 Accumulated error trend lines (in terms of percentual MAEs) for SML2010 and Metro Traffic data sets for four methods under comparison

both variants, MAE-based uncertainty weights without CR (upper row) and with CR (lower row), leading to very similar error trend lines performance. However, the difference in pruning behavior becomes apparent when looking at the right plots: for the sole atypical error based pruning case (w/o CR), new operating conditions (as induced by the new test campaigns towards the end of the stream) induce a high pruning rate, because a lot of older members trained on older test campaigns before are not able to predict well the new campaigns and thus induce atypical high MAEs; for the diversity-based pruning variant, which correctly keeps the number of members low during the first, more stable phase of the stream, the new test campaigns trigger an increase of members, because the new ones trained on the new test campaigns are diverse to the older ones. This should make more clear the different behaviors and thus pros and cons of the two pruning variants — the latter can be assumed to be more robust subject to cyclic drifts, the former more flexible during new arising system states.

Finally, the performance speed is listed in Table 1 for the various sequential ensembling approaches as well as for the single EFS model in terms of seconds needed to process a single sample through prediction + adaptation in average, whereas 'CR' means that contradiction resolving for output prediction was conducted (as discussed in Sect. 2.3).

From this table, it can be realized that for the artificial data set intensive pruning leads to a reduction of about four fifth of the original time used when switching contradiction resolving on (CR) (best variant leading to lowest error trends), while the addition of diversity-based pruning even further decreases the time by another two thirds down to 0.062 ms for processing a single sample in average (through

prediction and adaptation). This is a remarkable speed-up, whereas the slowest variants (MAE-based weights with CR and no pruning) already lies in the low range of around 1 ms processing time. Furthermore, intensive and diversity-based pruning leads to an even significantly faster ensemble model than a single EFS model: this is basically because the single EFS model becomes heavier and heavier over time due to the evolution of new rules etc., whereas in the ensemble all members are typically weak and if these are only a few (due to pruning in steady phases etc.), it is clear that they end up in a faster stream processing. For the engine test bench data set, the situation is similar as intensive and diversity-based pruning can reduce the computation time by about two thirds compared to the case when no pruning is switched on, and leading to about a 10 times faster processing of samples than the single EFS model.

4.3 Further comparison on various real-world data sets

In order to achieve a wider comparison of our new approach with related works on several real-world data sets, we investigated the usage of a large-span EFS ensembling method, which has been recently published in Lughofer et al. (2021). It is termed as on-line bagged evolving fuzzy systems (OB-EFS), operating on a set of base models which are all incrementally evolved and updated with new incoming samples in parallel. So, all the base models successively expand their knowledge and no weak and flexible members on single chunks (as in our case) are generated. This may be at the expense of computation times (as can be seen below), but may increase robustness of the overall predictions, as

inheriting the robustness aspects of conventional bagging due to a specific sampling scheme. We will thus report computation times versus model accuracy to check over the tradeoff. Furthermore, we performed a comparison again with single EFS model and with two different modeling techniques from the field of artificial intelligence: OS-ELM (Lan et al. 2009), which is an on-line sequential learning method for extreme learning machines (Sun et al. 2013) (updating parameters and neuronal structures on the fly in single-pass manner), and MLPs (multi-layer perceptron neural networks) (Haykin 1999) with up to 5 layers being trained on single chunks of data. For all methods, the essential parameter(s) are tuned on the first data chunk (of 500 samples) based on a grid search scenario coupled with cross-validation (Stone 1974)—the parameters leading to lowest CV-error are selected and used for the on-line update process on the further stream. This somehow mimics an initial off-line modeling step for parameter tuning, which is very fast in our case due to the low number of initial samples.

The data sets under study have been randomly selected with an emphasis on application diversity from the UCI repository¹ (for regression problems—please see the description there for further details) and are listed in Table 2 indicating the number of samples and features as well as a short comment about the application task behind.

All the data streams have been used in its raw format appearance without performing any pre-processing or dimensionality reduction.

