Information Sciences 304 (2015) 54–79

Contents lists available at ScienceDirect

Information Sciences

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

Autonomous data stream clustering implementing split-and-merge concepts – Towards a plug-and-play approach

Edwin Lughofer [a,*], Moamar Sayed-Mouchaweh [b]

[a] Department of Knowledge-based Mathematical Systems/Fuzzy Logic Laboratorium Linz-Hagenberg, Johannes Kepler University Linz, Austria
[b] Ecole des Mines de Douai, Computer Science and Automatic Control Lab EMDouai-IA, Douai, France

ARTICLE INFO

Article history:
Received 1 August 2014
Received in revised form 26 November 2014
Accepted 10 January 2015
Available online 29 January 2015

Keywords:
Autonomous evolving clustering
On-line data streams
Incremental split-and-merge
Cluster heterogeneity
Cluster homogeneity
Parameter sensitivity

ABSTRACT

We propose a new clustering method, which is dynamic in the sense that it updates its structure (cluster partition) permanently based on new incoming data samples. As it basically operates in single-pass and sample-wise manner without using time-intensive re-training phases, it is applicable for extracting clusters from on-line data streams. The approach builds upon an extended, dynamic version of *evolving vector quantization*, generalizing it to prototype clusters with convex (ellipsoidal) shapes in arbitrary positions. It includes incremental split-and-merge techniques in order to resolve *cluster fusion* and *cluster delamination* effects. The merging process is guided by geometrical criteria indicating overlapping clusters with joint *homogeneity* areas, and implements a fast fusion of two clusters fulfilling the criteria. The splitting process relies on concepts of seeking for *heterogeneity* (disjoint density areas) within already extracted clusters. This is based (1) on components (mixture models) identification while measuring their reliability (a) with the *Bayesian information criterion (BIC)* and (b) in terms of a proper convergence of the EM algorithm and (2) on an extended form of the Welch test for verifying whether the identified components are statistically independent – the extension concerns the integration of their mixing proportions. Finally, split-and-merge operations are able to automatically compensate inappropriate settings of the method's learning parameter, thus still being able to extract the desired cluster structure. Thus, our approach presents an attempt towards a parameter-free evolving clustering algorithm. This is an essential property as several trials with different settings are often not desired (in case of huge streams) or not possible at all (in case of on-line streams). Based on high-dimensional real-world clustering streams, our approach will be compared with alternative incremental as well as batch prototype-based clustering methods. The comparison will be based on the *quality* of the final obtained cluster partitions, the deviation of the extracted number of clusters to the real number, the *sensitivity* with respect to the most influencing learning parameters and the *plug-and-play capability* of the methods.

1. Introduction

1.1. Motivation and state-of-the-art

In today's real-world systems, the clustering of data plays an essential role in various fields of applications such as data compression, data mining, grouping and similarity analysis, classification and also system identification [70,37,41]. Within

* Corresponding author.
E-mail address: edwin.lughofer@jku.at (E. Lughofer).

http://dx.doi.org/10.1016/j.ins.2015.01.010
0020-0255/

the latter two, clustering is applied to supervised problems in order to extract patterns and local regions for a more detailed representation of functional dependencies for regression [7,4,66] and classification [55,31]. Upon a wide variety of different types of clustering, such as density-based (e.g. [32,47]), hierarchical (e.g. [39,96]), fuzzy [70], spectral- or graphical-based (e.g. [84,17], or [49]) (surveys can be found in [37] or [94]), *prototype-based clustering* ([14], first part of [30]) is one of the most widely applied choices, as it delivers prototypes in form of cluster centers which can be seen as representatives for the local data clouds they model. In a data mining context, the further processing of models may then continue based on the more compact and transparent representation. This usually increases the understandability of the models as well as the computational resources they require to assign new queries to clusters (classes) or to update the whole cluster structure. This is also because the appearance of prototype-based clusters can be characterized by compact mathematical representations (formulas), rather then a loose collection of partition sets. In case of data coming from different local sites, prototypes are very useful to be able to extract a joined optimized spatial representation of the data groups, see e.g. [75,74,72].

Updating the model structure is often necessary due to the increasing complexity and the on-line characteristics of real-world systems. This requires significant dynamics in the modeling process in order to account for changes and expansions in the operation modes/conditions [79] and also for drift occurrences [63,83]. Evolving cluster models (ECM), an important subfield of evolving intelligent systems [2], are designed to handle these dynamics and process the huge amount of on-line data streams [36,12] or even very large data bases (VLDBs)[1] in a fast adaptive manner. The usage of prototype-based clustering variant in an evolving context is even more popular than in batch mode, as data compression and fast handling of data stream representatives often play an important role [6,87]. Furthermore, (evolving) prototype-based clustering enjoys a wide attraction in the field of various machine learning and soft computing techniques for achieving an appropriate degree of non-linearity when extracting classifiers or regression models from data [20,5]. For instance, in RBF neural networks design [82], prototype-based clustering is sometimes used for obtaining ideal positions of neurons, according to the real distribution of local data clouds [46]. Another example is in the field of data-driven design of fuzzy systems, where prototype-based clustering is used to obtain rules for a local partitioning [73,7]; in evolving fuzzy systems, which are incrementally trained from data, often evolving prototype-based cluster models are employed, coupled with a projection concept [76] in order to form the fuzzy sets appearing in the rules' antecedent parts, see [58] for a comprehensive survey. Such clusters are also often advantageous for an enhanced interpretation of (fuzzy) models [62].

An important aspect for an appropriate handling of data streams in evolving prototype-based cluster models is the usage of *incremental learning steps*, which offer a reasonable balance between plasticity and stability, according to the plasticity–stability dilemma [1]; the former is for introducing sufficient flexibility, especially in case of high dynamics or drift situations, the latter for guaranteeing convergence, preventing highly fluctuating models (clusters) – see also [43]. An essential property of incremental single-pass training is that it is linear in terms of the number of samples. This is opposed to so-called *wrapper approaches* (such as [90,21]), which perform a sort of re-training over recent past batches of the data. An essential problem therein is to determine an appropriate size of the batch. Training sets that are too large could stall the learning process and prevent the stream from being processed at an acceptable speed. Training sets that are too small will induce models that are poor at generalizing to new examples.

Several incremental learning methods for prototype-based clustering (mostly in single-pass manner) have been proposed in recent literature, for instance single-pass k-means [33], dynamic fuzzy k-nearest neighbors clustering [44], a recursive variant of subtractive clustering termed as *eClustering* [3], a recursive Gustafson–Kessel approach [29] and an evolving version of it [34], evolving neural-type models based on neural gas [88], evolving self-organizing maps (ESOM) [26], or the approach in [92] within the application of time series data, to name a few – for a recent overview and comprehensive list of references, see [15,36]. Most of these extract ellipsoidal clusters in main position (axes-parallel). Some others (such as [29,18]) can model rotations in the data clouds, but are not equipped with fast merging operations. Thus, they are not able to resolve unnecessary overlaps or to dynamically compensate inappropriately chosen learning parameters (leading to over-clustering effects). In [85,42,23], incremental approaches for learning Gaussian mixture models are presented, which can be equivalently treated as ellipsoidal clusters in arbitrary position (by taking the inverse covariance matrix as shape descriptor and the Mahalanobis distance radius as contour descriptor). However, they require the learning of additional weight parameters for the overall model representation (weighted sum). Furthermore, either no splitting options are integrated (in [85]) or the merging process is quite complicated and time-intensive ([42,23] or [86]). In [9], an approach was presented which tackles this issue by integrating a block-wise split-and-merge concept into a k-means algorithm. For a whole block of samples, only one cluster can be split or two can be merged, i.e. the number of clusters either decrease or increase by only 1 or remain the same. Furthermore, it employs feature-based clustering (for grouping similar time series together), rather than a sample-based one.

1.2. Our approach

1.2.1. Methodology

We present a new approach which is based on the established *evolving vector quantization (eVQ)* method as published in [56] and successfully applied as fast learning engine for evolving fuzzy systems [57] and classifiers [65]. It extends *eVQ* by evolving and incrementally updating ellipsoidal clusters in arbitrary position rather than being restricted to clusters in main

[1] http://en.wikipedia.org/wiki/VLDB.

position (axis-parallel ellipsoids), thus termed as *eVQ-A*. This yields more flexibility in terms of modeling rotated data clouds with arbitrary orientations occurring in data streams (see Section 2). In [87], also evolving vector quantization concepts are exploited. However, these are used in the context of compressing data streams with the support of so-called 'volume prototypes'. These prototypes indeed appear as multi-dimensional ellipsoids in arbitrary position, however do not necessarily represent compact and disjoint data clouds uniquely by single clusters. Thus, this also often results in inevitable overlapping prototypes within the same cloud (see Figs. 15 and 16 in [87]). In the context of the approach in [87], this is not considered as an undesired property, as the purpose for using these types of clusters is more to extract small micro-clusters, which, joined together, form one cluster of arbitrary shape. A similar consideration goes to the approach in [18], where Gaussian mixture models are grown for the same purpose (conducting micro-clustering).

In our approach, we are aiming for grouping the data into local regions (density areas) as well as possible, following the natural appearance of local data clouds and aiming for distinct, compact and unique clusters (one for each density area) with unique prototypes (centers). Thus, in our approach, we are primarily focussing on *prototype-based* stream clustering *supported by convex shapes*, where centers = prototypes are expected to be well defined and nicely lying within their corresponding sample clouds (appearing in convex form). If there are occurring more complex clusters in the stream, then our method will automatically produce more clusters than actually required splitting the real cluster shapes into ellipsoidal pieces – in a similar manner as achieved in [87]. Over-clustering is then usually caused, but distinct objects in the data set will still be split up into distinct clusters.

