Available online at www.sciencedirect.com

Fuzzy Sets and Systems 163 (2011) 1 – 23

www.elsevier.com/locate/fss

On-line incremental feature weighting in evolving fuzzy classifiers

Edwin Lughofer

Johannes Kepler University Linz, Department of Knowledge-based Mathematical Systems, Altenbergerstrasse 69, A-4040 Linz, Austria

Received 15 August 2009; received in revised form 25 August 2010; accepted 31 August 2010
Available online 15 September 2010

Abstract

In this paper, we present an approach for addressing the problem of dynamic dimension reduction during on-line training, evolution and updating of evolving fuzzy classifiers (EFC). The basic idea of our approach is that, instead of permanently changing the list of most important features with newly loaded data blocks, we generalize the concept of incremental feature selection to an incremental feature weighting approach: features are assigned weights in [0,1] according to their importance level. These weights are permanently updated during on-line mode and guarantee a smooth learning process in the evolving fuzzy classifiers, as they change softly and continuously over time. In some cases, when the weights become (approximately) 0, an automatic switching off of some features and therefore a (soft) dimension reduction is achieved. Two novel incremental feature weighting strategies are proposed in this paper, one based on a leave-one-feature-out, the other based on a feature-wise separability criterion. We will describe the integration concept of the feature weights in the evolving fuzzy classifiers, using single and multi-model architecture, where *FLEXFIS-Class SM* and *FLEXFIS-Class MM* serve as training engines. The whole approach of integrated incremental feature weighting in evolving fuzzy classifiers will be evaluated based on high-dimensional on-line real-world classification scenarios and based on data from the Internet. The results will show that incremental feature weighting in EFC in fact helps to reduce curse of dimensionality and therefore guides the evolving fuzzy classifiers to a higher on-line predictive power.
© 2010 Elsevier B.V. All rights reserved.

Keywords: Incremental feature weighting; Dynamic dimension reduction; Separability criteria; Evolving fuzzy classifiers; *FLEXFIS-Class*; Single and multi-model architecture; On-line classification

1. Motivation and state of the art

Data-driven fuzzy classifiers, i.e. fuzzy classification models purely extracted out of data, are nowadays used in many fields of applications such as decision making [1], decision support systems in medicine [2], fault and novelty detection [3], classification in EEG signals [4], image classification [5,6] and processing [7]. This is mainly because they are offering a powerful tool which is able to model non-linear dependencies between features and still providing some interpretable insight at the same time. Various important training algorithms for building up data-driven fuzzy classifiers approaches, also applying different model architectures, can be found in [8–13].

A significant problem when learning fuzzy classifiers from data is the so-called curse of dimensionality effect, especially when the number of features in a classification problem, compared to the number of available training samples, is high. Availability of sufficient training samples in classification tasks is a severe problem as the annotation,

E-mail address: edwin.lughofer@jku.at.

0165-0114/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.fss.2010.08.012

i.e. assignment of class labels in order to achieve a supervised training set ready-made for sending it into the classifier training algorithm, is quite time-intensive. A high dimensionality of the data set on the other hand usually decreases the predictive performance of classifiers, especially of non-linear classifiers, on new unseen samples significantly [14,15] (also known as *curse of dimensionality*). Therefore, during the last decade feature selection methods in conjunction with training of fuzzy classifiers for reducing the curse of dimensionality were developed. For instance in [16] features are selected according to the ranking obtained by applying mutual information. Another attempt for reducing dimensionality in fuzzy classifiers is presented in [17], where the best features for a fuzzy integral classifier [18] are elicited by a specific interaction index, comparing the importance of feature pairs with the importance of their sums. A fuzzy entropy measure was applied in [19] and a weighted similarity measure in [20] (applied to classification of web pages) for discarding noise-corrupted, redundant and unimportant features. An interesting approach for reduction of the feature space is presented in [21,22], where feature selection is integrated into a multistage genetic learning process, which determines a set of feature subsets by means of the chromosomes in the final population with the best fitness value.

Another important issue in classification scenarios is the usage of incremental updates of classifiers during on-line processes at the systems, ideally in single-pass mode without using any prior data samples (in order to achieve fast updates with low virtual memory demand). This may guide the classifiers to more predictive accuracy, especially when the original set of training samples was low or new operating conditions or system states arise suddenly (causing extrapolation situations in the original model). An attempt to tackle this issue was made in [23], where two different incremental training methods, *FLEXFIS-Class* [24,25] and *eClass* [26] are compared with respect to empirical performance and extended to include different classifier architectures (single model, multi-model in two variants). Another incremental fuzzy classification approach is demonstrated [27], where a generalized fuzzy min–max neural network is updated during on-line and hyper-boxes (updated in this network) are generalized to represent explicit fuzzy classification rules. Another incremental learning approach is presented in [28], where the concept of fuzzy pattern matching [29], using a transformation probability–possibility to construct densities of possibilities [30], is used as classifier.

Now, it is a challenge to apply an adaptive feature selection process during the incremental on-line update of the fuzzy classifiers in order to compensate different operation modes, changing characteristics of the process, etc., in the input structure of the classifier. This is because at the beginning of the whole learning process specific features may turn out to be much more important than later on. In principle, someone may apply some incremental feature selection techniques such as [31–33] synchronously to the update of the fuzzy classifiers. This may serve as a by-information about the most essential features, but the problem remains how to integrate this permanently changing information into the evolving fuzzy classifiers in a smooth way. In fact, the problem is how to change the input structure of the models appropriately without causing discontinuities in the learning process. In fact, features can be dynamically and on-line exchanged in the importance list, but this requires an exchange in the input structure of the fuzzy models as well (e.g. Feature #1 replaces Feature #2 in the list of five most important features after a new data block as becoming more important than Feature #2). However, parameters and rules were learned based on the inputs used in the past. As usually two inputs show different distributions and different relations with other features in the joint feature space, it is not reliable to exchange one input with another one by simply exchanging it in the model definition and continue with the old parameter setting learned for the other input. Furthermore, the inclusion of features weights gives rise which features are more important and which ones less and hence serves a kind of additional information to the users and experts, providing more insight into the process and interpretability of the classifiers.

2. Our approach

In this paper, we propose an alternative approach for dynamically reducing curse of dimensionality during the incremental on-line training phase of fuzzy classifiers in a smooth way. Therefore, we exploit the generalization concept of feature selection, called feature weighting, which assigns weights lying in [0,1] to the features according to their importance level; i.e. features which are more important for the discrimination of two or more classes are assigned weights lying near 1, features with a low importance are assigned weights lying near 0. If the weight of a feature approaches 0, the feature will not have any impact in the learning as well as classification process. Therefore, a soft dimension reduction can be achieved through the concept of feature weighting. Soft here means that features with very low weights have little impact and are almost but not completely discarded in the high-dimensional feature space. The key point and advantage among a crisp feature selection approach is now that these weights can be updated continuously during the incremental learning phase with newly loaded data samples in a life-long learning mode (all samples equally weighted): the update with one single sample changes the weights of all features just slightly. This causes a smooth learning behavior of the evolving fuzzy classifiers when integrating the features into their training (opposed to crisp feature selection resp. abrupt weighting when using weights in $\{0, 1\}$). Note that incremental feature weighting approaches were investigated and developed in connection with other learning methods such as [34] (for weighted distance calculations between objects in images), [35] (for integrating feature weights into Naive Bayes text classifiers) or [36] (for incremental concept evolution according to drift). Within the concept of evolving fuzzy systems, incremental feature weighting was, to our best knowledge, not studied so far. In this paper, two novel approaches for incremental feature weighting will be demonstrated, both based on Dy–Brodley's separability criterion [37] (Section 3): a leave-one-feature-out separability criterion by excluding each one of the p features leading to p measures, telling us how good the separation remains when each one of the p features is excluded from the feature space (Section 3.4), and a faster feature-wise (single dimension-wise) criterion. We will present update formulas for both criteria (Section 3), and mention two possibilities for improving the stability of the leave-one-feature-out approach in Section 4. Furthermore, we will demonstrate how to integrate the feature weights into the evolving fuzzy classifiers using *FLEXFIS-Class SM* and *FLEXFIS-Class MM* [25] approaches (Section 5). Hereby, we integrate them during the training as well as the classification phase. In Section 6, we will evaluate the impact of including the incrementally updated features weights into the evolving fuzzy classifiers by comparing the novel variants (features weights included) with the conventional incremental training variants (without including any feature weights) in terms of predictive power on new on-line samples. Also, we will study the required computational performance for updating the evolving fuzzy classifiers and feature weights with one single sample as well as how many features are out-weighted during incremental training phase, i.e. how intensively dimensionality is reduced. This will be based on two (noisy and high-dimensional) real-world data sets from surface inspection scenarios (CD Imprint and food production) and on one data set from the Internet (spam-base).

3. Incremental feature weighting — two variants

3.1. Basic concept

The concept of (on-line incremental) feature weighting is an attempt to decrease the over-fitting of evolving fuzzy classifiers by reducing the curse of dimensionality. In principle, it is a generalization of the (incremental) feature selection concept [38,39], assigning continuous weights to features in $[0,1]$ instead of crisp weights in $\{0, 1\}$. The continuous weights can be seen as importance levels of the features within a classification scenario, i.e. their impact for discriminating two or more classes. Whenever feature weights are approaching towards 0, it means that their importance is relatively low compared to others which have values near 1. In extreme cases, during the feature weighting process, it may happen that some features are assigned values of 0 ($\rightarrow$ automatically switching off features) and values of 1 ($\rightarrow$ full features inclusion as in conventional classifier training processes). In the first case, a kind of regularization effect is achieved, which usually tries to shrink the weights of features towards 0 [14].

The main advantage of feature weighting over feature selection becomes clear when applying it within an incremental on-line learning scenario. Incremental feature selection abruptly change the weights from 0 to 1 or vice versa, requiring changes in the input structure of the classifiers which is hard or almost impossible to compensate in a smooth continuous way during on-line learning mode without forcing a time-intensive re-training step with all data seen so far. In the context of incremental learning, serving as engine for updating and evolving fuzzy classifiers (mostly for on-line demands or in case of huge-data bases), the feature weights should be also permanently and synchronously updated with the fuzzy classifiers. This is because, the importance levels of the features may change based on the dynamics of the system such as parameter changes, upcoming new operating conditions, drift occurrences, or simply because the number of data samples included so far into the models is low (compared to the dimensionality of the data space). In Sections 3.4 and 3.5, we present two possibilities how to extract feature weights incrementally in single-pass and sample-wise manner. That means one single sample is sent into the update algorithms and immediately discarded, afterwards. This endeavors fast computation speed and a low virtual memory demand. Both possibilities are based on an incremental update of a specific kind of separability criterion demonstrated in the subsequent section.

