Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 41 (2008) 995–1011
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/pr

Extensions of vector quantization for incremental clustering

Edwin Lughofer*

Johannes Kepler University Linz, Altenbergerstrasse 69, A-4040 Linz, Austria

Received 9 October 2006; received in revised form 29 June 2007; accepted 12 July 2007

Abstract

In this paper, we extend the conventional vector quantization by incorporating a vigilance parameter, which steers the tradeoff between plasticity and stability during incremental online learning. This is motivated in the adaptive resonance theory (ART) network approach and is exploited in our paper for forming a one-pass incremental and evolving variant of vector quantization. This variant can be applied for online clustering, classification and approximation tasks with an unknown number of clusters. Additionally, two novel extensions are described: one concerns the incorporation of the sphere of influence of clusters in the vector quantization learning process by selecting the 'winning cluster' based on the distances of a data point to the surface of all clusters. Another one introduces a deletion of cluster satellites and an online split-and-merge strategy: clusters are dynamically split and merged after each incremental learning step. Both strategies prevent the algorithm to generate a wrong cluster partition due to a bad a priori setting of the most essential parameter(s). The extensions will be applied to clustering of two- and high-dimensional data, within an image classification framework and for model-based fault detection based on data-driven evolving fuzzy models.

Keywords: Vector quantization; Clustering; Incremental learning; New winning cluster selection strategy; Removing cluster satellites; Split-and-merge strategy; Image classification framework; Fault detection; Evolving fuzzy models

1. Introduction

Nowadays clustering plays an important role, whenever data bases or data sets should be divided into local areas and new observations be classified based on these local areas (denoting different states or patterns within the system). Other application tasks for clustering include the rule extraction for fuzzy models or forming neurons for neural networks. Data compression for huge data bases can be carried out as well. This is possible due to the fact that a cluster always can be seen as a group of data that are more similar to each other than data belonging to other clusters, see Ref. [1]. In this sense, a cluster center represents also a compact information about a local area or distinct data cloud within the data. A popular and widely applied clustering algorithm is vector quantization [2], which moves cluster centers denoted as code-book vectors towards accumulation points in the data set, a more detailed description follows in Section 2. In literature there are several extensions of vector quantization proposed, the most famous ones are the self-organizing maps (SOM) going back to Kohonen [3] and neural gas network [4,5]. With the usage of both, different topologies can be exploited (such as circular or grid ones), where reference vectors which are neighbors in the network topology should possess similar weight vectors, i.e. cluster centers. An extension for supervised learning tasks is the learning vector quantization (LVQ) approach [6], which is for the purpose to generate reference vectors which are then used for classification of new samples (usually by nearest-neighbor approach).

In nowadays industrial systems a special emphasis is given on incremental clustering processes. This is because in various applications quite often online measurements are recorded resulting in data streams. These data streams can roughly be thought of as an ordered sequence of data items, where the input arrives more or less continuously as time progresses [7,8]. Hence, the data streams must generally be processed in an online manner to guarantee that results are up-to-date (e.g. a clustering structure) and that queries can be answered with a small time delay (e.g. online classification statements based on

* Tel.: +43 7236 3343 435; fax: +43 7236 3343 434.
E-mail address: edwin.lughofer@jku.at.

0031-3203/$30.00
doi:10.1016/j.patcog.2007.07.019

clusters). This circumstance demands a self-automatic fast incremental and evolving procedure for building up clusters on a sample per sample basis, usually without using any prior data. Different approaches have been presented in literature such as incremental mixture models [9,10], online k-means algorithm [11] or a recursive calculation of cluster potentials [12] (a kind of incremental extension of subtractive clustering [13]).

In this paper an incremental and evolving variant of vector quantization is demonstrated, which builds up and updates clusters sample per sample with new incoming data. In this sense, it also omits the pre-definition of the number of clusters, which has to be sent as parameter into conventional vector quantization. Second, a different distance strategy is incorporated by taking the distance of new incoming points to the range of influence of clusters and not to the cluster centers themselves. Of course, the range of influence in each direction is also calculated incrementally. Third, a satellite deletion strategy is proposed to remove not significant clusters (satellites) after the complete learning process. This can be not only applied in connection with the incremental clustering approach described in this paper, but also in connection with arbitrary crisp clustering techniques which generate cluster centers. Fourth, a split-and-merge strategy is described, which guides the incremental clustering process to cluster partitions with a high quality (i.e. to cluster structures representing the clustered nature of data streams quite well), even though an undesired setting of the most essential vigilance parameter was carried out in advance (before starting the whole learning process). In this sense, it omits a strong dependency of a reasonable performance of the whole approach on an appropriate parameter setting, which is hardly possible to guess in advance. This strategy can be generically applied after each incremental learning step for all incremental clustering variants, which update cluster centers and ranges of influence. All these issues will be demonstrated in different subsections within Section 3. In Section 4 the extensions of vector quantization will be compared with conventional vector quantization and some other well-known clustering methods based on well-known high-dimensional clustering data sets with respect to the quality of the obtained clusters measured by a well-known cluster validation index. A performance comparison of the various clustering methods based on image classification and on evolving high-dimensional fuzzy models, which are further used in an online fault detection framework for obtaining high fault detection rates, rounds off the Evaluation section.

2. Vector quantization

The purpose of vector quantization [2] originally stems from encoding discrete data vectors to compress data which have to be transferred quickly e.g. for online communication channels. The prototypes, called code-book vectors here, are the representatives for similar/nearby lying data vectors. From the mathematical point of view, vector quantization is basically a simplified version of k-means clustering in sample-mode adaptation, i.e. it is capable to update the parameters point-wise. This should be not confused with a form of incremental clustering, where no iterations over the complete data set are possible. Let the dimensionality of the data to be clustered be p. The amount of code-book vectors (clusters) C has to be parameterized a priori, where each cluster has p parameters corresponding to the p components of each cluster center. With these notations and assuming that the input data are a priori normalized due to their range (this is necessary as distances between multi-dimensional data vectors are calculated), the algorithm for vector quantization can be formulated as in Algorithm 1.

Algorithm 1. Vector quantization.

1: Choose initial values for the C cluster centers $\vec{c}_i, i = 1, \ldots, C$, e.g. simply by taking the first C data points as cluster centers.
2: Fetch out the next data sample of the data set.
3: Calculate the distance of the selected data point to all cluster centers by using a pre-defined distance measure. Commonly, Euclidean distance is used.
4: Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances → winning cluster represented by its center c_{win}.
5: Update the p components of the winning cluster by moving it towards the selected point $\vec{x}$:

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

where the step size $\eta \in [0, 1]$ has to be chosen a priori and should be chosen appropriately. A value for η equal to 1 would move the winning cluster exactly to the selected data point in each iteration, a value near 0 would not give a significant change for the cluster center over all iterations, so a value between 0 and 1 is a more reasonable choice.
6: If the data set contains data points which were not processed through steps 2–5, goto step 2.
7: If any cluster center was moved significantly in the last iteration, say more than ε, reset the pointer to the data buffer at the beginning and goto step 2, otherwise stop.

It has to be remarked that indeed mostly the Euclidean norm is used resulting in ellipsoidal clusters parallel to the main axes, but others are also possible: for instance, with Mahalanobis distance ellipsoidal clusters in general position can be achieved [14,15]. Furthermore, for a good convergence an adaptive learning gain η is recommended [16], which decreases with the number of iterations.

3. Extensions of vector quantization

In this section necessary and recommended extensions are demonstrated to be able to apply vector quantization for online clustering tasks and to data sets with an unknown number of

clusters. Moreover, an alternative distance strategy for omitting more cluster centers in wide data clouds and a cluster satellite deletion technique are demonstrated. The latter one assures that cluster artifacts which can come up during the incremental clustering process (e.g. due to significant movements of cluster centers) are deleted. Furthermore, a split-and-merge strategy will be proposed, which overcomes the drawback of generating incorrect cluster partitions due to a priori, inappropriately set parameters.

3.1. Vector quantization in incremental mode

Algorithm 1 cannot be reasonably applied for online processes, where incremental clustering is demanded, meaning clustering techniques which update their parameter for each newly loaded data block or even for each single data sample without taking into account prior data. This is because Algorithm 1 iterates over the loaded data buffer several times. If this would be carried out for each incremental learning step i.e. for each actual loaded data block separately, the cluster centers would only represent a reliable partition of this data block and forget the older data completely. Indeed, it is theoretically possible to collect all the data points recorded or loaded so far, keep it in the virtual memory and perform a re-estimation of the cluster centers from time to time. This procedure, however, would result in an inacceptable computation performance, which could be verified in Ref. [17], when training fuzzy models from data with the help of vector quantization for input/output space partitioning and rule generation (see also Section 4.3).

Moreover, the number of clusters has to be known in advance, which can be a significant drawback, especially in case of high-dimensional data sets as the number of clusters can usually not be seen. Furthermore, in the case of online clustering the number of clusters are never known in advance as usually the data have to be processed through the algorithm as these are loaded (e.g. for online streams in data bases) or recorded (e.g. for online measurement systems). It should be noticed that for the offline case this problem can be indeed solved by applying cluster validation indices [18] and choosing that partition with that number of clusters which optimizes the cluster validation index. However, for the online case the drawback still remains. It could be also inspected that the convergence of conventional vector quantization may be weak, when taking, for instance, the first C data points as the C cluster centers to start, see Section 4.1.

Hence, for omitting these drawbacks the idea of adaptive resonance theory (ART) network [19] is exploited. These networks consist of neurons which are able to adapt to new information without forgetting or overwriting already learned relationships, so as to overcome the famous *stability/plasticity dilemma*. In ART networks, especially in the ART-2 algorithm, this conflict is solved by the introduction of a *vigilance* parameter, which controls the tradeoff between adaptation of already learned clusters and generation of new clusters. In this sense, for each new data point the following condition is checked:

$$\|\vec{x} - \vec{c}_{win}\|_A \geqslant \rho \quad \text{and} \quad \vec{x} \text{ is not faulty} \tag{2}$$

with $\vec{x}$ the actual data point, $\vec{c}_{win}$ the winning cluster and A the norm-inducing distance. If this condition is fulfilled, the prototype c_{C+1} of the new (the $C+1$th) cluster is set to the actual data point. In fact, it is true that the problem of a priori defining the number of clusters C is shifted to finding a good value for the *vigilance* parameter ρ. However, a better assessment for the value of this parameter can be achieved, when clustering is applied onto data normalized into the hypercube $[0, 1]^p$: it denotes the maximal distance of a new data point to the cluster partition obtained so far, such that no new cluster needs to be set. In a trial and error tuning phase with quite a lot different data sets it turned out that the following choice of this parameter should be preferred:

$$\rho = 0.3\frac{\sqrt{p}}{\sqrt{2}}. \tag{3}$$

This fixed setting can made flexible by the split-and-merge strategy which will be discussed in Section 3.4. The dependency of ρ on the p-dimensional space diagonal, i.e. $\sqrt{p}$, can be explained with the *curse of dimensionality* effect: the higher the dimension, the greater the distance between two adjacent data points, see Ref. [20]; therefore, the parameter ρ should get the larger so as to prevent the algorithm to generate too much clusters and causing strong overfitting effects. The second part of condition (2) assures that the new data point does not represent a faulty situation during data recordings, simply denoted as '$\vec{x}$ is not faulty'. In this sense, a faulty data point or a faulty point is defined as a measurement or a data sample which is recorded during a faulty situation at a system, e.g. a broken interface, an overheated sensor and so on. Indeed, it is very hard to decide whether a new incoming data point lying far outside previously loaded data clouds denotes a new operating condition, which should be in fact included into the clustering process, or a faulty situation, which should be not included. It is worthy to mention that in a fault detection framework [21,22] (where the extended version of vector quantization was applied for training fuzzy models) faulty points could be detected and marked before and hence omitted in the clustering process. Another possibility for excluding such points is to wait for a certain amount of data points occurring in the same newly gathered local area and to compare this local area with respect to its range of influence, density and number of data points with the other already obtained local areas (= clusters) so far. If it is completely different, a fault is more likely than a new operating condition, as various operating conditions usually possess more similar ranges, densities, etc. In this case, no new cluster would be generated. With these notations, vector quantization in incremental mode can be described as in Algorithm 1.