Furthermore, our approach is equipped with a dynamic (single-pass) cluster merging strategy. This does not only resolve significant overlaps arising over time as clusters may move together [64], but also compensates slight overlaps or touching clusters, which form a reciprocal homogeneity of their shape and direction (see Section 3). Moreover, a splitting strategy resolves the situation that two density areas are crystallized over time within one cluster. Compared to [60], dynamic cluster merging and splitting was conducted for *prototype-based convex clusters in main position*, whereas in our new approach, termed as *eVQ-AMS* (short for *evolving Vector Quantization for Arbitrary ellipsoids with Merge-and-Split functionality*), all the merging and splitting operations are developed for *prototype-based convex clusters in arbitrary position*. This yields a higher flexibility whenever rotated data clouds appear as will be verified in Section 6. From the mathematical point of view, the difference lies in handling fully-occupied inverse covariance matrices instead of diagonal ones, which makes all the operations regarding cluster evolution and update strategy, the merging process as well as the splitting strategy more sophisticated. Moreover, we will perform a deeper study on the *plug-and-play capability* of our method (which has been not achieved in [60]), see also the paragraph below.

As merging criterion, we apply geometrical criteria indicating cluster overlaps and joint cluster homogeneity: only when overlapping clusters form a joint homogenous regions, a merge is suggested. For the merging process itself, a fast fusion of two clusters, i.e. of their inverse covariance matrices and prototypes, is implemented. The criterion whether to split an updated cluster or to continue with the current partition relies on concepts of seeking for an undesired heterogeneity (disjoint density areas) within the cluster. This is based (1) on components (mixture models) identification while measuring their reliability (a) with the *Bayesian information criterion (BIC)* and (b) in terms of a proper convergence of the EM algorithm and (2) on an extended form of the Welch test for verifying whether the identified components are statistically independent – the extension concerns an integration of their mixing proportions. The *split point* is thereby elicited as the saddle (cut) point of two statistically different components.

Finally, the merging and splitting options are not only for the purpose to obtain more homogenous cluster partitions, but also to compensate an inappropriate setting of the learning parameter – a threshold in our case controlling cluster evolution versus cluster update, causing either over-clustering (threshold is set too low) or under-clustering effects (threshold is set too high). In this sense, our approach can be seen as a step towards a *parameter-free* evolving clustering method. In an incremental on-line learning scenario, this is an essential property, as usually evolving methods are preferred to directly run on new data streams with little pre-parameterization or even to act in a kind of autonomous *plug-and-play* manner, not allowing any parameter optimization steps within time-intensive re-training and re-validation phases [79,50] – an issue which is also important in case of using (incremental) clustering techniques for *Big Data* [22] (only one scan of the data base possible). Moreover, it is not possible to fully optimize learning parameters at any certain point of time for future on-line data stream samples.

1.2.2. Evaluation strategy

We compare our new approach with original *eVQ*, several other incremental, evolving as well as batch learning clustering approaches based on two-dimensional as well as high-dimensional streaming clustering sets. In these, the cluster partitions as well as features ranges show an evolving behavior over time and the number of clusters present in the whole stream is known. The comparison is also based on the obtained cluster partition quality measured in terms of a well-known separability criterion. The batch clustering approaches serve as a kind of hard benchmark as having the advantage to see all data samples at once. Furthermore, a sensitivity analysis will be included in order to verify the impact of the essential learning parameter(s) in all methods on the cluster partition quality. In this context, we will show that the essential learning parameter (steering cluster evolution versus update) in our method will have a quite tolerant variation range for resulting in exactly the same cluster partition. This range is much narrower in case of corresponding learning parameters in the other methods. Additionally, we observe the *plug-and-play* capability of the method by applying the optimized learning parameters from the two-dimensional streaming data onto the high-dimensional ones (without any tuning steps).

2. Evolving cluster ellipsoids in arbitrary position

Recall the original (conventional) *eVQ* algorithm as published in [56]:

(1) For each new incoming sample $\vec{x}$, it elicits the cluster whose center coordinates are closest to it; this is denoted as the *winning cluster (neuron)*.
(2) It checks whether the new incoming sample matches the current cluster partition: the *distance* of the new point to the winning cluster is compared against a threshold (vigilance) $vigi = fac * \frac{\sqrt{p}}{\sqrt{2}}$, with p the dimensionality of the feature space and *fac* a multiplication factor in $]0,1]$.
(3) If it exceeds this threshold, a new cluster is evolved by setting its center to the sample $\vec{c} = \vec{x}$ and initializing its spread vector $\vec{\sigma} = \vec{0}$.
(4) If it does not exceed the threshold, the center and spread of the winning cluster are updated. The update of the center is performed by minimizing the expected squared quantization error (as done in conventional vector quantization [38]):

$$E = \int \|\vec{x} - \vec{c}\| p(\vec{x}) d\vec{x} \tag{1}$$

with $p(\vec{x})$ a continuous probability density function. Its approximation scheme can be derived as follows [51]:

$$\vec{c}_i(N+1) = \vec{c}_i(N) + \eta_i \delta_{win;i}(\vec{x} - \vec{c}_i(N)) \tag{2}$$

with $\delta_{win;i}$ only equal to 1, whenever $win = i$, thus in each incremental update cycle only the winning cluster is updated. According to [38], η_i should be a small learning rate that satisfies the classical Robbins-Monro conditions, i.e. $\sum_{N=1}^{\infty} \eta(N) = \infty$, but $\sum_{N=1}^{\infty} \eta_i^2(N) < \infty$. Therefore, and due to some nice convergence properties in evolving clustering [56] and in evolving fuzzy systems design when using *eVQ* as rule learning engine [57,61], the learning rate was chosen to decrease with the number of samples by $\eta_i = 1/k_i$, with k_i the number of samples belonging to cluster i so far, i.e. the number of samples for which c_i was the winning cluster. In this sense, each cluster has its own learning rate and thus its own flexibility/stability for incremental movements according to its current support.

The distance calculations in above itemization points are performed by the Euclidean distance, thus triggering prototype-based clusters with ellipsoidal shapes in main position. Thus, their principal axes are parallel to the input feature axes. Hence, the update of the spread can be performed for each input dimension separately as conducted by the recursive variance formula including rank-1 modification [77]. The clustering, however, may suffer from inappropriate representations of data clouds (density areas) which are not following the trend of the coordinate axes, see the solid ellipsoid in Fig. 1.

By introducing the Mahalanobis distance as defined by [68]:

$$d(\vec{x}, \vec{c}) = \sqrt{(\vec{x} - \vec{c})^T Cov^{-1} (\vec{x} - \vec{c})} \tag{3}$$

with Cov^{-1} the inverse covariance matrix, one can achieve ellipsoidal clusters in arbitrary position, i.e. the necessary rotation required according to the direction of the data cloud, see the dotted ellipsoid in Fig. 1 for an example. An arbitrary ellipsoid (cluster) in multi-dimensional space is then defined by all samples fulfilling the following condition

$$\sqrt{(\vec{x} - \vec{c})^T Cov^{-1} (\vec{x} - \vec{c})} \leqslant r \tag{4}$$

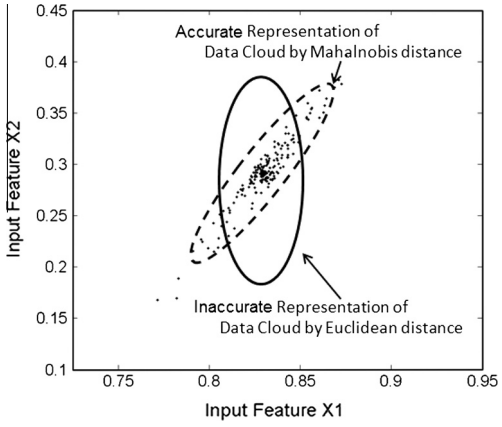

Fig. 1. Representation of an ellipsoidal data cloud by a cluster using Euclidean distance (solid ellipsoid) and by a cluster using Mahalanobis distance (dotted ellipsoid).

with r the Mahalanobis distance radius. This changes the formal description of a cluster i, as it is not defined by $(\vec{c}_i, \vec{\sigma}_i, vigi)$ any longer, but by $(\vec{c}_i, Cov_i^{-1}, r_i)$. We will see below that, instead of a global vigilance *vigi* applied on cluster centers, the Mahalanobis distance (tolerance) radius r will decide whether a new cluster should be evolved or not. More flexibility will be introduced due to different r_i's for different clusters, according to their support.

It is important to note that the Mahalanobis distance is *scale-invariant* [67], thus no normalization of the features is required to get valid distance calculations (which is opposed to Euclidean distance, for which this normalization is indispensable). Thus, in case of conventional *eVQ* the ranges of the features have to be estimated and updated through the whole stream, which prevents the usage of the method from scratch: some initial data is needed to obtain reliable estimates of the ranges and to avoid over-clustering effects, see also [58], Section 3.1.1.2. (discussion on the range problematic). Another strong property of the Mahalanobis distance is that it automatically integrates the spread of a cluster: a new sample may be far away from a cluster center, but close to the principal surface spanned by the Mahalanobis distance radius (range of influence), thus matching the current partition. In such cases, applying Euclidean distance as done in conventional *eVQ* may evolve unnecessary clusters close to or lying even inside already existing clusters. Finally, (inverse) covariance matrices (used in Mahalnobis distance calculations) are known to suffer from curse of dimensionality less than mathematical constructs inducing more complex shapes (e.g. one-Class SVMs [80,69] or general convex hulls), as they can be also equipped with banding or tapering procedures having nice convergence properties [11]. In this sense, our approach may be beneficial over more complex ones also for arbitrary shape clusters (inducing several ellipsoidal micro-clusters for one bigger cluster) in case of high-dimensional problems. In the results section, we will show that our approach can work well in high-dimensional spaces without employing any dimension reduction algorithm.

Adopting and extending the steps as in above itemization to the general case, the core components in *eVQ-A (evolving Vector Quantization for Arbitrary Ellipsoids)* can be formulated as follows:

(1) For each new incoming sample $\vec{x}$, the closest cluster is elicited in terms of Mahalanobis distance using Eq. (3); this is denoted as the *winning cluster (neuron)*, represented by $(\vec{c}_{win}, Cov_{win}^{-1}, r_{win})$ with

$$win = argmin_{i=1,\ldots,C}\,(d(\vec{x}, \vec{c}_i)) \tag{5}$$

with C the number of clusters in the current partition. Additionally, the cluster nearest to the winning cluster is elicited, denoted as $(\vec{c}_{near}, Cov_{near}^{-1}, r_{near})$:

$$near = argmin_{i=1,\ldots,C, i\neq win}\,(d(\vec{c}_i, \vec{c}_{win})) \tag{6}$$

(2) It is checked whether the new incoming sample matches the current cluster partition: the *Mahalanobis distance* of the new sample to the winning cluster is compared against the tolerance radius r_{win} (defining the hyper-ellipsoidal contour), which is dynamically updated:

$$r_{win} = fac * p^{1/\sqrt{2}} * \frac{1.0}{(1 - 1/(k_{win}+1))^m} \tag{7}$$

with *fac* an a priori defined parameter, steering the tradeoff between stability (update of an old cluster) and plasticity (evolution of a new cluster). This is the only sensitive parameter in our method, and an inappropriate setting will be compensated by the dynamic merging and splitting options as explained in the subsequent sections; an extended sensitivity analysis on this parameter will be conducted in Section 6.