3.2. Separability criterion

For assigning weights to the features, we exploit a well-known criterion for measuring the discriminatory power of a feature set in classification problems is the so-called Fisher's interclass separability criterion which is defined as [15]

$$J = \frac{\det(S_b)}{\det(S_w)} \tag{1}$$

where S_b denotes the between-class scatter matrix measuring how scattered the cluster means are from the total mean and S_w the within-class matrix measuring how scattered the samples are from their class means. The goal is to maximize this criterion, i.e. achieving a high between-class scatter (variance), while keeping the within-class scatter (variance) as low as possible. S_w can be expressed by the sum of the covariance matrices over all classes, i.e. when assuming K classes by

$$S_w = \sum_{j=1}^{K} \Sigma_j \tag{2}$$

with Σ_j the covariance matrix for the jth class:

$$\Sigma_j = \frac{1}{N_j} \sum_{k=1}^{N_j} (X_{k,.} - \bar{X}_j(N_j))^T (X_{k,.} - \bar{X}_j(N_j)) \tag{3}$$

with X the regression matrix containing N_j samples from class j as rows and p features as columns, $\bar{X}_j(N_j)$ the mean value vector of all the features for the N_j samples. In case of K classes, the matrix S_b is defined as

$$S_b = \sum_{j=1}^{K} N_j (\bar{X}_j - \bar{X})^T (\bar{X}_j - \bar{X}) \tag{4}$$

with N_j the number of samples belonging to class j, $\bar{X}_j$ the center of class j (i.e. the mean value of all samples belonging to class j) and $\bar{X}$ the mean over all data samples (for all features).

However, this criterion has the following shortcomings [37]:

- The determinant of S_b tends to increase with the number of inputs, hence also does the separability criterion (1), preferring higher dimensionality of the input space.
- It is not invariant under any non-singular linear transformation [40]. This means that once m features are chosen, any non-singular linear transformation on these features changes the criterion value. This implies that it is not possible to apply weights to the features or to apply any non-singular linear transformation or projection and still obtain the same criterion value.

Hence, we apply the following criterion (Dy–Brodley's measure) [37], which does not suffer from the deficiencies mentioned in the itemization above:

$$J = \operatorname{trace}(S_w^{-1} S_b) \tag{5}$$

with trace(A) the sum of the diagonal elements in A. The term within the trace operator is nothing else than the between class scatter S_b normalized by the average class covariance. Hence, the larger the value of $\operatorname{trace}(S_w^{-1} S_b)$ is, the larger the normalized distance between clusters is, which results in better class discrimination.

3.3. Incremental update of separability criterion

Regarding incremental capability of this criterion, it is obviously sufficient to update the matrices S_w and S_b and then to compute (5). Both matrices can be updated during incremental mode in single-pass and sample-wise manner.

The matrix S_b can be updated by simply updating N_j (the number of samples falling into class j) through counting and $\bar{X}_j$ as well $\bar{X}$ by incrementally calculating the mean:

$$\bar{X}(N+m) = \frac{N\bar{X}(N) + \sum_{k=N}^{N+m} X_{k,.}}{N+m} \tag{6}$$

Hence, the old mean is multiplied by the older number of samples, the new samples added in sum and this divided by the whole number of samples $(N+m)$ seen so far. The same update mechanism applies to the mean over samples falling into class j, $\bar{X}_j$.

For S_w, we need to update the covariance matrices of each class. When including m new samples from class j, we obtain the following for the jth class:

$$\begin{aligned}\Sigma_j(new) = {} & \frac{1}{N_j+m}\sum_{k=1}^{N_j}(X_{k,.} - \bar{X}(N_j+m))^T(X_{k,.} - \bar{X}(N_j+m)) \\ & + \frac{1}{N_j+m}\sum_{k=N_j+1}^{N_j+m}(X_{k,.} - \bar{X}_j(N_j+m))^T(X_{k,.} - \bar{X}_j(N+m))\end{aligned}$$

Then we obtain (leaned on [41])

$$\begin{aligned}\Sigma_j(new) = {} & \frac{1}{N_j+m}\sum_{k=1}^{N_j}((X_{k,.} - \bar{X}(N_j)) + c(\bar{X}(N_j) - \bar{X}(m)))^T \\ & \times((X_{k,.} - \bar{X}_j(N_j)) + c(\bar{X}_j(N_j) - \bar{X}_j(m))) \\ & + \frac{1}{N_j+m}\sum_{k=N_j+1}^{N_j+m}((X_{k,.} - \bar{X}_j(N_j)) + c(\bar{X}_j(N_j) - \bar{X}_j(m)))^T \\ & \times((X_{k,.} - \bar{X}_j(N_j)) + c(\bar{X}_j(N_j) - \bar{X}_j(m)))\end{aligned} \tag{7}$$

with $c = m/(N+m)$. By developing the products within the sum terms using $(A+B)^T(A+B) = A^TA + B^TB + A^TB + B^TA$ one can finally derive the update of the covariance matrix for class j with m new samples from class j:

$$\Sigma_j(new) = \frac{1}{N_j+m}\left(N_j\Sigma_j(old) + m\Sigma_{j;pnew} + \frac{N_jm}{N_j+m}(\bar{X}_j(N_j) - \bar{X}_j(m))^T(\bar{X}_j(N_j) - \bar{X}_j(m))\right) \tag{8}$$

with $\Sigma_{j;pnew}$ the covariance matrix on the m new samples for class j. Note that in case of $m=1$, i.e. feeding the N_j+1th sample x_{N_j+1} into the update, the covariance matrix update becomes

$$\Sigma_j(new) = \frac{1}{N_j+1}\left(N_j\Sigma_j(old) + \frac{N_j}{N_j+1}(\bar{X}_j(N_j) - x_{N_j+1})^T(\bar{X}_j(N_j) - x_{N_j+1})\right) \tag{9}$$

as the covariance matrix on one new sample is equal to 0.

In [42] it is reported that an even more stable calculation can be achieved by including the effect of mean changes over the new data samples $\Delta\bar{X}_j(N_j+m) = \bar{X}_j(N_j+m) - \bar{X}_j(N_j)$ as a rank-one modification. In case of $m=1$ (sample-wise update), this leads us to an alternative update of the covariance between features x_1 and x_2 for class j:

$$\begin{aligned}(N_j+1)cov_j(x_1,x_2)(new) = {} & N_jcov_j(x_1,x_2)(old) + (N_j+1)\Delta\bar{x_1}(N_j+1)\Delta\bar{x_2}(N_j+1) \\ & + (\bar{x_1}(N_j+1) - x_1(N_j+1))(\bar{x_2}(N_j+1) - x_2(N_j+1))\end{aligned} \tag{10}$$

3.4. Leave-one-feature-out approach (LOFO) for assigning feature weights

Now, the question remains how to use this criterion for assigning feature weights. Assuming that the full dimensionality of the data is p, the idea is now to calculate (5) p times, each time one of the p features is discarded $\rightarrow$

leave-one-feature-out approach. In this sense, we obtain p different between-class and within-class variances as well as p different values for (5), $J_1, \dots, J_p$, which can be again updated synchronously and independently in incremental mode. A statement on the relative importance of features can be made, when sorting these values in decreasing order: the maximal value of these indicates that the corresponding discarded feature is the least important one, as the feature was discarded and still a high value is achieved. In the same manner, the minimal value of these indicates that the corresponding discarded feature is the most important one, as dropping the value of the separability criterion (when applied to the remaining $p-1$ features) more significantly than any of the others. Note that when using all features usually achieves the highest values of (5), but not necessarily as there can be a kind of curse of dimensionality effect in case of a low number of samples. Hence, we do not apply a comparison with the value obtained using all p features. However, a relative comparison is valid, as we always process $p-1$ features, i.e. having the same dimensionality when comparing the values of Dy–Brodley's separability measure. Then, the weight of feature j, λ_j, is achieved by

$$\lambda_j = 1 - \frac{J_j - \min_{1,\dots,p}(J_j)}{\max_{j=1,\dots,p}(J_j)} \tag{11}$$

This guarantees that the feature causing the minimal value of J_j (hence most important one) gets a weight of 1, whereas all others are relatively compared to the minimum: in case when they are close the weight will also be almost 1, in case when they are significantly higher, the weight will be forced towards 0. A problem of this feature weight assignment strategy is given in case of a high number of inputs in real-world application scenarios. Then, it is quite likely that the deletion of one feature does not decrease the value of (5) significantly compared to when using all features (also not for those with high discriminatory power). This is because the likelihood that a similar or even redundant feature is present in the data set is quite high. This finally means that the normalized feature weights according to (11) are all close to 1. Hence, in case of a high number of input, we suggest to normalize to the full range of [0,1] by

$$\lambda_j = 1 - \frac{J_j - \min_{1,\dots,p}(J_j)}{\max_{j=1,\dots,p}(J_j) - \min_{1,\dots,p}(J_j)} \tag{12}$$

hence the feature with the weakest discriminatory power (and therefore maximal J_j) is assigned a weight value of 0 and the feature with strongest discriminatory power a weight of 1.

3.5. *Single dimension-wise (SD) approach for assigning feature weights*

The problem with the leave-one-feature-out approach is that it requires a quite high computation time, as for each single sample falling into class L, the covariance matrix Σ_L and the between-class scatter matrix for class L need to be updated, which demands an update of the whole S_w resp. S_b by summing up over all classes. Updating the covariance matrix Σ_L requires $p + 2p^2$ operations (mean vector update for each dimension + component-wise multiplication of column vector with row vector to form the new covariance information and summation of two matrices (old covariance matrix with new covariance information)), updating the between-class scatter for class L requires $p+p^2$ operations due to similar considerations (no matrix addition is required, hence p^2 instead of $2p^2$). For calculating the new S_b and S_w matrices and assuming K classes, K p times p matrices are summed up, achieving a complexity of $2Kp^2 + 3p^2 + 2p$. Calculating criterion (5) from the updated S_w and S_b matrices requires another $2p^3 + p$ operations (matrix inversion as well as matrix multiplication requires p^3, the trace p operations as summing up the diagonal entries of a p times p matrix). The complexity for calculating one J_p is therefore $O((2K+3)p^2+2p^3+3p) \approx O((2K+3)p^2+2p^3)$, hence the overall complexity for all J_j's is $O((2K+3)p^3 + 2p^4)$. For reducing this complexity, we apply a greedy-based approach for approximating Dy–Brodley's separability criterion. This is done by calculating the between-class and within-class scatter for each feature separately, so by not neglecting possible good interrelations between some features for a better discriminatory power among classes. As reported in [43], this approach gives a reasonable approximation of the original separability criterion. At least, features with a high proportion of between-class scatter to within-class scatter are for sure important features also when joined with others for building a classifier. Indeed, features with a low proportion of between-class scatter to within-class scatter may improve separability of classes when joined with others; however, this improvement is usually stronger for features having higher single discriminatory power. By using the feature-wise approach we obtain scalar values for S_b and S_w, i.e. for the ith feature:

$$S_b(i) = \sum_{j=1}^{K} N_j(\bar{X_{j;i}} - \bar{X_i})^2, \quad S_w(i) = \sum_{j=1}^{K} Var_{j;i} \tag{13}$$

with $\bar{X_i}$ the mean value of the ith feature, $\bar{X_{j;i}}$ the mean value of the ith feature over class j samples and $Var_{j;i}$ the variance of the ith feature over class j samples. The single separability criterion for feature i is simply calculated by

$$I_i = \frac{S_b(i)}{S_w(i)} \tag{14}$$

The mean value over features for class j samples can be updated by the incremental mean as in (6), the variance of a features x_i is updated in the same manner as the covariance in (10) by setting $x_i = x_1 = x_2$. The update is quite fast as it requires only linear complexity in the number of features ($O(p)$) for each single sample. The final feature weights are obtained by

$$\lambda_j = \frac{I_j}{\max_{j=1,\ldots,p} I_j} \tag{15}$$

Someone may argue that two redundant features are given the same weight (no matter which weight criterion is applied), hence not masking out any of the two features (by assigning a weight of 0). On the other hand, three pro-arguments exist for this strategy:

(1) it is from interpretability point of view intuitive and even advantageous to assign equal weights as it would be unnatural to give one redundant feature a weight with value 0 and the other a weight with value greater than 0 according to its importance w.r.t. the others;
(2) in the incremental learning context two features, originally redundant, may become un-redundant after some time — hence, it is natural to start with equal weights for two redundant features at the beginning and change these smoothly and slightly away from each other; on the other hand, if forcing one redundant feature to have weight of 0, then it may cause a discontinuous jump to a significant higher value after new data blocks are loaded;
(3) it is not really counter-productive in terms of preventing any curse of dimensionality reduction, as deleting one of two completely redundant features does not help to improve curse of dimensionality in the sense that decision boundaries are easier to find. This is because samples of redundant features are lying along straight lines.