Table 3 compares the area under the accumulated error curve (AUC) for the various methods on all data set (the lower the better). This measure is simply the sum over all accumulated mean absolute errors (MAEs) of the overall model/ensemble, as being calculated by (21) with $M = N$ and without any subscript to achieve a global MAE for the whole model/ensemble. Furthermore, the computation times are provided in seconds needed for the whole stream mining process until termination.

It can be clearly recognized that our new approach OS-FS can outperform single EFS models as well as machine learning techniques OS-ELM and MLPs in terms of AUC for all data streams (showing lower AUC values)—the only exception is the appliances energy data, showing values of related ML techniques in cursive font, where OS-FS produces around 20% higher accumulated error curves. In most of the cases, the out-performance is more than 10% (i.e. a 10% lower accumulated error trend in average), especially compared to the faster single-pass OS-ELM approach, which is remarkable. Compared to OB-EFS, only in four out of ten cases an out-performance in terms of AUC could be achieved, whereas in two cases the AUC is nearly the

same. This confirms the improved robustness of OB-EFS ensembles as being claimed in Lughofer et al. (2021) (with respect to single EFS models), especially compared to single EFS models in the case of shorter streams. On the other hand, OB-EFS requires around 100 times more computation time than OS-FS which is extremely fast, processing single samples with a speed way below milli-seconds, and which can compete with OS-ELM (5 times slightly better, 5 times slightly worse). The slower performance of OB-EFS is due to large-scale ensembles, where mostly around 20 strong base models are processed and permanently updated and its knowledge expanded. This means, OB-EFS suffers from the same issue as single EFS models, i.e. to become heavier and heavier over time within the base model components with more and more knowledge included (although members are automatically pruned when not needed). Single EFS models can close this gap achieving computation times significantly below those of OB-EFS, but still slower ones than OS-FS in all cases, while not producing any better AUCs. Thus, in the case of larger streams, OS-FS is the only reliable fuzzy systems training/update option—and compared to OS-ELM and MLPs the computation time is similar, but with being the winner in terms of AUC in nine out of ten times.

Figure 8 shows the accumulated error trends in comparison among the methods for two examples which are potentially including a drift, the left one starting at around Sample 600 and the right one starting at around Sample 6000.

While OS-ELM suffers most from the drift as showing a significant increase in both cases, leading to a doubling of the error trend line in the left case, single EFS model performs a bit better (but still with an increase in error) and both ensembling variants are pretty robust against the drift: in the left case, the increase in error is low for both variants and even lower for OS-FS than for OB-EFS (as coming from a higher error before), in the right case no increase in the error is visible, thus both are not affected at all by the drift. Indeed, OB-EFS can outperform OS-FS a bit regarding slightly lower error trend lines (especially towards the end of the streams), but, as discussed above, at the expense of around 100 times slower computation times.

5 Conclusion

We proposed a new on-line sequential ensembling scheme for fuzzy systems with chunk-wise training of single (smaller) ensemble members (OS-FS) in order to cope with abrupt and gradual drifts in a more flexible manner, decreasing prediction error trends over data chunks. We could verify this by successfully evaluating our approach on two data sets, one including regular input space expansions (shifts), the other including target concept drifts in abrupt and

¹ <https://archive.ics.uci.edu/ml/index.php>.

gradual form, and this even with a cyclic re-occurrence of an older concept. The results showed significantly improved performance (lower error trends) over classical single EFS models in both cases, and this by even lower on-line computation times for processing stream samples, especially by significantly lower ones when pruning options are used. Furthermore, the concept for resolving contradictions could further boost the accuracy of the models. Thus, our approach can be seen as a promising extension of current EFS modeling approaches [as widely discussed in the recent survey in Skrjanc et al. (2019)]. Furthermore, our approach is able to deal with delayed target values in a natural and autonomous manner (with the chunk-size for ahead prediction depending on the delay of the target measurements), whereas most of the current single EFS models rely on one-step-ahead prediction with immediate updating, afterwards [at least, they have been mostly empirically evaluated in this way (Skrjanc et al. 2019)].