The second product term in (7) compensates for the curse of dimensionality (as increasing with p), based on the same considerations as done in *eVQ*, see [56]. The slight difference to the approach used there is that here we are taking the $\sqrt{2}$-root of the dimensionality instead of the conventional square-root. We then automatically obtain a close approximation of the quantile function of the χ^2 distribution, i.e. $\chi^2_p(\alpha)$, with p the degrees of freedom and α the significance level. This quantile function can be used as boundary of the prediction interval, see e.g. [52]. If samples are falling outside this interval, they are most likely (statistically) not belonging to the corresponding clusters, thus are candidates for evolving a new cluster. In this sense, it serves as *statistical tolerance region* for multivariate normal distributions as also motivated and used in [87].

Finally, our threshold corresponds to a statistically motivated tolerance that a sample is part of the region with a certain significance level α: higher values of *fac* are correlated with lower α's. The reason why we use the approximation and not the real critical distribution value is the much lower computational cost resp. the avoidance of large, cumbersome (hard-coded) look-up tables. The closeness of the approximation can be visualized in Fig. 2, where for different values of p (along the x-axis) the corresponding tolerance region values (along the y-axis) are plotted. The dotted line indicates the χ^2 critical value with 5% significance level, the solid line our approximation.

Additionally, we include a third term in Eq. (7), which compensates for the uncertainty in clusters with low support/significance (density), increasing the radius in this case and keeping it on the original level for compact, dense clusters. The symbol m steers the degree of this density-based impact and is set to 4 as default value (also used in all experiments below). An explanation of this issue is underlined in Fig. 3, showing two clusters with equal sizes but different densities: obviously, in the right case the evolution of a cluster is more intuitive than in the left one, although in both

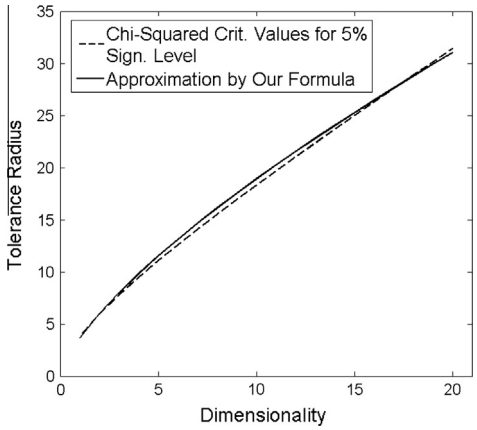

Fig. 2. Increasing $\chi^2_p(\alpha)$ critical values for $\alpha = 5\%$ significance level when increasing the dimensionality (degrees of freedom): the original in dotted line, our approximation in solid line.

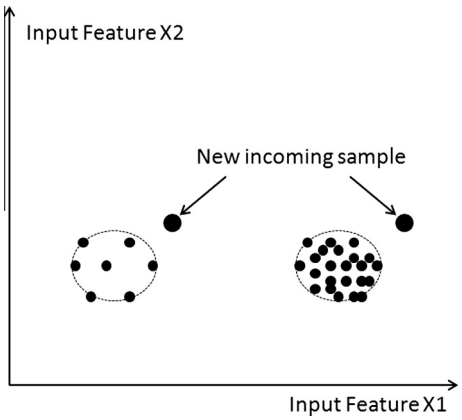

Fig. 3. Two situations where the Mahalanobis distance of a new sample to the nearest cluster is the same, but appearing with a different matching degree as the right cluster is much more dense than the left one – intuitively, in the right case it is more appealing to evolve a cluster than in the left one.

cases the distance of the sample to the cluster is the same. Furthermore, clusters with low support cannot be seen as heterogenous enough such that samples falling near to them can be safely handled as new clouds. Thus, our strategy enforces wide acceptance regions for clusters with low support. As k_{win} starts to grow, the third term in Eq. (7) approaches 1, leaving only the (approximated) statistical oriented tolerance region.

(3) If the shortest distance to all clusters is higher than r_{win}, i.e.

$$\min_{i=1,\ldots,C} \sqrt{((\vec{x}-\vec{c}_i)^T Cov_i^{-1}(\vec{x}-\vec{c}_i))} > r_{win} \tag{8}$$

is fulfilled, a new cluster is evolved by setting its center to the sample $\vec{c}_{C+1} = \vec{x}$ and initializing its inverse covariance matrix to

$$Cov_{C+1}^{-1} = diag\left(\frac{1}{frac} * \vec{range}\right) \tag{9}$$

with *frac* denoting a small fraction of the input ranges contained in $\vec{range}$, i.e. restricting the range of the influence of the initial cluster to a small area: we set *frac* to 100 in all our tests and experiments. Once some clusters are identified, the inverse covariance matrix of the new cluster can be set to the weighted average of the inverse covariances of already existing clusters

$$Cov_{C+1}^{-1} = \frac{\sum_{i=1}^{C} Cov_i^{-1}}{C} \tag{10}$$

making it independent from possibly up-coming range changes.

(4) If (8) is not fulfilled, the components of c_{win} are updated: updating the center $\vec{c}_{win}$ with the new sample $\vec{x}$ can be done in the same manner as in conventional *eVQ* as this describes a movement of the center to the new point, which is independent from the applied distance measure, thus:

$$\vec{c}_{win} = \vec{c}_{win} + \eta_{win}(\vec{x} - \vec{c}_{win}) \tag{11}$$

with $\eta_{win} = 1/k_{win}$ as discussed above after Eq. (2). The difference lies in the update of the spread as it is characterized by the inverse covariance matrix. A possibility is to update the covariance matrix in recursive (almost) exact manner, as done in [71,16,59]. However, this requires a matrix inversion step after each incremental learning cycle, which sometimes can get unstable [59]. Thus, we are opting for a direct update of the inverse covariance matrix through the following investigations [8]: setting $\alpha = \frac{1}{k_{win}+1}$ with k_{win} the number of samples belonging to cluster c_{win} so far (we neglect here the index *win* in all *Cov*-occurrences for the purpose of transparency):

$$Cov^{-1}(k+1) = ((1-\alpha)Cov(k) + \alpha(\vec{x}-\vec{c})(\vec{x}-\vec{c})^T)^{-1} = \left(I + \frac{\alpha}{1-\alpha}Cov^{-1}(k)(\vec{x}-\vec{c})(\vec{x}-\vec{c})^T)\right)^{-1}\frac{Cov^{-1}(k)}{1-\alpha} \tag{12}$$

By expanding the inverse and setting $\gamma = \frac{\alpha}{1-\alpha}$, we obtain

$$\left(I - \gamma Cov^{-1}(k)(v \cdot v^T) + \gamma^2 Cov^{-1}(k)(v \cdot v^T)Cov^{-1}(k)(v \cdot v^T) - \gamma^3 \cdots\right)\frac{Cov^{-1}(k)}{1-\alpha} \tag{13}$$

By rearranging and substituting $v^T Cov^{-1}(k)v$, we obtain

$$\begin{aligned} &= \frac{Cov^{-1}(k)}{1-\alpha} - \frac{\beta}{1-\alpha}Cov^{-1}(k)(v \cdot v^T)Cov^{-1}(k)(1 - \beta\lambda + \beta^2\lambda^2 - \cdots) \\ &= \frac{Cov^{-1}(k)}{1-\alpha} - \frac{\beta}{1-\alpha}\frac{Cov^{-1}(k)(v \cdot v^T)Cov^{-1}(k)}{1+\beta\lambda} \end{aligned} \tag{14}$$

After rearranging the infinite sum in the braces, the following update formula is obtained:

$$Cov^{-1}(k+1) = \frac{Cov^{-1}(k)}{1-\alpha} - \frac{\alpha}{1-\alpha}\frac{(Cov^{-1}(k)(\vec{x}-\vec{c}))(Cov^{-1}(k)(\vec{x}-\vec{c})^T}{1+\alpha((\vec{x}-\vec{c})^T Cov^{-1}(k)(\vec{x}-\vec{c}))} \tag{15}$$

Additionally, we apply a decremental stage on the nearest cluster to the winning cluster by moving away its center from the current sample $\vec{x}$:

$$\vec{c}_{near}(N+1) = \vec{c}_{near}(N) - \eta_{near}(\vec{x} - \vec{c}_{near}(N)) \tag{16}$$

with $\eta_{near} = 1/k_{near}$ a decreasing learning gain according to the support of the cluster. Eq. (16) enforces a movement of the center, thus a shift of the whole cluster away from the current sample $\vec{x}$. This increases the likelihood of avoiding overlaps and thus sharpens the contour boundaries of nearby lying clusters.

In Fig. 4, we demonstrate the difference between conventional *eVQ* (first three rows) and the new version *eVQ-A* for arbitrary ellipsoidal shaped clusters (last row) on a two-dimensional clustering stream data set from the Internet, in which the (number of) clusters are known. The left column represents the clustering results after the first 1000 samples, the right one after additional 1000 samples. The first three rows correspond to different parameterizations of *eVQ*: the vigilance set to 0.22, 0.25 and 0.28. Obviously, in all these three cases *eVQ* is not able to extract correct cluster partitions (too less clusters in case of 0.28, too much clusters in case of 0.22 and misplaced clusters in case of 0.25). Especially, it is not being able to sufficiently model the rotated ellipsoidal data cloud in the lower right corner because of being restricted to axis-parallel shapes. Indeed, when increasing the vigilance to 0.28 only one cluster can be extracted for this data cloud, however when updating with the next data block the samples from other clusters are joined to form a new big but misplaced cluster. When decreasing the vigilance below 0.22 a more over-clustered situation as in the first row was the case. When increasing it above 0.28 even more distinct clouds are joined to the same clusters → partition cannot be correctly modeled with eVQ. *eVQ-A* achieved correct clustering results for a *wider range* of the tolerance radius (from $fac = 3.5$ to $fac = 5.0$). In the results section, we will demonstrate the lower sensitivity and better cluster partition quality of *eVQ-AMS* over *eVQ* on various streaming data sets.

3. Single-pass on-line merging functionality

In order to polish up evolved cluster partition ensuring clear distinct clusters, it is necessary to provide a merging functionality due to the following considerations:

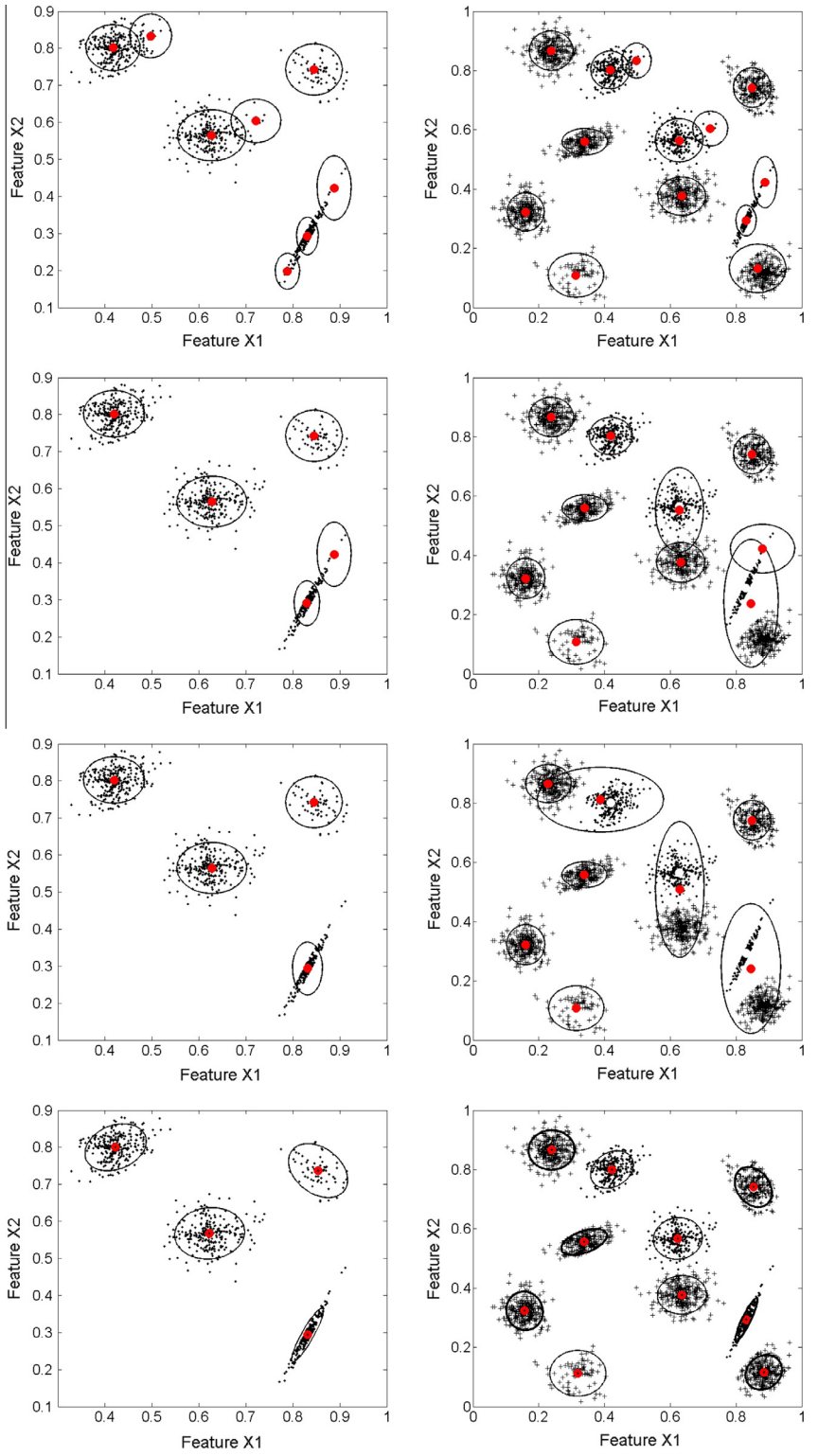

Fig. 4. Left: clusters evolved on first 1000 samples with conventional eVQ and vigilance = 0.22 (first row), conv. eVQ and vigilance = 0.25 (second row), conv. eVQ and vigilance = 0.28 (third row) and new eVQ-A variant with Mahalanobis radius of 3.5, 4.0, 4.5, 5.0 (last row), all triggering the same results; right: the same with the next 2000 samples (marked as pluses) updated, the old samples shown as dots.

- Cluster may move together over time due to the nature of the data stream, filling up holes in-between two original distinct data clouds (also termed as *cluster fusion or coalescence*). An example is given in Fig. 5. Indeed, due to the built in decremental learning stage conducted by Eq. (16), enforcing movements of runner-up clusters away from winning clusters, the likelihood decreases that cluster move together and get overlapping. However, the problem can still arise whenever the support of the runner-up cluster is high such that it is almost converged (thus frozen) and the winning cluster moves towards it (due to low support and samples filling up the gap inbetween these two).
- An inappropriately chosen tolerance region (*fac* in Eq. (7)): if it is set too small, superfluous clusters may be evolved close to existing clusters representing the outer spread of a (noisy) data cloud. When then more samples are filling up the outer spread of the cloud, the evolved cluster gets overlapping with the original one.

Our merging approach consists of two steps, extending the concepts in [60] (there specifically deduced for axis-parallel ellipsoids), to ellipsoidal clusters in arbitrary position:

(1) Identifying the cluster pair(s) which should be merged together.
(2) Merging cluster pairs $(\vec{c}_i, Cov_i^{-1}, r_i)$ and $(\vec{c}_k, Cov_k^{-1}, r_k)$ to a new cluster $(\vec{c}_{md}, Cov_{md}^{-1}, r_{md})$.

The first step is again split up into two steps, namely (1a) measuring the degree of overlap of an updated cluster with all other clusters in the current cluster partition and (1b) respecting the homogeneity of two significantly over-lapping clusters.

We applied the Bhattacharyya distance [10,28] for calculating the overlap degree, which is defined for multi-variate Gaussian distributions as

$$olap(i,k) = \frac{1}{8}(\vec{c}_i - \vec{c}_k)^T Cov^{-1}(\vec{c}_i - \vec{c}_k) + \frac{1}{2}\ln\left(\frac{\det Cov^{-1}}{\sqrt{\det Cov_i^{-1}\ \det Cov_k^{-1}}}\right), \tag{17}$$

with $Cov^{-1} = (Cov_i^{-1} + Cov_k^{-1})/2$ – note that, due to the arbitrary ellipsoidal shape of our clusters, their contours in the p-dimensional space (according to r) are equivalent to the characteristic (supporting) spread of multi-variate Gaussian distributions [86]. Thus, Bhattacharyya distance is valid to be adopted one-to-one to our cluster partitions by using the extracted (inverse) covariance matrices (see preliminary section). The nice property is that the distance delivers exactly 0 if two ellipsoids are touching, > 0 when they are overlapping (the closer the value to 1, the bigger is the overlap) and < 0 when they are disjoint. Thus, a feasible threshold for cluster merging candidates is 0, omitting the introduction of an additional parameter. This is essential and not achievable with other overlap measures (such as Kullback–Leibler divergence [53], for instance), as we want to avoid parameters as far as possible in order to approach a plug-and-play method.

After each incremental update cycle with one single sample, the maximal overlap degree of the updated cluster $(\vec{c}_{win}, Cov_{win}^{-1}, r_{win})$ to any other cluster is calculated (assuming C cluster in the partition):

$$olap_{max} = \max_{i=1,\ldots,C/win} olap(win, i) \tag{18}$$

If this exceeds the threshold set to 0 (for touching case), the two clusters $(\vec{c}_k, Cov_k^{-1}, r_k)$ with $k = argmax_{i=1,\ldots,C}\ olap(win, i)$ and $(\vec{c}_{win}, Cov_{win}^{-1}, r_{win})$ are merging candidates. In order to restrict computation time, we only inspect *olap* for the winning cluster and perform the merging with the cluster having maximal overlap. Fig. 6 shows some examples of disjoint (a), slightly over-lapping (b) and strongly over-lapping ellipsoids (c), indicating the values of the Bhattacharyya distance in the caption.

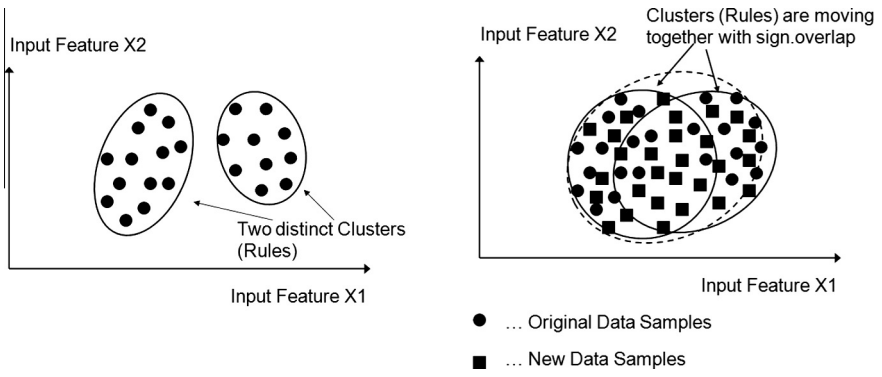

Fig. 5. Left: initial situation favors two clusters (low sample significance) and right: new samples improve the significance and make clear that there exists only one cluster.

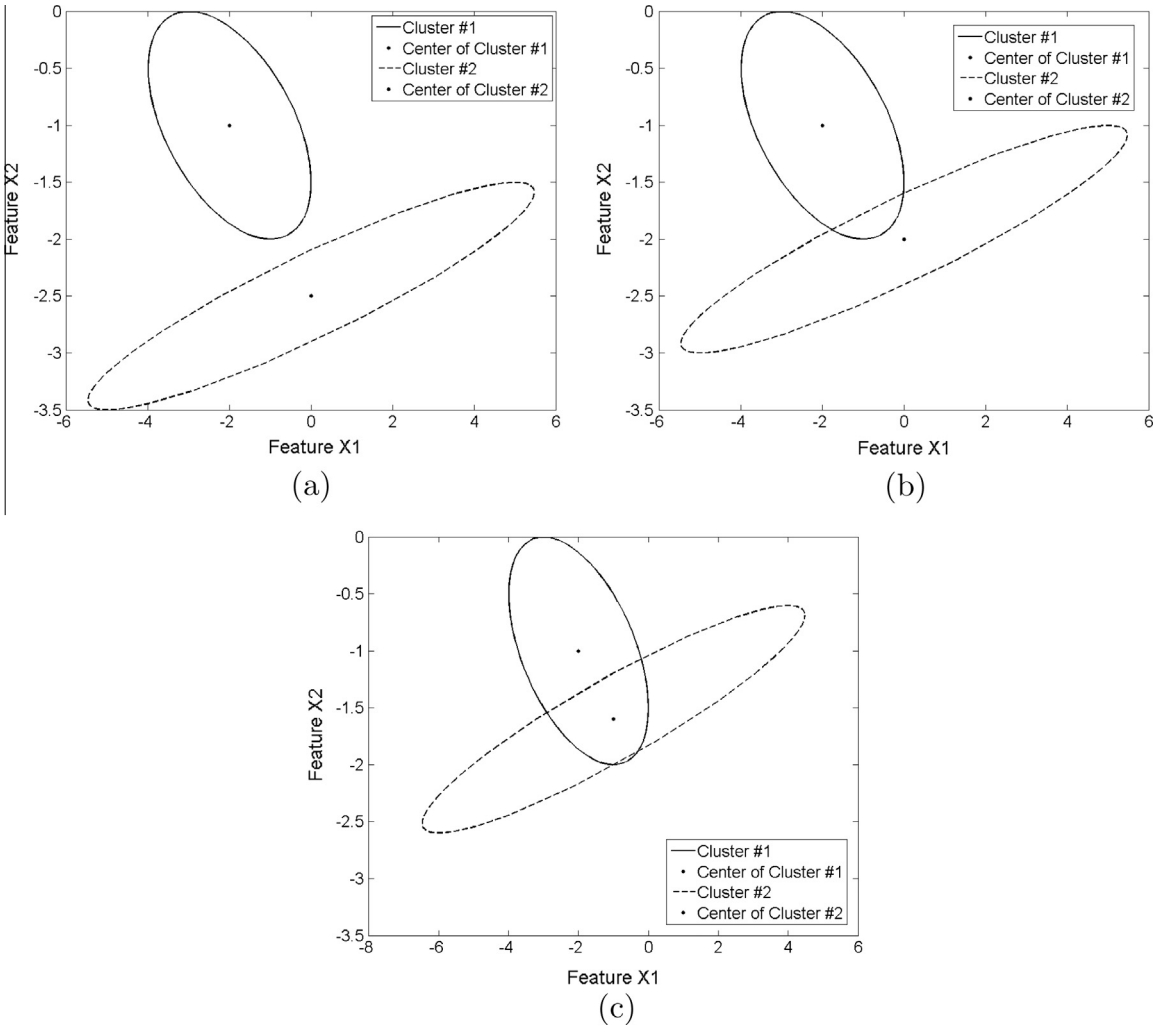

Fig. 6. (a): Disjoint ellipsoidal clusters (distance = −0.13); (b): slightly over-lapping ellipsoidal clusters (distance = 0.12); and (c): strongly over-lapping ellipsoidal clusters (distance = 0.37).

The final decision whether such clusters are merged depends on the reciprocal homogeneity of their shape and direction. In particular, two clusters can be merged whenever their joint distribution follows the "behavior" of their single distributions. In order to make this more clear, we provide two cluster touching examples in Fig. 7. In the left case, the two clusters form a homogeneous combination and thus can be merged. In the right case, they are following a different trend in the two-dimensional space and therefore it is not convenient to merge this two clusters. In both cases, the merged clusters are highlighted with dotted ellipsoids. Obviously, in the right case, an undesired blow-up of the cluster volume can be observed, as the range of influence of the cluster marks a hyper-ellipsoids implicitly including a large empty space into which no samples fall inside. Thus, our homogeneity condition expresses the change in the volume of the merged cluster w.r.t. the original clusters: an undetermined high volume blow-up indicates an inconsistency in the joint cluster homogeneity. Thus, we hypothetically merge the two clusters $(\vec{c}_k, Cov_k^{-1}, r_k)$ and $(\vec{c}_{win}, Cov_{win}^{-1}, r_{win})$ to one (for formulas see below) and compare the volume of the new cluster with the volume of the old ones; if the following condition is fulfilled:

$$V_{merged} \leqslant p(V_{win} + V_k) \tag{19}$$

the two clusters are merged. V denotes the volume of an arbitrary ellipsoid in the high-dimensional space [48] (we used the ellipsoidal toolbox from MATLAB[2] [54] for a very fast implementation):

$$V_k = \frac{2 * \prod_{j=1}^{p} (r_k/\lambda_{kj}) * \pi^{p/2}}{\Gamma(p/2)} \tag{20}$$

with λ_{ij} the jth eigenvalue corresponding to the jth eigenvector (ellipsoidal axis direction).

[2] http://code.google.com/p/ellipsoids/.

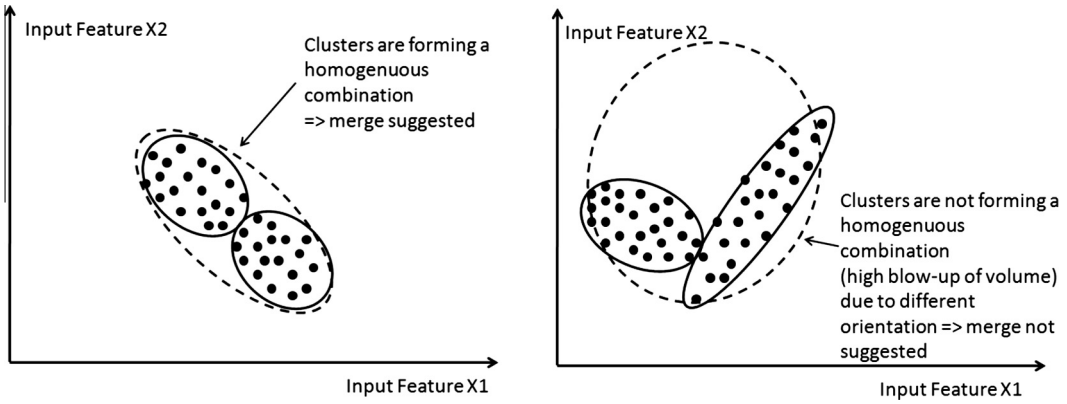

Fig. 7. Left: clusters are touching and forming a homogeneous combination (→ merge suggested); right: clusters are touching but required to appropriately model and keep the shapes and direction of the two data clouds (→ merge not suggested).

The merging operation itself is based on the idea that the more supported cluster should have a higher influence than the less supported one. In other words, the center of the merged cluster should lie closer to the center of the more supported cluster, and its inverse covariance matrix should be more similar to the inverse covariance matrix of the more supported cluster. Thus, a weighted average of the centers of the two overlapping clusters is obvious by choosing the cluster supports as weights. A similar concept is considered for the inverse covariance matrix, which is stabilized according to the rank-1 modification concept in the recursive update of the covariance matrix. Thus, we obtain:

$$\vec{c}_{md} = \frac{k_{win}\vec{c}_{win} + k_k\vec{c}_k}{k_{win} + k_k} \tag{21}$$

for center merging and

$$Cov_{md}^{-1} = \frac{k_{win}Cov_{win}^{-1} + k_kCov_k^{-1} + \frac{k_{win}k_k}{k_{win}+k_k} diag\left(\frac{1}{\left((\vec{c}_{md}-\vec{c}_{old})^T(\vec{c}_{md}-\vec{c}_{old})\right)}\right) \cdot * I}{k_{win} + k_k} \tag{22}$$

for inverse covariance merging with I the identity matrix, $(\vec{c}_{md} - \vec{c}_{old})$ a row vector with $old = win$ if $k_{win} = min(k_{win}, k_k)$, otherwise $old = k$, and $diag$ the vector of diagonal entries. The division in the numerator $(1/\ldots)$ (rank-1 modification) is done component-wise for each matrix element; moreover, $k_{md} = k_{win} + k_k$.

Fig. 8 visualizes an interesting example in case when different data clouds are touching along their outer samples: obviously, still five clusters are visible and thus should be extracted. This is achieved by our method as shown in the left image, although the merging option as discussed above is used: the reason is that the principal contours of the ellipsoids (defined by their covariance matrices and shown in solid line in the figure) are not touching at all, as representing the core density area of the clouds in compact form. When moving the samples defining the middle cluster more to the upper right part (as shown in the middle figure), the clusters get slightly overlapping and thus are merged to one cluster by our procedure as shown in the right image of Fig. 8. In this case, the merging is correct, as two clear density areas in this part of the feature space cannot be recognized any longer.

3.1. Comments on the computational costs

In every incremental learning step the Mahalanobis distance of a new sample to all C clusters is calculated. As Mahalanobis requires a complexity of $O(p^2)$ with p the input dimensionality, this complexity is $O(Cp^2)$. In the evaluation section, we will see for a huge stream that the native version of our extended variant (*eVQ-A*) looses only a little in computation time compared to *eVQ*. Clearly, the major bottleneck when conducting a merging operation is the calculation of the Bhattacharyya distance, which requires $O(p^4)$ complexity, as calculating the determinant of three matrices, thus $O(Cp^4)$ in sum, as the newly updated cluster is compared with all remaining ones in terms of overlap degree. Thus, we will check the overlap degree of an updated cluster only whenever its support is an integer multiple of M, where we used $M = 10$ in all experiments, reducing computation time by about a factor of 10.

Another option is to substitute the Bhattacharyya distance with a simpler, approximative measure, for instance by checking whether any of the ellipsoid outer points of the updated cluster lies inside any other cluster. The ellipsoid outer points $v_{win;i}$ and $v^*_{win;i}$ in the ith eigen-direction $e_{win;i}$ of cluster win are given by [27]:

$$\begin{aligned} v_{win;i} &= \vec{c}_{win} + \frac{r_{win}}{\lambda_{win;i}} u_{win;i} \\ v^*_{win;i} &= \vec{c}_{win} - \frac{r_{win}}{\lambda_{win;i}} u_{win;i} \end{aligned} \tag{23}$$

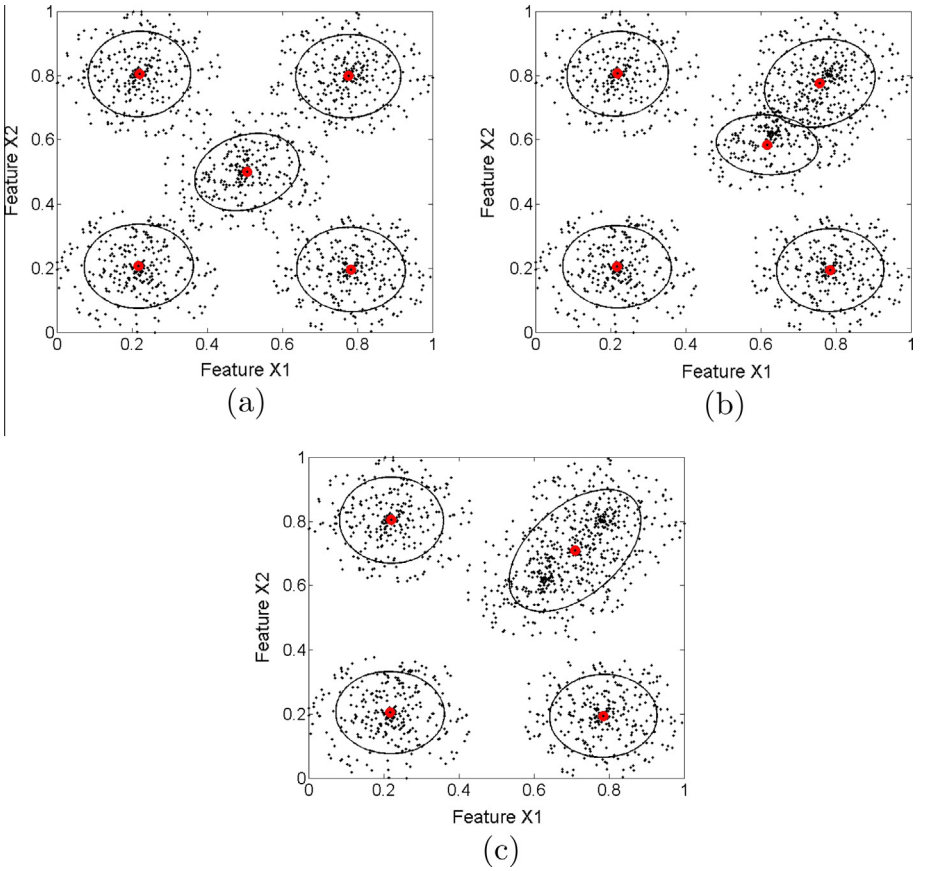

Fig. 8. (a): Initial data set containing 5 clusters which are touching at their outer spreads (but core density areas still disjoint) → 5 clusters correctly extracted by our method; (b): when moving the samples defining the middle cluster more to the upper right part, there are not two clear density areas any longer (clusters get slightly overlapping); and (c): the two clusters in the upper right part automatically merged by our approach.

with $u_{win;i}$ the normalized eigenvector to length 1. Checking whether Eq. (4) holds for all points $v_{win;i}$, $i = 1,\ldots,p$ and $v^*_{win;i}$, $i = 1,\ldots,p$ with $\vec{c} = \vec{c}_k$ and $r = r_k$ for $k = 1,\ldots,C/win$, leads to $O(2(C-1)p) = O(Cp)$ operations, whereas a fast eigenvalue decomposition can be conducted with complexity $O(p^{2.367})$ [24], leading to an overall complexity of $O(Cp^{3.367})$. This is an approximative measure in the sense that ellipsoids may still overlap along their contours away from the extreme points which cannot be resolved by this variant, but the extent of this overlap is restricted.

4. On-line split functionality

Opposed to the problem that clusters may coalesce over time and thus get touching or overlapping, there may be also cases where the data clouds may split up within one cluster. This is called the *cluster delamination effect*. The reason for this can be due to the nature of a stream, i.e. at the beginning of the stream flow the appearance of some data samples seem to form one homogenous cluster, which, however turns out to be non-homogenous later, i.e. internally there may appear two (or even more) density areas within one cluster. Such a case is demonstrated in Fig. 9 (left: original situation, one cluster seems to be favorable; right: data sample distribution after further stream flow). Another reason can be simply due to an inappropriate (too high) setting of *fac* in Eq. (7), which steers the wideness of the statistical χ^2 based tolerance region – such a case is visible in Fig. 11(a) below, examine Cluster #4. Although it is obvious that a new density area emerges on the left hand side, the samples are lying not sufficiently away from the surface of the already available cluster (w.r.t. to a too high *fac*).

In order to resolve such unpleasant cases properly, we develop a dynamic split procedure (in connection with *eVQ-AM*, thus termed as *eVQ-AMS*), which

(1) decides when a cluster has to be split,
(2) performs the splitting operation into two parts,
(3) and ideally acts in incremental manner.

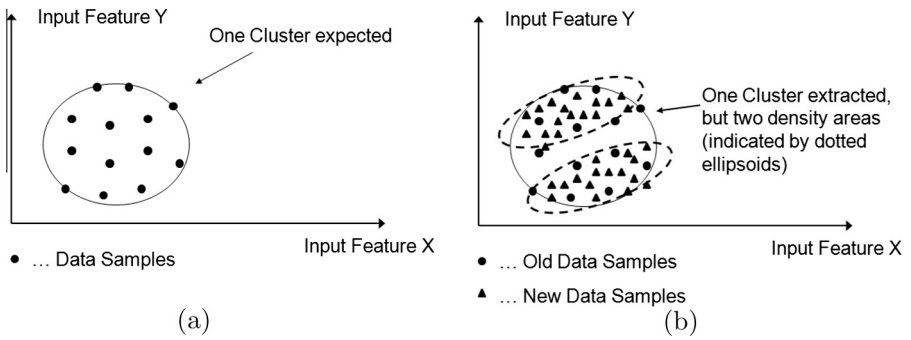

Fig. 9. (a) Original situation favors one cluster over the input domain and (b) when updating with new samples (triangular markers), the cluster contains two heterogenous density areas → split desirable.

Obviously, a cluster can be split when the density within is heterogenous, i.e. there are at least two distinct areas with a density of each which is higher than the density between these areas. Of course, this cannot be recognized based on the cluster shape and surface alone, i.e. a cluster with a large volume does not necessarily indicate to contain several density areas. Thus, a structure is needed for tracking the heterogeneity of clusters. We investigate a density-based approach, which we develop single dimension-wise independently in order to prevent the curse of dimensionality effect (which is usually very severe in case of density exposures in the high-dimensional case, see also [45]). Obviously, the principal components directions explaining the highest variance, i.e. the elements of $V = (pri_1, pri_2, \ldots, pri_p)$ with

$$Cov = WQV^T \tag{24}$$

with Q diagonal matrix and $W = V^T$, are candidates for selecting the orientations within a cluster, along which heterogeneity is sought. The problem is that an eigen-decomposition of a cluster covariance matrix Cov usually needs $O(p^3)$ complexity, resp. at least $O(p^{2.376})$ [24], and this should be ideally done after every incremental learning step in order to check the updated (winning) cluster for structural change. Thus, for the sake of computational complexity, we select the original input features as split directions candidates. All subsequent operations and methodologies can be equivalently applied to the principal component directions by using the scores of the samples $Scores = X^T W$ instead of the original sample values X (of the current sub-sample set, see below).

In order to be able to build up an appropriate density structure along each dimension (and also to be able to split one cluster appropriately once a split is suggested), we require a set of sub-samples for each cluster. For sub-sampling the data, we apply the reservoir sampling technique according to [89], in which we keep never more than N_{sub} samples in total. Let therefore S_{win} denote a representative sub-sample of the updated cluster $(\vec{c}_{win}, Cov_{win}^{-1}, r_{win})$ with $|S_{win}| \leqslant N_{sub}$ (in all our experiments we used $N_{sub} = 300$) – note that in each incremental learning cycles we only check whether the updated cluster (nearest to the current sample) need to be split, thus all subsequent density estimations are restricted to N_{sub} samples. Then, we can build up histograms along each dimension and look for distinct peaks and valleys in-between in order to get an impression whether there is an (upcoming) implicit heterogeneity contained in these along a single dimension. However, often histograms suffer from noise, i.e. peaks are not unique and may appear more frequently due to small density fluctuations. Furthermore, the optimal number of bins cannot be easily decided.

Hence, our strategy relies on estimating a uni-dimensional Gaussian mixture model (GMM) in the ith dimension $(i = 1, \ldots, p)$:

$$GMM_{i,k}(x) = \sum_{j=1}^{k} \pi_j \Phi_j \tag{25}$$

one time with $k = 1$ component, one time with $k = 2$ components and comparing their qualities. The symbol Φ denotes the univariate normal distribution:

$$\Phi_j = \frac{1}{\sigma_j \sqrt{\pi}} e^{-0.5 \frac{(x-\mu_j)^2}{\sigma_j^2}} \tag{26}$$

and π_j the mixing proportion. The estimation of mixtures does not only represent a smoothed density structure, but is also in-line with our general cluster representation (following multi-variate Gaussians). The estimation of the Gaussian mixture model is conducted by the expectation maximization algorithm [25], which may not converge whenever two components are selected, but only one can be reliably extracted. In such a case, automatically a split is not suggested. Furthermore, whenever the quality in terms of the BIC (Bayesian Information Criterion) [81] of the one-component model $GMM_{i,1}$ is higher than of the two-component $GMM_{i,2}$, again automatically a split is not suggested. The BIC of a uni-dimensional Gaussian mixture for the ith feature (direction) can be calculated as:

$$BIC_i(GMM_{i,k}) = -2 * loglik_k + N_{sub} * d_k \tag{27}$$

with d the degrees of freedom = number of parameters, which are 2 for one component (mean, variance). The log-likelihood over all samples n_j belonging to component j given by (proof is left to the reader)

$$loglik_k = \sum_{j=1}^{k} \left(n_j \ln(\pi_j) - n_j \ln(\sigma_j) - n_j \ln\left(\sqrt{2\pi}\right) - 0.5 \sum_{x_i \in comp_j \subset S_{win}} \frac{(x_i - \mu_j)^2}{\sigma_j^2} \right) \tag{28}$$

If $k = 1,\ comp_j = S_{win}$, if $k = 2,\ comp_1$ is containing those samples for which the first component has a higher likelihood, and $comp_2$ those ones for which the second component has a higher likelihood. The higher the log-likelihood (subject to the punishment term $N_{sub} * d_k$), the lower the BIC, thus the better the model. The punishment term punishes more complex models, thus decreasing the likelihood of a split.

Whenever a two-component GMM achieves a higher quality than a one-component one, a statistical test is made whether the two means (subject to the corresponding variances) are significantly different, by also respecting the mixing proportions π_j. In case of almost equal mixing proportions (meaning that the Gaussians have equal heights in the mixture), the means have not to be that distinct as in case of very different proportions. In the later case, it means that the heterogeneity is only weakly present due to a few samples lying apart from the dense area. In order to intensify these clues, we visualize a two-dimensional example in Fig. 10, where two clusters (indicated by 3 and 4) are under examination: Cluster #3 may contain two (noisy) density areas, which could be split somewhere in the middle along Feature X2 (in fact we know that it de facto contains two clouds due to the available cluster structure information), Cluster #4 contains some 'outer points' (indicated with an arrow), which are starting to drag the cluster away from its original density area and to expand its spread. The evolution of a new cluster in this case would be helpful (and correct), however due to a high parametrization of *fac* (5.0, which was ideal for another data set), no evolution takes place. When looking at the distribution (represented by histograms for visualization purposes) of these two clusters (Fig. 10(b) and (c)) and the fitted Gaussian mixture models (black line), a split would be favorable in case of Cluster #3 (as it represents two internal homogenous dense areas with almost equal weights). In Cluster #4 the situation is different as one GMM is indeed distinct to the other but much more tiny in its density. The latter aspect can be reflected in the relative percentage of the mixing proportion as defined by:

$$Rel_{prop} = \frac{\min(\pi_1, \pi_2)}{\max(\pi_1, \pi_2)} \tag{29}$$

In case of half-half situation, i.e. $p_1 = p_2 = 0.5$, Eq. (29) becomes 1. Now, when additional samples come in lying near Cluster #4, the second component along Feature X1 becomes more apparent, see Fig. 11. In this case, a split is already more considerable than in Fig. 10.

A statistical test of equal means between two distributions can be conducted with the so-called *t-test* or also called *student test* [19]. In our case, as dealing with Gaussians, we use ∞ degrees of freedom, as then the student distribution equals to a Gaussian one. Furthermore, the variances of the two components are assumed to be different (and thus must be estimated separately), thus we apply a specific version of the *t-test*, the so-called *Welch test* [91] (which also can be found in [13]), where the test indicator (t-statistics) I for feature (direction) i becomes:

$$I_i = \frac{(\mu_0 - \mu_1)^2}{\sigma_0^2 + \sigma_1^2} > crit \tag{30}$$

with $\mu_0,\ \sigma_0$ the mean value and the variance of the first component and $\mu_1,\ \sigma_1$ the mean and variance of the second one. From statistical tables, someone may obtain in case of a 5% significance level ($\alpha = 0.05$), $crit = 1.96$, as $P(X = |\mu_0 - \mu_1| < 1.96) = 0.975 = 1 - \alpha/2$. In case of 1% significance level, we obtain $crit = 2.57$, which we will use in Section 6 for all experiments (in order to assure very little false splits).

However, the test as in Eq. (30) only holds in case of two Gaussians with equal support. In our case using GMM, we are confronted with different mixture proportions which is reflected in Eq. (29). Obviously, the higher the discrepancy between the two components, the lower I should get. Finally, we weight the relative proportion into the test statistics achieving its weighted version $I_i(w)$:

$$I_i(w) = Rel_{prop} \frac{(\mu_0 - \mu_1)^2}{\sigma_0^2 + \sigma_1^2} > crit \tag{31}$$

Note that in the two examples above, with $crit = 2.57$ no split of Cluster #4 along Feature X1 is suggested for the situation in Fig. 10, but in Fig. 11.

Compiling all the considerations together, leads to the following condition whether the previous updated cluster $(\vec{c}_{win}, Cov_{win}^{-1}, r_{win})$ should be split:

$$\exists_i \in \{1, \ldots, p\} \quad (Converged(EM_i, 2) == true \wedge BIC_i(GMM_1) > BIC_i(GMM_2) \wedge I_i(w) > crit) \tag{32}$$

Note: the model with *lower* BIC tends to have a better quality (representation of the data) and thus should be preferred. The second argument in *Converged* function indicates the run-through with two components parameterized. If more than one feature fulfills this three-way criterion, the split is conducted along that feature with highest $I_i * (w)$, $i^* = argmax_{i=1,\ldots,p}(I_i(w))$.

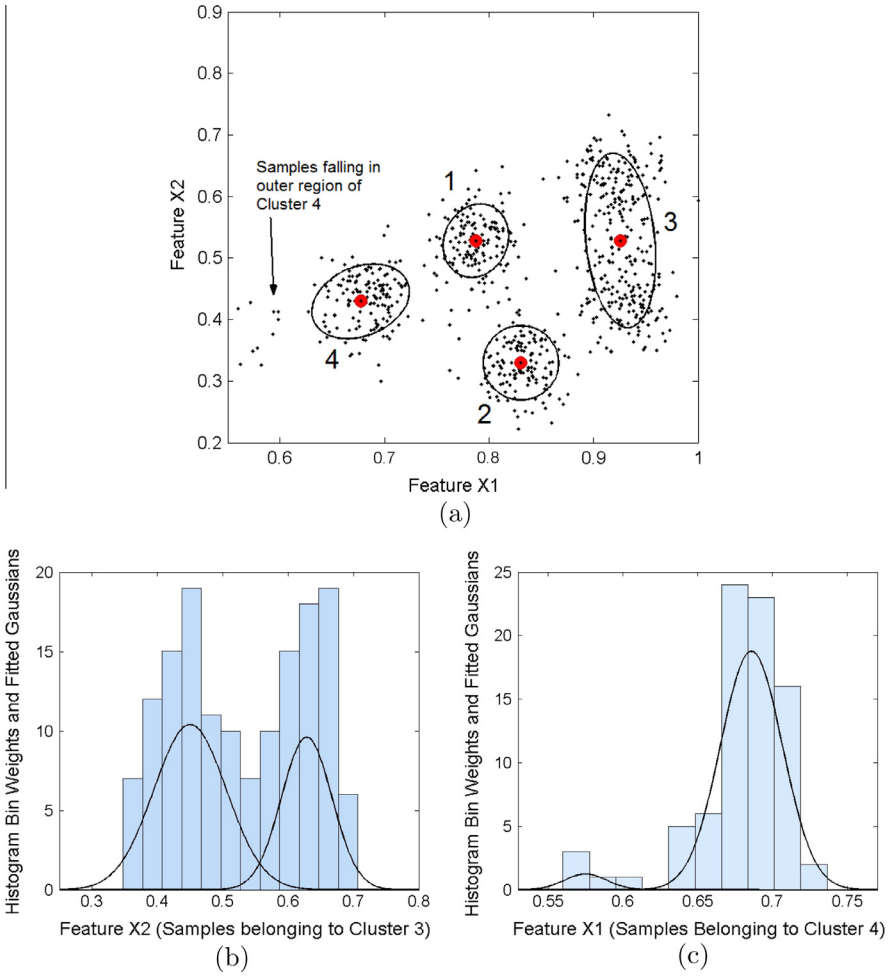

Fig. 10. (a): The cluster structure after 760 samples, Cluster #3 a potential candidate for containing two heterogenous density areas along Feature X2 (compare with its histogram and fitted GMM in (b)), Cluster #4 containing some 'outer' points (as no new cluster evolved due to a too high tolerance region in Eq. (7)), its corresponding histogram along Feature X1 and fitted GMM in (c) – please note that the histograms are just for visualization purposes (to make the heterogenous density regions more clear) and are not used within the algorithm (only the fitted GMMs).

The open question is now how the split should be conducted. Intuitively, it is clear that the cutting point of the two components in the GMM, (μ_1, σ_1, π_1) and (μ_2, σ_2, π_2), which lies in-between the two centers, represent a reliable split point. The cutting point(s) respecting the two mixing proportions can be calculated as follows (proof is left to the reader):

$$cut_{1,2}(i^*) = \begin{cases} \frac{\mu_1+\mu_2}{2} + \frac{\sigma^2(\ln(\pi_1)-\ln(\pi_2))}{\mu_1-\mu_2} & \sigma_1 = \sigma_2 \\ \frac{\mu_2\sigma_1^2-\mu_1\sigma_2^2 \pm \sqrt{(\mu_1\sigma_2^2-\mu_2\sigma_1^2)-(\sigma_1^2-\sigma_2^2)*d}}{\sigma_1^2-\sigma_2^2} & otherwise \end{cases} \tag{33}$$

with $d = 2\sigma_1^2\sigma_2^2(\ln(\pi_1) - \ln(\pi_2)) + \mu_2^2\sigma_1^2 - \mu_1^2\sigma_2^2$. Then, the sub-sample set S_{win} is split into two portions along feature i^*, yielding S_{win1} and S_{win2}:

$$S_{win1} = \{\vec{x} | x_{i^*} \leqslant cut_{1,2}(i^*)\} S_{win2} = \{\vec{x} | x_{i^*} > cut_{1,2}(i^*)\} \tag{34}$$

and the mean values and (inverse) covariances of these two sets estimated, yielding $(\vec{c}_{win1}, Cov_{win1}^{-1}, r_{win1})$ and $(\vec{c}_{win2}, Cov_{win2}^{-1}, r_{win2})$. r_{win1} and r_{win2} are given by Eq. (7) according to the new data samples k_{win1} and k_{win2} belonging to the two clusters. These can be calculated as the relative percentage of the amount of data samples belonging to the original (non-split) cluster, i.e. by:

$$k_{win1} = k_{win} \frac{|S_{win1}|}{|S_{win1}| + |S_{win2}|} k_{win2} = k_{win} \frac{|S_{win2}|}{|S_{win1}| + |S_{win2}|} \tag{35}$$

The resulting split of the heterogenous situation in Fig. 11(a) is visualized in Fig. 11(c).