Algorithm 2. Vector quantization in incremental mode (VQ-INC).

1: Initialize the number of clusters to 0
2: Collect a few tens of data samples and estimate the ranges of all p variables
3: Take the next incoming data point (online case) or fetch out a data sample from a data matrix randomly or ordered (offline case) and normalize it according to the ranges, let us call it $\vec{x}$
4: **if** number of clusters $=0$ **then**
5: Set $i=1$
6: Set the first center c_1 to the actual data point, hence $\vec{c}_1=\vec{x}$
7: goto step 17
8: **end if**
9: Calculate the distance of the selected data point to all cluster centers by using a predefined distance measure. Commonly, Euclidean distance is used
10: Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances $\rightarrow$ winning cluster represented by its center c_{win}
11: **if** $\|\vec{x}-c_{win}\|_A \geqslant \rho$ and $\vec{x}$ is not faulty **then**
12: Set $i=i+1$
13: Set $\vec{c}_i=\vec{x}$
14: goto step 17
15: **end if**
16: Update the p components of the winning cluster by moving it towards the selected point $\vec{x}$, as in Eq. (1)
17: **if** $\vec{x}$ is not faulty **then**
18: Update the ranges of all p variables
19: **end if**
20: If the data matrix still contains uncovered data (offline case) or new incoming data points are still available (online case) goto step 3, otherwise stop.

From this definition it can be realized that each (newly loaded) data point is processed only once through the update process. This is quite reliable (opposed to conventional vector quantization) as cluster centers are already initialized in new local data clouds and hence restricted to move therein (due to the fact that a movement is carried out only within a radius smaller than or equal to ρ). For this algorithm, the learning gain η cannot be decreased by the amount of iterations as no iterations take place. On keeping it constant, it would result in a bad convergence of the algorithm, as no matter how many data points belong to one cluster (i.e. for which this cluster was the winning cluster) the shift of the center would be always to the same extent, causing a fluctuating cluster structure. A possibility for preventing this situation can be accomplished by steering η with the amount of data points belonging to each cluster in a monotonic decreasing way:

$$\eta_i = \frac{0.5}{k_i} \quad \forall i \tag{4}$$

with k_i the number of data points belonging to cluster i. This is implemented in this way in Algorithm 2 and a reasonable choice as in k-means clustering algorithm [23] the step size is also normalized by this number, whereas original vector quantization is a simplified version of k-means clustering. Please note that there are also some synergies to the CEM algorithm (applied for incremental learning of Gaussian mixture models), which generalize the update of the centers by multiplying the inverse of the covariance matrix with the learning gain, see also Ref. [9]. An estimation of an initial range is done by the first few dozen of data samples, by which the data samples are normalized before being processed through the core part of Algorithms 2 and 3. After that the ranges are updated further for each new incoming fault-free sample (step 18). As it is not straightforward to decide, whether a new data sample represents a faulty or a non-faulty situation (see comments above), an alternative would be to apply 5%—respectively, 95%—quantiles (calculated incrementally) for estimating the range.

3.2. An alternative distance strategy

The problem with vector quantization in incremental mode is that in the case of wider data clouds or points belonging together widely spread over the input space Algorithm 2 tends to generate more clusters than necessary and hence performs an 'overclustering' and incorrect partition of the input space. This fact is underlined in the left image of Fig. 1, where obviously three clusters are the optimal case, but five clusters are generated. This is because for the big data pattern three clusters are produced instead of one.

The reason for this unpleasant occurrence lies at hand: Algorithm 2 (as well as conventional vector quantization) compares each new incoming point with all the cluster centers, which can happen to be far away even if the data point is close to its spanned range of influence represented by those data points already belonging to the cluster. Therefore, an obvious overcoming of this drawback can be achieved by calculating the ranges of influence during the incremental learning process and taking the distance of new points to these ranges instead of to the cluster centers. Whenever the Euclidean distance is used as distance measure (which is more or less the most common choice), axis-parallel ellipsoids are triggered as clusters, whose range of influence (as 2σ-area) can be calculated in incremental mode by exploiting recursive variance formula [24]:

$$k_{win}\sigma^2_{win,j}(new) = (k_{win}-1)\sigma^2_{win,j}(old) + k_i\Delta c^2_{win,j} + (c_{win,j}-x_j)^2 \quad \forall j, \tag{5}$$

where $\Delta c_{win,j}$ is the distance of the old prototype to the new prototype in the jth dimension of the cluster nearest to the actual point $\vec{x}$ and k_{win} is the amount of data points lying nearest

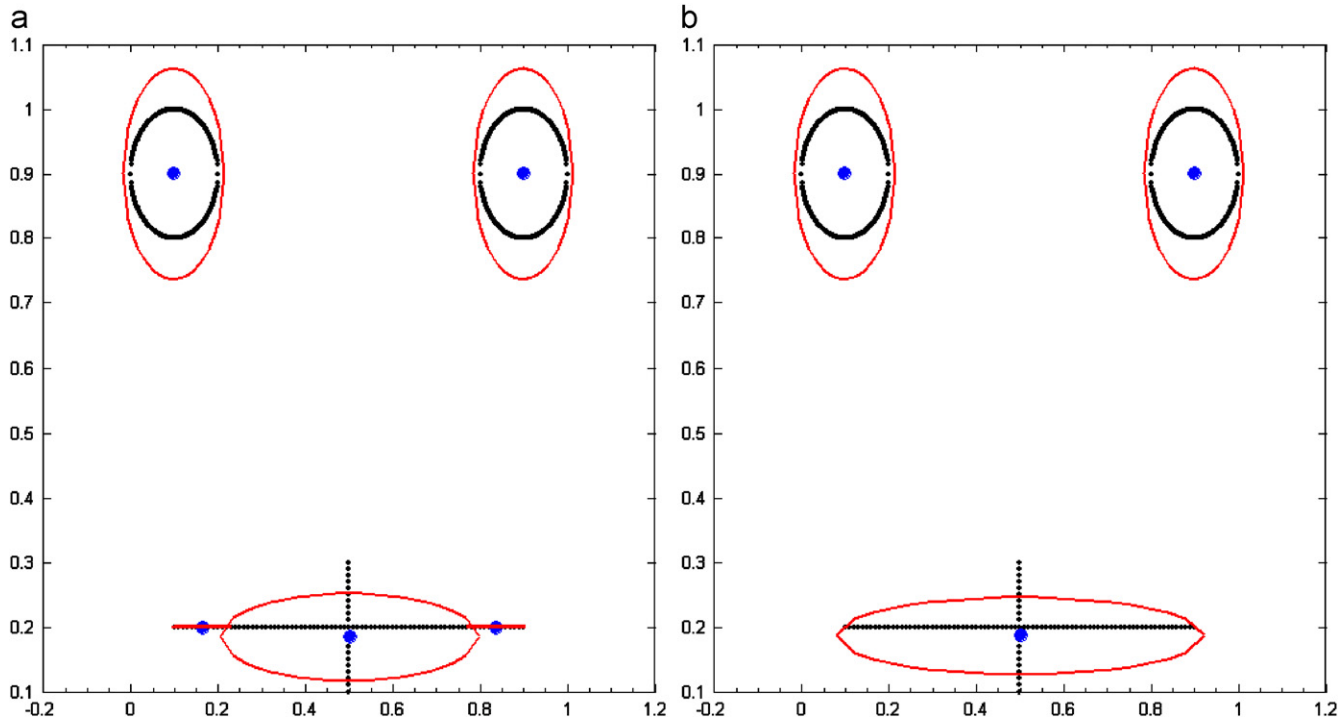

Fig. 1. (a) Clustering obtained by Algorithm 2 → too many clusters; (b) clustering obtained by its extended version with new distance strategy (Algorithm 3) → clusters are fine.

to cluster c_{win} and can therefore be simply updated through counting. For the distance of the new data point to the surface of the multi-dimensional ellipsoid spanned by a cluster we take the distance along the direction from the actual point towards the cluster center.

Lemma 1. *Let therefore* $\sum_{j=1}^{p}(x_j - c_{ij})^2/\sigma_{ij}^2 = 1$ *be a multi-dimensional ellipsoid of the ith cluster in main position,* σ_{ij} *the variance of the data belonging to the ith cluster in dimension j, then the Euclidean distance of the new data point* $(q_1, \ldots, q_p)$ *to the surface along the direction towards the cluster center* c_{ij} *is given by*

$$dist = (1-t)\sqrt{\sum_{j=1}^{p}(q_j - c_{ij})^2} \tag{6}$$

with

$$t = \frac{1}{\sqrt{\sum_{j=1}^{p}(q_j - c_{ij})^2/\sigma_{ij}^2}}. \tag{7}$$

Proof. Let $\vec{x} = \vec{c}_i + t(\vec{q} - \vec{c}_i)$ be the straight line in the multi-dimensional space between the new point $(q_1, \ldots, q_p)$ and the ith cluster center. Then the crossing point of this straight line with the multi-dimensional ellipsoid is obtained by setting the parameter vector into the ellipsoid equation. Hence $\sum_{j=1}^{p} t^2(q_j - c_{ij})^2/\sigma_{ij}^2 = 1$, and, by solving this equation after t, we achieve Eq. (7). The Euclidean distance between the crossing point and $(q_1, \ldots, q_p)$ is computed by summing up the squared differences $(q_j - c_{ij} - t(q_j - c_{ij}))^2 = (q_j(1-t) - c_{ij}(1-t))^2$ for all j and taking the root, resulting in Eq. (6). □

In fact, distance (6) is only computed for the actual data point $\vec{q}$, if it is lying outside the ranges of influence of all clusters, i.e. the condition

$$\exists i \quad \sum_{j=1}^{p}\frac{(q_j - c_{ij})^2}{\sigma_{ij}^2} \leqslant 1 \tag{8}$$

is not fulfilled. Otherwise, the usual distance strategy is applied to all clusters, in whose range of influence the actual data point lies inside. This leads us to the extended version of vector quantization in incremental mode, as described in Algorithm 3.