Further results on several data streams from a wider range of applications scenarios showed significantly improved performance in terms of lower accumulated error trends compared to single EFS models and related AI modeling techniques (NNs in form of ELMs and MLPs), while being faster than single EFS models and approximately as fast as related AI modeling techniques. Large-scale EFS ensembles (OB-EFS), which permanently update base models with new incoming samples, could indeed improve the robustness of the models due to a specific sampling scheme [as claimed in Lughofer et al. (2021)], but produced around 100 times higher computation times and are thus only reliably applicable for smaller streams—especially for these ones the robustness improved most. A reason for the low computation times way below requiring milli-seconds for processing single samples are the particular pruning options within OS-FS which lead to slim ensembles which finally contained members actually really needed for performing reliable predictions with diversity among the members—in this sense, unnecessary replications of models in the case of stationary phases as well as undesired models with unpleasant error trends could be omitted. A further reason is the re-training of fresh weak members (small fuzzy systems) based on single chunks just containing a few hundred samples, rather than updating heavy models containing all the different information in a non-stationary stream (and thus embedding a higher number of rules). All in all, OS-FS can be seen as a flexible and high-speed fuzzy systems training alternative from regression streams with solid performance.

Acknowledgements The first author acknowledges the support by the “LCM—K2 Center for Symbiotic Mechatronics” within the framework of the Austrian COMET-K2 program.

Funding Open access funding provided by Johannes Kepler University Linz.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Al-Mahasneh AJ, Anavatti S, Garratt M (2019) Evolving general regression neural networks for learning from noisy datasets. In: Proceedings of the 2019 IEEE Symposium Series on computational intelligence, Xiamen, China, pp 1473–1478
- Ando T (2010) Bayesian model selection and statistical modeling. CRC Press, Boca Raton
- Angelov P (2011) Autonomous machine learning (alma): generating rules from data streams. In: Special International Conference on complex systems. 2011, pp 249–256
- Angelov P (2012) Autonomous learning systems: from data streams to knowledge in real-time. Wiley, New York
- Angelov P, Gu X (2017) Autonomous learning multi-model classifier of 0 order (almmo-0). In: Proceedings of the evolving and intelligent systems (EAIS) conference 2017. IEEE Press, Ljubljana, Slovenia, pp 1–7
- Angelov P, Lughofer E, Zhou X (2008) Evolving fuzzy classifiers using different model architectures. *Fuzzy Sets Syst* 159(23):3160–3182
- Angelov P, Filev D, Kasabov N (2010) Evolving intelligent systems—methodology and applications. Wiley, New York
- Ashfahani A, Pratama M, Lughofer E, Ong YS (2020) DEVDAN: deep evolving denoising autoencoder. *Neurocomputing* 390:297–314
- Babuska R (1998) Fuzzy modeling for control. Kluwer Academic Publishers, Norwell
- Bauer F, Lukas M (2011) Comparing parameter choice methods for regularization of ill-posed problems. *Math Comput Simul* 81(9):1795–1841
- Brazdil P, Giraud-Carrier C, Soares C, Vilalta R (2009) Metalearning. Springer, Berlin
- Cernuda C, Lughofer E, Mayr G, Röder T, Hintenaus P, Märzinger W, Kasberger J (2014) Incremental and decremental active learning for optimized self-adaptive calibration in viscose production. *Chemom Intell Lab Syst* 138:14–29
- Chiu S (1994) Fuzzy model identification based on cluster estimation. *J Intell Fuzzy Syst* 2(3):267–278
- Collins M, Schapire R, Singer Y (2002) Logistic regression, adaboost and bregman distances. *Mach Learn* 48(1–3):253–285
- Das A, Pratama M, Zhang J, Ong J (2020) A skip-connected evolving recurrent neural network for data stream classification under label latency scenario. In: Proceedings of the AAAI Conference on artificial intelligence 34(04), pp 3717–3724
- Davila C (1994) An efficient recursive total least squares algorithm for FIR adaptive filtering. *IEEE Trans Signal Process* 42(2):268–280