3.6. A small empirical study on Iris data

Here we provide a small evaluation of the on-line feature weighting strategies based on the Iris data set (from UCI repository) to demonstrate how this approach works. The Iris data set is a well-known data set including four features and three classes and enjoys great popularity in the pattern recognition community for comparing classification approaches. The goal is to recognize the species of flowers (setosa, versicolor and virginica) based on some length and width criteria of their blossoms. Fig. 1 represents a plot of the four features ((a)–(d)) included in this data set, whereby the x-axis represents the sample number. From this figure we visually can imagine the discriminatory power of the single features as the first 50 samples belong to Class #1, the next 50 samples to Class #2 and the last 50 samples to Class #3. In Features #1 and #2 the classes are mixed up, hence having a quite low discriminatory power, whereas Features #3 and #4 possess a high one (based on these two the classes can be already separated pretty well — e.g. thresholds of 2.5 and 5.0 in Feature #3 already divides the three classes with almost 0 error).

Now we apply feature weighting criteria according to (12) and (15) in order to see how the features are weighted. The outcome of the two methods (in batch mode) is presented in Table 1. From this table someone can recognize that Features #1 and #2 (sepal length and width) are giving lower weights by all three feature weighting variants than Features #3 and #4, which is a reasonable choice according to Fig. 1. Furthermore, someone gets an impression about the distribution of the weights along the different features: the feature-wise and the normalized leave-one-feature-out approaches (Columns #4 and #3) perform similar as pushing the weights of the unimportant features (sepal length

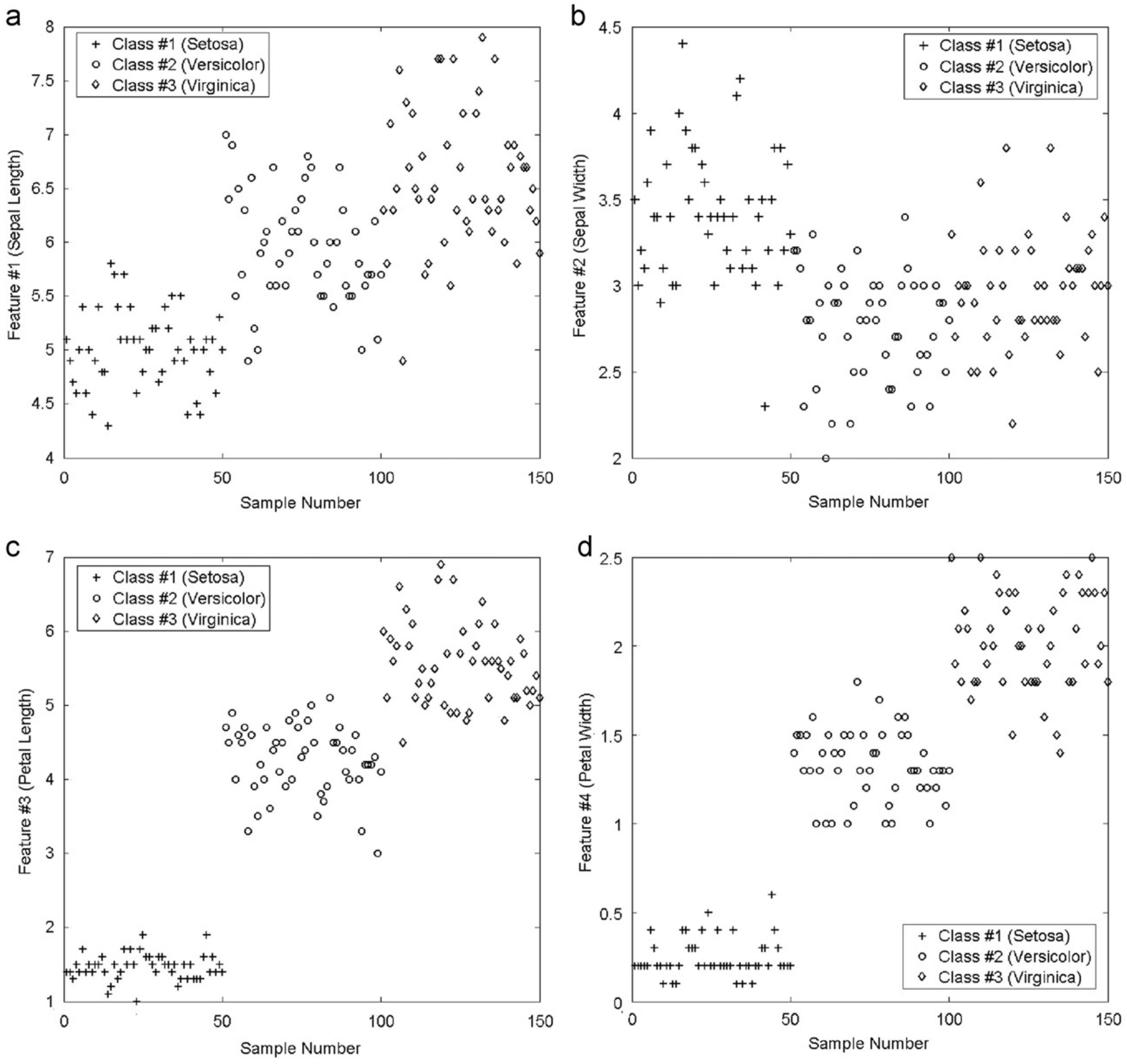

Fig. 1. Four Features (a)–(d) plotted versus the class information (plus, circle, diamond).

Table 1
Feature weights obtained for Iris data set according to the (un-normalized and normalized) leave-one-feature-out and the feature-wise criterion.

Feature	Leave-one-out u./inc.	Leave-one-out n./inc.	Feature-wise/inc.
Sepal length	0.72/0.72	0/0	0.1/0.11
Sepal width	0.84/0.85	0.41/0.42	0.04/0.04
Petal length	1.0/1.0	1.0/1.0	1.0/1.0
Petal width	0.89/0.89	0.61/0.61	0.81/0.82

and width) towards zero, while maintaining high values of the important ones (petal length and width). The un-normalized leave-one-feature-out approach tendentially produces high feature weights for all features, as the weights are seen relative to the drop obtained in Dy–Brodley's separability criterion when discarding the most important

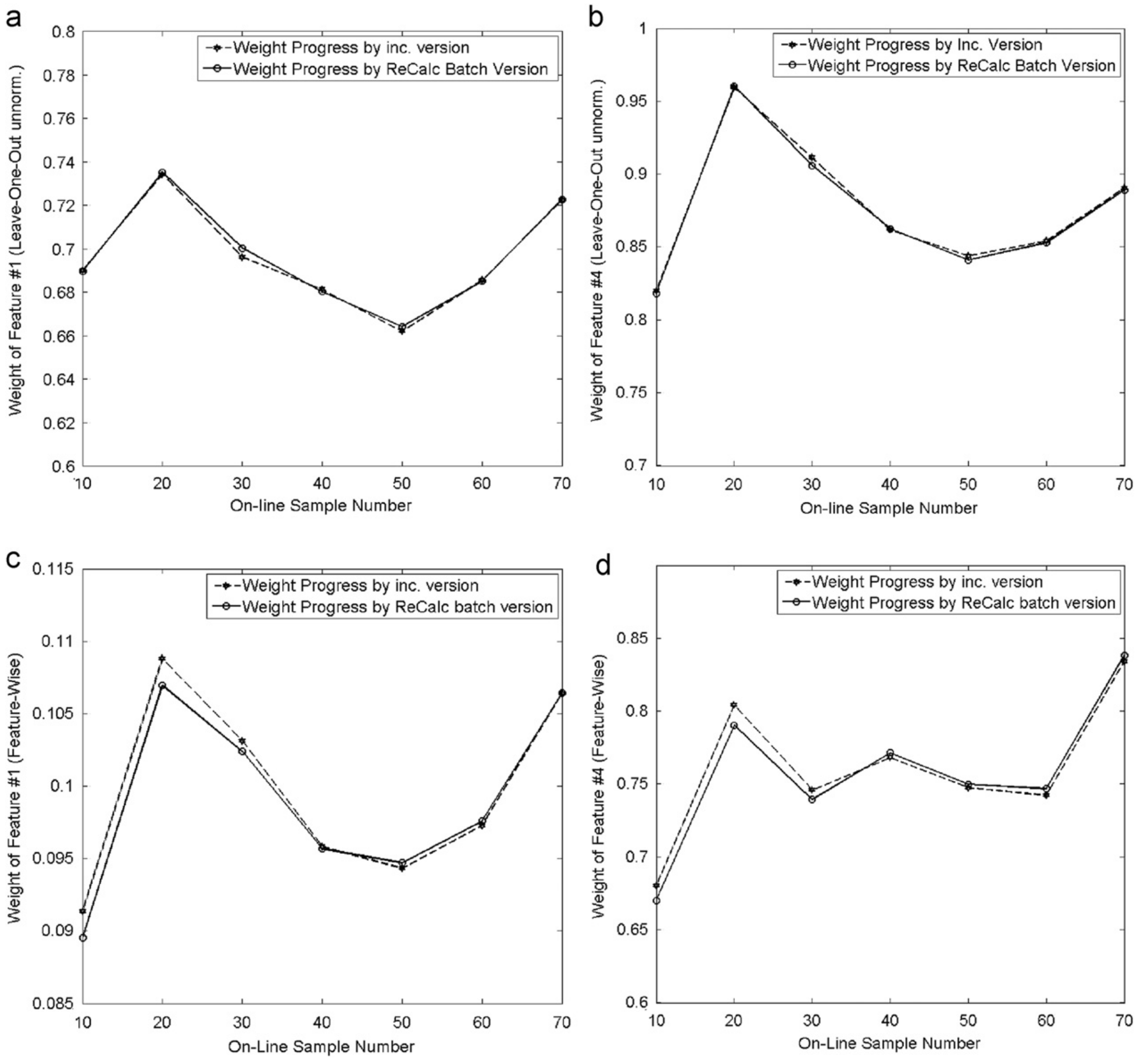

Fig. 2. Progress of the feature weights for Features #1 (left) and #4 (right) during the incremental update (dotted lines) compared with batch re-trained ones (solid lines); upper row: leave-one-feature-out criterion; lower row: single feature-wise criterion.

feature (petal length). In higher-dimensional problems, the weights are expected to have even higher values in a smaller range below 1. According to this observation, we have some favor for the single feature-wise separability criterion, as (1) performing much faster than the leave-one-feature-out criterion and (2) reducing the curse of dimensionality stronger than the leave-one-feature-out approach (unimportant features are giving very low weights and hence are almost masked out). Now, we send the samples in single-pass mode into the on-line incremental versions (by updating between- and within-class scatter information) and obtain the results as shown in after the slashes in Table 1. Clearly, the incremental versions are able to approximate the weight of the batch version quite closely (values are almost the same).

Furthermore, we also examine the behavior of the feature weight updates during the incremental learning and compare these with the re-calculated versions of un-normalized leave-one-feature-out and feature-wise criteria. The results are shown in Fig. 2, and someone can realize that almost the same progress can be observed. This finally means that our incremental approaches converge to or even behave in the same way as the batch approaches (when re-trained on all seen samples so far). Hence, we can speak about a robust incremental feature weighting approach (robustness in terms of convergence of the incremental learning approach to the batch solution, see also [24]).

4. Improving stability of incremental leave-one-feature-out approach

The within-class scatter matrix may become singular or nearly singular in case when there are redundant features in an on-line data stream; with growing dimensionality of the input space, the likelihood of redundant features present in the data increases. Then, calculating the inverse in (5) leads to an unstable and wrong criterion J. One possibility to circumvent this is to apply Tichonov regularization by adding αI to S_w with α chosen by a heuristic approach not requiring any past samples. The on-line regularization strategy based on heuristic setting of regularization parameter α was presented in [44] and can be summarized as follows:

- Compute the condition of the matrix S_w by $cond(S_w) = \lambda_{max}/\lambda_{min}$ with λ_{max} the largest and λ_{min} the smallest eigenvalue.
- If $cond(S_w) > threshold$, the matrix is badly conditioned and we set

  $$\alpha = \frac{2\lambda_{max}}{threshold}$$

  with *threshold* a large value, for instance 10^{15}.
- Else, we set $\alpha = 0$.

However, this yields a high computational effort, as the condition of a matrix and its eigenvalues needs significant resources (these are computed for each on-line sample).

An alternative is to delete redundant features at the beginning of the whole training process and to continue with the remaining features. Redundant features are identified by finding linear dependencies between two or more features: for doing so, a possibility is to build up linear regression models by using one feature as target and a subset of the remaining ones as input (selected with a variable selection method, we suggest to apply an orthogonal version of forward selection as described in [45]). In case of p features, we therefore obtain p models as follows:

$$\begin{aligned}
Feat_1 &= f_1(Feat_{1,1}, Feat_{2,1}, \ldots, Feat_{L_1,1})\\
Feat_2 &= f_2(Feat_{1,2}, Feat_{2,2}, \ldots, Feat_{L_2,2})\\
&\ldots\\
Feat_p &= f_p(Feat_{p,1}, Feat_{p,2}, \ldots, Feat_{L_p,p})
\end{aligned} \tag{16}$$

with f_1 to f_p linear models, where $Feat_{i,j}$ denotes the jth input feature and L_i the number of input features for approximating the ith feature; in general $Feat_{1,i} \neq Feat_{1,j}$, $Feat_{2,i} \neq Feat_{2,j}$, … for $i \neq j$, and $L_i \neq L_j$, so the selection of the L_i features included in the various models is always different (as depends on the target feature) and also using different (number of) inputs.

We inspect the qualities of the models in terms of a quality measure such as r-squared-adjusted. The target features of models with a high quality can be seen as redundant to a linear combination of the input features in these models and hence can be deleted. Cross-deletion have to be checked appropriately such that no target feature is deleted which is needed as input for another regression model with high quality where the target was already deleted. This is achieved by the following steps:

(1) Input: list of features L.
(2) Build up feature relation models as in (16) for the p features.
(3) Calculate the quality of the p models with quality measure lying in [0, 1] (r-squared, r-squared-adjusted).
(4) Delete all models which have a lower quality than a pre-defined threshold (we used 0.97 for all experiments).
(5) Sort the p_2 remaining models according to their quality in descending order.
(6) Delete the target feature in the model with highest quality from the list of features: $L = L - \{feat_1\}$ (as it can be approximated by a linear combination of others with high quality).

(7) For $i = 2, \dots, p_2$, take the target feature from ith model and look whether it is an input feature in any of the former $j = 1, \dots, i-1$ models (with higher qualities):

- If no, delete this target feature from the list of features: $L = L - \{feat_i\}$ (as it can be approximated by a linear combination of others with high quality).
- If yes, keep it in the feature list.

(8) Output: new feature list L where the redundant features are omitted.

5. Integrating feature weights into evolving fuzzy classifiers

The next step is how to include the feature weights permanently updated by the approaches as discussed in Section 3 into the fuzzy classifier's training/updating process. In principle, there are three possible strategies:

(1) Inclusion of the feature weights during the classification process only, i.e. when classifying new instances.
(2) Inclusion of the feature weights during the training process when building up the classifiers.
(3) Inclusion of the feature weights during training and classification process.

In the first option, the classifiers are trained as in the usual setting (without feature weights) and the weights are only included when classifying new samples. Thus, no decrease of the curse of dimensionality is achieved in the sense that the complexity of the classifiers is reduced, i.e. fewer rules are generated. However, by ignoring features with low weights during classification, classification accuracy may be improved when predicting the classes of new samples. An example for underlining this is visualized in Fig. 3 for a two-dimensional data set containing two classes and three rules (shown as clusters in the two-dimensional space): the second feature (y-axis) has almost no discriminatory power, whereas the first feature (x-axis) can discriminate the classes almost perfectly. Hence, when calculating the distance to the centers of the three clusters = rules for a new incoming sample (dark dot), the second feature has almost no impact, so the winning cluster (nearest cluster) turns out to be the bottom left cluster, which only contains samples from class '□'. In this sense, this class is correctly assigned to the new sample. If the feature weights would be not included in the classification process (inference of the classifiers), the nearest rule to the new sample would be the upper middle one, where class ' ○ ' has a majority → this class would be falsely assigned to the new sample (from the distribution of the classes it is somewhat clear that samples appearing at the far left side of the x-axis belong to class '□'). Thus, even for simple two-dimensional data sets, feature weights may be important for guiding classifiers to higher correct classification rates.

When using the second option, the classifier can be prevented from generating such rules as indicated by the upper middle cluster in Fig. 3 (note that within the incremental learning context the bottom two clusters were included in the first data block and the upper middle cluster in the second data block). The weight of the second feature is (1) either so low that the distance to the lower rule centers is always close enough such that no rule is generated in-between the two clusters (just the centers obtained from the first data block are moved a bit together and towards the center of

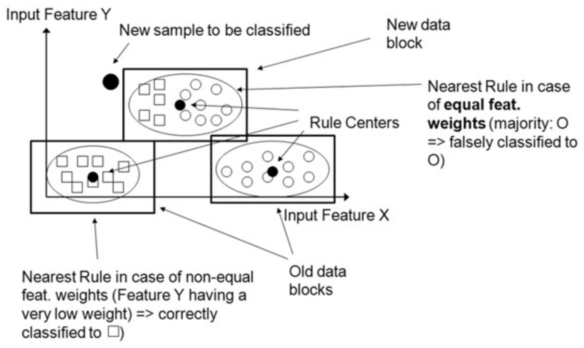
Input Feature Y
New sample to be classified
New data block
Nearest Rule in case of **equal feat. weights** (majority: O => falsely classified to O)
Rule Centers
Input Feature X
Nearest Rule in case of non-equal feat. weights (Feature Y having a very low weight) => correctly classified to □)
Old data blocks

Fig. 3. Impact of including feature weights when classifying new samples according to the winner-takes-it-all classification scheme (majority class in the nearest rule is assigned as class label).

Feature X) or (2) the hole between the lower rule centers is wide enough such that a new cluster may be generated (in order to cover the data space sufficiently); however, in this case a new rule will be generated in-between these two (close to the *x*-axis) rather than way up along the second feature (or in general along the features with lower weights). In a winner-takes-it-all concept, a new sample will be closest to this center when falling in-between the other two rules and not when appearing on either left or right side of the *x*-axis with a high value in Feature #2.

The third option is a combination of the two above. In usual cases, the second option should decrease the complexity of the classifiers by not generating misleading rules, hence the first option should get superfluous. Nevertheless, in order to be on the save side, we suggest to use a combination of both (double-coupled). In the subsequent section we will demonstrate how an integration of feature weights is accomplished in the evolving fuzzy classification methods *FLEXFIS-Class SM* and *FLEXFIS-Class MM*.

5.1. Integration in FLEXFIS-Class SM

FLEXFIS-Class SM, firstly introduced in [25], extended in [23], exploits the classical fuzzy classifier architectures, which comes with singleton labels in the consequent parts of the rules, see [9,8,46]. The single antecedent parts of the rules, each corresponding to one dimension in the feature space, are connected by a t-norm [47] to obtain the rule activation and class label confidence degrees (one rule antecedent part is connected with one unique consequent class label). In *FLEXFIS-Class SM* Gaussian membership functions with product t-norm operator are used. Hence, during the classification phase, feature weights can be simply included when eliciting the rules' activation degrees, that is by computing the t-norm over all antecedent parts, where the fulfillment degree of each part is multiplied with the corresponding feature weight (belonging to this dimension):

$$\mu_i(\vec{x}) = \prod_{j=1}^{p} e^{-(1/2)((x_j-c_{ij})^2/\sigma_{ij}^2)\lambda_j}, \quad i = 1, \ldots, C \tag{17}$$

with p the dimensionality of the feature space, C the number of rules, c_{ij} the center of the jth antecedent part in the ith rule and σ_{ij} the width of the jth antecedent parts in the ith rule. This means that the fuzzy set membership degrees of unimportant features (with feature weights near 0) are 1, hence serving a 'don't care parts' as not influencing the whole rule activation when taking the product over these degrees. For important features (feature weights near 1), the (nearly) actual fuzzy set membership degrees are taken, influencing the rule activation degrees. This concept can be generalized to arbitrary membership functions μ_{ij} by using the following expression for calculating the rule activation degrees:

$$\mu_i(\vec{x}) = \prod_{j=1}^{p} ((\mu_{ij} - 1)\lambda_j + 1), \quad i = 1, \ldots, C \tag{18}$$

This guarantees 'don't care entries' (values of 1) in case of unimportant features and original μ_{ij}'s in case of important ones, interpolating linearly for features with weights in-between 0 and 1. In this sense, a similar effect as shown in Fig. 3 is achieved with the difference that here not the distances to the nearest cluster centers with respect to weighted features are calculated, but the rule activation degrees including the ranges of influences of the rules in each direction. The expression in (18) can be used for any other t-norm, for which values of 1 represent 'don't care parts' (for instance minimum would be such an option).