Algorithm 3. Extended version of vector quantization in incremental mode (VQ-INC-EXT).

1: Initialize the number of clusters to 0
2: Collect a few tens of data samples and estimate the ranges of all p variables
3: Take the next incoming data point (online case) or fetch out a data sample from a data matrix randomly or ordered (offline case) and normalize it according to the ranges, let us call it $\vec{x}$
4: **if** number of clusters $= 0$ **then**
5: Set $i = 1$
6: Set the first center c_1 to the actual data point, hence $\vec{c}_1 = \vec{x}$, set $\vec{\sigma}_1 = \vec{0}$ and the number of data points belonging to the first cluster to 1 ($k_1 = 1$)

7: goto step 26
8: **end if**
9: **if** The actual data point lies inside any cluster's range of influence, i.e. the condition Eq. (8) is fulfilled for at least one i (where σ_{ij} are taken as $\max(\sigma_{ij}, \varepsilon)$ with $\varepsilon > 0$ to stay numerically stable) **then**
10: Calculate the distance of the selected data point to all those cluster centers fulfilling Eq. (8) by using Euclidean distance measure
11: Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances → winning cluster c_{win}
12: Set $\min_{dist} = 0$
13: **else**
14: Calculate Eq. (6) for all clusters
15: Elicit the cluster center which is closest to the data point by taking the minimum over all calculated distances → winning cluster c_{win}
16: Set $\min_{dist}$ as the minimum over all distances
17: **end if**
18: **if** $\min_{dist} \geqslant \rho$ and $\vec{x}$ is not faulty **then**
19: Set $i = i + 1$
20: $\vec{c}_i = \vec{x}$, $\vec{\sigma}_i = \vec{0}$, $k_i = 1$
21: goto step 26
22: **end if**
23: Update the p components of the winning cluster by moving it towards the selected point $\vec{x}$ as in Eq. (1) with η_{win} as in Eq. (4)
24: Update the range of influence in each direction by using Eq. (5)
25: Increase the number of data points belonging to the winning cluster by 1, i.e. $k_{win} = k_{win} + 1$
26: **if** $\vec{x}$ is not faulty **then**
27: Update the ranges of all p variables
28: **end if**
29: If the data matrix still contains uncovered data (offline case) or new incoming data points are still available (online case) goto step 3, otherwise stop.

Fig. 1 demonstrates the impact of this extended version of vector quantization. While Algorithm 2 generates new clusters for data points lying near the range of influence of another cluster (Fig. 1(a)), the extended version performs better and extends the range of influence (drawn as the 2σ-range in each dimension) of the nearby lying cluster (Fig. 1(b)). The cluster centers are visualized as big dark data dots.

It should be noticed that Algorithm 3 can be extended straightforwardly to the case when Mahalanobis norm is used, which triggers ellipsoidal clusters in general position. For this, the recursive covariance formula, which is similar to the recursive variance formula, can be exploited for updating the covariance matrix (as its inverse is the Mahalanobis-norm-inducing matrix) of each cluster. The incremental calculation scheme of the covariance between the two variables x and y is defined as

$$cov(N+m) = cov(N) + N\Delta\bar{x}(N+m)\Delta\bar{y}(N+m) + \sum_{k=N}^{N+m}(x(k) - \bar{x}(N+m))(y(k) - \bar{y}(N+m)) \quad (9)$$

with $\bar{x}$ the mean value of variable x and $\Delta\bar{x}(N+m)$ the deviation between the mean value of x based on $N+m$ samples and the mean value based on the first N samples. Furthermore, Lemma 1 needs to be extended with an appropriate distance calculation from a new data point to the surface of an ellipsoidal in general position.

3.3. Satellite deletion

In the case of online learning the incremental training scheme in Algorithm 3 may lead to undesirable tiny clusters, called cluster satellites, which lie very close to significantly bigger ones. An example is demonstrated in Fig. 2(a) (the data 'clusterdemo' taken from MATLAB), where the tiny cluster is obviously superfluous. The reason for this unpleasant effect is the following (and could be also observed in other examples): at the beginning of the incremental training process the first points forming the bottom cluster appear at the upper region of this cluster, the cluster had a very narrow range of influence at this stage (see small ellipsis inside and surrounding a bigger dark dot). Afterwards a data point at the lower region came in and formed a new cluster, as being too far away from the small ellipsis in the upper region. This cluster forming at this stage of the incremental (online) learning process was correct, as it seemed that a new cluster arises there. Afterwards, newly loaded data filled up the big hole between these two tiny clusters, causing a movement and an expansion of the upper cluster. Finally, it turned out that they in fact belong to one cluster. This causes an 'overclustering' effect. This should be prevented as it leads to false information when, for instance, performing classifications based on the clustering. In this sense, Algorithm 4 was developed for deleting cluster satellites, which can be applied after Algorithm 3, at the end of the whole learning process, as no prior loaded data are needed.

Algorithm 4. Satellite deletion.

1: **For** all $i = 1, \ldots, C$ clusters generated by Algorithm 3 perform the following steps:
2: **if** $k_i/N < 1\%$ where k_i is the amount of data belonging to the ith cluster and N the number of data points loaded in sum **then**
3: Mark the ith cluster for cutting out, as it only captures outliers
4: Continue with next cluster

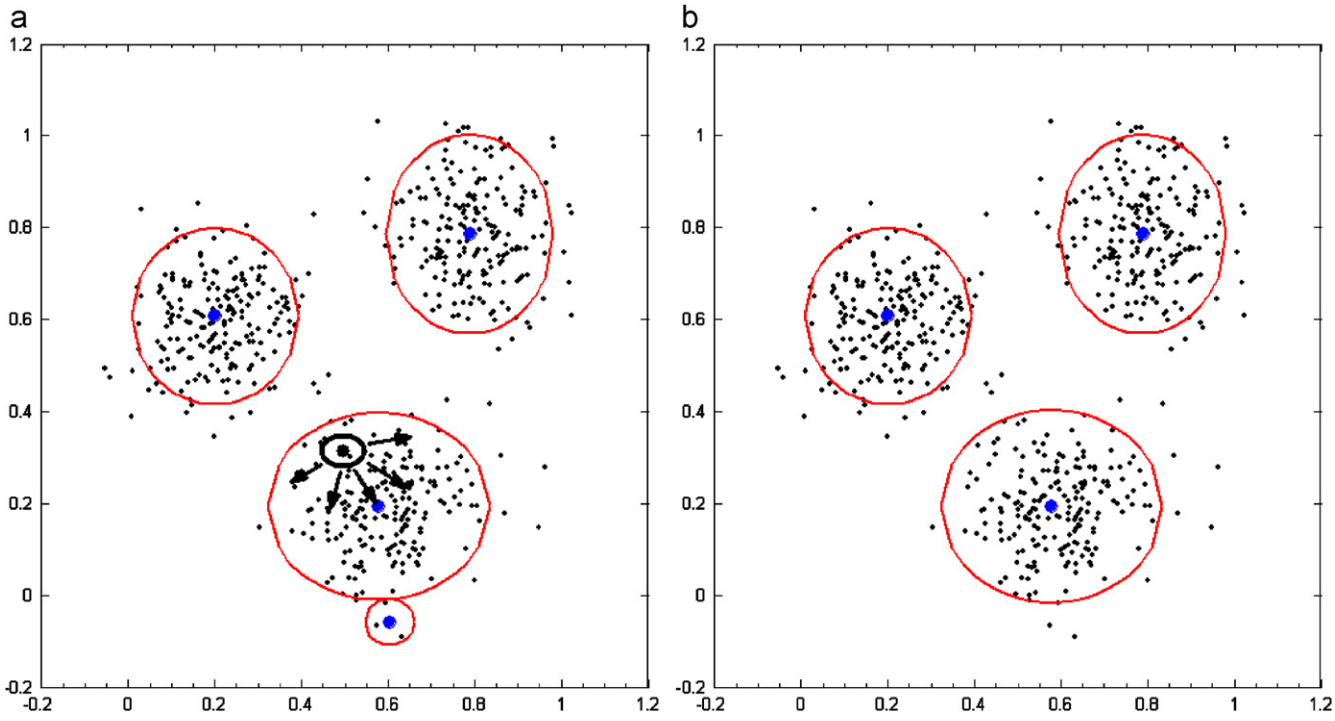

Fig. 2. (a) An unpleasant satellite cluster by applying Algorithm 3; (b) the unpleasant satellite cluster removed by applying Algorithm 4.

5: **end if**
6: **if** $k_i/N < low\_mass$ **then**
7: **if** The ith cluster center lies inside the range of influence of any other cluster **then**
8: Elicit the closest centerc_{win} (like in step 15 of Algorithm 3)
9: If $\sum_{j=1}^{p}\sigma_{ij}/\sum_{j=1}^{p}\sigma_{win,j} < \varepsilon$, where σ_{ij} is the length of the jth axes of the ellipsoid spanned by the ith cluster, then mark the ith cluster for cutting out
10: **end if**
11: **else**
12: Calculate the distance of the ith cluster center to the surface of all other clusters after Eq. Eq. (6) in Lemma 1
13: Elicit the cluster which is closest to the cluster center of the ith cluster, let us say cluster win with $dist_{win}$
14: If $dist_{win} < thr$ and $\sum_{j=1}^{p}\sigma_{ij}/\sum_{j=1}^{p}\sigma_{win,j} < \varepsilon$, where σ_{ij} is the length of the jth axis of the ellipsoid spanned by the ith cluster, then mark the ith cluster for cutting out
15: **end if**
16: **End For**
17: Cut out all marked clusters.

This strategy is based on the investigations, what characterizes in fact a satellite cluster, namely (1) a low mass i.e. a small fraction of data points belonging to it, (2) the cluster center has to be close or inside the range of influence of the (most) adjacent cluster, and (3) the cluster has to be significantly smaller in its range of influence than the (most) adjacent cluster. Moreover, clusters with a very low mass will be marked for cutting out immediately (see steps 2 and 3), as they usually denote outliers. The marking and not directly cutting out the other candidates ensures that satellite clusters of satellite clusters are cut out too.

Note that in Algorithm 4 satellite clusters are completely cut out. This is opposed to the situation where two or more significant clusters move together and an intrinsic penetration of spheres or ellipsoids can be observed. In this case, which can quite often occur during incremental learning steps, well-known cluster merging algorithms can be applied, for instance, Refs. [25,26].

3.4. A split-and-merge strategy during incremental learning

Even though the satellite deletion strategy as described in the previous section deletes tiny, not really distinct clusters as may be generated by Algorithm 3, an incorrect cluster structure still may arise. The reason for this unpleasant effect is a bad setting of the parameter ρ. This parameter has to be defined in advance (before starting the whole incremental training process) and is the essential parameter controlling the tradeoff between generating new clusters and adapting already existing ones. A favorable choice of this parameter is given by formula (3), which is based on our best knowledge that it performed well on various high-dimensional real-world data sets. However, a feasible setting depends strongly on the nature and characteristics of

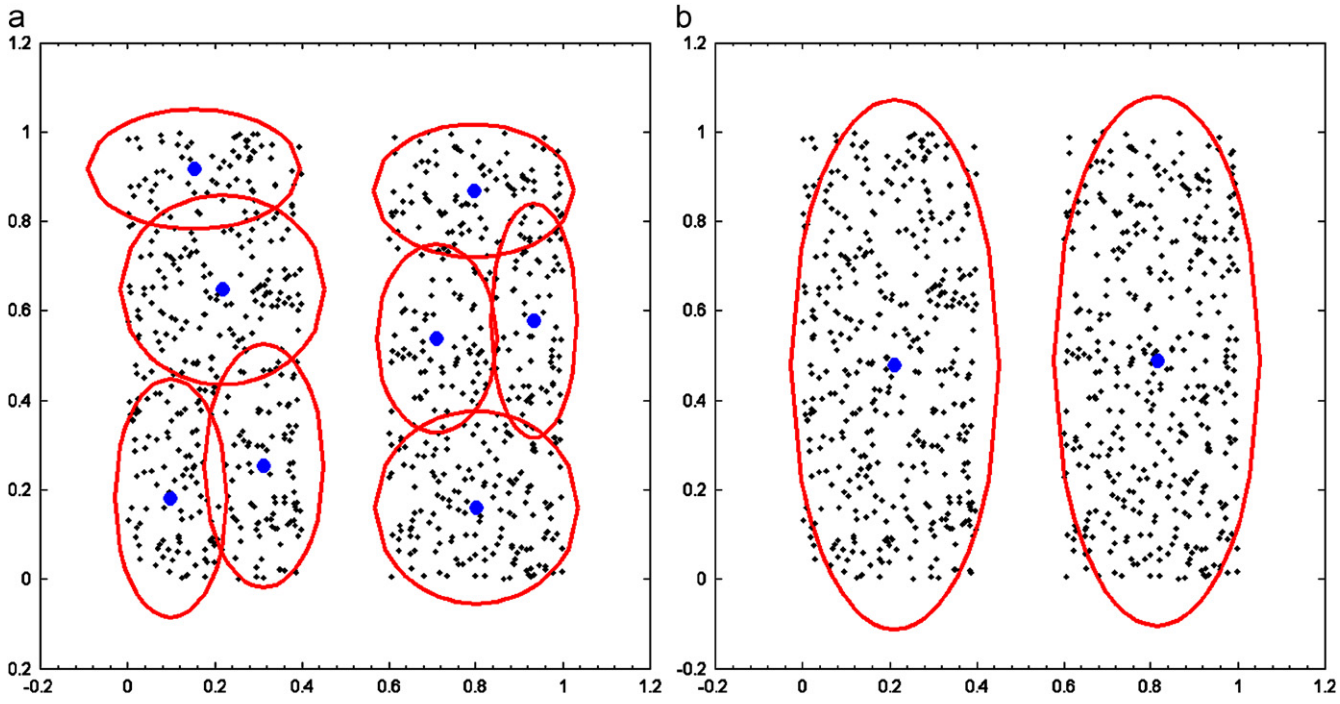

Fig. 3. (a) An incorrect clustering structure obtained by standard setting of ρ as in Eq. (3); (b) correct clustering for the same data when increasing ρ to $0.7\sqrt{p}/\sqrt{2}$.

the data, which is usually not known in advance, when loaded buffer-wise or sample-wise as data streams into the memory. In Fig. 3(a) an example is given, where the default setting of ρ as in Eq. (3) leads to an undesired clustering, as for each of the two big data clouds four clusters are generated. In this sense, a higher value of ρ ($0.7\sqrt{p}/\sqrt{2}$) would have been a better choice resulting in the correct cluster structure as visualized in Fig. 3(b).

For solving this problem for the online case, a split-and-merge strategy of clusters is proposed in this paper (see also Ref. [27]): this strategy is based on the idea that a not-optimal clustering structure which may arise during the incremental learning process is prevented by merging clusters grown together (e.g. as in Fig. 3) or by splitting big clusters including more than one distinct data cloud. After each (sample-wise) incremental learning step, the updated cluster is first split into two halves with respect to each axis and the quality of the obtained clustering structures is calculated with a cluster validation index, which does not need any prior raw data, respectively, with such an index which can be calculated incrementally and synchronously to the cluster update(s). Furthermore, the updated cluster is merged with the closest cluster and the new clustering structure obtained in this way is again validated by the same validation index. The same validation is done for the original clustering obtained after the incremental learning step. The best performing clustering structure out of these three is kept and updated with the next incoming sample. Algorithm 5 gives a concise formulation of this strategy (with C the number of clusters). Please note that in Ref. [28] an online merging strategy for Gaussian mixture models (concurrently seen as clusters) is carried out as well; however, (1) there it is based on some other criteria, i.e. not using cluster validation indices, but multivariate statistical tests and (2) it requires new batches of data containing significant amount of samples (as new mixture models are learned from scratch of each batch) and cannot cope with sample-wise incremental learning.

Algorithm 5. Split-and-merge strategy.

1: Calculate the quality of the actual cluster partition obtained by Algorithm 3 $\rightarrow cl\_qual_1$
2: Split the winning cluster (represented by c_{win}) into two equal halves with respect to each axis and calculate the quality of the obtained p cluster partitions $\rightarrow cl\_qual_2, \ldots, cl\_qual_{p+1}$
3: **if** $C \geqslant 3$ **then**
4: Merge the winning cluster (represented by c_{win}) with the closest cluster. The closest center is obtained in the same way as through steps 9–17 inAlgorithm 3 with $\vec{x} = c_{win}$
5: Calculate the obtained cluster partition $\rightarrow cl\_qual_{p+2}$
6: $max\_ind = \arg\max_{i=1,\ldots,p+2}(cl\_qual_i)$
7: Overwrite the actual cluster partition with that cluster partition represented by max_ind, update the number of clusters C (in case of a conducted split $C = C + 1$, in case of a conducted merge $C = C - 1$)
8: **end if**
9: **if** $C = 2$ **then**
10: Determine if any clustering structure is present at all in the data set (loaded so far)

11: if no, merge the two clusters to one, set $C = C - 1$; if yes, do nothing
12: **end if**.

If this algorithm is carried out after each sample which triggered a cluster update (and not a generation of a new cluster), i.e. immediately after step 25 in Algorithm 3, it is guaranteed that always the clustering structure with the highest quality value is transferred to the next incremental learning step. This means that if the cluster validation index always prefers the more precise clustering structure, a significant improvement of the performance of VQ-INC-EXT with a fixed starting value of ρ can be expected. A suitable cluster validation index is the PS index [29] as in its crisp form it does not require any prior data for calculating the quality of the clustering, just the number of data points $k_.$ belonging to each cluster (resp. those data points which formed the different clusters during the incremental learning process so far):

$$PS(C) = \sum_{i=1}^{C} PS_i \tag{10}$$

with

$$PS_i = \frac{k_i}{k_{max}} - \mathrm{e}^{-\min_{k\neq i}(\|c_i - c_k\|)/\beta_T}, \tag{11}$$

where

$$k_{max} = \max_{i=1,\ldots,C} k_i, \tag{12}$$

$$\beta_T = \frac{\sum_{i=1}^{C}\|c_i - \bar{c}\|^2}{C} \tag{13}$$

and $\bar{c}$ the mean value of all cluster centers. The higher the $PS(C)$ gets, the better the data are partitioned into C clusters. From this definition it is clear that the PS index only needs the cluster centers and the number of data points belonging to each center. Both information are updated by Algorithm 3, such that the PS index can be re-calculated newly after each incremental learning step without requiring any additional update information.

Regarding splitting the winning cluster into two equal axis-parallel halves with respect to dimension j, the following formulas are applied (which do not use any prior data):

$$c_{win,j}(new1) = c_{win,j}(old) + \sigma_{win,j}(old),$$

$$c_{win,j}(new2) = c_{win,j}(old) - \sigma_{win,j}(old),$$

$$\sigma_{win,j}(new1) = \sigma_{win,j}(new2) = \frac{\sigma_{win,j}(old)}{2},$$

$$k_{win}(new1) = k_{win}(new2) = \frac{k_{win}(old)}{2}. \tag{14}$$

The number of data points belonging to the original cluster is divided by 2 and assigned to each one of the two new clusters. This is probably the deficiency of the whole split strategy, as the data points may be unequally distributed within the two halves of the original cluster. A reasonable merging of two clusters can be carried out in the following way ($\forall j = 1, \ldots, p$):

$$c_{win,j}(new) = \frac{c_{win,j}(old)k_{win} + c_{close,j}(old)k_{close}}{k_{win} + k_{close}},$$

$$\sigma_{win,j}(new) = \max(\sigma_{win,j}(old), \sigma_{close,j}(old)) + \left(1 - \frac{\max(k_{win,j}, k_{close,j}) - \min(k_{win,j}, k_{close,j})}{k_{win,j} + k_{close,j}}\right) \times \min(\sigma_{win,j}(old), \sigma_{close,j}(old)),$$

$$k_{win,j}(new) = k_{win,j}(old) + k_{close,j}(old).$$

This guarantees that the center of the merged cluster is not placed exactly in the middle of the two centers belonging to the original clusters, but shifted towards the center of that cluster which is more significant, i.e. which was formed by more data points and hence possesses a higher weight. A similar consideration is valid for the width of the new cluster in each direction (the less significant cluster should influence the new width by only a fraction of the more significant one).

In the case when only two clusters are present, cluster merging as in step 4 in Algorithm 5 is not carried out, as the PS index is not able to discriminate between one or two clusters as optimal partitions. Hence, in this case it has to be determined if a cluster structure is present at all in the data set (loaded so far). This can be achieved with the Hopkins index [30], whose determination can be proceeded in the following way: first, m data samples $R = \{r_1, \ldots, r_m\}$ are randomly selected from the convex hull of the data space of the data set X, i.e. by generating random points within the ranges of each variables of X. Then, further points $S = \{s_1, \ldots, s_m\}$ are randomly selected directly from the data set X, i.e. $S \subset X$. For those sets R and S, the distances $d_{r_1}, \ldots, d_{r_m}$, respectively, $d_{s_1}, \ldots, d_{s_m}$ of R, respectively, S to the nearest points in X are determined. Then the Hopkins index h is defined by

$$h = \frac{\sum_{i=1}^{m} d_{r_i}^p}{\sum_{i=1}^{m} d_{r_i}^p + \sum_{i=1}^{m} d_{s_i}^p} \tag{15}$$

with p the dimensionality of the data set. The incremental calculation of the Hopkins index is quite straightforward, as for each newly loaded data point the distances to R and S can be calculated and the new minimal distances obtained. It has to be taken into account, however, that sometimes the sets R and S need to be changed a bit (R according to possible range extensions in some variables and S according to new data points).

Remark. Another decision criterion for merging two clusters generated by Algorithm 3 lies in the exploitation that two (axis-parallel) ellipsoidal clusters enclose some minimal degree of overlap. If this is the case, it is very likely that the two clusters should be merged to one for a better representation of the data partitioning.

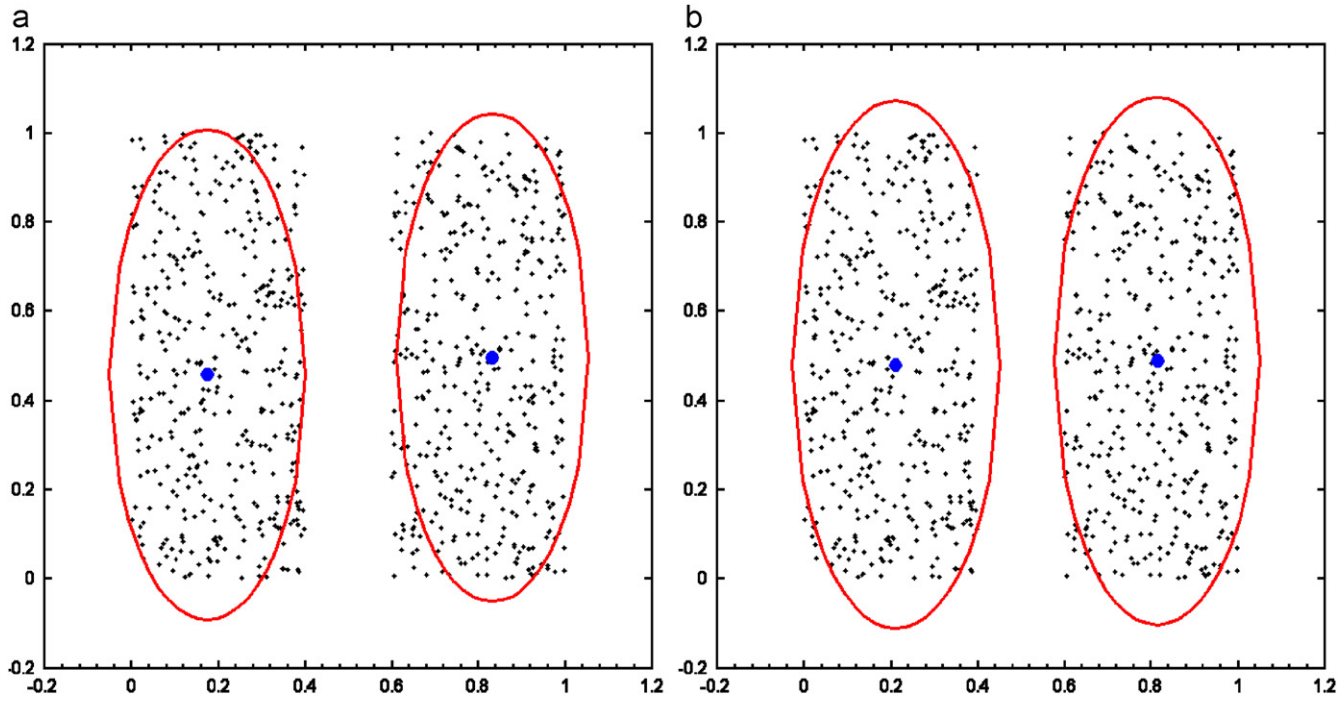

Fig. 4. (a) Clustering structure obtained by standard setting of ρ as in Eq. (3) and split-and-merge strategy as demonstrated in Algorithm 5; (b) clustering structure for the same data when increasing ρ to $0.7\sqrt{p}/\sqrt{2}$ by manual tuning (which is not possible for online clustering processes).

In Fig. 4(a) the obtained cluster partition is shown when applying the conventional parameter setting of ρ as in Eq. (3) together with the split-and-merge algorithm after each incremental learning step. In Fig. 4(b) the cluster partition obtained by a setting of $\rho = 0.7\sqrt{p}/\sqrt{2}$ is shown again for comparison. From this figure it can be recognized that the split-and-merge strategy guides the algorithm to the correct cluster partition, even though the conventional setting of ρ after Eq. (3) would produce a totally incorrect partition, see Fig. 3(a). In this sense, with the split-and-merge strategy it is possible to overcome an inappropriate value of ρ set at the beginning of the incremental training process.

4. Evaluation

In this section we evaluate the following new algorithms:

- VQ-INC: The basic incremental version of vector quantization as described in Algorithm 2.
- VQ-INC-EXT: The extended incremental version of vector quantization based on a new winner selection strategy as described in Algorithm 3.
- evolving Vector Quantization (eVQ): VQ-INC-EXT combined with satellite deletion (Algorithm 4) and split-and-merge strategy (Algorithm 5).

This evaluation demonstrates the performance on some clustering benchmark data sets, on image classification based on extracted clusters in the feature space as well as the applicability for generating high-dimensional fuzzy models by cluster projection onto the axes. For the first case the new approach is compared with conventional vector quantization as well as some well-known clustering methods based on the cluster validation index by Yang and Wu, the PS index [29]. For image classification (into good and bad images) the obtained misclassification rates are compared when using different clustering and batch classification methods for training an image classifier. For the high-dimensional fuzzy modelling tasks the obtained model complexities and accuracies with respect to achieved fault detection rates are compared between the different clustering methods.

Note that all the applied data sets except the online measurements from the engine test bench (Section 4.3) are offline data sets and were converted to pseudo-streams when applying VQ-INC, VQ-INC-EXT, respectively, eVQ. This is carried out in online simulation procedures by taking sample per sample from the data sets and sending them into these algorithms. In this sense, a comparison with reputable batch mode clustering methods (e.g. such as *k*-means) applied onto the data sets as a whole serves as a valuable feedback regarding feasibility of the new incremental techniques. A summary of the application cases, the used data sets and the applied methods as well as some batch methods for comparison is demonstrated in Table 1. A detailed description of the experimental setup, the parameter setting and performance analysis for each of these cases will be given in the subsequent sections.

4.1. Results on clustering data sets

In Table 2 a comparison between different clustering methods (columns) performing on various data sets (rows) containing

Table 1
Overview of application cases, data sets and applied methods described throughout the Evaluation section

Application	Data set	Applied new methods	Methods for comp.
Two-dim. clustering	Data set in Fig. 1	VQ-INC, VQ-INC-EXT	VQ, *k*-means, subclust.
	Clusterdemo data	VQ-INC, VQ-INC-EXT	VQ, *k*-means, subclust.
	Deviation image data	VQ-INC-EXT	*k*-Means, subclust.
High-dim. clustering	Wine data, iris data	VQ-INC, VQ-INC-EXT	VQ, *k*-means, subclust.
Image classification	Online recorded CD-imprint images	VQ-INC, VQ-INC-EXT, eVQ	*k*-Means, subclust., CART, DA, NN
Online FD	Offline data set (engine test bench)	VQ-INC, VQ-INC-EXT, eVQ with consequ. est.	Subclust. with consequ. est.
	Online measurements (engine test bench)	VQ-INC, VQ-INC-EXT, eVQ with FLEXFIS	Subclust. with consequ. adapt.

Table 2
Quality of clusters denoted by the PS index obtained by five different clustering methods including VQ-INC-EXT

Method	Parameter setting	Two-dim. data set 1	Two-dim. data set 2	Iris data	Wine data
k-Means	# of cl. = 3	1.9379/3	2.098/3	1.38/3	1.57/3
Subclust.	Best roc	1.934/2	2.024/3	1.10/3	1.75/3
VQ	# of cl. = 3	1.9370/3	2.214/3	−0.16/3	1.31/3
VQ-INC	ρ by Eq. (3)	0.004/5	−0.5841/6	−0.03/5	−8.59/5
VQ-INC-EXT	ρ by Eq. (3)	1.935/3	2.167/3	1.14/3	1.77/3

cluster structures is demonstrated. The entries ahead the slashes in the matrix correspond to the value obtained from the PS index: the higher this value, the better the quality of the clusters, i.e. the better the centers are set. If, for example, the correct number of clusters present (and known) in the data set is missed, this index will deliver a significant lower value than in the case of the optimal amount. The entries after the slashes are the number of clusters produced by the methods, whereas in all the cases the three clusters are the optimal choice.

For comparison reasons all the clustering methods in Table 2 are crisp clustering methods, which produce centers and hence are mainly applicable for finding clearly separable data clouds in a data set. It should be noticed that for *k*-means [23] and conventional vector quantization [2] the number of clusters has to be set in advance (here to three as the clusters/classes were known a priori), which is not the case for subtractive clustering [13] and the extended versions of vector quantization proposed in this paper. The essential parameter 'radius of cluster' in subtractive clustering is iteratively changed along a pre-defined grid (ranging from 0.05 to 0.9 with step size 0.05) and the best clustering structure kept as the final structure. The corresponding 'radius of cluster' is denoted as 'best roc' in Table 2 (second column). The first three methods (*k*-means, subtractive clustering and VQ) are popular well-accepted batch mode algorithms and hence serve as a welcome benchmark for the two incremental variants of VQ demonstrated in this paper. The chosen data sets include two two-dimensional ones, the first one is the same as that visualized in Fig. 1, the second one is the obtained set when extracting the first and the third column in the 'clusterdemo' data. The latter possesses three clusters, where two are partially merged, but still clearly visible to human eyes. Obviously, for both data sets the new Algorithm 3 can compete with all the other clustering procedures.

For an evaluation based on high-dimensional data the two famous data sets *iris* and *wine* data (available in the UCI repository[1])are applied. While *iris* is a four-dimensional data set, the *wine* data includes 13 dimensions; both data sets contain three classes and hence the number of clusters is set to 3 for *k*-means and conventional vector quantization. For the *iris* data set the performance of Algorithm 3 is second among all methods and slightly worse than the best one, namely *k*-means, whereas VQ-INC totally fails as producing an unacceptable number of clusters (namely 5). For the *wine* data the new extended version of vector quantization can even outperform all the other methods with respect to the cluster quality. This is quite a strong result for the satellite deletion strategy as for this data set VQ-INC-EXT originally produced 13 different clusters, where 10 represented just outliers and no significant clusters. These could be removed by applying the satellite deletion strategy as proposed in Algorithm 4. Again Algorithm 2 could not find the correct number of clusters and produced an unacceptable result. It should be noticed that for all data sets the standard parameter settings for η_i and ρ as presented in Eqs. (4) and (3) are applied together with satellite deletion.