- Ding J, Wang H, Li C, Chai T, Wang J (2017) An online learning neural network ensembles with random weights for regression of sequential data stream. *Soft Comput* 21:5919–5939
- Dovzan D, Logar V, Skrjanc I (2015) Implementation of an evolving fuzzy model (eFuMo) in a monitoring system for a waste-water treatment process. *IEEE Trans Fuzzy Syst* 23(5):1761–1776
- Efron B, Tibshirani R (1993) An introduction to the bootstrap. Chapman and Hall/CRC, Boca Raton
- French RM (1999) Catastrophic forgetting in connectionist networks. *Trends Cogn Sci* 3(4):128–135
- Friedman J (1991) Multivariate adaptive regression splines. *Ann Stat* 19(1):1–67
- Gama J (2010) Knowledge discovery from data streams. Chapman & Hall/CRC, Boca Raton
- Harrel F (2001) Regression modeling strategies. Springer, New York
- Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining inference and prediction, 2nd edn. Springer, New York
- Hastie T, Tibshirani R, Friedman J (2010) Regularized paths for generalized linear models via coordinate descent. *J Stat Softw* 33(1):1–22
- Haykin S (1999) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall Inc., Upper Saddle River
- Iglesias J, Sesmero M, Lopez E, Ledezma A, Sanchis A (2020) A novel evolving ensemble approach based on bootstrapping and dynamic selection of diverse learners. In: Proceedings of the Evolving and Adaptive Intelligent Systems Conference (EAIS) 2020. Bari, Italy, pp 1–8
- Ikonomovska E, Gama J, Sebastiao R, Gjorgjevik D (2009) Regression trees from data streams with drift detection. In: Gama J, Costa VS, Jorge AM, Brazdil PB (eds) Lecture Notes in Computer Science (LNCS vol 5808). Discovery science. Springer, Berlin, pp 121–135
- Jang JS (1993) ANFIS: adaptive-network-based fuzzy inference systems. *IEEE Trans Syst Man Cybern* 23(3):665–685
- Khamassi I, Sayed-Mouchaweh M, Hammami M, Ghedira K (2017) Discussion and review on evolving data streams and concept drift adapting. *Evol Syst* 9(1):1–23
- 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
- Kuncheva L (2004) Combining pattern classifiers: methods and algorithms. Wiley-Interscience (John Wiley & Sons), Chichester
- Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Mach Learn* 51:181–207
- Lan Y, Soh Y, Huang GB (2009) Ensemble of online sequential extreme learning machine. *Neurocomputing* 72(13–15):3391–3395
- Leite D, Skrjanc I (2019) Ensemble of evolving optimal granular experts, OWA aggregation, and time series prediction. *Inf Sci* 504:95–112
- Lemos A, Caminhas W, Gomide F (2011) Multivariable Gaussian evolving fuzzy modeling system. *IEEE Trans Fuzzy Syst* 19(1):91–104
- Lughofer E (2011) Evolving fuzzy systems—methodologies advanced concepts and applications. Springer, Berlin
- Lughofer E (2016) Evolving fuzzy systems—fundamentals, reliability, interpretability and useability. In: Angelov P (ed) Handbook of computational intelligence. World Scientific, New York, pp 67–135
- 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, Kindermann S (2010) SparseFIS: data-driven learning of fuzzy systems with sparsity constraints. *IEEE Trans Fuzzy Syst* 18(2):396–411
- Lughofer E, Pratama M (2018) On-line active learning in data stream regression using uncertainty sampling based on evolving generalized fuzzy models. *IEEE Trans Fuzzy Syst* 26(1):292–309
- Lughofer E, Sayed-Mouchaweh M (2019) Predictive maintenance in dynamic systems—advanced methods decision support tools and real-world applications. Springer, New York
- Lughofer E, Macian V, Guardiola C, Klement E (2011) Identifying static and dynamic prediction models for NOx emissions with evolving fuzzy systems. *Appl Soft Comput* 11(2):2487–2500
- Lughofer E, Cernuda C, Kindermann S, Pratama M (2015) Generalized smart evolving fuzzy systems. *Evol Syst* 6(4):269–292
