Pattern Recognition 45 (2012) 884–896

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Hybrid active learning for reducing the annotation effort of operators in classification systems

Edwin Lughofer

Department of Knowledge-based Mathematical Systems/Fuzzy Logic Laboratorium Linz-Hagenberg, Johannes Kepler University of Linz, Austria

A R T I C L E I N F O

Article history:
Received 27 January 2010
Received in revised form
3 August 2011
Accepted 7 August 2011
Available online 12 August 2011

Keywords:
Active learning
Reduction of annotation effort
Unsupervised selection criteria
Certainty-based selection
On-line update of classifiers

A B S T R A C T

Active learning is understood as any form of learning in which the learning algorithm has some control over the input samples due to a specific sample selection process based on which it builds up the model. In this paper, we propose a novel active learning strategy for data-driven classifiers, which is based on unsupervised criterion during off-line training phase, followed by a supervised certainty-based criterion during incremental on-line training. In this sense, we call the new strategy *hybrid active learning*. Sample selection in the first phase is conducted from scratch (i.e. no initial labels/learners are needed) based on purely unsupervised criteria obtained from clusters: samples lying near cluster centers and near the borders of clusters are expected to represent the most informative ones regarding the distribution characteristics of the classes. In the second phase, the task is to update already trained classifiers during on-line mode with the most important samples in order to dynamically guide the classifier to more predictive power. Both strategies are essential for reducing the annotation and supervision effort of operators in off-line and on-line classification systems, as operators only have to label an exquisite subset of the off-line training data resp. give feedback only on specific occasions during on-line phase. The new active learning strategy is evaluated based on real-world data sets from UCI repository and collected at on-line quality control systems. The results show that an active learning based selection of training samples (1) does not weaken the classification accuracies compared to when using all samples in the training process and (2) can out-perform classifiers which are built on randomly selected data samples.

1. Introduction

1.1. Motivation

In classification problems, the main problem is to gather the labeling information for recorded data samples (measurements, images, signals). That is because labeling usually requires a high workload for experts or system operators, which have to look through all the collected samples and give them labels in order to indicate to which classes the samples belong. Finally, this means that for the company where the expert/operators are employed, it may cost significant man-hours and hence significant money. For instance, in an image classification system for surface inspection [1], a human has to examine the images, showing the surfaces of production items, in order to be able to annotate them into 'good' (showing no faulty occasions) and 'bad' (showing faulty occasions) ones—based on this labeling information, machine learning classifiers are trained. In audio (inspection) systems, classification of audio parts, such as for instance the discrimination of audio segments into speech or music [2] or the recognition of music genres [3], is an essential aspect and often required. An expert has to hear through a high number of music samples and characterize segments therein whether they belong to either speech or music or to a concrete music genre (=class), which could be quite time-consuming. Labeling is even more time-intensive and especially requires deeper expert knowledge in case of physical/chemical/biological measurements with little or without any context. For on-line learning scenarios, such as adapting classifiers during on-line operation modes in order to account for changing system dynamics and to achieve an improved predictive performance over time, the labeling problem is even more severe, as often data is recorded with a high frequency which mostly prohibits the operator(s) to give feedback onto the classifiers decision (a quality information). This feedback is absolutely necessary as within the incremental learning context it is only possible to further improve the classifier when having the real class label available [4,5], at least for some representative points [6]; if using the own classification outputs, the classifier would train its own errors again and again into its structure, which after some time would lead to a decreased predictive performance.

In fact, in software rating tools equipped with a comfortable user and human–machine interaction interface the user gets a catchy visualization of the data to be classified and can perform

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

the ratings with single mouse-clicks—for instance see [7] where such a rating tool is used for image classification scenarios at surface inspect systems or [8] for more general visual interactive systems—such systems are more and more replacing traditional simple industrial front-ends. However, still it is a high effort to label sufficient number of samples (1) especially in multi-class problems including many classes, where each class needs a significant number of representative for building a reasonable classifier on these [9], also to avoid imbalanced learning problems [10] and (2) especially giving feedback during on-line mode in real-time.

1.2. State-of-the-art

A possible approach to reduce the annotation effort of operators in classification systems is the so-called active learning strategy [11,12]. Active learning can be understood as any form of learning in which the learning algorithm has some control over the input samples based on which it builds up the model. In natural systems (such as humans), this phenomenon is exhibited at both high levels (for instance active examination of objects) and low, subconscious levels (for instance the work on infant reactions to "Motherese" speech demonstrated in [13]). In industrial systems, this property of controlling the inputs is exactly what someone can exploit for reducing the workload of the operators (labeling effort, designing excitation signals). In particular, in an active learning approach the learning process may iteratively query unlabeled samples to select the most informative samples to annotate and update its learned models. Therefore, the central part of active learning is a data selection strategy in order to avoid unnecessary and redundant annotation and to concentrate on really new and important information to be included into the classifiers.

Regarding an appropriate data selection, there are two basic strategies:

- Certainty-based selection [14,15]
- Committee-based selection [16,17]

In the former approach, after training an initial classifier with some labeled samples, the certainties in the predictions on new (unlabeled) samples obtained from the initial classifier are measured. These certainties in the predictions can be mostly obtained as output confidence levels directly from the classifiers: for instance in case of rule-based classifiers the maximal membership degree of a rule (representing a class) compared to the sum of the membership degrees to all other rules or to the second nearest rule can be taken as confidence level—see [4]. In case of decision trees, the terminal nodes usually contain the relative frequency of the classes and thus can be used as confidence levels on new samples falling into these. In case of Naive Bayes classifiers, the a posteriori probability is a good measure for the confidence in the output class. For the samples where the lowest confidence weight is responded (i.e. the lowest certainty from the classifier reported), the expert or operator is asked to provide his/her labeling. This procedure is somewhat intuitive as an unconfident response indicates that the sample either lies around the decision boundary (conflict state) or in the extrapolation region of the original sample range (ignorance state) [18]. Fig. 1 visualizes an example, where the #1 query lies in the safe region, where neither ignorance nor conflict occurs, the #2 query lies in the conflict region where the two classes overlap and the #3 query lies in the ignorance region where none of the two classes was covered by training samples. In case when setting up a linear classifier, different possible decision boundaries can be established as indicated by straight lines in Fig. 1. This underlines the

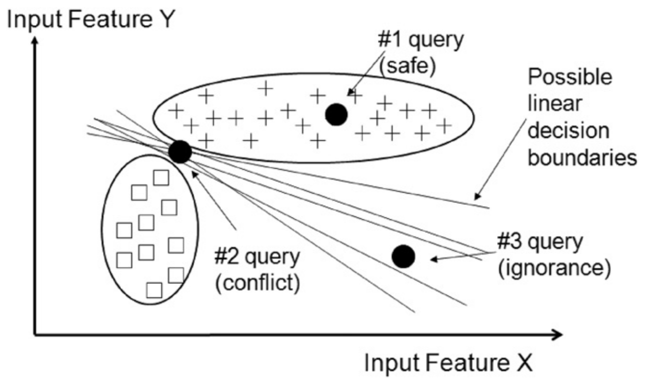

Fig. 1. Conflict and ignorance states in case of two query samples (#2 and #3), not being able to provide a confident decision to any of the two classes, the #1 query point lying in the save region; note the different possible decision boundaries in case of setting up a linear classifier.

uncertainty to predict #3 query to one of the two classes. When using (non-periodic) non-linear classifiers, this situation may become even worse, as it is additionally often unpredictable how the decision boundaries behave outside the range of the training samples. In both cases (conflict and ignorance), it is most welcome to update the classifiers in order to intensify its decision boundary, increasing its certainty or to correctly extend its structure by including the new data region.

The committee-based selection approach applies a distinct set of classifiers, which are all used for predicting new samples. Those samples whose assigned labels differ most among all classifiers are presented to the experts for annotation. In principle, this is equivalent to the situation that in case when different operators may label a sample differently (and eventually discuss and quarrel about it), a supervising operator (with more experience and know-how) has to make a final decision (here we have different classifiers and one operator). An example of a committee-based approach is the co-SVM approach demonstrated in [19] and applying two SVM classifiers for image retrieval purposes. The drawback of committee-based selection is that it is basically applicable in off-line mode (the diversity of predictions have to be examined for all collected training samples and compared against each other). On the other hand, certainty-based selection is applicable in an off-line as well as on-line learning process. The difference is basically that in the off-line phase some training samples are pre-labeled and, if the classification accuracy is not high enough, additional ones are presented to the expert, and a new classifier re-trained on all samples. This is done in a step-wise iterative procedure, parsing over the complete data set. In the on-line phase dynamic data streams characterized by data samples or data blocks are continuously arriving on-line over time, feedback from the experts is requested immediately in uncertain prediction cases from the evolved classifiers—for instance for all samples having a confidence lower than 0.7; this threshold is finally a matter of a tradeoff between the speed of increase of predictive accuracy and labeling effort of the operators. An example of a certainty-based approach is demonstrated in [20], where it is applied for spoken language understanding. Furthermore, certainty-based learning is not restricted to classification, but also applicable for identification/approximation problems, as there the local error bars and adaptive local error bars [21] can be used to detect regions where the model is still quite unsure as having wide error bands.

Active learning in connection with neural networks and support vector machines are studied in [22,23], respectively. In [24] an active learning approach is demonstrated, which is based on statistical models. This approach relies on selecting those samples from a data base in an attempt to minimize the variance of a learner over its own predictions; an optimal guidance for sample selection can be achieved for mixtures of Gaussians and locally weighted regression. Thereby, it is assumed that an initial learner is available, which was trained on some initial training samples, and active learning is done within a batch off-line modeling context. The batch approach in [25] requires an initial learner and selects that training sample, which, when joined with the other training samples and its label estimated by a posterior output distribution, most decreases the error of the re-trained learner. An enhanced active learning approach is proposed in [26], where feedback on features in addition to the labeling instances are included and which was successfully applied to text categorization, improving the classifier accuracies significantly. The basic assumption therein is that the expert is able to point to the most important features, based on which a re-weighting of the features can be achieved.

1.3. Our approach – the basic concept

Although active learning strategies as described in the previous section are mechanisms to reduce the annotation effort or the design of excitation signals for experts significantly, still they may require significant number of samples for the initial model training step (e.g. [24] or [25] require an initial learner to reduce its prediction variance). This is because the model already has to have a reasonable accuracy in order to give reliable predictions on new unlabeled training samples and hence in order to be able to decide which samples should be selected for further labeling. However, in real-world application scenarios, it is possible that the number of initial labeled data is small or even zero (for instance, there is no expert which can do the labeling or it is simply too time-consuming). Furthermore, in case of dynamic systems with changing behavior, an incremental update of classifiers is required in order to include new operating conditions and concept drifts on-the-fly. In this case, active learning in an on-line setting helps to reduce operators' efforts for providing feedback on classifier's predictions upon which the classifier needs to be updated. For such cases, we propose a novel active learning strategy, which is called hybrid active learning as coming with two stages:

- In the first stage, this strategy tries to select the most informative samples based on unsupervised clustering. Hence, no pre-labeling at all is required, as the selection is fully done based on cluster partitions obtained from the off-line batch data set.
- Once an initial classifier is set up, a certainty-based active learning approach is applied for new incoming on-line samples (during on-line training mode), and the classifier updated based on selected samples in purely incremental manner (without re-training phases).

In this sense, the approach is given the name *hybrid active learning*, as it combines unsupervised with supervised active learning within an initial off-line and further on-line learning context. Note that in [27] also an unsupervised active learning approach is demonstrated, which, opposed to our approach, requires pre-labeled examples (based on random selection from an initial collection): the annotations are used to assign labels to whole clusters obtained from the clustering process (using hierarchical graph-theoretic clustering). Furthermore, [27] does not include any on-line active learning stage coupled with incremental updateable classifiers, but conducts the whole active learning sample selection in batch mode.

2. Unsupervised sample selection for off-line training

2.1. Clustering phase

We use the *eVQ = evolving Vector Quantization* approach as demonstrated in [28], but with an initial number of clusters which is set to the number of possible classes at the system (it can be assumed that this number K can be gained a-priori from expert knowledge). This means, instead of evolving new clusters on demand based on the characteristics of the data (when processing the off-line data set as single samples into the algorithm), the clusters are kept fixed during the whole learning process, their centers initialized at the beginning (e.g. by using the first K samples) and shifted according to

$$\vec{c}_{win}^{(new)} = \vec{c}_{win}^{(old)} + \eta(\vec{x} - \vec{c}_{win}^{(old)}) \tag{1}$$

with a decreasing learning gain η which decreases with the number of iterations over the whole data set rather than with single samples forming the clusters (as done in original *eVQ*). The whole algorithm is called iterative vector quantization and described in Algorithm 1.

Algorithm 1. Iterative vector quantization (for unsupervised active learning)

```
(1)  Input: Number of clusters = number of classes $K$.
(2)  Choose initial values for the $K$ cluster centers
     $\vec{c}_i, i = 1, \ldots, K$, e.g. simply by taking the first $K$ data
     samples as cluster centers.
(3)  Fetch out the next data sample $\vec{x}$ from the data set.
(4)  Calculate the distance of the selected data sample to all
     cluster centers by using a pre-defined distance measure.
     We apply Euclidean distance measure.
(5)  Elicit the cluster center which is closest to the data sample
     by taking the minimum over all calculated distances $\rightarrow$
     winning cluster represented by its center $c_{win}$.
(6)  Update the $p$ components of the winning cluster by moving
     it towards the selected sample $\vec{x}$ as in (1) with $\eta$ a learning
     gain, decreasing with number of iterations, i.e. $\frac{0.5}{\#iter}$.
(7)  If the data set contains data samples which were not
     processed through steps 3 to 6, goto step 3.
(8)  If any cluster center was moved significantly in the last
     iteration, say more than $\varepsilon$, reset the pointer to the data
     buffer at the beginning and goto step 3, otherwise stop.
(9)  Estimate for each cluster which samples belong to it (for
     which samples it is the nearest one): yields a set of index
     vectors $index_i, i = 1, \ldots, K$.
(10) Estimate the spreads $\sigma_{i,j}^2$ for each cluster $i$ and each
     dimension $j$ by applying variance formula over samples
     of $index_i$.
(11) Output: cluster partition with $K$ clusters: centers
     $\vec{c}_1, \ldots \vec{c}_K$ and spreads $\vec{\sigma}_1, \ldots, \vec{\sigma}_K$.
```

The reason why we choose an initial number of clusters rather than estimating it automatically from the data is that we want to select the most informative samples based on the natural position of the classes in the feature space. Hence, we use the same # of clusters as classes are present in the data. In fact, if classes do not completely overlap, we are able to characterize the different class regions by different clusters.

We want to emphasize that for the subsequent sample selection strategy in principle any clustering approach extracting centers and ranges of influence can be applied. In the evaluation section, we will study the impact of using a different prototype-based clustering (in particular, the well-known k-means algorithm [29]) in order to see the impact of the chosen clustering algorithm on sample selection and furthermore classifiers accuracies when trained from selected samples.

2.2. Selection strategy from the extracted clusters

From the cluster partition, the selection is now performed based on the following two criteria:

- The first criterion is to select those samples which are close to the cluster (class) centers, also denoted as center-based selection. The reason for this is that these samples usually characterize the inner core parts of the classes and can be seen as highly informative where the classes have the highest density and their intrinsic location in the feature space. Closeness to the centers can be either expressed by any distance measure d in the normalized feature space or by a membership degree μ how strong samples belong to the clusters—as, e.g. achieved in fuzzy clustering approaches such as fuzzy c-means [30]. In the first case, those samples $\vec{x}$ are candidates for selection for which
$$d(\vec{c}_i,\vec{x}) \le thresh_i \quad \forall i=1,\ldots,K \tag{2}$$
with $thresh_i$ a small value dependent on the average width of the ith cluster (averaged over all dimensions):
$$thresh_i = 0.1\frac{1}{p}\sum_{j=1}^{p}\sigma_{i,j} \tag{3}$$
Hereby, we take care of a balanced class/cluster distribution, that is, to select the same number of samples from each cluster candidate set. Usually, a percentage of samples to be annotated is pre-defined, based on which the number of selected samples per cluster can be calculated: for instance if 20% samples (of the whole training set) are wished to be annotated, and four classes are possible, then for each cluster those 5% samples (of the whole training set) having minimal distance to the center are selected. In the second case, those samples are candidates for selection for which the membership degree μ to any cluster is high, e.g. higher than 0.9.
- The second criterion is to select those samples which are close to the border of the clusters (classes), also denoted as border-based selection. The reason for this concept is that these samples characterize the border-line between the classes, i.e. the region where the decision boundary is expected going to be drawn by the classifier. Hence, it is necessary to feed the classifier training algorithm with a significant number of these samples. Border samples are characterized by lying near the surfaces of the ellipsoidal clusters; that is the case when distance to the surfaces is low. In case of Euclidean measure, the distance can be calculated by
$$dist = (1-t)\sqrt{\sum_{j=1}^{p}(x_j-c_{ij})^2} \tag{4}$$
with
$$t = \frac{1}{\sqrt{\sum_{j=1}^{p}\frac{(x_j-c_{ij})^2}{\sigma_{ij}^2}}} \tag{5}$$
and p the dimensionality of the feature space, $\vec{c}_i$ the center of the ith cluster and x_j the current data sample. In case of Mahalanobis distance measure, the distance is calculated by (6)
$$t = \frac{1}{\sqrt{\sum_{j=1}^{p}(\sum_{k=1}^{p}(x_k-c_{ik})cov_{kj})(x_j-c_{ij})}} \tag{6}$$
with cov_{kj} the covariance between variable k and j, is low (refer to [28] for the proof of these formulas). The problem with this approach (we call it border-based selection variant 1) is that not only border samples in-between clusters (classes) are selected but also in cluster border regions where only one class is really significant. Furthermore, if two classes (clusters) strongly overlap, the surface-oriented border samples may lie significantly away from the decision boundary (see Fig. 2 below). Hence, we suggest to calculate membership degrees of each sample to all existing clusters, hence obtaining a membership matrix U where μ_{ij} is the membership degree of the ith sample to the jth cluster (class). Hereby, we calculate the membership degree μ_{ij} in the same manner as done in the fuzzy c-means training algorithm [30]:
$$\mu_{ij} = \left(\sum_{k=1}^{K}\frac{\|\vec{x}_i-\vec{c}_j\|^{2/(m-1)}}{\|\vec{x}_i-\vec{c}_k\|^{2/(m-1)}}\right)^{-1} \tag{7}$$
with m a fuzzy factor, usually set to 2. Then, for each sample $\vec{x}_i, i=1,\ldots,N$ the two clusters with maximal membership degrees are elicited by:
$$L1 = arg\ max_{j=1,\ldots,K}(\mu_{ij}) \quad L2 = arg\ max_{j=1,\ldots,K\backslash L_1}(\mu_{ij}) \tag{8}$$
and the deviation of membership degree T_i calculated by:
$$T_i = |\mu_{i,L1}-\mu_{i,L2}| \tag{9}$$
Finally, the samples with the lowest T_i score are selected for annotation, as these belong to two clusters (classes) with almost equal membership and hence can be seen as border-line samples along the classes.

In sum, k_1 samples from center-based and k_2 samples from border-based selection strategy will be selected, where k_1+k_2 denote the # of pre-defined samples which should be used in the annotation process.

2.3. A simple 2-dimensional example

A visualization of the three selection strategies (center-based, border-based in two variants) is presented in Fig. 2(a)–(c) for the Wine data set from UCI repository (showing the two most important features, 'total phenols' and 'alcohol'). The selected samples by each strategy are surrounded by a circle. From (b) we can clearly recognize that also border-line samples in uninteresting regions are selected, whereas in (c) only the border-line samples in the regions between the clusters (indicated as ellipsoids) are selected. Also, it is interesting to see that the second variant of border-based selection automatically selects more samples in the region where the two clusters (for the circle and diamond classes) overlap, i.e. where the decision boundary is harder to learn for the classifier, and a low number where the boundaries are easier to learn (as low class overlap). First experiments on classifiers trained from the selected samples by our unsupervised active learning approach show (1) that the obtained classifiers achieve higher accuracies than initial learners obtained from randomly selected samples (as used in many state-of-the-art approaches before active learning can start), and (2) that the obtained classifiers can even compete with classifiers trained on all samples. Some detailed results will be presented in Section 4.

3. Supervised sample selection during on-line adaptation

3.1. Strategy for sample selection

Once an initial classifier is trained based on the selected samples, it is now a challenge in many industrial systems to

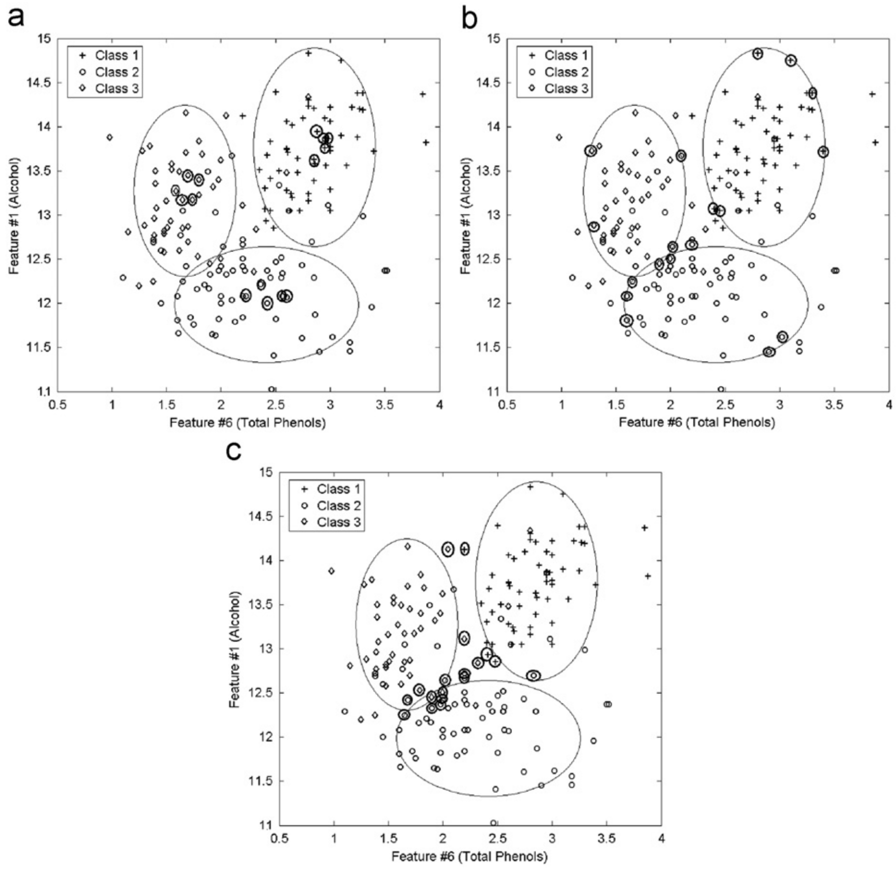

Fig. 2. (a) Samples selected by center-based selection strategy, (b) samples selected by border-based selection strategy variant 1, (c) samples selected by border-based selection strategy variant 2; the ellipsis indicate the extracted clusters.

update and evolve the classifiers on demand during on-line operation phase. This usually guarantees a higher performance of the classifiers and the prevention of extrapolation cases due to upcoming new operating conditions and systems states—see for instance [4], where image classifiers are updated regularly by new incoming feature vectors (characterizing images showing the surface of production items) or [31], where incremental trainable fusion methods are used for increasing the performance of incrementally updated base classifiers and also of initially batch trained fusion methods. Incremental training methods for various data mining and pattern recognition techniques have been reported in [5] (incremental SVMs), [32] (ID4 and ID5 as incremental decision tree learners), Hoeffding trees [33], incremental AdaBoost (AdaBoost.M1) [34] using the concept of incremental ensembling in Learn C++ [35], EFuNN [36] and DENFIS [37] as evolving neuronal approaches for classification and regression or FLEXFIS [38] as evolving Takagi–Sugeno fuzzy systems.

Here, we propose an active learning concept for incremental learning which is based on the certainty-based selection strategy [14,15]. Therefore, we define the confidence level of a classifier's output class response $conf_{out}$ to lie in the interval [0,1], where 0 denotes that the classifier is completely unconfident in its decision and 1 that it is completely confident. In most cases, we may get confidence levels for all K classes included in the data set $conf_i$ for $i = 1,\ldots,K$. Depending on the classifier's final decision, we may calculate a weighted final confidence out of the $conf_i$'s or use the winner-takes-it-all scheme (class with highest confidence is output class and automatically provides $conf_{out}$). Based on the final confidence output, the operator is requested to give a feedback on the classifier's decision:

- If the classifier is very certain in its decision ($conf_{out}$ near 1), then it is not necessary for the user to provide a feedback, as certain decisions are usually obtained for samples lying significantly far away from the decision boundaries and uniquely representing one class. The classifier is not updated in this case in order to prevent the inclusion of miss-classified samples (it is possible that the classifier could produce a classification error with high confidence, which would then mislead the classifier and intensify the error if included in the update).
- If $conf_{out} < thresh_{conf}$, where $thresh_{conf}$ is a user-defined parameter, the classifier is quite uncertain in its decision, hence the operator's response urgently requested, in which class the current sample actually falls. Also, when the operator agrees with the classifiers final output class, the classifier is updated in order to increase its certainty in the region where the current sample lies. We used different values for the threshold in the results section, see below, finally it is a matter of a tradeoff between feedback and annotation effort for the operators and classification performance: if $thresh_{conf}$ is lower, then the annotation effort for operators is lower as well, however, the accuracy may suffer, as the classifier is updated with less samples (how much is lost will be verified in Section 4.5); on the other hand, if $thresh_{conf}$ is higher, for much more samples the feedback is requested and the classifier updated more regularly, but the effort for the operators also increases.

3.2. Evolving classifiers used during on-line adaptation

We give a short introduction on the evolving classifiers we use for the experimental setup (Section 4.4) to verify the online active learning stage as demonstrated in the previous section. Hereby, we apply two methods:

- An evolving clustering-based classifier called *eVQ-Class* [39] (short for *evolving vector quantization for classification*), as it extends conventional VQ (vector quantization [40]) with incremental learning capabilities, and forming a classifier around the extracted cluster partitions with the help of a hit matrix.
- An evolving fuzzy classifier variant called *FLEXFIS-Class* [41] (applied for evolving image classifiers in [4], short for *FLEXible Fuzzy Inference Systems for Classification*), which exploits Takagi–Sugeno fuzzy systems to set up a multi-model fuzzy classification architecture (based on the idea of regression by indicator matrix [42]).

3.2.1. eVQ-Class

eVQ-Class is deduced from the unsupervised evolving vector quantization approaches (*eVQ*) as demonstrated in [28] by using the incremental update of the winning cluster as proposed in (1) for each new incoming sample in the case when the new sample fits to the current cluster partition. Thereby, the learning gain η_i (dependent on the ith cluster) is decreased with the number of samples belonging to the ith cluster, rather than with the number of iterations over the whole data set (as done in the conventional vector quantization method). The point why this is a plausible strategy is that clusters are evolved from scratch and the maximal movement horizon of each newly evolved cluster is bounded by a radius which is smaller than a vigilance parameter ρ, usually set to a fraction of the square diagonal in the unit hypercube (*eVQ-Class* takes normalized samples in range [0,1]). This is because otherwise a new cluster is born. In this sense, a kind of regularization effect is achieved, allowing only local movements. This makes more iterations over the latest incoming data block not necessary (as done in conventional *VQ*). This is a very welcome aspect as with more iterations the danger is high that the partition of older samples are completely forgotten. This is usually not wished in life-long learning tasks, except in drift cases [43]. Fig. 3 visualizes the update progress achieved by *eVQ* on ten samples $x1$–$x10$: in some cases a new cluster is evolved (a new center set to one of the data samples), in others already existing clusters updated (moving c's and expanding ranges of influences indicated by ellipses).

The extension to the classification case is simply achieved by introducing a hit matrix H, whose entries h_{ij} contains the number of data samples belonging to class j and falling into cluster i. In this sense, each row of the matrix contains the single relative frequencies of each class falling into the corresponding cluster. Note that the incremental update of this matrix is straightforward by incrementing the entries h_{ij}, whenever a sample is attached to cluster i and class j. If a new cluster is evolved for a sample falling into class j, a row is appended to H, where the single entries are all set to 0, except the jth which is set to 1. If a new class (the $K+1$th) is introduced by a new sample falling into cluster i, a column is appended whose entries are all set to 0, except the ith which is set to one. A hit matrix is necessary as the clusters are usually not clean, i.e. single clusters do not represent single classes, even though when using the class label entry in the clustering process, especially in high-dimensional cases: one additional feature (i.e. the label entry) would change the outcomes of distance calculations just very slightly compared to when not using this feature (an interpretation of the curse of dimensionality effect).

The classification phase in *eVQ-Class* is based on the 'winner-takes-it-all' approach, where the nearest cluster is seen as the best representative to label the current sample. Hence, similarly within the training process, the first variant simply takes the Euclidean distance of the sample to be classified to all the cluster centers and the cluster with minimal distance elicited, say the ith. The final class response L is given by that class appearing most frequently in the nearest cluster:

$$L = \arg \max_{k=1,\dots,K} h_{ik} \tag{10}$$

The confidence in the final output class depends on the degree of conflict to any of the existing classes. Fig. 4 shows and example, where a conflict occurs between two classes represented by two clusters: in fact, the query point lies somewhere in the middle, i.e. very close to the decision boundary. Therefore, a confidence output of 0.5 would be appropriate, favoring none of the two classes over the other.

If using in Fig. 4 only the hit matrix from the nearest cluster as basis for the calculation, the certainty would be 1, as the nearest cluster (the left is a bit nearer than the right) is clean, i.e. containing only samples from one class. Therefore, we propose

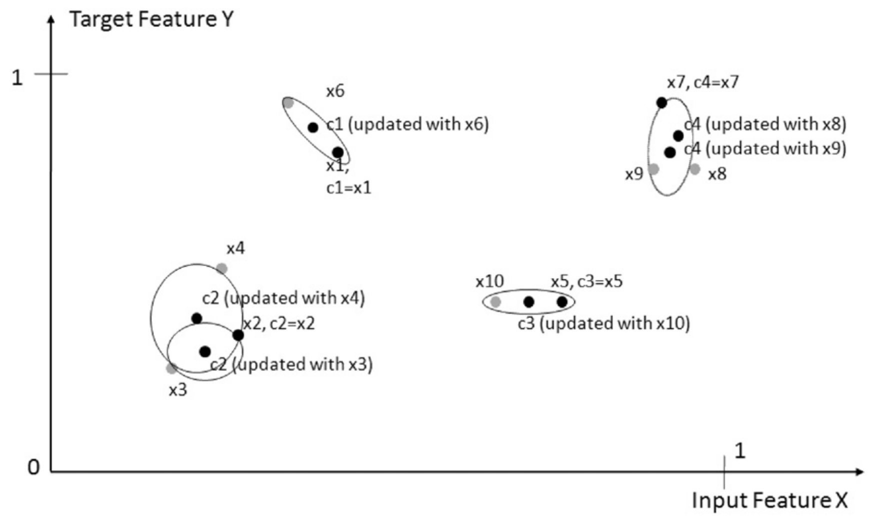

Fig. 3. Update progress of eVQ through data samples $x1$–$x10$, the centers indicated as $c1$–$c4$.

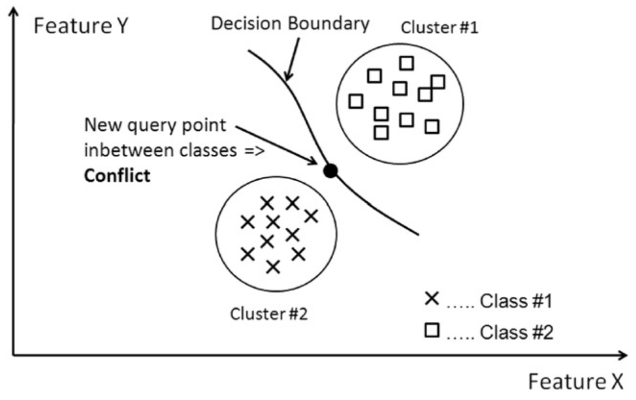

Fig. 4. Conflict case as the new query point lies exactly in-between two clusters representing two different classes.

the following confidence calculation:

$$conf_L = \frac{\mu_L h^*_{1,L} + \mu_{L2} h^*_{2,L}}{\mu_L + \mu_{L2}} \tag{11}$$

with

$$h^*_{1,L} = \frac{h_{1,L}}{h_{1,L} + h_{1,L2}} \quad h^*_{2,L} = \frac{h_{2,L}}{h_{2,L} + h_{2,L2}} \tag{12}$$

the relative frequency (weight) of the output class L in the two nearest rules supporting the two classes (h^*_1 belongs to the nearest cluster supporting class L, h^*_2 belongs to the nearest cluster supporting another class $L2$), μ_L the membership degree of the current sample to the nearest cluster supporting class L and μ_{L2} the membership degree of the current sample to the nearest cluster supporting class $L2$, i.e. in which the $L2$th class is the most frequent one. The membership degree of a sample to an ellipsoidal cluster can be expressed by multivariate Gaussians $\mu = e^{(-1/2)\sum_{j=1}^{p}(x_j - c_{ij})^2/\sigma_{ij}^2}$ with p the dimensionality of the feature space. This has the effect that conflicts as shown in Fig. 4 are modeled appropriately by responding a confidence degree near 0.5. In case when a sample falls into the center of one of the clusters shown in Fig. 4, it would automatically cause a membership value of the other cluster much less (near 0), which would reduce the calculation to $conf_L = h^*_{1,L}$, achieving a comparison of the frequency in the most frequent class with the second most frequent class (in this case an appropriate value of 1 would be responded).

In the active learning scenario, this confidence level is used for comparing against a pre-defined threshold $conf_{thr}$ as proposed in Section 3.1.

3.2.2. FLEXFIS-class

FLEXFIS-Class MM [44], applied for evolving image classifiers during on-line operation at surface inspection systems [4], exploits a multi-model architecture integrating K Takagi–Sugeno fuzzy regression models [45] 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 [46]. An indicator vector sent into the kth TS fuzzy model contains 1 in the target, if the current sample belongs to the kth class, otherwise it contains 0. Hence, for each class a regression surface is triggered, which goes towards 1 in the region of the feature space, where the corresponding class is present and towards 0 in the other parts. 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 $K>2$ [4]. 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 [47]).

The training of each TS fuzzy model in *FLEXFIS-Class MM* is based on a combination of a robust connection of incremental and evolving clustering in the antecedent space and a recursive weighted least-squares estimator for the linear consequent parameters. The weighted approach is necessary, as internally a local learning scheme is performed which estimates the consequent parameter for each rule separately (and independently from the others). This has some favorable properties regarding numerical stability, training speed as well as consistency and interpretability of the consequent functions, see [48, Chapter 2]. With robust connection it is meant that the recursive least square (RLS) estimator is still able to converge to a near optimal solution, although rules and fuzzy sets are shifted during the incremental learning phase (hence structural changes does not 'disturb' the RLS estimator significantly), also see [38].

Classification of new samples is conducted by eliciting that model producing the maximal output and taking the corresponding class as label response, according to a one-versus-rest classification scheme:

$$L = class(\vec{x}) = \arg\max_{m=1,\dots,K} \hat{f}_m(\vec{x}) \tag{13}$$

The confidence *conf* of the overall output value $L = m \in \{1,\dots,K\}$ is elicited by normalizing the maximal output value with the sum of the output values from all K models:

$$conf = \frac{\max_{m=1,\dots,K} \hat{g}_m(\vec{x})}{\sum_{m=1}^{K} \hat{g}_m(\vec{x})} \tag{14}$$

where $\hat{g}_m(\vec{x}) = \hat{f}_m(\vec{x}) + |\min(0, \min_{m=1,\dots,K} \hat{f}_m(\vec{x}))|$. This assures that all output values from all TS fuzzy systems are forced to be positive, and hence the confidence value well defined. Note that usually the output values of one TS fuzzy model are lying in [0,1], but could exceed this range in extrapolation cases (samples not included in the training data) or in regions where the sample density of the corresponding class was quite low. In an active learning scenario, this confidence information is used for comparing against a pre-defined threshold as proposed in Section 3.1. In the extreme case, the regression output value from all TS fuzzy models is nearly the same: then, the final class decision is near to a random guess and its confidence approaches $1/K$.

4. Evaluation

This section demonstrates the evaluation of the aforementioned approaches on unsupervised and supervised active learning based on two high-dimensional real-world problems, originally recorded at on-line production processes in surface inspection systems, and on three data sets from the Internet (UCI repository). The first part of this section is dedicated to the off-line learning phase, where unsupervised criteria are used for eliciting the most informative feature vectors; the second part describes the results achieved during on-line adaptation and evolution of (initially batch trained) classifiers.

4.1. Characteristics of the data sets

The two real-world application scenarios are characterized as follows:

- 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,

Table 1
Characteristics of the applied data sets.

Data set	# Tr. samples	# Test samples	# Feat.	# of classes
CD imprint	1024	510	74	2
Plates	4940	2471	62	8
Iris	150	150	4	3
Yeast	989	495	8	10
Spambase	2300	2300	57	2

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, aggregated features (in sum 74) 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 [49] 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.

- Pitch circle plates: there the problem is to detect various occurrences on the surface of these plates, such as dirt, scratches, porosities or splinters of glass; opposed to the CD imprint data, here no master images are calculated, but the original grey-level images are used; first, the objects are found by algorithms being able to put circumscribing ellipsoids over the regions of interest (usually identified by light and intensity indicators); second, object features (62 in sum) are extracted from these regions of interest (e.g. statistical measures from the grey-level histograms, shape descriptors, size of circumscribing ellipsis, intensity features).

Data sets from both application scenarios were collected on-line and stored in the same order on hard disc (into feature matrices) as recorded—this is necessary for a one-to-one simulation of on-line adaptation and evolution of the classifiers, see Section 4.4. Additionally, we used the following data sets (all known as widely applied application cases from the UCI repository[1]) [50] in order to complement the two data sets:

- Spam-base: the data represent feature vectors extracted from text in emails for discriminating between emails containing spam and emails containing no spam.
- Yeast: contains samples for predicting protein localization sites in eukaryotic cells.
- Iris: contains samples from the species of three different flowers (setosa, versicolor and virginica) based on some length and width criteria of their blossoms—the goal is to distinguish between these three flower classes.

A summary of the characteristics of all applied data sets is given in Table 1; these data sets comprise a range of the number of classes from 2 to 10, of the number of features from 4 to 74 and of the number of instances from 300 to 7411.

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

4.2. Experimental setup off-line phase

For the off-line phase, we divided the complete data set into training samples and test samples (for the concrete splits see Table 1). The whole training set or parts thereof were used in a 10-fold CV cross-validation phase in order to elicit the optimal parameter setting (CV repeated multiple times over a parameter grid). This setting was used for final training of the classifiers on all data folds used in the CV process, the obtained classifiers were evaluated with respect to their expected predictive accuracy by using the separate test data. In case when no active learning was switched on, the complete training data set was used in the CV procedure; in case when active learning was switched on, the most informative samples based on the supervised center- and border-based selection criterion demonstrated in Section 2 were used in the CV procedure. Therefore, we used various percentages of the whole data set as most informative samples, namely 10%, 20%, 30% and 50%. Half of these samples (i.e. 5%, 10%, 15% resp. 25%) were selected by center-based selection, the other half by border-based selection. Additionally, we applied a random selection of 10%, 20%, 30% and 50% of the whole data sets in order to see the improvement of our active learning scheme over randomly selected samples for initial learners (reducing the annotation effort of operators to the same extent as active learning selection does). We conducted center- and border-based selection based on clusters extracted from iterative VQ (as described in Algorithm 1) and k-means algorithm [51]. The number of clusters was set to the number of classes present in the data set. No parameters or thresholds for tuning are present in center- and border-based selection methods.

4.3. Results off-line phase

Table 2 shows the accuracies and variances (sensitivity over the CV folds) of the classifiers when trained with the different portions of training data (selected by our CBS (center- and border-based method) and by random selection) and using the CART approach for building decision trees [52]. These accuracies (plus variances) are compared with the accuracies (plus variances) when using the complete data set in the 10-fold cross-validation procedure. The pruning level in CART approach is optimized according to a best parameter grid search scenario, where the pruning level is varied from 0 (no pruning) to the maximal tree depth minus 1 (maximal pruning) and the quality of each setting is measured by the 10-fold CV classification rate—in Table 2, we report the best setting (highest accuracy) for each selection variant. For the CD imprint data set we report the (lowest) miss-detection rate, i.e. the number of samples which were classified as 'good', but are in fact 'bad', i.e. show broken surfaces of production items. This is because for this data set this number is very important, as denoting misses of bad items during the fault detection process which are then unpleasantly delivered to the customers.

From this table, we can observe the following results:

- Our CBS (center- and border-based selection) method can clearly out-perform random selection for almost all training data portions in all data sets in terms of both, final CV accuracy and variance of the accuracies over all folds. Thereby, our selection strategy based on k-means clustering (CBS-k) performs similar as based on iterative vector quantization (CBS-VQ).
- Random selection sometimes shows a quite unnatural behavior with respect to accuracy improvement over the percent of selected training data: for instance for the CD imprint data set the miss-classification rate is 9.26% when selecting only 10% of

Table 2
Comparison of classification performance (10-fold CV rates ± standard deviation) achieved by CART on several data sets when (1) *p*% randomly selected (RS) samples are sent into training of CART, (2) *p*% samples selected after center- and border-based selection strategy using iterative VQ (CBS-VQ) as well as *k*-means (CBS-*k*) are sent into training of CART and (3) all training data samples are sent into CART (last row); *p* is varied from 10% to 50%; for CD imprint data the miss-detection rates (=relative number of bad samples which are not detected as bad and hence denote a 'miss') are shown, for the other the classification rates.

% Samples	Meth.	CD	Plates	Iris	Yeast	SpamB
10						
	RS	9.26 ± 18.84	65.31 ± 3.85	50 ± 52.50	52.22 ± 15.76	81.82 ± 5.67
	CBS-VQ	15.55 ± 11.64	79.79 ± 6.22	80 ± 28.71	57.78 ± 11.48	90.00 ± 5.59
	CBS-*k*	16.26 ± 12.02	75.11 ± 5.55	82.3 ± 19.77	51.88 ± 14.52	88.75 ± 5.81
20						
	RS	14.89 ± 18.71	68.27 ± 3.52	86.67 ± 23.31	44.74 ± 12.89	83.78 ± 6.11
	CBS-VQ	11.15 ± 9.36	80.20 ± 5.10	90 ± 16.1	61.67 ± 15.15	89.78 ± 5.56
	CBS-*k*	11.81 ± 8.95	75.55 ± 5.25	89.51 ± 14.33	60.22 ± 14.83	89.12 ± 4.94
30						
	RS	17.36 ± 12.15	75.00 ± 4.39	80.00 ± 16.87	47.93 ± 6.18	85.74 ± 7.33
	CBS-VQ	10.17 ± 8.14	81.16 ± 2.96	87.50 ± 13.18	58.28 ± 11.31	88.24 ± 3.67
	CBS-*k*	9.68 ± 5.98	79.17 ± 3.62	90.78 ± 11.49	57.45 ± 13.92	89.78 ± 5.21
50						
	RS	19.24 ± 8.94	72.67 ± 2.69	95.71 ± 9.64	51.84 ± 8.72	88.42 ± 4.09
	CBS-VQ	8.19 ± 6.43	84.27 ± 2.38	94.29 ± 12.05	54.90 ± 5.04	90.88 ± 3.34
	CBS-*k*	9.12 ± 7.62	81.84 ± 3.11	92.95 ± 11.72	55.11 ± 11.33	90.55 ± 3.76
All (100%)		15.77 ± 6.46	77.96 ± 2.21	94.00 ± 7.34	56.84 ± 4.71	90.48 ± 1.57

the training data and much higher when selecting 20%, 30%, 50%—this is in unconformity with someone's intuition about monotonicity between the number of selected training samples and CV accuracies.

- When using the complete training data set (last row), the accuracies are generally higher than when using a selected subset of samples, however, the discrepancy to a 50% CBS selection is not that big. This means that by reducing the workload of the operators by about 50%, we still can achieve reasonable accuracies of the classifiers; in some cases (for instance in case of spam-base data or yeast data set), a significant reduction of the workload (by about from 70% to 80%) is plausible as still achieving high classification rates when using 30% or only 20% of the data.
- The selected class distribution in case of CD imprint data (good/bad) when using our method CBS-VQ was at 30% samples: 60.87/39.13%, at 50% samples: 64.98/35.03%, at 20% samples: 59.34/40.66%, at 10% samples: 55.26/44.74%; when using the complete data set, only around 16.78% are falling into the 'bad' class—therefore, it is no surprise that the miss-detection rate could be even increased when selecting a more balanced distribution for training; this also means that our CBS method is a reasonable technique for pre-selecting a balanced class distribution for validation and training purposes (as taking values from different clusters which may be indeed not strictly clean but softly associated with classes).

Table 3 can be seen as an extension of Table 2, as it reports the accuracies on a separate test set after final training of the CART classifier with the optimal parameter setting (achieving highest CV classification rate). This table shows similar occurrences as Table 2, underlining the observations drawn and reported in above itemization. Finally, we studied the dependence of our results on the selected classification method (CART in our case), as someone may argue that our method may favor certain classification techniques. Hence, we followed the same experimental setup as was conducted for producing results in Table 3, however, applied widely used *k*-NN [53] and SVMs classifiers [54,55] instead of CART. Therefore, to find the optimal parameters for final training, we varied *k* from 1 to 50 (from 1-NN to 50-NN), resp. *C* (a kind of regularization parameter used as penalty parameter in the error term) from 2^{-5} to 2^{15} in steps of +2 for

Table 3
Same as Table 2, but results are reported when performing a final training on all selected data with best parameter setting (elicited during 10-fold CV) and an evaluation on a separate validation test data set.

% Samples	Meth.	CD	Plates	Iris	Yeast	SpamB
10						
	RS	29.58	54.75	62.67	40.46	74.62
	CBS-VQ	17.36	60.50	74.67	48.28	80.00
	CBS-*k*	19.78	61.00	75.72	45.34	78.37
20						
	RS	22.83	60.58	75.33	40.46	84.10
	CBS-VQ	11.25	63.24	80.00	50.16	88.23
	CBS-*k*	12.66	63.55	80.00	49.45	86.11
30						
	RS	20.58	62.00	75.33	46.46	88.01
	CBS-VQ	10.65	65.14	96.00	52.46	89.21
	CBS-*k*	10.33	62.78	95.25	51.23	89.67
50						
	RS	15.43	62.72	88.00	53.74	88.05
	CBS-VQ	3.86	72.11	94.67	54.75	89.75
	CBS-*k*	6.77	70.42	95.25	53.89	89.75
All (100%)		7.72	76.57	98.00	56.97	91.27

the exponent and γ (steering the width of the kernel) from 2^{-15} to 2^3 in steps of +2 for the exponent when choosing RBF kernel as kernel function. These parameters were chosen according to recommendations in the guide how to use LibSVM in an ideal way as described in [56]. The results are reported in Tables 4 and 5 and basically follow the trend of the results produced with CART.

In order to underline this observation, Fig. 5 visualizes the accuracy trend-lines of center- and border-based selection for CART, *k*-NN and SVMs classifiers on the separate test data set when successively increasing the number of selected samples (from 10% to 50% along the x-axis).

Center-based selection can out-perform random selection for all training data portions, no matter which classification method is applied. Moreover, random selection shows an unexpected behavior when using SVMs classifier, as the miss-detection rate is increased when increasing the percentage of selected samples from 10% to 30%. The horizontal line at the bottom of the images shows the miss-detection rate achieved when using the complete training data set.

Table 4
Same as Table 3, but using k-NN instead of CART for building up a classifier.

% Samples	Meth.	CD	Plates	Iris	Yeast	SpamB
10						
	RS	33.44	57.14	85.33	43.27	82.67
	CBS-VQ	16.08	65.32	92.67	48.28	79.89
	CBS-*k*	14.56	62.93	88.82	47.88	82.34
20						
	RS	30.80	64.91	91.33	45.66	78.87
	CBS-VQ	16.80	66.76	94.00	54.14	83.06
	CBS-*k*	16.55	65.67	94.00	52.70	84.11
30						
	RS	15.66	69.00	92.67	47.47	87.01
	CBS-VQ	14.15	65.31	96.00	54.33	87.40
	CBS-*k*	13.78	67.45	95.25	54.65	86.23
50						
	RS	7.40	66.67	96.00	50.30	88.05
	CBS-VQ	5.79	67.75	97.33	55.56	85.58
	CBS-*k*	6.45	68.11	96.00	56.35	87.72
All (100%)		5.55	80.45	96.67	53.74	90.14

Table 5
Same as Table 3, but using SVMs instead of CART for building up a classifier.

% Samples	Meth.	CD	Plates	Iris	Yeast	SpamB
10						
	RS	19.61	57.14	90.00	48.55	82.71
	CBS-VQ	19.94	65.32	95.33	50.30	87.56
	CBS-*k*	21.82	64.23	94.00	48.34	86.64
20						
	RS	25.08	64.91	91.67	53.33	87.41
	CBS-VQ	17.36	66.76	95.33	57.05	90.96
	CBS-*k*	17.88	66.34	95.33	54.12	88.25
30						
	RS	26.69	69.00	94.67	53.13	91.31
	CBS-VQ	15.43	65.31	96.67	56.36	92.66
	CBS-*k*	13.92	66.75	95.33	55.33	89.41
50						
	RS	7.72	47.67	94.67	54.57	91.31
	CBS-VQ	5.47	56.37	95.33	56.36	92.23
	CBS-*k*	7.81	62.32	96.00	52.43	91.67
All (100%)		0.32	66.33	96.00	59.60	92.70

4.4. Experimental setup on-line evolution phase

For testing the supervised active learning strategy during on-line update of the classifiers, we used the CD imprint and the Plates data sets, as both were stored onto the hard disc as recorded during on-line phase. This means that the order in the off-line feature matrix is the same as during on-line production phase. In this sense, we are able to achieve a one-to-one simulation of a real on-line training and evolution phase of the classifiers. For setting up initial classifiers, we used the first half of the training data set (see Table 1), where we performed an elicitation of an optimal parameter setting (based on a 10-fold CV procedure): for both classifiers (*eVQ-Class* and *FLEXFIS-Class*), we varied the vigilance parameter ρ, responsible for controlling the evolution of a new rule versus the update of a cluster partition, from $0.1\sqrt{p}/\sqrt{2}$ to $0.9\sqrt{p}/\sqrt{2}$ in steps of 0.1 for the multiplication factor (p the dimensionality of the input feature space). For simulating the on-line adaptation and evolution of classifiers, we used the second half of the training data: for the conventional update, all training samples in the second half were used; in the context of our active learning scenario only those samples from this second half were used for which the classifiers were not really certain whether they belong to the final output class or to another class (measured in terms of confidences after 11 and 14). Therefore, we applied four different thresholds on the confidence levels: 0.6, 0.7, 0.8 and 0.9. For the sake of simplicity, we used and report a threshold with value 1 when referring to the full update with all samples in the (second half of the) data set. For eliciting the accuracies of our classifiers during the incremental phase, we used the separate test data sets as indicated in Table 1 and examined the performance behavior over time.

4.5. Results on-line evolution phase

Tables 6 and 7 report the results obtained when applying the on-line active learning strategy on the CD imprint and the Plates data set, respectively. The second column denotes the results when applying *eVQ-Class*, the third column when applying *FLEXFIS-Class*. Rows from #3 to #7 show the results when using different thresholds in the active learning scheme: all samples for which the certainties in their predictions were below these thresholds (according to the confidence outputs of the corresponding classifiers) were used for parameter adaptation and structure evolution of the classifiers. This means that Row #3 reports the results when using all training data (conventional update without any active learning). Row #2 is for comparison purposes when the classifiers are not evolved with the new on-line samples, i.e. kept static after initial batch training.

When comparing these values (from the static kept fixed classifiers) with the updated ones, it is obvious that updating the classifiers in fact leads to clear improvement of the finally obtained accuracies on separate (on-line) test data, i.e. in terms of expected prediction accuracy for future production cycles: in particular, a full update leads to a performs boost of 3.5% in case of *eVQ-Class* and of 6.5% in case of *FLEXFIS-Class*. This means, classifier update and evolution can be seen as one key feature for improving the performance of classifiers during on-line operation mode. Comparing Rows from #4 to #7 denoting active learning scenarios with the results on full training (Row #3), we can see that in case of the plate data set a bit of this performance boost of 3.5% resp. 6.5% is lost when updating with less and less samples (as thresholds decrease):

1. No lose in case when updating only with samples for which the classifiers had 0.9 or lower certainty levels in their output class responses.
2. Small lose of 2.0% resp. 0.5% when updating with data predicted with certainty levels smaller 0.8, still leaving a 1.5% gain over static classifiers in case of *eVQ-Class* and a 6% gain over static classifiers in case of *FLEXFIS-Class*
3. Small lose of 2.0% resp. 1% when updating with data predicted with certainty levels smaller 0.7, still leaving a 1.5% gain over static classifiers in case of *eVQ-Class* and a 5.5% gain over static classifiers in case of *FLEXFIS-Class*
4. Small lose of 2% when updating with data predicted with certainty levels smaller 0.6, still leaving a 1.5% gain over static classifiers in case of *eVQ-Class* and a 4.5% gain over static classifiers in case of *FLEXFIS-Class*

For the CD imprint data set, the accuracies stay quite high in case of thresholds 0.9–0.7 and close to the accuracies when using all data for updating the classifiers. A severe drop by over 13% can be observed when using a threshold of 0.6 and the *eVQ-Class* approach for training.

Furthermore, we plot the evolution of the test accuracies over time (with more and more samples loaded into the classifiers update) for the plates data set in Fig. 6 for the thresholds 1 = full training (a), 0.9 (b), 0.8 (c) and 0.7 (d). Obviously, the improvement of the accuracies from the beginning of the incremental learning phase (left side) to its end (right side) is very similar in all scenarios: therefore, we can conclude that the improvement in

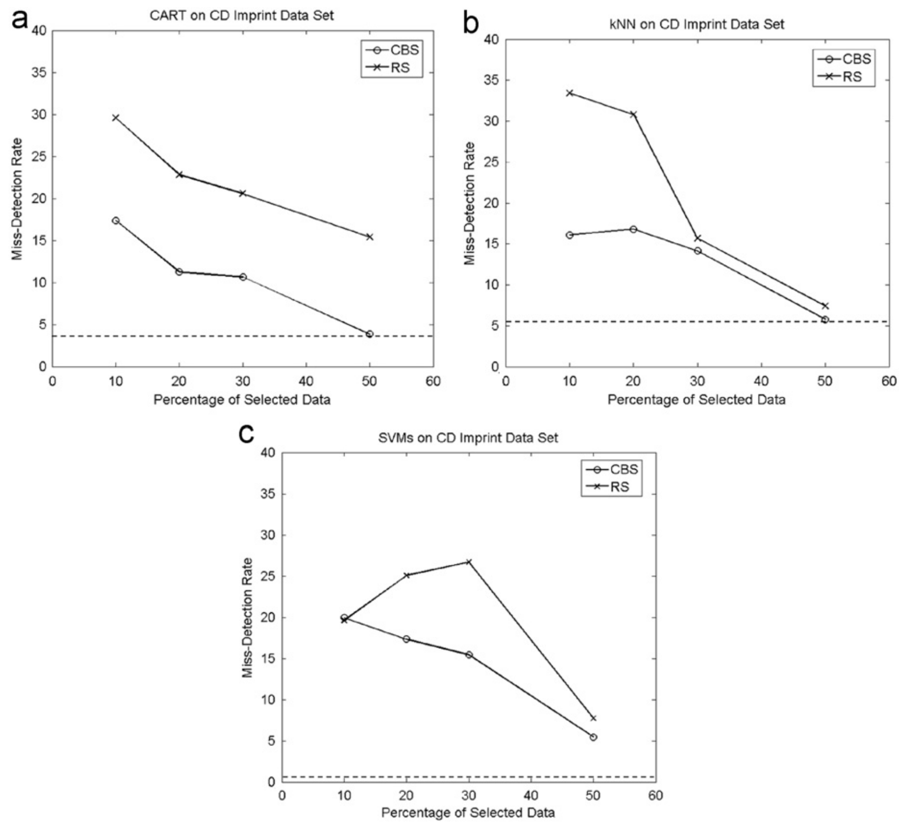

Fig. 5. Accuracy trend-lines with increasing percentage of selected training samples for CART (a), *k*-NN (b) and SVMs (c) classifiers; the horizontal lines show the accuracies achieved when using the complete training set for validation and final training.

Table 6
Comparison of the accuracies (in %) between classifiers initially built on the first half of the training data (second row) and sample-wise evolved classifiers with different portions of on-line data according to various thresholds in the active learning scenario for the CD imprint data set.

Training option	*eVQ-Class*	*FLEXFIS-Class MM*
Initial training	75.69	79.41
Threshold 1.0 (update on all samples)	89.61	91.57
Threshold 0.9	89.61	88.82
Threshold 0.8	89.61	88.82
Threshold 0.7	89.61	88.04
Threshold 0.6	75.98	87.06

Table 7
Same as Table 6, but using the 'Plates' data set.

Training option	*eVQ-Class*	*FLEXFIS-Class MM*
Initial training	67.99	61.98
Threshold 1.0 (update on all samples)	71.34	68.43
Threshold 0.9	71.34	68.43
Threshold 0.8	69.41	67.99
Threshold 0.7	69.41	67.34
Threshold 0.6	69.41	66.55

the full training is mainly because of the update based on those samples for which the classifier is not really certain. This finally means that our active learning strategy is able to reduce the workload of the operators significantly while synchronously weaken the predictive performance of the classifiers just slightly.

5. Conclusion

In this paper, we presented a novel approach for training data selection in classification problems, which comes with two strategies: completely unsupervised selection prior to classifier training in batch off-line phase and supervised selection for adaptation and evolution of the classifiers during on-line operation mode. The former is based on a clustering process, where samples near the cluster prototypes and along the cluster boundaries are selected. The latter exploits a certainty-based selection based on the certainties of the classifiers in their predictions=output response class labels. Both strategies are very useful in the sense that they reduce the workload of operators significantly. In fact, for high-dimensional real-world classification problems it could be verified that, by selecting only a percentage of 20–30%, still highly qualitative classifiers could be achieved whose performance is similar as those trained on the complete data (which was not possible when randomly selecting the same percentages). An essential point is that this could be achieved in connection with three different widely applied classification techniques (CART, kNN and SVMs). The on-line update of the classifiers shows similar accuracy improvement lines over the number of loaded on-line samples when using either all training data samples for classifier updates or only a small portion of these according to the proposed active learning selection strategy. An interesting future research direction could be the automatic calculation of the optimal number of clusters contained in the data (e.g. using cluster validation indices) supporting the idea that a class may appear in different regions of the feature space. This would probably require an extended sample selection strategy according to the multi-cluster-based class representation.

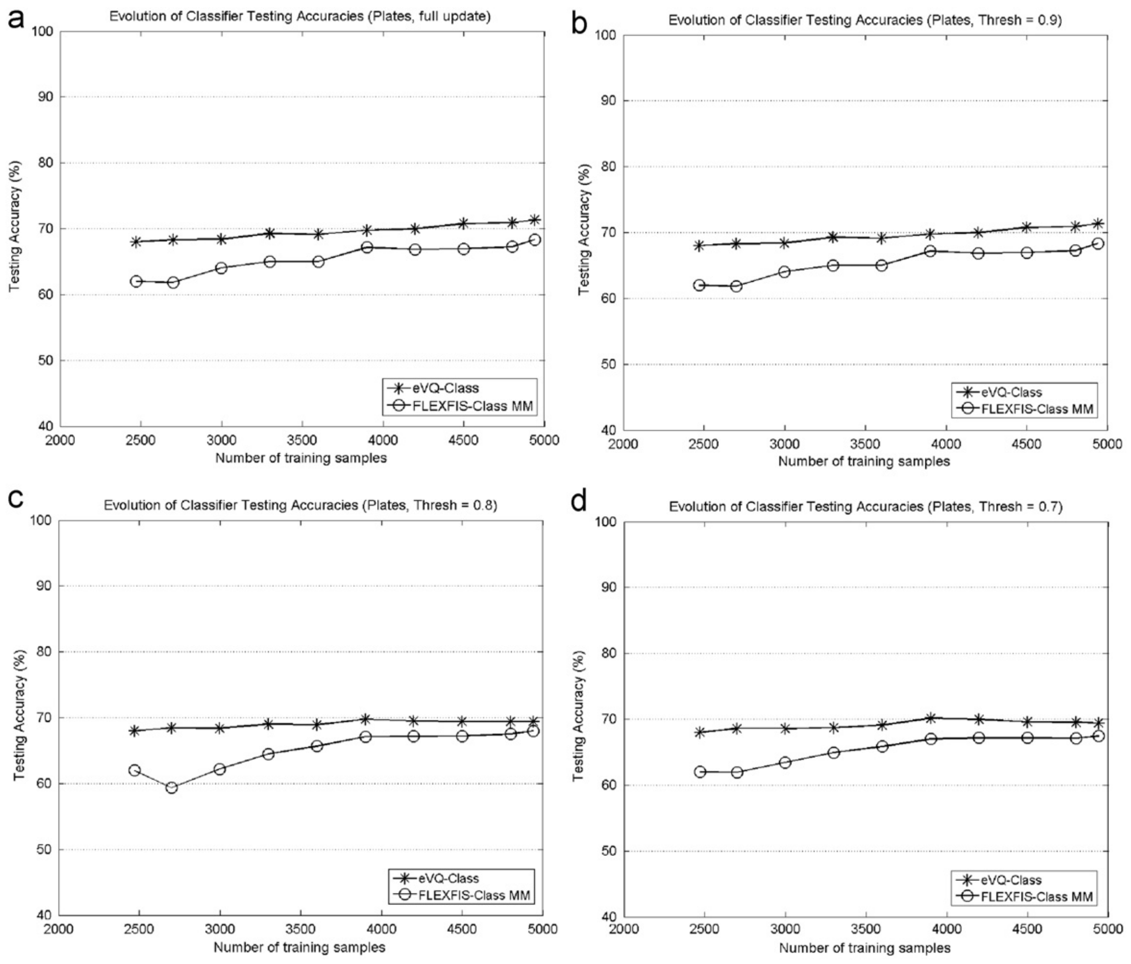

Fig. 6. Evolution of test accuracies (a) in case of full update of the classifiers (with all on-line samples); (b) in case of using active learning approach applying a threshold of 0.9 on the certainty levels; (c) in case of using active learning approach applying a threshold of 0.8 on the certainty levels; (d) in case of using active learning approach applying a threshold of 0.7 on the certainty levels.

Acknowledgments

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

References

[1] 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.
[2] M. Carey, E. Parris, H. Lloyd-Thomas, A comparison of features for speech, music discrimination, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) 1999, IEEE Press, Phoenix, USA, 1999, pp. 149–152.
[3] J. Bergstra, N. Casagrande, D. Erhan, D. Eck, B. Kegl, Aggregate features and ADA-boost for music classification, Machine Learning 65 (2006) 473–484.
[4] E. Lughofer, On-line evolving image classifiers and their application to surface inspection, Image and Vision Computing 28 (7) (2010) 1065–1079.
[5] C. Diehl, G. Cauwenberghs, SVM incremental learning, adaptation and optimization, in: Proceedings of the International Joint Conference on Neural Networks, vol. 4, Boston, 2003, pp. 2685–2690.
[6] A. Bouchachia, Incremental induction of classification fuzzy rules, in: IEEE Workshop on Evolving and Self-Developing Intelligent Systems (ESDIS) 2009, Nashville, USA, 2009, pp. 32–39.
[7] E. Lughofer, J.E. Smith, P. Caleb-Solly, M. Tahir, C. Eitzinger, D. Sannen, M. Nuttin, On human–machine interaction during on-line image classifier training, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 39 (5) (2009) 960–971.
[8] M. Costabile, D. Fogli, P. Mussion, A. Piccinno, Visual interactive systems for end-user development: a model-based design methodology, IEEE Transactions on Systems, Man and Cybernetics, Part A: Cybernetics 37 (6) (2007) 1029–1046.
[9] J. Zhou, H. Peng, C.Y. Suen, Data-driven decomposition for multi-class classification, Pattern Recognition 41 (1) (2008) 67–76.
[10] H. He, E. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge and Data Engineering 21 (9) (2009) 1263–1284.
[11] D. Cohn, L. Atlas, R. Ladner, Improving generalization with active learning, Machine Learning 15 (2) (1994) 201–221.
[12] D. Mackay, Information-based objective functions for active data selection, Neural Computation 4 (4) (1992) 305–318.
[13] A. Fernald, P. Kuhl, Acoustic determinants of infant preference for motherese speech, Infant Behavior and Development 10 (3) (1987) 279–293.
[14] D. Lewis, J. Catlett, Heterogeneous uncertainty sampling for supervised learning, in: Proceedings of the Eleventh International Conference on Machine Learning, New Brunswick, New Jersey, 1994, pp. 148–156.
[15] C. Thompson, M. Califf, R. Mooney, Active learning for natural language parsing and information extraction, in: Proceedings of Sixteenth International Conference on Machine Learning, Bled, Slovenia, 1999, pp. 406–414.
[16] I. Dagan, S. Engelson, Committee-based sampling for training probabilistic classifier, in: Proceedings of Twelfth International Conference on Machine Learning, 1995, pp. 150–157.
[17] I. Muslea, Active Learning with Multiple Views, Ph.D. Thesis, University of Southern California, 2000.
[18] J. Hühn, E. Hüllermeier, FR3: a fuzzy rule learner for inducing reliable classifiers, IEEE Transactions on Fuzzy Systems 17 (1) (2009) 138–149.
[19] J. Cheng, K. Wang, Active learning for image retrieval with co-SVM, Pattern Recognition 40 (2006) 330–334.
[20] G. Tur, D.-H. Tür, R. Schapire, Combining active and semi-supervised learning for spoken language understanding, Speech Communication 45 (2) (2005) 171–186.
[21] E. Lughofer, C. Guardiola, Applying evolving fuzzy models with adaptive local error bars to on-line fault detection, in: Proceedings of Genetic and Evolving Fuzzy Systems 2008, Witten-Bommerholz, Germany, 2008, pp. 35–40.

[22] K. Fukumizu, Statistical active learning in multilayer perceptrons, IEEE Transactions on Neural Networks 11 (1) (2000) 17–26.
[23] S. Tong, D. Koller, Support vector machine active learning with application to text classification, Journal of Machine Learning Research 2 (2001) 45–66.
[24] D. Cohn, Z. Ghahramani, M. Jordan, Active learning with statistical models, Journal of Artificial Intelligence Research 4 (1) (1996) 129–145.
[25] N. Roy, A. Mccallum, Toward optimal active learning through sampling estimation of error reduction, in: In Proceedings of the Eighteenth International Conference on Machine Learning, Morgan Kaufmann, 2001, pp. 441–448.
[26] H. Raghavan, O. Madani, R. Jones, Active learning with feedback on both features and instances, Journal of Machine Learning Research 7 (2006) 1655–1686.
[27] W. Hu, W. Hu, N. Xi, S. Maybank, Unsupervised active learning based on hierarchical graph-theoretic clustering, IEEE Transactions on Systems Man and Cybernetics—Part B: Cybernetics 39 (5) (2009) 1147–1161.
[28] E. Lughofer, Extensions of vector quantization for incremental clustering, Pattern Recognition 41 (3) (2008) 995–1011.
[29] J. Hartigan, Clustering Algorithms, John Wiley & Sons, New York, USA, 1975.
[30] J. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Kluwer Academic/Plenum Publishers, USA, 1981.
[31] D. Sannen, E. Lughofer, H.V. Brussel, Increasing on-line classification performance using incremental classifier fusion, in: Proceedings of the International Conference on Adaptive and Intelligent Systems (ICAIS'09), Klagenfurt, Austria, 2009, pp. 101–107.
[32] P. Utgoff, Incremental induction of decision trees, Machine Learning 4 (2) (1989) 161–186.
[33] P. Domingos, G. Hulten, Catching up with the data: research issues in mining data streams, in: Proceedings of the Workshop on Research Issues in Data Mining and Knowledge Discovery, Santa Barbara, CA, 2001.
[34] H. Mohammed, J. Leander, M. Marbach, R. Polikar, Can adaboost.m1 learn incrementally? A comparison to learn++ under different combination rules, in: Proceedings of the Artificial Neural Networks Conference 2006, Athens, Greece, 2006, pp. 172–179.
[35] R. Polikar, L. Upda, S. Upda, V. Honavar, Learn++: An incremental learning algorithm for supervised neural networks, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 31 (4) (2001) 497–508.
[36] N.K. Kasabov, Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning, IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics 31 (6) (2001) 902–918.
[37] N.K. Kasabov, Q. Song, DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction, IEEE Transactions on Fuzzy Systems 10 (2) (2002) 144–154.
[38] E. Lughofer, FLEXFIS: a robust incremental learning approach for evolving TS fuzzy models, IEEE Transactions on Fuzzy Systems 16 (6) (2008) 1393–1410.
[39] E. Lughofer, Evolving vector quantization for classification of on-line data streams, in: Proceedings of the Conference on Computational Intelligence for Modelling, Control and Automation (CIMCA 2008), Vienna, Austria, 2008, pp. 780–786.
[40] R. Gray, Vector quantization, IEEE ASSP Magazine 1 (2) (1984) 4–29.
[41] 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.
[42] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, second ed., Springer, New York, Berlin, Heidelberg, 2009.
[43] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Machine Learning 23 (1) (1996) 69–101.
[44] 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.
[45] 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.
[46] N. Draper, H. Smith, Applied Regression Analysis. Probability and Mathematical Statistics, John Wiley & Sons, New York, 1981.
[47] L. Wang, J. Mendel, Fuzzy basis functions universal approximation and orthogonal least-squares learning, IEEE Transactions on Neural Networks 3 (5) (1992) 807–814.
[48] E. Lughofer, Evolving Fuzzy Systems—Methodologies, Advanced Concepts and Applications, Springer, Berlin, Heidelberg, 2011.
[49] 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.
[50] UCI machine learning repository: center for machine learning and intelligent systems (1987–2011). ⟨http://archive.ics.uci.edu/ml/datasets/⟩.
[51] A. Jain, R. Dubes, Algorithms for Clustering Data, Prentice Hall, Upper Saddle River, New Jersey, 1988.
[52] L. Breiman, J. Friedman, C. Stone, R. Olshen, Classification and Regression Trees, Chapman and Hall, Boca Raton, 1993.
[53] R. Duda, P. Hart, D. Stork, Pattern Classification, second ed. Wiley-Interscience (John Wiley & Sons), Southern Gate, Chichester, West Sussex, England, 2000.
[54] V. Vapnik, Statistical Learning Theory, Wiley and Sons, New York, 1998.
[55] B. Schölkopf, A. Smola, Learning with Kernels—Support Vector Machines, Regularization, Optimization and Beyond, MIT Press, London, England, 2002.
[56] C.-W. Hsu, C.-C. Chang, C.-J. Lin, A practical guide to support vector classification, 2006.

Edwin Lughofer received his Ph.D. degree from the Department of Knowledge-Based Mathematical Systems, University Linz, where he is now employed as post-doctoral fellow. From 2002 to 2010, he has participated in several international research projects, such as the EU-projects DynaVis: www.dynavis.org, AMPA and Syntex (www.syntex.or.at). In this period, he has published around 60 journal and conference papers in the fields of evolving fuzzy systems, machine learning and vision, clustering, fault detection, image processing and human–machine interaction, including a monograph on 'Evolving Fuzzy Systems' (Springer, Heidelberg). In these research fields he acts as a reviewer in peer-reviewed international journals and as (co-)organizer of special sessions and issues at international conferences and journals. He is a member of the editorial board and associate editor of the international Springer journal 'Evolving Systems'. Research visits and stays include the following locations: Department of Mathematics and Computer Science (Philipps-Universität Marburg), Center for Bioimage Informatics and Department of Biological Sciences, Carnegie Mellon University, Pittsburgh (USA), Faculty for Informatics (ITI) at Otto-von-Guericke-Universität, Magdeburg (Germany), Department of Communications Systems InfoLab21 at Lancaster University (UK).

In 2006 he received the best paper award at the International Symposium on Evolving Fuzzy Systems, and in 2008 the award at the 3rd Genetic and Evolving Fuzzy Systems workshop. In 2007, he received a Royal Society Grant for know-how exchange with Lancaster University in the field of Evolving Fuzzy Systems. In 2010 he initiated the bilateral FWF/DFG Project 'Interpretable and Reliable Evolving Fuzzy Systems' and is currently key researcher in the national K-Project 'Process Analytical Chemistry (PAC)'.