In Fig. 5 another two-dimensional example is presented, which stems from a deviation image (see also next section) containing 17 objects which should be extracted fully automatically from the binarized image. The dark dots represent the foreground pixels whose gray levels are laid over a specific threshold in the original gray-level deviation image. Extracting the correct objects is an important issue whenever features should be extracted and processed further for classifying images (see also next section). From this figure it can be realized that VQ-INC-EXT (lower) extracts more feasible clusters (16 in sum) than *k*-means (upper left) and subtractive clustering (upper right) when choosing the best parameter setting for all these methods: for *k*-means this is the number of objects present in the image (a priori known as 17), for subtractive

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

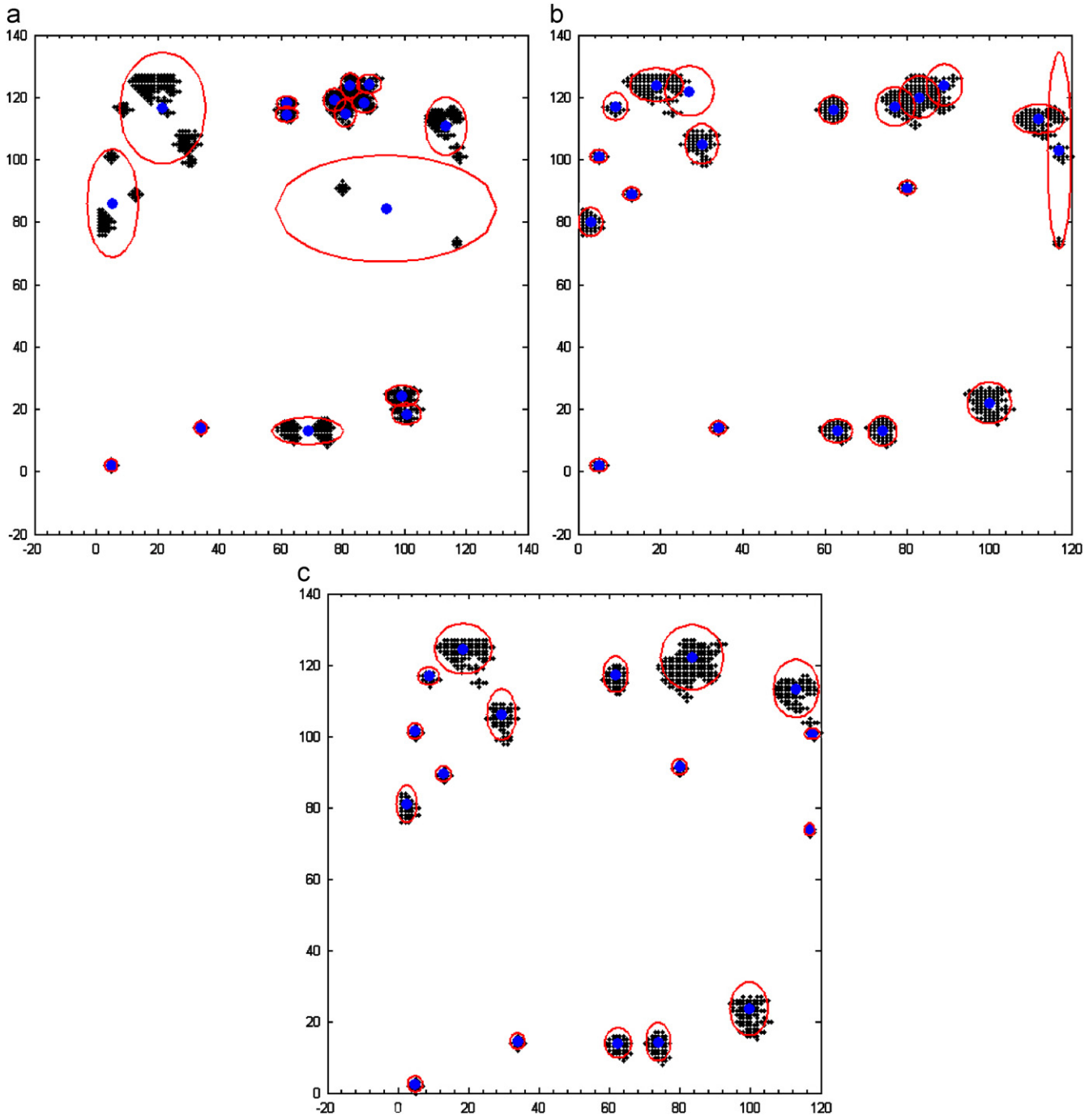

Fig. 5. (a) Extracted objects by applying k-means with 17 number of clusters; (b) extracted objects by subtractive clustering when choosing the best parameter value for radius of influence; (c) extracted objects by VQ-INC-EXT when choosing $\rho = 0.1\sqrt{p}/\sqrt{2}$ as best parameter setting.

clustering the optimal radius of influence was 0.05 and for VQ-INC-EXT the optimal setting of ρ was $0.1\sqrt{p}/\sqrt{2}$. No better cluster structures could be achieved when changing these parameters in each of these three methods.

4.2. Evaluation on image classification based on cluster extraction in the feature space

In this section another evaluation on the new clustering methods VQ-INC, VQ-INC-EXT and eVQ is demonstrated based on image classification into good or bad images. The principal idea of the whole classification framework is shown in Fig. 6. In the low-level processing part newly recorded images are compared pixel-wise with a master image (by subtraction) and the obtained deviation images are used for further processing: first a binarization (for obtaining black/white images from the gray-level images) is performed and afterwards various features are extracted from the recognized objects in the binarized images. These features are collected and stored in feature

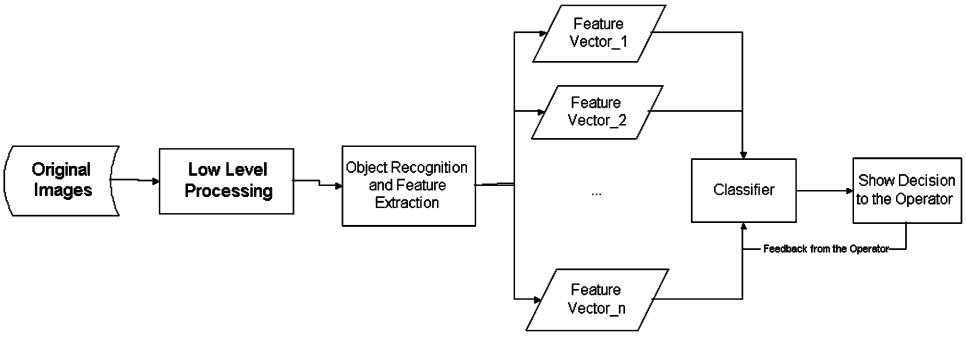

Fig. 6. Image classification framework.

vectors. A classifier is built up based on the feature vectors and based on some images rated by operators during a test or installation phase of the system. The ratings can be carried out on image level or on object level, the latter causing a higher effort for the operator(s). In our specific case for image data from a compact disc production process only a rating on image basis was available, such that aggregated feature vectors, containing features describing the collection of objects in an image as a whole, served as most appropriate information about the characteristics of the images. Examples of aggregated features are the number of objects, the average size of the objects or the maximal density of objects in an image. The images represent CD imprints and may show faulty occasions which came up during the production process. Someone may assume that if a deviation image to the master image contains some pixels at all (so it significantly deviates), it can be immediately classified as a 'bad' image, i.e. an image which contains an error from the production process. However, this is generally not true as some deviations simply are not visible in the original image or do not reflect an error at all (e.g. a small shift of the compact disc in the tray would cause an arc-type object in the deviation image which does not denote an error on the print itself). In this sense, features characterizing different shapes, densities and sizes of objects in an image are needed for a discrimination between bad and good images.

The different clustering approaches already applied in the previous section for clustering data sets are again applied here for clustering the aggregated feature space and building up a nearest-neighbor classifier based on the prototypes (centers) of the obtained clusters: the prototype of each found cluster is marked as a fault-free region of the feature space, when most of the feature vectors representing the cluster are labelled with 0, otherwise it is marked as a faulty region. In this sense a cluster is represented by its center c and a label index l. A new incoming feature vector (extracted from currently recorded/loaded deviation image) is then classified as good or bad according to the value of l of the nearest cluster center. The classification results regarding mis-classification rates are visualized in Table 3. Mis-classification rate is calculated in both directions, i.e. number of samples which are classified as faults, but are actually good ones, plus number of samples which are classified as fault free, but are actually bad ones, relative to the number of samples as a whole. This is done in a 10-fold cross-validation step [34], where the 10 different mis-classification estimates are averaged to the final mis-classification rate. Moreover, the final results stated in Table 3 are obtained by iterative clustering in batch mode, i.e. 10-fold cross-validation is carried out multiple times with different parameter settings of the most essential parameters in the different clustering methods, leading to parameters responsible for the best results: for k-means and VQ the most essential parameter is the number of clusters, for subtractive clustering it is the radius of influence, for VQ-INC and VQ-INC-EXT it is the vigilance parameter ρ. For eVQ the default parameter setting of ρ (see Eq. (3)) is used.

Table 3
Comparison of clustering algorithms when applied for image classification, in the last four rows other well-known classification methods are listed

Method	Mis-class rate (%)	Best param value
k-Means	10.96	# of clusters = 29
Subtractive clustering [13]	5.03	Radius of cluster = 0.2
VQ-INC	5.03	$\rho = 0.1\frac{\sqrt{p}}{\sqrt{2}}$
VQ-INC-EXT	5.01	$\rho = 0.2\frac{\sqrt{p}}{\sqrt{2}}$
eVQ	6.91	Default setting of ρ
Decision trees (CART) [31]	6.71	Optimal pruning
Discr. analysis [20]	6.33	Quadratic boundaries
Prob. neural networks [32]	4.9	Spread = 0.15
Gaussian mixture models [33]	7.86	# of models of p. class = 13

From this table it is clear, that VQ-INC(-EXT) can compete with all other clustering methods, also when using it for offline batch mode training, i.e. by iteratively selecting different settings of ρ and performing incremental clustering and classification. However, in an online data stream mining process, this search procedure for obtaining the best parameter setting is not possible, as data samples are presented just once to the methods. Hence, it was tested how VQ-INC(-EXT) performs, when using a default setting of ρ as proposed in Eq. (3). The accuracy decreased significantly to 11.45% (for VQ-INC-EXT) resp. 11.55% (for VQ-INC) falling behind the worst performing classifier (k-means). In this sense, the importance of eVQ is underlined, which always starts with the default setting and evolves, merges and splits clusters on demand. Note that in Table 3 also the results of some other well-known and widely used batch mode classification methods are stated for comparison