- Lughofer E, Pratama M, Skrjanc I (2018) Incremental rule splitting in generalized evolving fuzzy systems for autonomous drift compensation. *IEEE Trans Fuzzy Syst* 26(4):1854–1865
- Lughofer E, Pratama M, Skrjanc I (2021) Online bagging of evolving fuzzy systems. *Inf Sci* 570:16–33
- Markovsky I, Huffel S (2007) Overview of total least-squares methods. *Sig Process* 87(10):2283–2302
- Marrs G, Black M, Hickey R (2012) The use of time stamps in handling latency and concept drift in online learning. *Evol Syst* 3(2):203–220
- Nelles O (2001) Nonlinear system identification. Springer, Berlin
- Pedrycz W, Gomide F (2007) Fuzzy systems engineering: toward human-centric computing. Wiley, Hoboken
- Polikar R (2006) Ensemble based systems in decision making. *IEEE Circ Syst Mag* 6(3):21–45
- Polikar R, Upda L, Upda S, Honavar V (2001) Learn++: an incremental learning algorithm for supervised neural networks. *IEEE Trans Syst Man Cybern Part C Appl Rev* 31(4):497–508
- Pratama M, Anavatti S, Er M, Lughofer E (2015) pClass: an effective classifier for streaming examples. *IEEE Trans Fuzzy Syst* 23(2):369–386
- Pratama M, Lu J, Lughofer E, Zhang G, Er M (2017) Incremental learning of concept drift using evolving type-2 recurrent fuzzy neural network. *IEEE Trans Fuzzy Syst* 25(5):1175–1192
- Pratama M, Pedrycz W, Lughofer E (2018) Evolving ensemble fuzzy classifier. *IEEE Trans Fuzzy Syst* 26(5):2552–2567
- Qin S, Li W, Yue H (2000) Recursive PCA for adaptive process monitoring. *J Process Control* 10(5):471–486
- Rayana S, Zhong W, Akoglu L (2016) Sequential ensemble learning for outlier detection: a bias-variance perspective. In: Proceedings of the 2016 IEEE 16th international conference on data mining (ICDM). Barcelona, Spain, pp 1167–1172
- Sannen D, Lughofer E, Brussel HV (2010) Towards incremental classifier fusion. *Intel Data Anal* 14(1):3–30
- Seni G, Elder J (2010) Ensemble methods in data mining: improving accuracy through combining predictions. Morgan and Claypool Publishers, Chicago
- Shaker A, Lughofer E (2014) Self-adaptive and local strategies for a smooth treatment of drifts in data streams. *Evol Syst* 5(4):239–257
- Sidhu P, Bathia M (2015) An online ensembles approach for handling concept drift in data streams: diversified online ensembles detection. *Int J Mach Learn Cybern* 6(6):883–909
- Siler W, Buckley J (2005) Fuzzy expert systems and fuzzy reasoning: theory and applications. Wiley, Chichester
- Skrjanc I, Iglesias J, Sanchis A, Lughofer E, Gomide F (2019) Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey. *Inf Sci* 490:344–368
- Stone M (1974) Cross-validatory choice and assessment of statistical predictions. *J R Stat Soc* 36(1):111–147
- Sun F, Toh KA, Romay M, Mao K (2013) Extreme learning machines: algorithms and applications (Adaptation, Learning, and Optimization). Springer, Heidelberg
- 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

- Tibshirani R (1996) Regression shrinkage and selection via the lasso. *J R Stat Soc* 58B(1):267–288
- Tschumitschew K, Klawonn F (2010) Incremental quantile estimation. *Evol Syst* 1:253–264. <https://doi.org/10.1007/s12530-010-9017-7>
- Zdsar A, Dovzan D, Skrjanc I (2014) Self-tuning of 2 DOF control based on evolving fuzzy model. *Appl Soft Comput* 19:403–418
- Zhou Z (2012) Ensemble methods: foundations and algorithms. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, Boca Raton
- Zou H, Hastie T (2005) Regularization and variable selection via the elastic net. *J R Stat Soc Ser B* 67(2):301–320

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Authors and Affiliations

Edwin Lughofer¹ · Mahardhika Pratama²

¹ Institute for Mathematical Methods in Medicine and Data Based Modeling, Johannes Kepler University Linz, Linz, Austria

² School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore