Evolving Single- And Multi-Model Fuzzy Classifiers with FLEXFIS-Class

Edwin Lughofer, Plamen Angelov and Xiaowei Zhou

Abstract— **In this paper a new method for training single-model and multi-model fuzzy classifiers incrementally and adaptively is proposed, which is called *FLEXFIS-Class*. The evolving scheme for the single-model case exploits a conventional zero-order fuzzy classification model architecture with Gaussian fuzzy sets in the rules antecedents, crisp class labels in the rule consequents and rule weights standing for confidence values in the class labels. In the multi-model case *FLEXFIS-Class* exploits the idea of regression by an indicator matrix to evolve a Takagi-Sugeno fuzzy model for each separate class and combines the single models' predictions to a final classification statement. The paper includes a technique for increasing the prediction quality, whenever a drift in a data stream occurs. An empirical analysis will be given based on an online, adaptive image classification framework, where images showing production items should be classified into good or bad ones. This analysis will include the comparison of evolving single- and multi-model fuzzy classifiers with conventional batch modelling approaches with respect to achieved prediction accuracy on new online data. It will also be shown that multi-model architecture can outperform conventional single-model architecture ('classical' fuzzy classification models) for all data sets with respect to prediction accuracy.**

Index Terms— **evolving fuzzy classifiers, single- and multi-model architecture, incremental training, regression by indicator matrix, process safety, data drift, image classification framework**

I. INTRODUCTION

In the contemporary industrial systems data-driven fuzzy classifiers are applied for decision making [1], fault detection or image classification tasks [2] quite often. The reason is not only a high predictive quality [3], which can be achieved when applying these types of classifiers, but also the good transparency in the form of linguistic rules that shows understandable dependencies between the features [4]. This is an essential point e.g. for finding reasons of a faulty system state. Furthermore, confidence values for models' decisions can be calculated easily based on fulfillment degrees of the rules. These confidence values usually support the operators, as they point out the trustworthiness of the classifier's decisions. In case when the data is recorded online with a high frequency, the training of fuzzy classifiers has to be carried out in an incremental and evolving manner on a sample-per-sample basis, because a complete rebuilding of the classifier with all the recorded data contradicts with the online demands. Moreover, some operators may overrule a decision of the classifier during online operation. Another reason that requires incremental and evolving learning techniques is the usually huge size of the data bases.

A number of papers exist that consider fuzzy rule-based classifiers [5] [6]. To our best knowledge, the evolving fuzzy and neuro-fuzzy modelling techniques such as *SAFIS* [7], *DENFIS* [8], *FLEXFIS* [9] etc. has not yet been applied to classification. The only incremental and evolving fuzzy approaches for classification that we are aware of are [10] and [11]. The first one introduces the *eClass* method which applies all-in-one fuzzy rule-base which contains sub-rule-bases per class and uses zero order (singleton) consequents. This concept is taken further in [11] where a first order multi-input-multi-output (MIMO) Takagi-Sugeno non-linear model is used.

In this paper a different approach is proposed, *FLEXFIS-Class*, which uses both, multi-model architecture based on the idea of regression by an indicator matrix (*FLEXFIS-Class MM*) and single-model architecture ('classical' fuzzy classification models) (*FLEXFIS-Class SM*). Both variants of *FLEXFIS-Class* are basically deduced from *FLEXFIS* [9], which serves as an evolving method for building up fuzzy regression models fully automatically and adaptively with new incoming data points (from measurement signals, data streams etc.). For both model architectures, the evolving mechanism for the rule antecedent parts takes place in the clusters space by using an evolving version of vector quantization [12] (the original VQ in [13]), including cluster evolution, an alternative winner selection strategy and updates of cluster surfaces synchronously to their centres. In the single-model case the evolution of the consequents (=single class labels) and rule weights is based on a plurality choice and relative frequency of classes in the different clusters (rules), which both can be updated sample-wise (Section II). In the multi-model case one Takagi-Sugeno fuzzy regression model [14] is trained for each separate class and their continuous outputs are aggregated to an overall classification statements (Section III). The incremental training of the consequents in the TS models (hyper-planes) is carried out by exploiting recursive weighted least squares [15] (also applied for consequent adaptation in [16]). Improving the fuzzy classifiers towards approximation accuracy (dealing with drifts in online data streams [17]) is described in Section IV. The paper is concluded in Section V with an evaluation of the proposed approaches within an online adaptive image classification framework and based on a pen-digit recognition data set from the UCI-repository. This evaluation includes a comparison of the impact of the two model architectures on prediction accuracy and model

Edwin Lughofer is with the Department of Knowledge-based Mathematical Systems, Johannes Kepler University of Linz, A-4040 Linz, Austria (e-mail: edwin.lughofer@jku.at)

Plamen Angelov and Xiaowei Zhou are with Department of Communication Systems, Infolab21, Lancaster University, Lancaster, LA1 4WA, United Kingdom (e-mail: p.angelov@lancaster.ac.uk and x.zhou3@lancaster.ac.uk)

1-4244-1210-2/07/$25.00 ©2007 IEEE.

complexity as well as a comparison with other well-known batch modelling methods.

II. FLEXFIS-Class Using (Classical) Single-Model Architecture

For the single-model architecture case we exploit the 'classical' architecture for fuzzy classifiers [5] [6], where the ith rule is defined in the following way:

$$\text{Rule}_i: \quad \text{IF } x_1 \text{ IS } \mu_{i1} \text{ AND...AND } x_p \text{ IS } \mu_{ip} \text{ THEN } y_i = c_i \tag{1}$$

where p is the dimensionality of the input space, $\mu_{i1},...,\mu_{ip}$ are the p antecedent fuzzy sets and c_i is the crisp output class label from the set $\{1,...,K\}$ with K the number of classes. Now, for each new incoming sample $\vec{x}$, the final crisp class label is obtained by taking the class label of that rule with the highest activation degree, i.e. by calculating

$$y = c_{i^*} \text{ with } i^* = \text{argmax}_{1 \leq i \leq C}\ \mu_i \tag{2}$$

with C the number of rules and μ_i the activation degree of rule i defined by (using product t-norm):

$$\mu_i = \prod_{i=1}^{p} \mu_{ij}(x_j)$$

Furthermore, we introduce K rule weights $w_{i1},...,w_{iK}$, which denote the confidence of the ith rule in all the K classes, whereas the highest weight corresponds to the output class c_i. Note that confidence values are of great help for a better interpretation of the model's final output and quite often demanded by operators in an industrial system, especially in case of incorrect model's feedback. Based on the rule weights an overall confidence $conf$ of the overall output class label $y = k \in \{1,...,K\}$ is calculated through the following weighted average:

$$conf = \frac{\sum_{i=1}^{C} w_{ik}\mu_i}{\sum_{i=1}^{C} \mu_i} \tag{3}$$

Note that in case of a two-class classification task, the single-model architecture can be made even slimer, as the rule weights can be directly encoded in the consequent part of the rules, serving as continuous output values between 0 = class#1 and 1 = class#2. If for example a rule has consequent value of 0.6, this means that it is confident in class#2 with a degree of 0.6 and in class#1 with a degree of $1 - 0.6 = 0.4$. Consequently, if the value of the rule with highest activation degree is greater or equal 0.5 the current data sample belongs to class#2, otherwise to class#1.

The adaptive and evolving training procedure for the rules' antecedent parts is done with the help of an incremental and evolving clustering procedure that is modified, evolving version of VQ [12] (eVQ), whose algorithm is summarised in the flowchart of Figure 1 (assuming normalized features). After each new incoming sample the updated/generated cluster in the high-dimensional data space is projected onto the axes to form one-dimensional Gaussian fuzzy sets, i.e. $\mu_{ij}(x_j) = e^{-\frac{1}{2}\frac{(x_j - c_{ij})^2}{\sigma_{ij}^2}}$. The fuzzy sets projected from one cluster on each axis form the antecedent part of one rule. In this sense, antecedent parts are permanently updated and new rules and fuzzy sets are born, whenever a new cluster is born according to Step 7a in Figure 1. Please note that the class labels are included during the whole learning procedure. The crisp and unique class label in the consequent part of the ith rule is elicited by counting the relative proportions of all the samples which formed the corresponding cluster to all the K classes and taking the maximal value of these proportions:

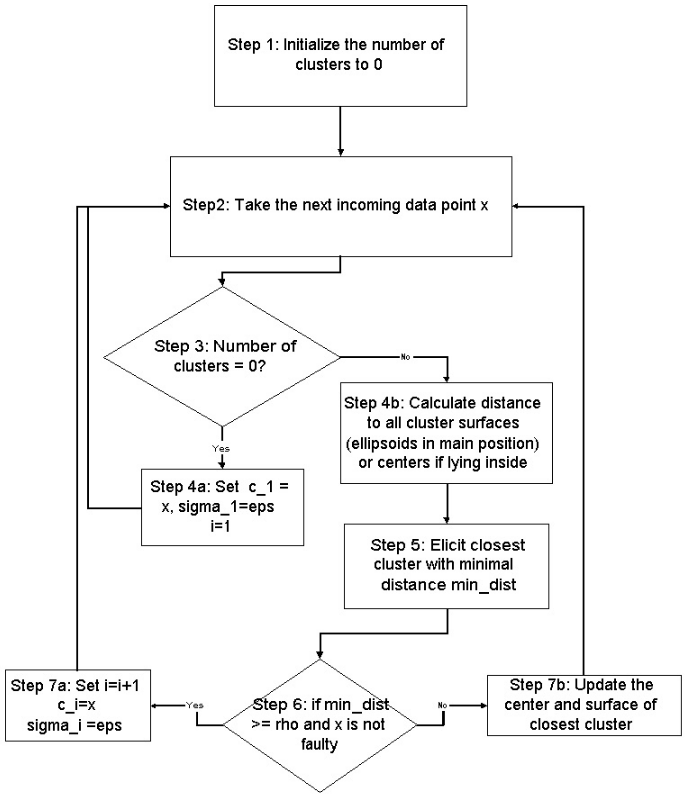
Step 1: Initialize the number of clusters to 0
Step2: Take the next incoming data point x
Step 3: Number of clusters = 0?
No
Yes
Step 4a: Set c_1 = x, sigma_1=eps i=1
Step 4b: Calculate distance to all cluster surfaces (ellipsoids in main position) or centers if lying inside
Step 5: Elicit closest cluster with minimal distance min_dist
Step 6: if min_dist >= rho and x is not faulty
Step 7a: Set i=i+1 c_i=x sigma_i =eps
Step 7b: Update the center and surface of closest cluster

Fig. 1. Workflow of evolving version of vector quantization [12] (assuming normalized features)

$$c_i = \max_{k=1}^{K}(w_{ik}) = \max_{k=1}^{K}(\frac{n_{ik}}{n_i}) \tag{4}$$

These values can be updated by counting n_i, the number of samples, which formed the antecedent part of the ith rule and n_{ik} the number of samples forming the antecedent part of the ith rule and falling into class k. The K rule weights $w_{i1},...,w_{iK}$ are also obtained through (4). Combining evolving antecedent and consequent learning yields *FLEXFIS-Class* with single-model architecture, *FLEXFIS-Class SM*:

Algorithm 1: **FLEXFIS-CLASS SM**

1) Collect N data points sufficient for the actual dimensionality of the problem, estimate the ranges of all features from these N points.
2) Generate an initial fuzzy classifier with these N points by applying eVQ as in Fig. 1 (with features normalized by their ranges) and eliciting the consequent labels as in (4) for all clusters (=rules).
3) Take the next incoming sample $\vec{x}$, elicit its class label, say k; either the data is pre-labelled or it can be for

instance labelled by calculating (2) when using all the fuzzy rules obtained so far and obtaining feedback from an operator if the class output is correct.
4) Proceed through evolving VQ as demonstrated in Figure 1 from Step 3 on without going back to Step 2 (with features normalized by their ranges).
5) Project updated or newly generated cluster onto the input feature axes, forming the complete antecedent part of the corresponding rule. Hereby, the jth dimension of the cluster prototype and width is assigned to the center and width of the Gaussian fuzzy set in the jth premise part.
6) **If** a new rule is born, set $n_i = 1$ and $n_{ik} = 1$.
7) **Else** Update the corresponding consequent label and confidence values in all K classes by setting $n_i = n_i + 1$ and $n_{ik} = n_{ik} + 1$ and using (4).
8) Update ranges of features
9) If new incoming data points are still available then goto Step 1; otherwise $stop$.

III. FLEXFIS-Class Using Multi-Model Architecture

FLEXFIS-Class MM is based on multiple Takagi-Sugeno fuzzy regression models [14] using a K indicator matrices for the K classes which are generated from the original classification matrix, containing different features in different columns and a label entry in the last column. This is done for the ith matrix by setting each label entry to 1, if the corresponding row belongs to the ith class, otherwise it is set to 0. The feature columns remain the same for all indicator matrices. In this sense, it is guaranteed that the regression surface is forced towards 1 in the region of those samples belonging to its corresponding class and otherwise it approaches 0.

The overall output from the fuzzy classifier is calculated by inferencing a new multi-dimensional sample $\vec{x}$ through the K Takagi-Sugeno models (fuzzy basis function networks [18], as a product t-norm in connection with Gaussian fuzzy sets):

$$\hat{f}_m(\vec{x}) = \sum_{i=1}^{C} l_i \Psi_i(\vec{x}) \qquad m = 1, ..., K \tag{5}$$

with the normalized membership functions

$$\Psi_i(\vec{x}) = \frac{e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j - c_{ij})^2}{\sigma_{ij}^2}}}{\sum_{k=1}^{C} e^{-\frac{1}{2}\sum_{j=1}^{p}\frac{(x_j - c_{kj})^2}{\sigma_{kj}^2}}} \tag{6}$$

and consequent functions

$$l_i = w_{i0} + w_{i1}x_1 + w_{i2}x_2 + ... + w_{ip}x_p \tag{7}$$

and then eliciting that model producing the maximal output and taking the corresponding class as label response, hence

$$y = class(\vec{x}) = \text{argmax}_{m=1,...,K}\hat{f}_m(\vec{x}) \tag{8}$$

In [19] it is maintained that regression by an indicator matrix only works well on two-class problems (0/1) and can have problems with multi-class problems, as then a complete masking of one class by two or more others may happen. However, opposed to linear regression by an indicator matrix [20], the TS fuzzy models are non-linear, i.e. not only the (locally linear) rule consequents but also the non-linear antecedent parts are trained based on the indicator matrix information, i.e. by taking the label column as the (regressing) target variable. In this sense, the approximation behaviour is non-linear which forces the surface of a model f_k going to 0 more rapidly in regions apart from class k samples as in the case of inflexible linear hyper-planes, therefore the masking is much weaker than in the pure linear case. In Section V-B an empirical verification of this point will be given based on a high-dimensional data set containing 10 classes, where the classification accuracy obtained by conventional batch modelling approaches (which do not suffer by the masking problem) can be approximated with *FLEXFIS-Class MM* quite well, whereas linear regression by an indicator matrix is far behind. An overall confidence value is elicited by normalizing the maximal output value by the sum of the output values from all K models.

Now the problem remains how to generate the K Takagi-Sugeno fuzzy models in incremental and evolving manner. This is done here with the original version of *FLEXFIS* [9], extended in [12] (*FLEXFIS-MOD*), compared with *eTS* in [21]. The basics of this algorithm are:

- It applies eVQ [12] for incremental clustering, see Figure 1
- The updated cluster centers and surfaces are projected onto the axes after each incremental learning step in order to adapt the fuzzy sets and rules.
- It uses recursive local learning by exploiting *recursive weighted least squares* [15], [16] for the linear consequent parameters.

The *FLEXFIS-Class MM* algorithm for building an evolving multi-model fuzzy classifier can be summarized in the following pseudo-code:

Algorithm 2: **FLEXFIS-CLASS MM**

1) Collect N data points sufficient for the actual dimensionality of the problem, estimate ranges of all features from the N points.
2) Generate K (initial) TS fuzzy models for K classes present with N data points by using batch mode *FLEXFIS* [9], i.e. eVQ as in Figure 1 (with normalized features by their ranges) and axis-projection first and least squares afterwards.
3) Take the next incoming data point $\vec{x}$; elicit its class label, say k; either the data is pre-labelled or it can be for instance labelled by calculating (2) when using all the fuzzy rules obtained so far and obtaining feedback from an operator if the class output is correct.
4) Update ranges of features
5) Update the kth TS fuzzy model by taking $y = 1$ as response (target) value and using *FLEXFIS(-MOD)* [9] [12]
6) Update all other TS fuzzy models by taking $y = 0$ as response (target) value and using *FLEXFIS(-MOD)* [9] [12]
7) If new incoming data points are still available then goto

Step 3; otherwise *stop*.

IV. Tracking Drift in the Data

More accuracy can be achieved for specific data streams, whenever a drift in the data appears after some time. For achieving a smooth tracking of a drift in the data, an exponential forgetting of older data points over time is applied. In *FLEXFIS-Class MM* this forgetting takes place exclusively in the linear consequent parameters of all the TS fuzzy models synchronously. This can be achieved by introducing a so-called forgetting factor λ and incorporating them into the *recursive weighted least squares* formula [22] (here for the ith rule of any model):

$$\hat{\vec{w}}_i(k+1) = \hat{\vec{w}}_i(k) + \gamma(k)(y(k+1) - \vec{r}^T(k+1)\hat{\vec{w}}_i(k)) \quad (9)$$

$$\gamma(k) = \frac{P_i(k)\vec{r}(k+1)}{\frac{\lambda}{\Psi_i(\vec{x}(k+1))} + \vec{r}^T(k+1)P_i(k)\vec{r}(k+1)} \quad (10)$$

$$P_i(k+1) = (I - \gamma(k)\vec{r}^T(k+1))P_i(k)\frac{1}{\lambda} \quad (11)$$

with $P_i(k) = (R_i(k)^T Q_i(k) R_i(k))^{-1}$ the inverse weighted Hesse matrix and $\vec{r}(k+1) = [1 \; x_1(k+1) \; x_2(k+1) \; \dots \; x_p(k+1)]^T$ the regressor values of the $(k+1)$th data point. Note that in the case of $\lambda = 1$ no forgetting takes place, while with decreasing λ the forgetting gets stronger and stronger. In this way, drifts in the trajectory of the TS fuzzy regression models can be tracked [21], which changes the overall output of the fuzzy classifier more rapidly. For recognizing a drift in the data the cluster ageing strategy as introduced in [23] and further developed in [24] can be exploited, as then slopes of the rule age curves tend to increase. A verification example for this point will be demonstrated in Section V-A.

When using single-model architecture the forgetting of the consequent class labels can be achieved by using a window with fixed size, over which n_i and n_{ik} are counted. Depending on the window size, the forgetting is faster or slower. However, this requires a batch storage of data samples in a buffer with the same size as the chosen window. A better alternative is to fix a value ϵ and add this for each counting cycle, i.e. after N samples belonging to the ith rule we get

$$n_i = \sum_{i=1}^{N}(1 + (i-1)\epsilon) \quad (12)$$

The same is carried out for n_{ik}. Depending on the size of ϵ older points are forgotten faster or slower. For instance, when setting $\epsilon = 1$ and the first three points belonging to the ith cluster (=rule) fall into class#1 and the next two into class#2, then we get $n_i = 1+2+3+4+5 = 15$ and $n_{i1} = 1+2+3 = 6$ while $n_{i2} = 4+5 = 9$, hence $c_i = 2$, as the last two data samples count more than the first three.

V. Application Examples

In this section an evaluation of the two evolving classification approaches, *FLEXFIS-Class SM* and *FLEXFIS-Class MM*, within two applications is demonstrated: a generic online adaptive image classification framework (from an EU-project) and recognition of pen-based hand-written digits. This evaluation includes tests on the accuracy and complexity of the evolved classifiers. Furthermore, the impact on the accuracy of forgetting consequent parameters in the multi-model case when a drift in the data set is observed, will be pointed out.

TABLE I
Comparison of *FLEXFIS-Class SM* and *MM* with batch classifiers when applying them onto CD imprint data

Method	**MC Rate Agg. / No. of Rules / No. of Inputs / Training Time**
FLEXFIS-Class SM	16.77% / 39 / 17 / 1.98s
FLEXFIS-Class MM	9.02% / 62 / 17 / 3.56s
FLEXFIS-Class MM + forg.	8.76% / 62 / 17 / 3.56s
CART [25]	8.76%
Probabilistic NN [26]	9.54%

A. Image Classification Framework

In this section an application example is given, which includes an automatically self-reconfigurable and adaptive fault detection framework for images which classifies each image as good or bad, and evolves the classifier upon operator's feedback and the data. The images are taken from an online production process with a high frequency with the aim to supervise the system, as they may show errors in a production process. This framework including pre-processing, segmentation and classification is shown in Fig. 2.

In principle, each type of image may be processed through the classification framework as shown in Fig. 2. The only assumption is that a master image is available: the purpose is to generate deviation images by subtracting newly recorded images from the master one in order to be able to classify the image into good or bad (depending on the structure and characteristics of the deviation pixels). For the evaluation of our approaches we applied image data from a CD-imprint production process, where faults due to weak colours, wrong palettes etc. should be detected within a process frequency of about one Hz. The data stream comprises 1164 images that were recorded one by one. Out of these images 776 served as training samples for building up the classification models in an evolving manner and the remaining 388 samples (test data) were used only for classification only (without an adaptation of the classification models, in order to be able to compare with the batch modelling approaches). 17 aggregated features were extracted, describing the distribution, density, shape etc. of the pixel fragments in the deviation images. The miss-classification rates on this test data set are demonstrated in Table I. From this table it can be recognized, that *FLEXFIS-Class MM* can compete with two well-known batch modelling methods, the decision tree-based classification method CART with optimal pruning strategy [25] and possibilistic neural networks [26], with respect to classification accuracy. By taking into account that the two renowned approaches access the whole data set information at once, the applicability and worthiness of *FLEXFIS-Class MM* should be clearly

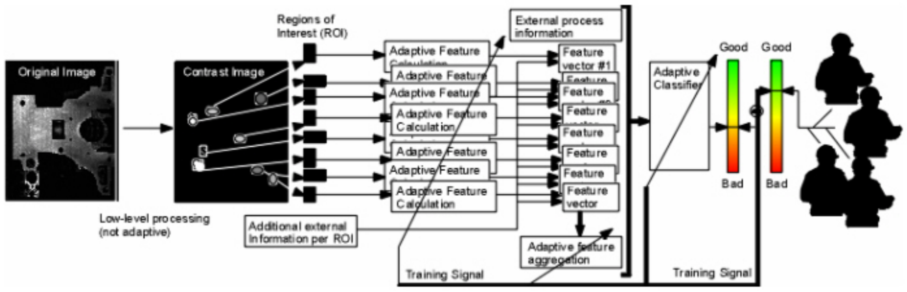
Original Image
Contrast Image
Regions of Interest (ROI)
Adaptive Feature Calculation
Feature vector #1
Feature vector
External process information
Adaptive Classifier
Good
Good
Bad
Bad
Low-level processing (not adaptive)
Additional external information per ROI
Adaptive feature aggregation
Training Signal
Training Signal

Fig. 2. Classification framework for classifying images into good and bad

underlined. From this table it is also clear that *FLEXFIS-Class MM* significantly outperforms *FLEXFIS-Class SM* in terms of accuracy. This is because Takagi-Sugeno models offer more flexibility, when regressing on the indicator matrix, than classical classification models which strongly depend on the incremental clustering procedure for evolving the antecedent parts of the rules. Regarding time complexity, *FLEXFIS-Class SM* is superior to *FLEXFIS-Class MM*. This is because in *FLEXFIS-Class MM* two TS fuzzy models need to be updated synchronously (one for class 'bad', one for class 'good'), there are more consequent parameters to update ($p+1$ per rule versus 1 per rule in case of *FLEXFIS-Class SM*) and the adaptation of the consequent parameters itself is much more complex, as a parameter vector and an inverse matrix needs to be updated, whereas in *FLEXFIS-Class SM* updating is done by simple counting. However, both can cope with the on-line demand of this system, as the image data is usually loaded with a frequency of about 1 Hz, whereas the training time in Table I is listed for the whole 776 training data samples.

The evolution of the accuracy over time was examined, when building up the classifier with *FLEXFIS-Class MM* step by step. Therefore, the fuzzy classification model was trained in incremental manner with 100, 200, 300, ..., 776 (all) data samples and the development of the miss-classification rate tracked, see Figure 3. One can see that the classifier improves with the number of samples, which shows the consistency with respect to the amount of information provided by the operator. Furthermore, cluster ageing as introduced in [23] and [24] was applied for detecting a potential drift in the data set (indicating that one or more operating conditions changed). This can be used as a trigger for exponential forgetting of older data points in the linear consequent adaptation within *FLEXFIS-Class MM* (see Section IV). In fact, it turned out that a small drift occurred for the last 80 data samples (as the slope of the age curves tend to increase), hence a forgetting with $\lambda = 0.99$ was initiated for the last 80 points. The results are listed in Row 3 of Table I; it can be seen that a some decrease of the miss-classification rate can be achieved. The complexity and computational performance of the classifier are practically the same without forgetting, (only a single parameter value is included in the weighted RLS update for linear consequent parameters, which does not affect the rule evolution and adaptation part). It should be also mentioned that a forgetting applied on the complete second half of the training data set worsened the miss-classification rates for both data sets. This indicates that a correct detection of a drift to trigger a forgetting is essential, when intending to improve the accuracy of the fuzzy classifier.

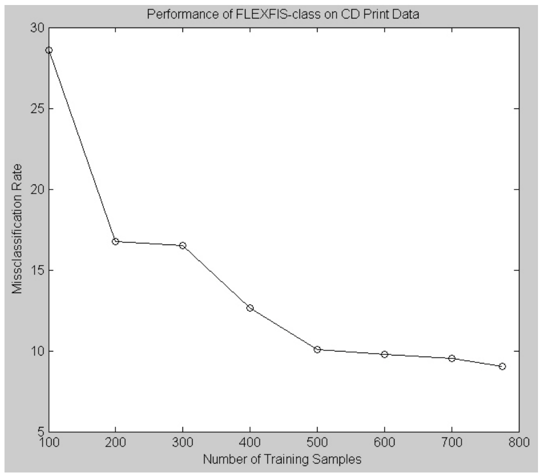

Fig. 3. Performance of *FLEXFIS-Class MM* on CD print data with increasing number of (online) training samples

B. *Pen-Based Recognition of Handwritten Digits*

This data set from the UCI repository[1] was created by collecting 250 samples from 44 writers, which was generated with the use of a pressure sensitive tablet with an integrated LCD display and a cordless stylus. The data set contains 16 features, 7494 training data samples and 3498 test data samples, containing ten different classes (for the ten digits) which are almost equally distributed in both, training and test data set. Principally, this data set is an off-line batch data set. However, we simulate it as an on-line pseudo-stream by performing a loading of data samples and evolve the fuzzy classifiers with each new incoming point separately.

Table II shows the results when applying the two model architectures for this data set. The conclusion is similar to that one for the image data set: *FLEXFIS-Class SM* is more transparent and less accurate, while *FLEXFIS-Class MM* produced ten TS fuzzy models with in sum 61 rules in total, which

[1] http://www.ics.uci.edu/ mlearn/MLRepository.html

TABLE II

COMPARISON OF EVOLVING VARIANTS TO WELL-KNOWN BATCH CLASSIFICATION METHODS ON HANDWRITTEN DIGITS DATA

Method	**Accuracy**
/ No. of Rules	**/ No. of Rules**
FLEXFIS-Class SM	88.77% / 16
FLEXFIS-Class MM	96.23% / 61
CART [25]	97.68%
k-NN	97.40%
Lin. regr. with indicator mat.	82.0%

is hardly interpretable, but could achieve an accuracy of over 96% which comes quite close to the results of the decision tree classifier CART [25] as well as of k-NN. Taking into account that the results for CART and k-NN were produced with the best parameter grid search procedure (for CART optimal pruning strategy, for k-NN the parameter k was varied from 1 to 20) while *FLEXFIS-Class MM* was started with the default parameter setting, it underlines even more the strength of *FLEXFIS-Class MM* on this data set. Compared to linear regression by an indicator matrix, which usually suffers from the masking problem when applied to a multi-classification task, *FLEXFIS-Class MM* can significantly outperform this method.

VI. CONCLUSION AND OUTLOOK

Two variants for evolving fuzzy classification schemes were presented, *FLEXFIS-Class SM* based on single-model architecture and *FLEXFIS-Class MM* based on multi-model architecture. The key issues of their algorithms were pointed out in detail in Sections II and III, whereas a concept for reacting on drifts in the data was demonstrated in Section IV, improving the accuracy of the model as shown in Section V-A. In the evaluation section two applications were described: i) classifying images into good or bad; ii) pen-based recognition of handwritten digits were described. Both methods produced reliable results; while *FLEXFIS-Class SM* produces a slim and transparent fuzzy model (with low number of rules and unique models), *FLEXFIS-Class MM* generates more complex models, but could achieve a higher accuracy, especially for the CD imprint data set. Furthermore, a reliable connection of cluster/rule ageing with forgetting of consequent parameters (as done in *FLEXFIS-Class MM*) could be achieved, triggering a benefit in terms of a better accuracy of the resulting classifier.

ACKNOWLEDGEMENTS

This work was supported by the European Commission (project Contract No. STRP016429, acronym DynaVis). This publication reflects only the author's view.

REFERENCES

[1] H. Maturino-Lozoya, D. Munoz-Rodriguez, F. Jaimes-Romera, and H. Tawfik, "Handoff algorithms based on fuzzy classifiers," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 6, pp. 2286–2294, 2000.

[2] R. Santos, E. Dougherty, and J. A. Jaakko, "Creating fuzzy rules for image classification using biased data clustering," in *SPIE proceedings series (SPIE proc. ser.) International Society for Optical Engineering proceedings series*. Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, 1999, pp. 151–159.

[3] J. Q. S. Marin-Blazquez, "From approximative to descriptive fuzzy classifiers," *IEEE Transactions on Fuzzy Systems*, vol. 10, no. 4, pp. 484–497, 2002.

[4] D. Nauck and R. Kruse, "Nefclass-x a soft computing tool to build readable fuzzy classifiers," *BT Technology Journal*, vol. 16, no. 3, pp. 180–190, 1998.

[5] J. Roubos, M. Setnes, and J. Abonyi, "Learning fuzzy classification rules from data," *Information SciencesInformatics and Computer Science: An International Journal*, vol. 150, pp. 77–93, 2003.

[6] D. Nauck, U. Nauck, and R. Kruse, "Generating classification rules with the neuro–fuzzy system NEFCLASS," in *Proc. Biennial Conference of the North American Fuzzy Information Processing Society NAFIPS'96*, Berkeley, 1996.

[7] N. S. H.-J. Rong, G.-B. Huang, and P. Saratchandran, "Sequential adaptive fuzzy inference system (safis) for nonlinear system identification and prediction," *Fuzzy Sets and Systems*, vol. 157, no. 9, pp. 1260–1275, 2006.

[8] N. K. Kasabov and Q. Song, "DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction," *IEEE Trans. on Fuzzy Systems*, vol. 10, no. 2, pp. 144–154, 2002.

[9] E. Lughofer and E. Klement, "FLEXFIS: A variant for incremental learning of Takagi-Sugeno fuzzy systems," in *Proceedings of FUZZ-IEEE 2005*, Reno, Nevada, U.S.A., 2005, pp. 915–920.

[10] C. Xydeas, P. Angelov, S. Chiao, and M. Reoullas, "Advances in eeg signals classification via dependant hmm models and evolving fuzzy classifiers," *International Journal on Computers in Biology and Medicine, special issue on Intelligent Technologies for Bio-Informatics and Medicine*, vol. 36, no. 10, pp. 1064–1083, 2006.

[11] P. Angelov, X. Zhou, and F. Klawonn, "Evolving fuzzy rule-based classifiers," in *2007 IEEE International Conference on Computational Intelligence Application for Signal and Image Processing*, Honolulu, Hawaii, USA, 2007, to appear.

[12] E. Lughofer and U. Bodenhofer, "Incremental learning of fuzzy basis function networks with a modified version of vector quantization," in *Proceedings of IPMU 2006, volume 1*, Paris, France, 2006, pp. 56–63.

[13] R. Gray, "Vector quantization," *IEEE ASSP Magazine*, pp. 4–29, 1984.

[14] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 15, no. 1, pp. 116–132, 1985.

[15] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, New Jersey 07458: Prentice Hall PTR, Prentic Hall Inc., 1999.

[16] P. Angelov and D. Filev, "An approach to online identification of Takagi-Sugeno fuzzy models," *IEEE Trans. on Systems, Man and Cybernetics, part B*, vol. 34, no. 1, pp. 484–498, 2004.

[17] A. Tsymbal, "The problem of concept drift: definitions and related work," Department of Computer Science, Trinity College Dublin, Ireland, Tech. Rep. TCD-CS-2004-15, 2004.

[18] L. Wang and J. Mendel, "Fuzzy basis functions, universal approximation and orthogonal least-squares learning," *IEEE Trans. Neural Networks*, vol. 3, no. 5, pp. 807–814, 1992.

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

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

[21] P. Angelov and E. Lughofer, "Data-driven evolving fuzzy systems using ets and flexfis: Comparative analysis," *to appear in International Journal of General Systems*, 2007.

[22] E. Lughofer and E. Klement, "Online adaptation of Takagi-Sugeno fuzzy inference systems," in *Proceedings of CESA'2003—IMACS Multiconference*, Lille, France, 2003, CD-Rom, paper S1-R-00-0175.

[23] P. Angelov and D. Filev, "Simpl_eTS: A simplified method for learning evolving Takagi-Sugeno fuzzy models," in *Proceedings of FUZZ-IEEE 2005*, Reno, Nevada, U.S.A., 2005, pp. 1068–1073.

[24] P. Angelov and X.-W. Zhou, "Evolving fuzzy systems from data streams in real-time," in *2006 International Symposium on Evolving Fuzzy Systems*, 2006, pp. 26–32.

[25] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees*. Boca Raton: Chapman and Hall, 1993.

[26] P. Wasserman, *Advanced Methods in Neural Computing*. New York: Van Nostrand Reinhold, 1993.