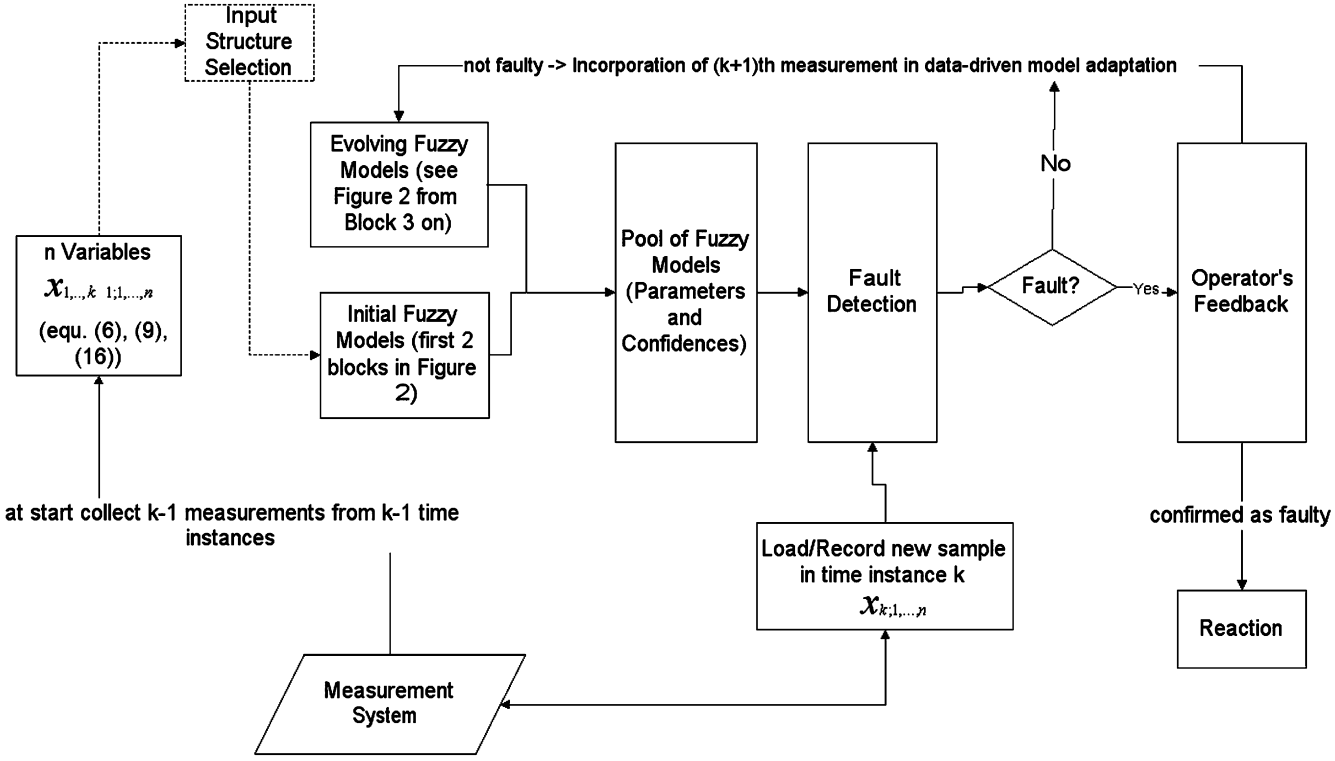

Fig. 7. Fault detection framework in an online measurement system, taken from Ref. [21].

(again achieved by using iterative 10-fold cross-validation with different parameter settings): decision trees by applying the CART approach with optimal pruning strategy [31], discriminant analysis with quadratic boundaries [20] and Gaussian mixture models (trained with EM algorithm [33] and optimized # of Gaussians) produced slightly higher mis-classification rates, whereas probabilistic neural networks [32] with an optimized spread of neurons produced slightly lower mis-classification rates.

4.3. Evaluation on fault detection based on data-driven high-dimensional fuzzy models

For demonstrating practical feasibility in an industrial process, VQ-INC-MOD is applied for generating high-dimensional fuzzy models from online measurement data. This is accomplished on the basis of cluster back-projection onto the one-dimensional space as it is also carried out in approaches such as FMCLUST [15] or genfis2 [13] implemented in MATLAB's fuzzy logic toolbox. For the offline training case, we exchange subtractive clustering in genfis2 one time with VQ-INC, the other time with VQ-INC-EXT and compare the performance of these three methods onto high-dimensional measurement data from an engine test bench with respect to quality and complexity of the models. For the online training case, a similar procedure is carried out, with the difference that conventional genfis2 is only applied for the initial model training and the incremental learning with new online data is conducted by rule consequent adaptation alone exploiting recursive least squares method [35]; VQ-INC and VQ-INC-EXT are connected with the evolving fuzzy modelling method FLEXFIS [17] for the online case.

The high-dimensional fuzzy models are not only compared by their prediction quality and complexity, but also by their fault detection performance when being applied as a fault-free reference situation in a fault detection framework for engine test benches, see Fig. 7. The purpose of this framework is to automatically detect as many system faults as possible, which usually include sensor overheatings, broken pipes or breakdowns of complete components or interfaces. For doing so, it is tried to extract as many dependencies between measurement channels as possible in the form of approximation models, which is done by fixing each channel as target and selecting the most important ones for obtaining a good approximation by an extended version of forward (variable) selection [36]. This guarantees a best probable coverage of channels involved in the fault detection process. For more details about the update strategy of the fuzzy models and the fault detection logic see Ref. [21]. Real faults were simulated at the test bench while recording the data for a specific diesel engine, which should be detected on the basis of the trained fuzzy models by calculating the deviation of actual measurements to the models. In sum, 70 measurement channels were measured, where up to 56 five-dimensional reasonable models could be extracted automatically from the data. This gives a good coverage of channels, when taking into account that some of the remaining 14 appear in the input side of the models. The question whether a model is reasonable or

not could be answered by measuring the quality of the model with *r*-squared-adjusted statistics, which is near 1 if an accurate model is the case and, near 0, if the model has a bad accuracy.

The accuracies of the models are calculated by two measures: a prediction quality measure lying in [0, 1], where a value near 0 denotes an expected bad prediction quality and a value near 1 an expected good one, and an ROC curve representing the sensitivity vs. specificity of the models when applying them as a fault-free reference in a fault detection framework for detecting system faults based on online recorded measurements. The sensitivity is represented by the detection rate, i.e. the percentage of faulty data points which are correctly detected, the specificity by the false detection rate, i.e. the percentage of fault-free data points which are wrongly suggested as faults. In Table 4 the number of correctly detected faults is stated in Column 4 beside the quality (Column 3) and the complexity (Column 2) of the fuzzy models, the latter measured in terms of the average number of rules within the different models. The chosen parameter setting is stated in Column 1, which could be tuned for the offline case (first part) and is set to the recommended default parameter setting for the online case (second part). The detection rates in this table are those ones obtained by setting

Table 4
Comparison of clustering algorithms in connection with fuzzy system training, first part: offline case, second part: online case

Method	Complexity	Quality	Det. rate
Subclust.	5.07	0.9201	70%/75%
VQ-INC	4.5	0.941	72.5%/75%
VQ-INC-EXT/eVQ	3.25	0.942	77.5%/87.5%
Subclust. + consequ. adapt.	9.43	0.793	53.75%/50%
VQ-INC + FLEXFIS	8.24	0.934	66.25%/75%
VQ-INC-EXT/eVQ + FLEXFIS	7.12	0.936	70%/75%

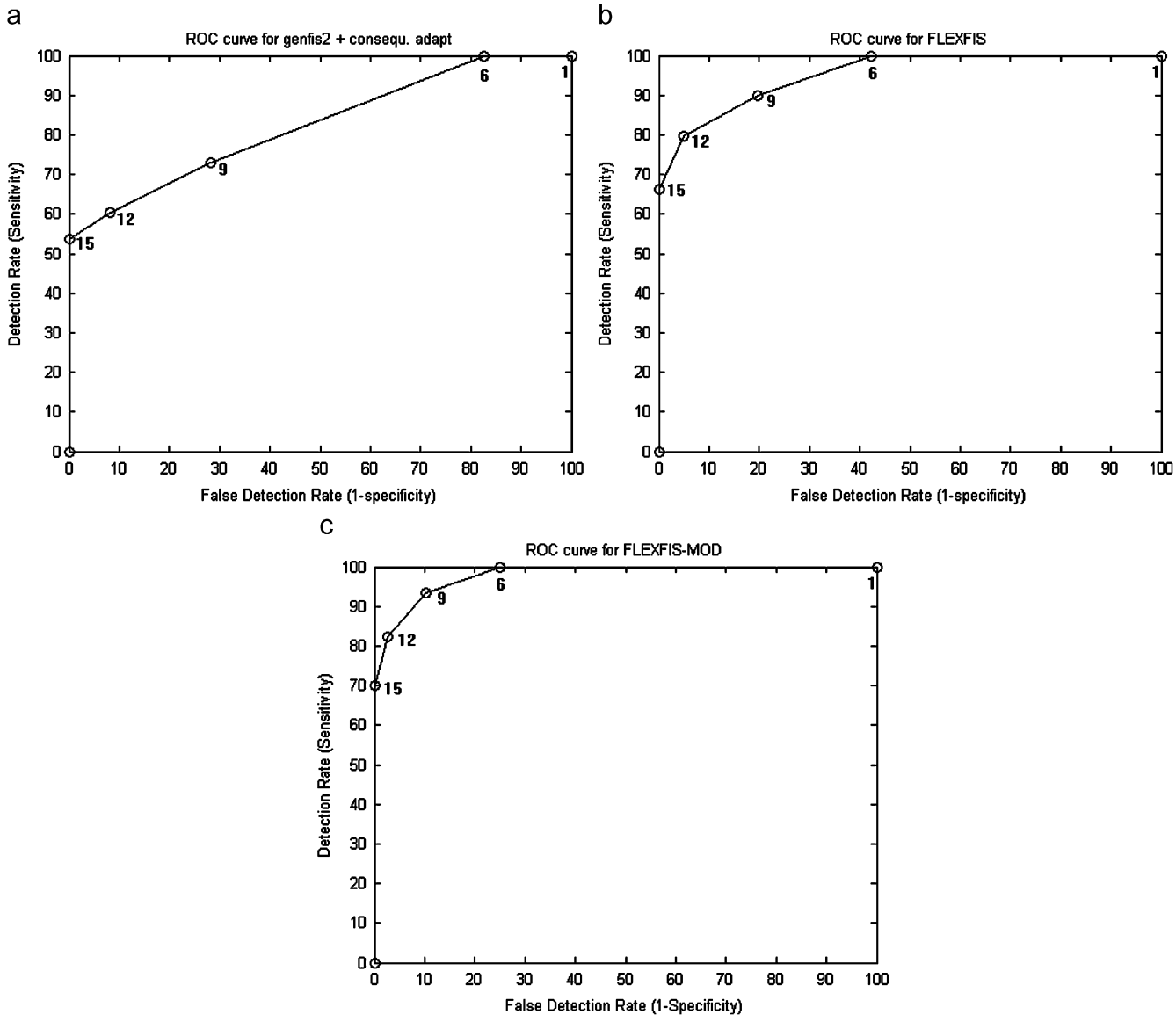

Fig. 8. ROC curves for (a) genfis2 applying subtractive clustering and consequent adaptation; (b) FLEXFIS applying VQ-INC; (c) FLEXFIS-MOD applying VQ-INC-EXT.

the threshold at a level, such that no over-detections occurred, compare ROC curves in Fig. 8. The table is split into two parts: the first one represents the offline case, i.e. models are generated in advance in an offline step (again with different parameter settings, see also Section 4.2) and new online measurements are checked; the second one demonstrates the online case, where fuzzy models have to be adaptively trained and extended (with the default parameter setting). A high-frequented re-building of all the 56 models is not possible, as this would slow down the whole process significantly, such that real-time performance cannot be achieved, see Ref. [17]. In the case of genfis2 using subtractive clustering an adaptation of linear rule consequent parameters alone has to be carried out, as subtractive clustering is a batch clustering variant with no cluster (= rule) adjoining strategy. This is different for VQ-INC and VQ-INC-EXT, which build up the clusters (and therefore rules and fuzzy sets) in an incremental manner. A stable connection of premise part and antecedent part learning in incremental manner is carried out as described in Ref. [17], denoted as *FLEXFIS*. From this point of view, it is quite obvious that subtractive clustering (in connection with genfis2) fails completely opposed to VQ-INC and VQ-INC-EXT (in connection with FLEXFIS); see quality and detection rates in the second part of Table 4 and also the ROC curves as shown in Fig. 8, which compare the methods with respect to specificity vs. sensitivity. Note that the closer the curve follows the left-hand border and then the top border of the ROC space, the better the fault detection performance of the method. For the offline case subtractive clustering performs better, but is still behind the other two approaches with respect to both quality and complexity. The complexity is measured by the average number of rules over the 56 fuzzy models. All the detection rates are measured on two bases: the first number corresponds to the measurement basis i.e. all by a fault affected measurements are counted (in sum 80), the second one corresponds to the fault bases, i.e. all different kinds of faults are counted (in sum eight). It should be noticed that for VQ-INC and VQ-INC-EXT again the standard values for the parameters η_i and ρ were applied, eVQ produced here the same results as VQ-INC-EXT.

5. Conclusion and outlook

A new extended version of vector quantization (VQ-INC-EXT) was proposed. It extends conventional vector quantization to an incremental variant and incorporates a new distance strategy, which takes into account the range of influence of clusters. In this sense, the generation of more clusters within wide data clouds is prevented. A satellite deletion strategy can be appended to any clustering technique producing cluster centers and the range of influence of clusters. This is for removing not really significant clusters or clusters representing just outliers in the data set, which can come up during incremental learning as the future data points are unknown at each learning step. Furthermore, the online split-and-merge strategy guides the incremental clustering process to more accurate cluster partitions, whenever a badly a priori setting of the vigilance parameter ρ was carried out. The connection of VQ-INC-EXT with satellite deletion and split-and-merge strategy leads to the evolving variant of vector quantization, denoted as eVQ. When inspecting the results in Sections 4.1–4.3, it can be realized that VQ-INC-EXT and eVQ can compete with or even outperform conventional clustering methods with respect to the accuracy and quality of the clusters (represented by a well-known cluster validation index), with respect to classification rates and with respect to accuracies of fuzzy models obtained via cluster projection for forming the premise part and least squares antecedent learning afterwards. This is quite a strong result, as the new method acts on the data set on a point-per-point basis and therefore can be applied for data streams in fast online applications or for huge data bases, which cannot be loaded into virtual memory at once.

As the split-and-merge strategy may suffer from computation speed (as PS index is calculated for several clustering partitions), a special emphasis for a possible future work is placed on an adaptation of the vigilance parameter itself based on changing characteristics of the incoming data. Another focus concerns strategies on reacting to drifts or shifts in the data. The first one changes the distribution of the underlying data smoothly over time, such that approaches for weighting older data points less than newer ones are required. The latter triggers abrupt and sudden changes of the data characteristics, such that a complete resetting of the cluster structure obtained so far may be required during online processing.

Acknowledgments

The author thanks Eyke Hüllermeier (University of Marburg) for valuable discussions and comments on the manuscript of this paper. This work was supported by the European Commission (project Contract No. STRP016429, acronym DynaVis). This publication reflects only the author's view.

References

[1] J. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Kluwer Academic/Plenum Publishers, USA, 1981.
[2] R. Gray, Vector quantization, IEEE ASSP Mag. (1984) 4–29.
[3] T. Kohonen, Self-Organizing Maps, second extended ed., Springer, Berlin, Heidelberg, Germany, 1995.
[4] T. Martinetz, S. Berkovich, K. Schulten, Neural gas network for vector quantization and its application to time-series prediction, IEEE Trans. Neural Networks 4 (4) (1993) 558–569.
[5] B. Fritzke, A growing neural gas network learns topologies, Adv. Neural Inf. Process. Syst. 7 (1995) 625–632.
[6] T. Kohonen, Self-Organization and Associative Memory, second ed., Springer, Berlin, 1987.
[7] L. Golab, M. Tamer, Issues in data stream management, SIGMOD Rec. 32 (2) (2003) 5–14.
[8] M. Garofalakis, J. Gehrke, R. Rastogi, Querying and mining data streams: you only get one look, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, ACM Press, New York, 2002, pp. 635–641.
[9] A. Same, C. Ambroise, G. Govaert, A mixture model approach for online clustering, in: COMPSTAT'2004 Symposium, Physica Verlag/Springer, 2004.
[10] O. Arandjelovic, R. Cipolla, Incremental learning of temporally-coherent gaussian mixture models, in: British Machine Vision Conference 2005, 2005.

[11] J. MacQueen, Some methods for classification and analysis of multivariate observations, in: Proceedings of Fifth Berkeley Symposium on Mathematics, Statistics and Probability, 1967, pp. 281–298.
[12] P. Angelov, Evolving Rule-Based Models: A Tool for Design of Flexible Adaptive Systems, Springer, Berlin, Germany, 2002.
[13] S. Chiu, Fuzzy model identification based on cluster estimation, J. Intell. Fuzzy Syst. 2 (3) (1994) 267–278.
[14] D. Gustafson, W. Kessel, Fuzzy clustering with a fuzzy covariance matrix, in: Proceedings of IEEE CDC, San Diego, CA, USA, 1979, pp. 761–766.
[15] R. Babuska, H. Verbruggen, Constructing fuzzy models by product space clustering, in: H. Hellendoorn, D. Driankov (Eds.), Fuzzy Model Identification: Selected Approaches, Springer, Berlin, 1997, pp. 53–90.
[16] O. Nelles, Nonlinear System Identification, Springer, Berlin, Germany, 2001.
[17] E. Lughofer, E. Klement, FLEXFIS: a variant for incremental learning of Takagi–Sugeno fuzzy systems, in: Proceedings of FUZZ-IEEE 2005, Reno, NV, USA, 2005, pp. 915–920.
[18] M. Halkidi, Y. Batistakis, M. Vazirgiannis, On clustering validation techniques, J. Intell. Inf. Syst. 17 (2/3) (2001) 107–145.
[19] G.A. Carpenter, S. Grossberg, Adaptive resonance theory (art), in: M.A. Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, MIT Press, Cambridge, MA, 1995, pp. 79–82.
[20] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer, New York, Berlin, Heidelberg, Germany, 2001.
[21] P. Angelov, V. Giglio, C. Guardiola, E. Lughofer, J. Luján, An approach to model-based fault detection in industrial measurement systems with application to engine test benches, Meas. Sci. Technol. 17 (2006) 1809–1818.
[22] E. Lughofer, E. Klement, J. Lujan, C. Guardiola, Model-based fault detection in multi-sensor measurement systems, in: Proceedings of IEEE IS 2004, Varna, Bulgaria, 2004, pp. 184–189.
[23] K. Alsabti, S. Ranka, V. Singh, An efficient *k*-means clustering algorithm, in: IPPS: 11th International Parallel Processing Symposium, IEEE Computer Society Press, Silver Spring, MD, 1998Available: ⟨citeseer.nj.nec.com/alsabti98efficient.html⟩ (Online).
[24] S. Qin, W. Li, H. Yue, Recursive PCA for adaptive process monitoring, J. Process Control 10 (2000) 471–486.
[25] R. Krishnapuram, C. Freg, Fitting an unknown number of lines and planes to image data through compatible cluster merging, Pattern Recognition 25 (4) (1992) 385–400.
[26] F.-H. Rhee, B.-I. Choi, A convex cluster merging algorithm using support vector machines, in: Proceedings of the 12th IEEE International Conference on Fuzzy Systems, vol. 2, 2003, pp. 892–895.
[27] J. Beringer, E. Hüllermeier, Online clustering of parallel data streams, Data Knowl. Eng. 58 (2) (2006) 180–204.
[28] M. Song, H. Wang, Incremental estimation of gaussian mixture models for online data stream clustering, in: Proceedings of the International Conference on Bioinformatics and its Applications, Fort Lauderdale, FL, USA, 2004.
[29] M. Yang, K. Wu, A new validity index for fuzzy clustering, in: Proceedings of the IEEE International Conference on Fuzzy Systems, Melbourne, Australia, 2001, pp. 89–92.
[30] A. Jain, R. Dubes, Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ, 1988.
[31] L. Breiman, J. Friedman, C. Stone, R. Olshen, Classification and Regression Trees, Chapman & Hall, Boca Raton, 1993.
[32] P. Wasserman, Advanced Methods in Neural Computing, Van Nostrand Reinhold, New York, 1993.
[33] R. Duda, P. Hart, D. Stork, Pattern Classification, second ed., Wiley-Interscience, Chichester, UK, 2000.
[34] M. Stone, Cross-validatory choice and assessment of statistical predictions, J. R. Stat. Soc. 36 (1974) 111–147.
[35] L. Ljung, System Identification: Theory for the User, Prentice-Hall PTR, Upper Saddle River, NJ, 1999.
[36] W. Groißböck, E. Lughofer, E. Klement, A comparison of variable selection methods with the main focus on orthogonalization, in: M. Lopéz-Díaz, M. Gil, P. Grzegorzewski, O. Hryniewicz, J. Lawry (Eds.), Soft Methodology and Random Information Systems, Advances in Soft Computing, Springer, Berlin, Heidelberg, New York, 2004, pp. 479–486.

About the Author—EDWIN LUGHOFER received his Master Degree in Technical Mathematics from the Department of Knowledge-Based Mathematical Systems at the Johannes Kepler University of Linz, Austria. From November 1997 onwards, he is employed at the Department of Knowledge-Based Mathematical Systems, where he participated in several research projects like, for instance, EU-Project Automatic Measurement Plausibility and Quality Assurance (AMPA), EU-Project Dynamically Adaptive Vision Systems (DynaVis), exchange of know-how in data-driven evolving fuzzy systems in cooperation with Lancaster University (sponsored by the Royal Society Grant, UK), technology transfer sponsored by the Upperaustrian Government and Strategic (National) Project Fast Model Prototyping (FMP). From 2002 onwards, he started worked on his PhD Thesis in the field of data-driven incremental learning and evolving fuzzy systems, which he finished in February 2005. His current research topics include data-driven modelling (especially evolving fuzzy systems and incremental algorithms), machine learning, data mining, fault detection, image processing and fuzzy control. Within these fields he has written and published several papers in cooperation with the Department of Communications Systems InfoLab21 at Lancaster University, Faculty for Informatics (ITI) at Otto-von-Guericke-Universität, Department of Telecommunication and Media Informatics, Budapest University of Technology and Economics, Department of Computer Science, The Australian National University, Spark Ignition Engines and Fuels Department, Istituto MotoriI C.N.R., CMT, Motores Térmicos Universidad Politécnica de Valencia, AVL List Gmbh Graz. In September 2006, he received the best paper award for the invited paper 'Process safety enhancements for data-driven evolving fuzzy models' at the International Symposium on Evolving Fuzzy Systems, Lake District, September 2006.