The training phase in *FLEXFIS-Class SM* takes place in the cluster space, where an evolving version of vector quantization, *eVQ* (see [48], also described in [25]) is applied for updating already existing and evolving new clusters on demand based on the characteristics of new incoming samples. Updating of already existing clusters takes place by a movement of the cluster centers with a learning gain decreasing over the number of samples which formed the cluster (for which the cluster was the winning = nearest one). This is done feature-wise and should be dependent on the importance of the features, enforcing small movements in the direction of unimportant features and large movements in the direction of important ones. Hence, we use the following weighted update of cluster (=rule) centers $\vec{c}$:

$$\vec{c}_{win}^{\,(new)} = \vec{c}_{win}^{\,(old)} + \eta_{win}\lambda I(\vec{x} - \vec{c}_{win}^{\,(old)}) \tag{19}$$

with I the identity matrix and η_{win} the decreasing learning. Generation of new clusters (=rules) is performed whenever a new sample is lying far away from existing cluster centers. In this case, the inclusion of feature weights in the distance calculation suppresses the generation of superfluous clusters in case when the distance with respect to unimportant features is high. For instance, the upper middle cluster in Fig. 3 would have been not generated when giving Feature #2 a low weight, according to its weak discriminatory power. Therefore, including the feature weights in the distance calculation is a key step for reducing curse of dimensionality. The weighted Euclidean distance measure (used in case of axes-parallel ellipsoidal clusters) between a sample $\vec{x}$ and a cluster center $\vec{c}$ is calculated by

$$dist = \sqrt{\sum_{j=1}^{p} \lambda_j (x_j - c_j)^2} \tag{20}$$

The weighted Mahalanobis distance measure (used in case of ellipsoidal clusters with arbitrary direction) between a sample $\vec{x}$ and a cluster center $\vec{c}$ is calculated by

$$mahal = \sqrt{(\lambda. * (\vec{x} - \vec{c}))\Sigma^{-1}(\lambda. * (\vec{x} - \vec{c}))} \tag{21}$$

with Σ^{-1} the inverse of the covariance matrix and $.*$ the component-wise product of two vectors.

5.2. Integration in FLEXFIS-Class MM

FLEXFIS-Class MM, firstly introduced in [25], extended in [23], exploits a multi-model architecture integrating K Takagi–Sugeno fuzzy regression models [49] for K classes. These are trained based on indicator matrices (off-line case), respectively, indicator vectors (on-line case), which follows the idea of linear regression by indicator matrix [50]. Opposed to the linear version, the non-linear version triggered by TS fuzzy models with more than one rule is able to circumvent the masking problem in case of $K > 2$. For the TS fuzzy models, it applies Gaussian membership function combined with product t-norm in the rules' antecedent parts (i.e. fuzzy basis function networks as introduced by Wang and Mendel in [51]). Hence, as in case of *FLEXFIS-Class SM*, during the classification phase feature weights can be simply included when eliciting the rules' activation degrees in the same manner as in (17). This weighting approach is performed when inferencing through each of the K TS fuzzy models and eliciting the final class response by a one-versus-rest classification approach:

$$L = class(\vec{x}) = \underset{m=1,\ldots,K}{\operatorname{argmax}} \hat{f}_m(\vec{x}) \tag{22}$$

with f_m the mth Takagi–Sugeno fuzzy model defined by

$$\hat{f}_m(\vec{x}) = \sum_{i=1}^{C_m} l_{i;m} \Psi_{i;m}(\vec{x}), \quad m = 1, \ldots, K \tag{23}$$

with the normalized weighted membership functions

$$\Psi_{i;m}(\vec{x}) = \frac{e^{-(1/2)\sum_{j=1}^{p}((x_j - c_{ij})^2/\sigma_{ij}^2)\lambda_j}}{\sum_{k=1}^{C_m} e^{-(1/2)\sum_{j=1}^{p}((x_j - c_{kj})^2/\sigma_{kj}^2)\lambda_j}} \tag{24}$$

and consequent functions

$$l_{i;m} = w_{i0;m} + w_{i1;m}x_1 + w_{i2;m}x_2 + \cdots + w_{ip;m}x_p \tag{25}$$

The interpretation of including feature weights in this way during classification is that un-important features do not change the rule activation degrees (low feature weight values triggering high fuzzy set membership values near 1), hence only the important ones are responsible for weighting the consequent hyper-planes in each of the K Takagi–Sugeno fuzzy models. This finally means that the influence of unimportant features for producing the final classification statement is more or less discarded.

The training phase of each Takagi–Sugeno fuzzy model is carried out separately and independently. As in case of *FLEXFIS-Class SM*, the rules and their antecedent parts are again generated by an evolving version of quantization *eVQ* [48]. Hence, the same procedure for including features weights in antecedent part learning as already described in Section 5.1 is applied, omitting superfluous clusters when the distances to already available clusters (rules) with respect to un-important features is high. When updating the consequent parameters $\hat{\vec{w}}_{i;m}$ of the ith rule in the mth model from the kth to the $k+1$th sample with *RFWLS = recursive fuzzily weighted least squares* (for local learning using least squares optimization function — see also [52]):

$$\hat{\vec{w}}_{i;m}(k+1) = \hat{\vec{w}}_{i;m}(k) + \gamma(k)(y(k+1) - \vec{r}^T(k+1)\hat{\vec{w}}_{i;m}(k)) \tag{26}$$

$$\gamma(k) = P_{i;m}(k+1)\vec{r}(k+1) = \frac{P_{i;m}(k)\vec{r}(k+1)}{\dfrac{1}{\Psi_{i;m}(\vec{x}(k+1))} + \vec{r}^T(k+1)P_{i;m}(k)\vec{r}(k+1)} \tag{27}$$

$$P_{i;m}(k+1) = (I - \gamma(k)\vec{r}^T(k+1))P_{i;m}(k) \tag{28}$$

the feature weights are included in $\mu_{i;m}$ as (17) (or (18)) are calculated, affecting the normalized rule membership degree $\Psi_{i;m}$ (appearing in the denominator of (27)). Note that $P_{i;m}$ denotes the inverse Hessian, $\hat{\vec{w}}_{i;m}$ the consequent parameter vector for the ith rule in the mth model and $\vec{r}(k+1)$ the regression vector containing the original samples plus an entry with value 1 for the intercept; $y(k+1) = 1$ whenever the class label of the current sample is m, otherwise $y(k+1) = 0$. Eq. (26) can be interpreted as the new estimate is achieved through adding a correction vector, multiplied by the difference of the new sample and the one-step-ahead prediction of the new sample. Formulas (26)–(28) are carried out for each rule separately (here they are reported for the ith rule) in order to achieve maximal flexibility for adding and pruning rules during incremental update phase (as does not affect the parameters of the other rules). RFWLS denotes an exact incremental learning method for linear consequent parameters, which means that for each time instant the algorithm converges within each iteration step to the optimal solution in the least squares sense [53].

The integration of feature weights into the recursive training of consequent parameters has the effect that tendentially more data samples will cause a higher rule activation degree or more rules which are active with more than ε (as unimportant features are masked out by 'don't care parts'), increasing the statistical significance/stability of the parameter vectors $\hat{\vec{w}}_{.;m}$ in all rules when being trained with (26)–(28). This can be again interpreted as an attempt to decrease over-fitting (by out-weighting features).

Finally, we want to point out that both, *FLEXFIS-Class SM* and *FLEXFIS-Class MM*, require an initial training phase (in batch mode), either with off-line samples or at the beginning of the on-line phase (if possible), see [25]; therefore, features weights can be also initially estimated and further updated during on-line phase (synchronously to the classifiers).

5.3. Comment on computational cost

The additional computation cost for including feature weights in the training as well classification phase of evolving fuzzy classifiers is very marginal. This is because the features are basically used as scalar multiplicators in already existing learning and classification schemes. Hence, the only significant additional computational cost comes from the incremental feature weighting process itself, which is significant for the leave-one-feature-out and quite low for the feature-wise weighting approach — see Section 3.5 for a complexity discussion. Furthermore, the intensity of the cost of incremental feature weighting will be empirically verified in Section 6.

6. Evaluation

This section demonstrates the impact of the incremental feature weighting during incremental on-line evolution of fuzzy classifiers. Thereby, the central aspect will be a comparison of conventional evolving fuzzy classifiers (without using any type of dimensionality reduction method) with extended evolving fuzzy classifiers (integrating incremental feature weighting concepts). First, we demonstrate the experimental setup (application scenarios, data sets, test procedure) we have used for performing this comparison, then we provide the results in terms of the progress of predictive

Table 2
Data sets from CD imprint, egg inspection and spam recognition and their characteristics.

	# Images	# Tr. samples	# Test samples	# Feat.	Class D.
CD imprint	1687	1024	510	74	16.78%/83.22%
Eggs	5341	2895	2302	17+2	79.83%/20.17%
Spam-base	NA	2300	2300	57	60.68%/39.32%

power when more and more samples are sent into the classifiers' update process and in terms of computation speed. We also examine how many features are actually out-weighted during the incremental update phase.

6.1. Experimental setup

We applied our novel approach to three different application scenarios:

- Inspection of CD imprints: the classification problem is to detect system failures (such as color drift during offset print, a pinhole caused by a dirty sieve, occurrence of colors on dirt, palettes running out of ink) when printing the upper-side of compact discs; the inspection is done visually with high-resolution cameras installed directly along the production line. The recorded images of CD imprints are compared with a fault-free ideal reference image, also called master image. This comparison results in a grey-level contrast image, where pixels denote deviations to the master and potential fault candidates. Once, regions of interest (= pixels forming one deviation region are grouped together) are found with a specific variant of hierarchical clustering (see [54]), aggregated features are extracted from the joint regions of interests (=objects) serving as image descriptors. Examples of such aggregated features are the number of objects, the maximal density of objects within a given radius or the average grey level over all objects in an image (see also [54] for details). These features are used as basis for a classifier training process (input) together with class labels 0 and 1, indicating good and bad images.
- Food production (egg inspection): the same procedure as in the case of CD imprints was carried out (comparison with ideal master image, finding regions of interests, extracting the same features, etc.), the task was to discriminate between dirt (= egg is still ok) and yolk (= egg is broken).
- Spam-base: a well-known and widely applied application case from the UCI repository[1]; the data represent feature vectors extracted from text in emails for discriminating between emails containing spam and emails containing no spam.

For the first two application scenarios, we recorded images during the real on-line production process and stored them onto the hard-disc in the same order as recorded. This means that the list of aggregated features extracted from the images appear in the same order in the feature matrix (row-by-row) as the images were recorded on-line. Hence, evolving the classifiers by incrementally sending the samples from the feature matrices into the training algorithms is a one-to-one simulation of the real on-line case. The third data set from the Internet, serving as well-known and widely applied benchmark data set, is used as pseudo-stream and sent sample-wise into the evolving fuzzy classification methods.

The obtained data sets had the following characteristics as shown in Table 2. The number of input features (=dimensionality of the learning problem) in case of eggs is listed as '$17 + 2$', which means that 17 aggregated features were extracted plus two features as outputs (number of counts for each class per image) from a preliminary object classifier trained on the single object features — see [55] for details.

All data sets are divided into a training set and a test set. The first half of the training set is used for an initial training and parameter optimization step, the second half for further adaptation and evolution of the classifiers to simulate the real on-line case. The test data set denotes the last part of the whole feature matrix, which is only used for calculating the accuracies of the classifiers on new on-line data as an estimation of the generalization performance of the classifiers (no further update of the classifiers on this data set is performed). The calculations of the accuracies on this separate test data set are carried out from time to time in order to observe the progress of classification performance with more

[1] http://archive.ics.uci.edu/ml/datasets/Spambase

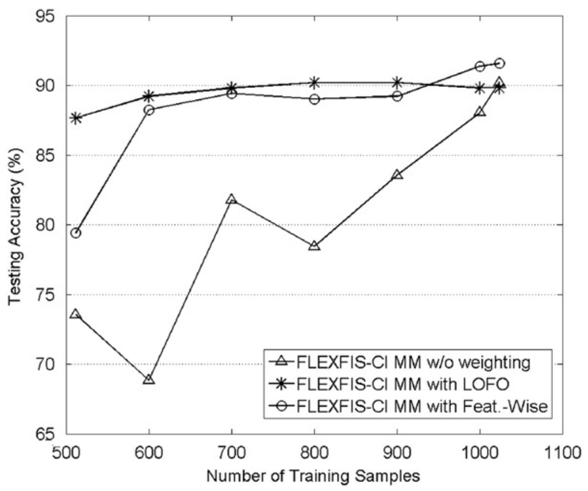

Fig. 4. Progress of accuracies on separate test data set during on-line evolution of fuzzy classifiers with *FLEXFIS-Class MM* for CD imprint data set, line with triangle markers: without feature weighting, line with star markers: with leave-one-feature-out criterion and line with circle markers: with feature-wise separability criterion.

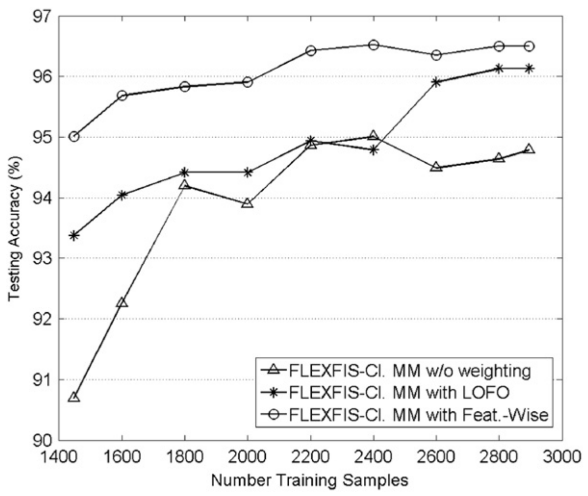

Fig. 5. Progress of accuracies on separate test data set during on-line evolution of fuzzy classifiers with *FLEXFIS-Class MM* for egg inspection data set, line with triangle markers: without feature weighting, line with star markers: with leave-one-feature-out criterion and line with circle markers: with feature-wise separability criterion.

and more data included into the fuzzy classifiers (through the incremental update process) — see Figs. 4, 5 and 6. Both methods *FLEXFIS-Class SM* and *FLEXFIS-Class MM* are applied on all data sets, one time with including the incremental feature weighting technique after leave-one-feature-out criterion, one time with including the incremental feature weighting technique after feature-wise criterion, one time without including any feature weighting approach.

6.2. Results

Fig. 4 presents the evolution of the accuracies (on separate on-line test data) during on-line update of the fuzzy classifiers with *FLEXFIS-Class MM* on CD imprint data. In these figures, the x-axis represents the number of training samples already sent into the update algorithm of the evolving fuzzy classifiers (which is the training samples for initial off-line training + the on-line samples already seen so far) and $f(x)$ the accuracy on the whole (separate) test data. The

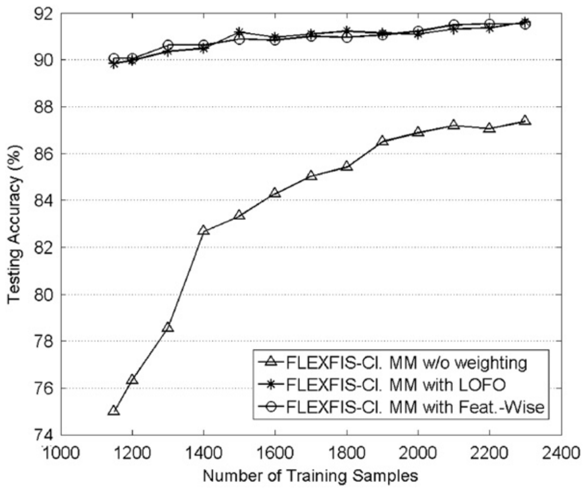

Fig. 6. Progress of accuracies on separate test data set during on-line evolution of fuzzy classifiers with *FLEXFIS-Class MM* for spam-base data set, line with triangle markers: without feature weighting, line with star markers: with leave-one-feature-out criterion and line with circle markers: with feature-wise separability criterion.

first figure (a) visualizes those obtained without applying any feature weighting strategy implicitly. The plots are starting after the initial batch training phase, where the accuracies of the classifiers are calculated for the first time (denoted by the first marker in the plots). Here, we can recognize that in fact the accuracy increases with more samples included into the incremental update; however, this increase is not as stable as in case when applying leave-one-feature-out feature weighing (line with star markers) or dimension-wise feature weighting (line with circle marker), as showing some down trends, especially for the first part of on-line samples (up to Sample #800). Furthermore, the levels of accuracy are higher in both incremental feature weighting variants, especially at the beginning of the whole update process, i.e. from the 512th sample on — note that the first 512 samples are used for initial training and feature weights are also included there. The bigger discrepancy in predictive performance at the beginning is no surprise, as curse of dimensionality (here we have 74-dimensional problem) is more severe when having less samples included into the evolving fuzzy classifiers. When more and more samples are loaded into the update of the fuzzy classifiers, the curse of dimensionality effect is usually getting weaker, hence the impact of feature weighting (or selection) diminishes (as gap between accuracy improvement lines is closed), but is still present (about 2% in case of single-wise feature weighting at the end of the whole incremental learning phase).

Figs. 5 and 6 show similar behavior between not including feature weights and inclusion of features weights when *FLEXFIS-Class MM* is applied to egg inspection and the spam-base data set. For egg inspection without feature weighting approach again more instabilities (significant downtrends in accuracies) during the incremental learning phase are achieved; when including feature weights the improvement of accuracies over the number of samples appears more monotonic, especially towards the end of the learning phase (where the approach without feature weighting does hardly further improve). In case of eggs, the data available for initial training is much larger than in case of CD imprints (1447 versus 512), and the number of features in both data sets is lower than in CD imprints (19 versus 74). Therefore, the proportion 'training samples to dimensionality of the feature space' is more beneficial in case of egg data, compared to CD imprint data set. This means that the impact of feature weighting (and reduction when weights are approaching 0) can be expected to be larger in case of CD imprints. This is reflected in the discrepancy of the accuracies between feature weighting and no feature weighting, especially at the beginning of the incremental learning phase: in case of eggs the discrepancy is 2.3% resp. 4.2%, whereas in case of CD imprint data the discrepancy is more than 5% resp. more than 10% — these gaps can be a bit closed but again leaving about 1.5% discrepancy for egg and about 2% discrepancy for CD imprint data set at the end of the complete learning phase (note the different scales in both images). Also in case of spam-base data, the discrepancy (which is severe at the beginning of the incremental learning phase as well) cannot be fully closed (still having about 4% difference at the end). Also an interesting and positive observation is that the much faster feature-wise separability criterion is at least as good as the leave-one-feature-out approach in all data sets and for

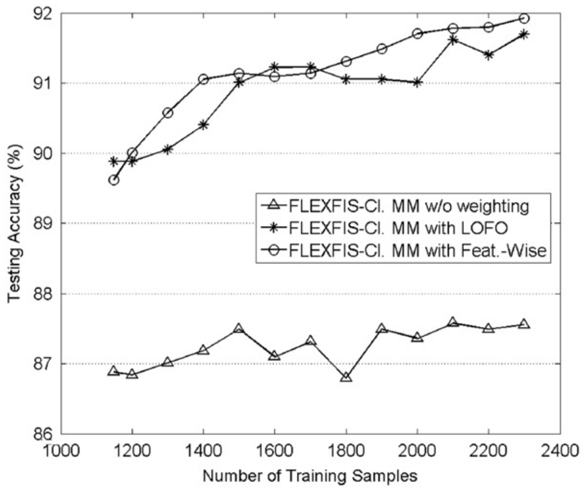

Fig. 7. Progress of accuracies on separate test data set during on-line evolution of fuzzy classifiers with *FLEXFIS-Class MM* for spam-base data set applied in reverse order, line with triangle markers: without feature weighting, line with star markers: with leave-one-feature-out criterion and line with circle markers: with feature-wise separability criterion.

both evolving fuzzy classification variants (in most cases it can even outperform the leave-one-feature-out approach, especially after the whole incremental training procedure is finished). This means the discriminatory power of single features alone are already sufficient in order to weight and select features appropriately and to increase accuracies of the evolving fuzzy classifiers.

For the spam-base data, we conducted an additional experiment by sending the training data (the pseudo-stream in this case) in reverse order into the incremental learning procedure. The intention of this is to elicit how much our incremental learning algorithm(s) are dependent on the order of the training data (note: from methodological point of view, there is a dependence according to *eVQ*, which is not order invariant, see [48]; the question remains how strong this effects the results). Fig. 7 visualizes the improvement line of the evolving fuzzy classifiers over the number of incrementally loaded samples when using the two weighting variants and when not using any weighting. Comparing with the results from Fig. 6 (original order), the results from the two weighting variants are almost equal (same behavior: starting with an initial accuracy of around 90% and improving this slightly to 92% throughout the incremental learning phase). For the non-weighting approach (Fig. 7 (a)), the situation is a bit different: obviously the initial data set (used for initial batch training) matches the separate test data set (used for eliciting the accuracies) better than in the original order, as the accuracy after initial training is much higher. However, what comes out here better is the impact of the incremental feature weighting during incremental learning phase: the accuracy of the evolved classifier stays approximately the same and 2–3% below 90% during the whole incremental learning phase when not using any feature weighting.

A summary of the obtained accuracies with and without feature weights in both variants (*FLEXFIS-Class SM* and *FLEXFIS-Class MM*) is demonstrated in Table 3. The first entry before the slash denotes the accuracy after initial training, the second entry the final accuracy obtained after the whole incremental learning process. From this table, we can also recognize that *FLEXFIS-Class MM* outperforms *FLEXFIS-Class SM* on all data sets, no matter whether and which feature weighting approach is included in the learning mechanisms. Furthermore, the table is split into two parts: the first part shows the accuracies obtained on the original order of the data, which in case of egg and CD imprint data sets is the true order as on-line recorded during production phase; the second part shows the mean accuracies obtained from 10 different shuffles of the data (not representing the natural on-line order but used for verification purposes). It is quite obvious that the first values (accuracies after initial training) for egg and CD imprint data sets are tendentially higher than in the non shuffle case, as images appearing at the end of the production process are already included in training in earlier phases. However, evolving fuzzy classifiers in connection with feature weighting still outperform those which do not include any weighting.

We performed a comparison between incrementally calculated and batch re-calculated feature weights during the on-line adaptation phase (simulated by using the second half of the training data in sample-wise manner). Fig. 8 shows

Table 3
Accuracies achieved on separate on-line test data set when applying feature weighting criteria and when applying conventional evolving fuzzy classifiers (without feature weighting), the entries before the slashes denote the accuracies after initial training, the entries after the slashes the accuracies after the whole incremental learning process; the first part denotes the accuracies obtained on the original order of the data, the second part shows the mean accuracies obtained aver 10 different shuffles of the data before the whole training process starts.

Methods	CD imprint	Eggs	Spam-base
Original on-line order			
FLEXFIS-Cl. SM	70.15/78.82	84.98/87.71	60.92/82.50
FLEXFIS-Cl. SM with LOFO	73.44/82.12	88.31/89.01	60.92/86.05
FLEXFIS-Cl. SM with feat.-wise	76.22/83.64	89.01/89.01	86.68/87.12
FLEXFIS-Cl. MM	74.23/90.31	90.55/94.62	75.10/87.55
FLEXFIS-Cl. MM with LOFO	87.12/89.91	93.12/96.00	89.75/91.87
FLEXFIS-Cl. MM with feat.-wise	79.55/91.93	95.01/96.41	90.18/91.87
Shuffled orders			
FLEXFIS-Cl. SM	78.82/78.82	87.71/88.01	64.88/82.50
FLEXFIS-Cl. SM with LOFO	76.22/83.23	87.71/90.11	65.22/85.72
FLEXFIS-Cl. SM with feat.-wise	78.82/83.64	88.85/91.55	85.55/88.23
FLEXFIS-Cl. MM	86.06/89.43	90.60/94.12	77.72/87.12
FLEXFIS-Cl. MM with LOFO	87.12/90.91	93.42/96.55	89.88/92.25
FLEXFIS-Cl. MM with feat.-wise	86.06/92.05	95.11/97.22	90.78/93.10

the evolution of two feature weights over time obtained by leave-one-feature-out variant for both cases: the solid lines with circle markers are denoting the incremental case, the solid lines with cross markers the batch (re-calculated) case. Obviously, for all three data sets (Figs. 8(a)–(c)), there is negligible differences between the two cases for both features (curves are lying almost exactly over each other, also compare the cross marker often included in the circle markers). In order to underline and confirm this situation, we calculated the mean absolute error (MAE) between re-calculated and incremental feature weights for all features. In fact, we calculate

$$MAE_{overall} = \frac{1}{p}\sum_{j=1}^{p}\left(\frac{1}{N}\sum_{i=1}^{N}|\lambda_{j;i}(recalc) - \lambda_{j;i}(inc)|\right) \tag{29}$$

with N the number of samples used for on-line adaptation and update phase (second half of training data set, see Table 2), p the dimensionality of the feature space, $\lambda_{j;i}(recalc)$ the jth feature weight re-calculated using the first i on-line samples and $\lambda_{j;i}(inc)$ the jth feature weight incrementally updated using the first i on-line samples. Furthermore, in order to observe the worst case scenario, we calculate the maximal MAE over all features by

$$MAE_{worst} = \max_{j=1,\ldots,p}\left(\frac{1}{N}\sum_{i=1}^{N}|\lambda_{j;i}(recalc) - \lambda_{j;i}(inc)|\right) \tag{30}$$

The values of both measures are reported in Table 4 for the three data sets and both feature weighting variants. From this table, someone can easily recognize that the deviations between re-calculated and incrementally calculated feature weights are negligible small (in large parts below 1%, only in case of CD imprint data in combination with leave-one-feature-out, the worst case is above 1%, note that the range of MAE is [0,1]). From this examination, we can also conclude that the performance of the evolving fuzzy classifiers would be (almost) the same either when using off-line or on-line feature weights.

Another interesting issue when doing on-line learning is the computational time needed during the adaptation phase. Therefore, in Table 5, we report computation times on all data sets needed for the two incremental feature weighting approaches and the computation time needed for *FLEXFIS-Class MM* resp. *FLEXFIS-Class SM*, the first number denotes the required computation time in sum (in seconds), the second number for one single sample. From this table, someone can realize that processing one single sample through either *FLEXFIS-Class SM* or *FLEXFIS-Class MM* and through one of the two feature weighting approaches, the maximal time required is 0.185 s (*FLEXFIS-Class MM*

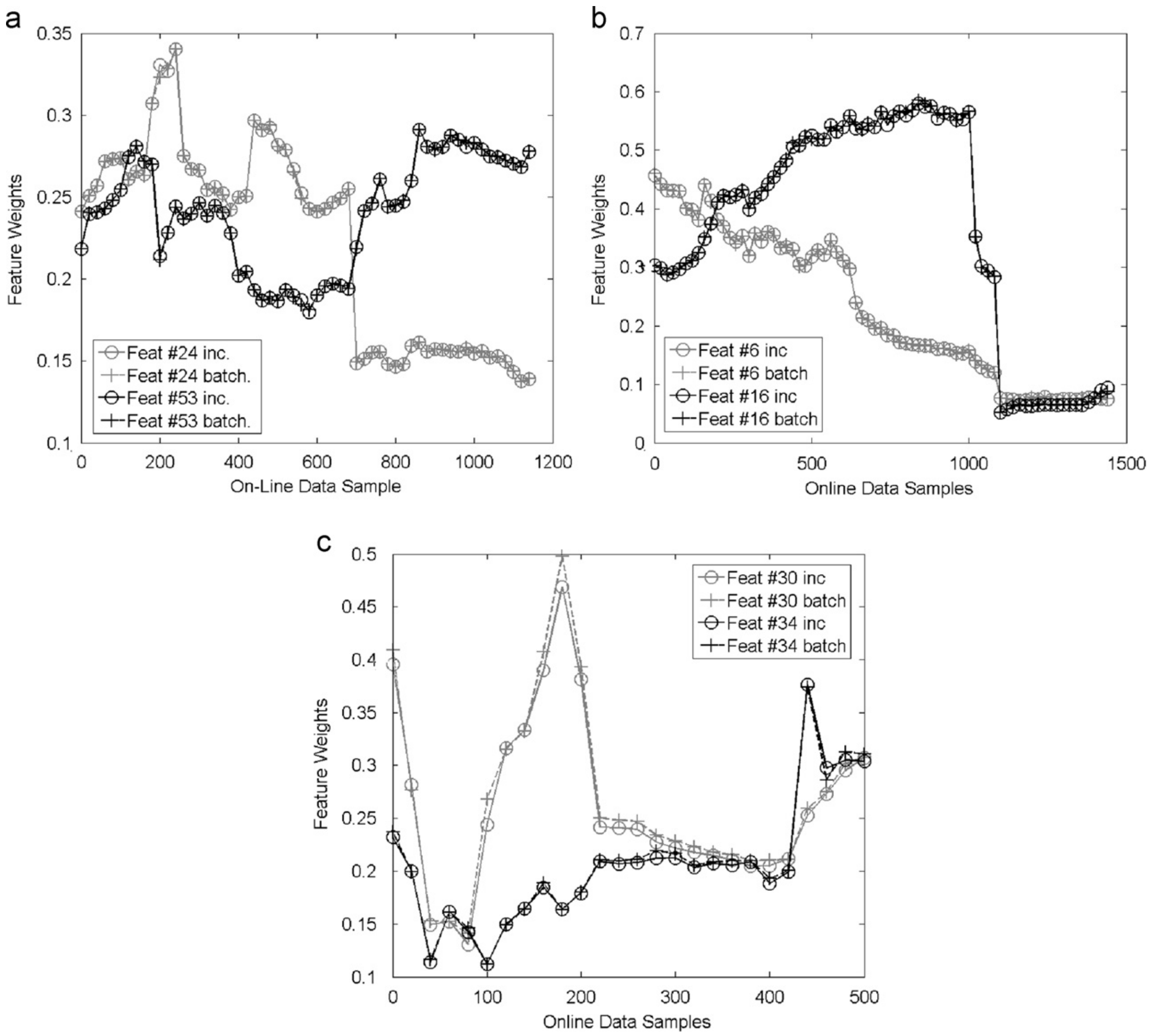

Fig. 8. Comparison of the evolution of feature weights for two features during the (on-line) adaptation phase; solid lines with circle markers denote the incrementally calculated feature weights, lines with cross markers the re-calculated feature weights; in all three data set cases (a)–(c) the deviation is negligible.

with leave-one criterion on CD imprint data set), hence being able to cope with real-time demands where samples are loaded with a frequency of 0.185 s. In case when using feature-wise weighting criterion (as in most cases yielding better accuracies than leave-one-feature-out approach), the worst-case performance is 0.03 s (for CD imprint data when using *FLEXFIS-Class MM*). This means that we are able to cope with very fast on-line processes in the area of milli-seconds.

Finally, as we claimed that our approach also performs a dynamic soft dimension reduction, we observed the feature weights finally obtained after the complete training when using feature-wise weighting criterion. Table 6 shows the number of feature weights whose values are > 0.1 compared to the number of features in the data set (second row); the third row shows the number of highly active features (with weights > 0.8). Someone can recognize that most of the features were assigned a weight below 0.1, hence are more or less switched off as not having any impact in the training and classification phases (in fact, no or very little contribution to cluster movements or distance calculations). For the egg data set, it is interesting to see that the four most active (therefore most important) features are describing the total area of the objects in an image, the size of the largest objects and information about the intensities of the objects. This

Table 4
Deviation in terms of mean absolute error between incrementally calculated and batch re-calculated feature weights for the three data sets and the two feature weighting variants; the entries before the slashes represent the values for the mean MAE over all features, the values after the slashes for the maximal MAE over all features (worst case).

Data set	Feature-wise	Leave-one-feature-out
Spam-base	9.75e−05/0.00042	0.000284/0.0011
Eggs	3.14e−05/2.11e−04	0.000849/0.0035
CD-imprint	5.36e−04/0.0025	0.005/0.0296

Table 5
Computation times in seconds of feature weighting approaches during incremental on-line phase, the entries before the slashes indicating the times for the whole on-line, the entries after the slashes for each single sample.

Methods	CD imprint	Eggs	Spam-base
Leave-one-feature-out	79.34/0.155	6.94/0.0048	64.35/0.056
Feature-wise	0.127/0.0002	0.162/0.00011	0.281/0.00024
FLEXFIS-class SM conv.	0.1563/0.00011	0.25/0.00017	0.2344/0.00016
FLEXFIS-class MM conv.	15.33/0.03	7.25/0.005	9.516/0.0083

Table 6
Number of features having a higher weight than 0.05 (second row) and than 0.8 (third row), compared to the number of features in the data sets (after the slashes).

Weight threshold	CD imprint	Eggs	Spam-base
0.05	29/74	7/19	30/57
0.8	5/74	4/19	3/57

may give some additional interpretation which features are essential for discriminating between yolk (faults) and dirt (no faults) on eggs.

7. Conclusion

In this paper, we proposed a concept for a dynamic and smooth integration of dimensionality reduction into evolving fuzzy classifiers in form of an incremental feature weighting algorithm. For feature weighting, we applied two criteria, one based on a leave-one-feature-out approach, the other based on a feature-wise (single dimension) approach, both exploiting the concept of separability criterion of features (jointly or independently) and both incrementally calculable during on-line phase in a robust manner (in the sense that they converge to the batch version). The feature weights are integrated into the classification phase when using single as well as multi-model fuzzy classifier architecture and into the learning phase when using *FLEXFIS-Class SM* as well as *FLEXFIS-Class MM* as evolving fuzzy classifier approaches. Based on the observations when applying evolving fuzzy classifiers to on-line classification scenarios, we can conclude that our incremental feature weighting approaches are able to guide evolving fuzzy classifiers to more predictive power during on-line operation mode than when feature weighting is not included in the training process. Thereby, both approaches perform approximately equally well in terms of predictive accuracy (for different data orders), whereas in terms of computation speed, the feature-wise approach outperforms the leave-one-feature-out approach significantly. Another important examination was that a high percentage of features were assigned a low weight, contributing definitively to the reduction of dimensionality. Future work includes the investigation and development of a non-linear feature weighting technique, ideally more closely coupled with the classifiers' architecture and training procedures. This may bring further improvement in terms of curse of dimensionality reduction and therefore accuracy of the classifiers, as currently some features which are not seen as important by our linear criterion may become important when performing a more combined non-linear criterion.

Acknowledgements

This work was funded partially by the Upper Austrian Technology and Research Promotion and partially by the Austrian fund for promoting scientific research (FWF, Contract number I328-N23, acronym IREFS). It reflects only the authors' views.

References

[1] H. Maturino-Lozoya, D. Munoz-Rodriguez, F. Jaimes-Romera, H. Tawfik, Handoff algorithms based on fuzzy classifiers, IEEE Transactions on Vehicular Technology 49 (6) (2000) 2286–2294.

[2] V. Agostini, G. Balestra, M. Norese, Fuzzy classifier based on muscle fatigue parameters, in: Proceedings of the 27th Annual International Conference of the Engineering in Medicine and Biology Society, 2005 (IEEE-EMBS 2005), 2006, pp. 2421–2424.

[3] D.P. Filev, F. Tseng, Novelty detection based machine health prognostics, in: Proceedings of the 2006 International Symposium on Evolving Fuzzy Systems, Lake District, UK, 2006, pp. 193–199.

[4] C. Xydeas, P. Angelov, S. Chiao, M. Reoulas, Advances in eeg signals classification via dependant hmm models and evolving fuzzy classifiers, International Journal on Computers in Biology and Medicine 36 (10) (2006) 1064–1083 (special issue on Intelligent Technologies for Bio-informatics and Medicine).

[5] D. Sannen, M. Nuttin, J. Smith, M. Tahir, E. Lughofer, C. Eitzinger, An interactive self-adaptive on-line image classification framework, in: A. Gasteratos, M. Vincze, J. Tsotsos (Eds.), Proceedings of ICVS 2008, Lecture Notes in Computer Science, vol. 5008, Springer, Santorini Island, Greece, 2008, pp. 173–180.

[6] R. Santos, E. Dougherty, J.A. Jaakko, Creating fuzzy rules for image classification using biased data clustering, in: SPIE Proceedings Series (SPIE Proceedings Series) International Society for Optical Engineering Proceedings Series, Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, 1999, pp. 151–159.

[7] Y.Y.T. Nakashima, G. Schaefer, H. Ishibuchi, A weighted fuzzy classifier and its application to image processing tasks, Fuzzy Sets and Systems 158 (3) (2006) 284–294.

[8] L. Kuncheva, Fuzzy Classifier Design, Physica-Verlag, Heidelberg, 2000.

[9] J. Roubos, M. Setnes, J. Abonyi, Learning fuzzy classification rules from data, Information Sciences—Informatics and Computer Science: An International Journal 150 (2003) 77–93.

[10] H. Ishibushi, K. Nozaki, N. Yamamoto, H. Tanaka, Selecting fuzzy IF–THEN rules for classification problems using genetic algorithms, IEEE Transactions on Fuzzy Systems 3 (3) (1995) 260–270.

[11] D. Nauck, R. Kruse, A neuro-fuzzy method to learn fuzzy classification rules from data, Fuzzy Sets and Systems 89 (3) (1997) 277–288.

[12] J. Hühn, E. Hüllermeier, FR3: a fuzzy rule learner for inducing reliable classifiers, IEEE Transactions on Fuzzy Systems 17 (1) (2009) 138–149.

[13] O. Cordon, M. del Jesus, F. Herrera, A proposal on reasoning methods in fuzzy rule-based classification systems, International Journal of Approximate Reasoning 20 (1) (1999) 21–45.

[14] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer Verlag, New York, Berlin, Heidelberg, Germany, 2001.

[15] R. Duda, P. Hart, D. Stork, Pattern Classification, second ed., Wiley-Interscience, Southern Gate, Chichester, West Sussex PO 19 8SQ, England, 2000.

[16] L. Sanchez, M. Suarez, J. Villar, I. Couso, Mutual information-based feature selection and partition design in fuzzy rule-based classifiers from vague data, International Journal of Approximate Reasoning 49 (2008) 607–622.

[17] L. Mikenina, H. Zimmermann, Improved feature selection and classification by the 2-additive fuzzy measure, Fuzzy Sets and Systems 107 (1999) 197–218.

[18] S. Cho, J. Kim, Combining multiple neural networks by fuzzy integral for robust classification, IEEE Transactions on Systems, Man, and Cybernetics 25 (2) (1995) 380–384.

[19] H. Lee, C. Chen, J. Chen, Y. Jou, An efficient fuzzy classifier with feature selection based on fuzzy entropy, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 31 (3) (2001) 426–432.

[20] Z. Mao-yuan, L. Zheng-ding, A fuzzy classification based on feature selection for web pages, in: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence 2004, 2004, pp. 469–472.

[21] J. Casillas, O. Cordon, M.D. Jesus, F. Herrera, Genetic feature selection in a fuzzy rule-based classification system learning process for high-dimensional problems, Information Sciences 136 (2001) 135–157.

[22] O. Cordon, F. Herrera, M.D. Jesus, P. Villar, A multiobjective genetic algorithm for feature selection and granularity learning in fuzzy-rule based classification systems, in: Proceedings of the IFSA World Congress and 20th NAFIPS International Conference 2001, 2001.

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

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

[25] E. Lughofer, P. Angelov, X. Zhou, Evolving single- and multi-model fuzzy classifiers with FLEXFIS-Class, in: Proceedings of FUZZ-IEEE 2007, London, UK, 2007, pp. 363–368.

[26] P. Angelov, X. Zhou, Evolving fuzzy-rule-based classifiers from data streams, IEEE Transactions on Fuzzy Systems 16 (6) (2008) 1462–1475.

[27] A. Bouchachia, Incremental induction of classification fuzzy rules, in: IEEE Workshop on Evolving and Self-Developing Intelligent Systems (ESDIS) 2009, 2009, pp. 32–39.

[28] M. Mouchaweh, A. Devillez, G. Lecolier, P. Billaudel, Incremental learning in fuzzy pattern matching, Fuzzy Sets and Systems 132 (2002) 49–62.

[29] D. Dubois, H. Prade, C. Testemale, Weighted fuzzy pattern matching, Fuzzy Sets and Systems 28 (1988) 313–331.

[30] D. Dubois, H. Prade, S. Sandri, On possibility/probability transformations, in: R. Lowen, M. Roubens (Eds.), Fuzzy Logic: State of the Art, Kluwer Academic Publishers, Dordrecht, 1993, pp. 103–112.

[31] Y. Li, On incremental and robust subspace learning, Pattern Recognition 37 (2004) 1509–1518.

[32] J. Ye, Q. Li, H. Xiong, H. Park, R. Janardan, V. Kumar, IDR, QR: an incremental dimension reduction algorithms via QR decomposition, IEEE Transactions on Knowledge and Data Engineering 17 (9) (2005) 1208–1222.

[33] N. Kasabov, D. Zhang, P. Pang, Incremental learning in autonomous systems: evolving connectionist systems for on-line image and speech recognition, in: Proceedings of IEEE Workshop on Advanced Robotics and its Social Impacts, 2005, Hsinchu, Taiwan, 2005, pp. 120–125.

[34] K. Lee, W. Street, Incremental feature weight learning and its application to a shape-based query system, Pattern Recognition Letters 23 (2002) 865–874.

[35] H. Kim, J. Chang, Integrating incremental feature weighting into naive Bayes text classifier, in: Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, IEEE Press, 2007, pp. 1137–1143.

[36] U. Hahn, M. Klenner, Incremental concept evolution based on adaptive feature weighting, in: E. Coasta, A. Cardoso (Eds.), Progress in Artificial Intelligence, Lecture Notes in Computer Science, vol. 1323, Springer, Berlin, Heidelberg, 1997, pp. 49–60.

[37] J. Dy, C. Brodley, Feature selection for unsupervised learning, Journal of Machine Learning Research 5 (2004) 845–889.

[38] I. Guyon, A. Elisseeff, An introduction to variable and feature selection, Journal of Machine Learning Research 3 (2003) 1157–1182.

[39] I. Guyon, S. Gunn, M. Nikravesh, L. Zadeh, Feature Extraction. Foundations and Applications, Springer Verlag, Berlin, Heidelberg, 2006.

[40] K. Fukunaga, Statistical Pattern Recognition, second ed., Academic Press, San Diego, CA, 1990.

[41] S. Pang, S. Ozawa, N. Kasabov, Incremental linear discriminant analysis for classification of data streams, IEEE Transactions on Systems, Man and Cybernetics—Part B: Cybernetics 35 (5) (2005) 905–914.

[42] S. Qin, W. Li, H. Yue, Recursive PCA for adaptive process monitoring, Journal of Process Control 10 (2000) 471–486.

[43] C. Eitzinger, M. Gmainer, W. Heidl, E. Lughofer, Increasing classification performance with adaptive features, in: A. Gasteratos, M. Vincze, J. Tsotsos (Eds.), Proceedings of ICVS 2008, Lecture Notes in Computer Science, vol. 5008, Springer, Santorini Island, Greece, 2008, pp. 445–453.

[44] E. Lughofer, S. Kindermann, Improving the robustness of data-driven fuzzy systems with regularization, in: Proceedings of the IEEE World Congress on Computational Intelligence (WCCI) 2008, Hongkong, 2008, pp. 703–709.

[45] W. Groißböck, E. Lughofer, E. Klement, A comparison of variable selection methods with the main focus on orthogonalization, in: M. Lopéz-Díaz, M. Gil, P. Grzegorzewski, O. Hryniewicz, J. Lawry (Eds.), Soft Methodology and Random Information Systems, Advances in Soft Computing, Springer, Berlin, Heidelberg, New York, 2004, pp. 479–486.

[46] D. Nauck, U. Nauck, R. Kruse, Generating classification rules with the neuro-fuzzy system NEFCLASS, in: Proceedings of the Biennial Conference of the North American Fuzzy Information Processing Society NAFIPS'96, Berkeley, 1996.

[47] E. Klement, R. Mesiar, E. Pap, Triangular Norms, Kluwer Academic Publishers, Dordrecht, Norwell, New York, London, 2000.

[48] E. Lughofer, Extensions of vector quantization for incremental clustering, Pattern Recognition 41 (3) (2008) 995–1011.

[49] T. Takagi, M. Sugeno, Fuzzy identification of systems and its applications to modeling and control, IEEE Transactions on Systems, Man and Cybernetics 15 (1) (1985) 116–132.

[50] N. Draper, H. Smith, Applied Regression Analysis. Probability and Mathematical Statistics, John Wiley & Sons, New York, 1981.

[51] L. Wang, J. Mendel, Fuzzy basis functions, universal approximation and orthogonal least-squares learning, IEEE Transactions on Neural Networks 3 (5) (1992) 807–814.

[52] E. Lughofer, On-line evolving image classifiers and their application to surface inspection, Image and Vision Computing 28 (7) (2010) 1063–1172.

[53] E. Lughofer, FLEXFIS: a robust incremental learning approach for evolving TS fuzzy models, IEEE Transactions on Fuzzy Systems 16 (6) (2008) 1393–1410.

[54] S. Raiser, E. Lughofer, C. Eitzinger, J. Smith, Impact of object extraction methods on classification performance in surface inspection systems, Machine Vision and Applications 21 (5) (2010) 627–641.

[55] C. Eitzinger, W. Heidl, E. Lughofer, S. Raiser, J. Smith, M. Tahir, D. Sannen, H. van Brussel, Assessment of the influence of adaptive components in trainable surface inspection systems, Machine Vision and Applications 21 (5) (2010) 613–626.