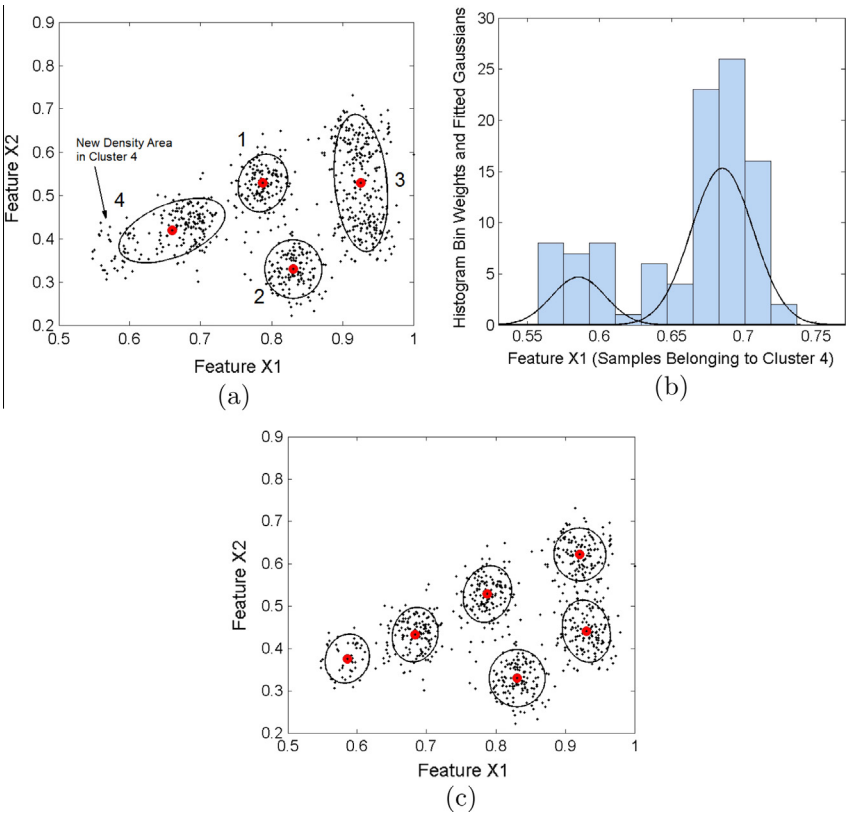

Fig. 11. (a): The cluster structure after 800 samples, Cluster #4 containing already a more distinct density area, its corresponding histogram along Feature X1 and fitted GMM in (b); and (c) represents the clusters after the splitting operation suggested by (32) and conducted by (34).

Regarding the computation costs of the splitting process, it is not that dramatic, as in each incremental learning cycle only one cluster has to be examined whether it should be split or not (the winning cluster $(\vec{c}_{win}, Cov_{win}^{-1}, r_{win})$). Furthermore, the *EM algorithm* requires a complexity of $O(Npk)$, see [78], with N the number of samples, p the dimensionality and k the number of components. In our case, k equals to a low value (either 1 and 2), $p = 1$ and $N = N_{sub}$ a restricted subset of samples. Thus, as performing $p * 2$ run-throughs of the *EM* algorithm (two for each feature), the computational complexity is $O(N_{sub}p)$. Calculating the test statistics $I(w)$ is negligible and calculating the BIC in (27) requires also $O(N_{sub}p)$ (due to the last summand in (28)). Calculating the split in (33) requires constant computational complexity and furthermore is only conducted when split is suggested, thus negligible. In sum, this turns into a complexity $O(N_{sub}p)$, where N_{sub} is small compared to the number of all stream samples. In case when using GMM fits along the principal components directions, it would require $O(N_{sub}p^{2.376}q)$ at least (as fastest eigen-decomposition requires $O(p^{2.376})$), with q the number of most important components under examination (often $q \ll p$).

Finally, it should be mentioned that the dynamic split procedure is another step towards a parameter-free incremental prototype-based clustering method. In fact, in the evaluation section below we will clearly demonstrate that in case of a very high setting of *fac* (that high that *eVQ-AM* without splitting would result in one global cluster), still the splitting procedure is able to extract the real clusters from the data streams. This is a very strong characteristics opposed to other methods, where usually the learning parameter suffers from high sensitivity and thus often has to be tuned for a new data set. In a data stream case, however, trials with different parameter settings are restricted or, in the real on-line case, even not possible.

5. AutoClust (eVQ-AMS) – the algorithm

In this section, we provide an overview on our data stream clustering algorithm by visualization of its workflow components, interactions and sequential processing in Fig. 12 by referring to the formulas needed in order to guarantee a successful run-through. Please note that the sub-sample buffer (for each cluster) is initialized when a new cluster is evolved and updated with new samples using reservoir computing and ring-buffer update strategy (always that buffer belonging to the winning cluster in case when no new cluster is evolved). In case of merging, the sub-sample buffers of the two clusters are merged and the recent N_{sub} samples kept resp. all samples kept if less than N_{sub} samples are present.

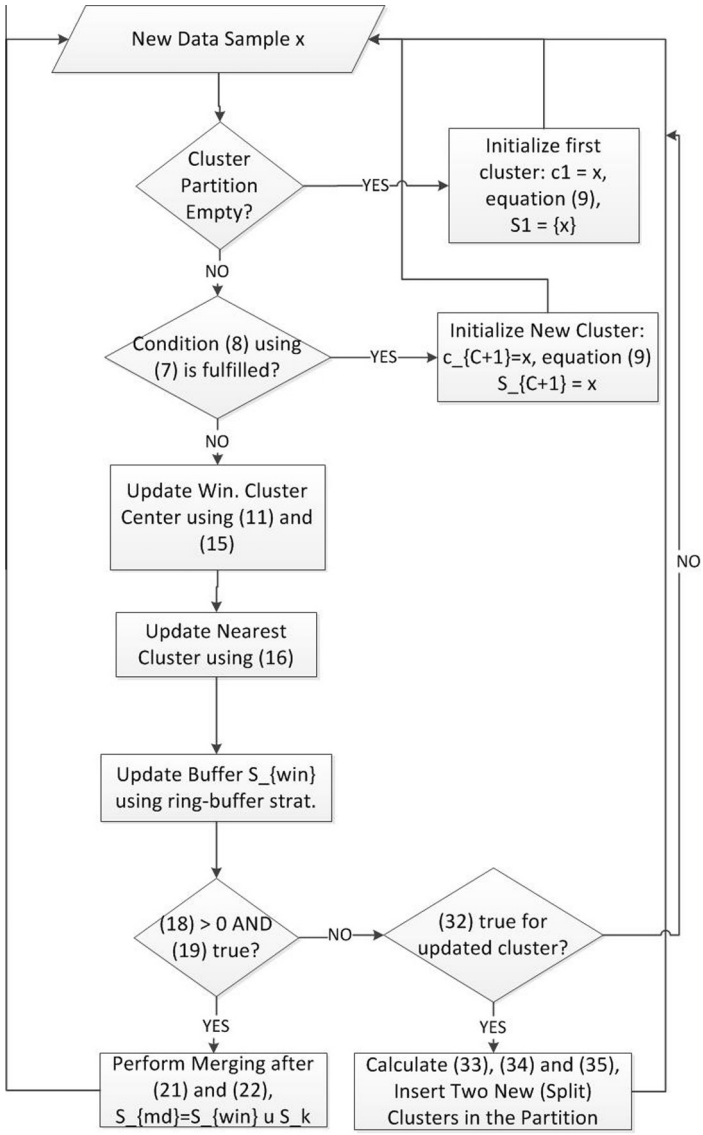

Fig. 12. Workflow of the *AutoClust (eVQ-AMS)* algorithm for data streams, the components and its processing order are shown when one single sample is loaded.

6. Experimental setup and results

For verifying the quality of our newly designed dynamic incremental algorithm, we applied the following data sets:

- 2-dimensional data sets in a data streaming context termed as S1, S2, S3, A1, A2 and R15, available in the Internet[3]; this data set includes not only new clusters evolving over time, but also dynamic feature range extensions. Furthermore, the data can be perfectly used for any visualization purposes of the appearances (shape, outlook) of the evolved clusters (as also conducted in the figures above): the meaningfulness of cluster partitions can be compared manually by human eyes, also because the real number of clusters contained in the data sets is known. Another nice feature of S1–S3 data sets is that successively more noise is added; the same is the case for A1–A2. While in S1/A1 the data clouds are quite distinct, in S3/A2 they appear much more blurred and intermixed, increasing the likelihood of cluster overlaps. R15 is an interesting set as 8 data clouds are arranged in a kind of inner circle, whereas 7 data clouds appear in a kind of outer circle surrounding the inner one.

[3] http://cs.joensuu.fi/sipu/datasets/.

Table 1
Characteristics of the clustering data sets used (samples, features, clusters).

	# Samples	# Feat.	# of cl.
S1	5000	2	15
S2	5000	2	15
S3	5000	2	15
A1	3000	2	20
A2	5250	2	35
R15	600	2	35
Gauss 6-dim	4051	6	9
Gauss 10-dim	6750	10	9
Yeast	1484	8	10
DIM032	1024	32	16
DIM064	2048	64	16
Animals	500 K	46	4
KDDCup	145,751	74	2000

- High-dimensional streaming clustering data sets termed as Gaussians 6-dim and 10-dim, DIM032, DIM064 and Yeast (from[4]) as well as huge streams such as Animals and KDDcup from UCI repository.[5] In all these data sets, the real number of clusters are known (thus the number of extracted clusters comparable with the real number), while Yeast, Animals and KDDcup have been recorded from real-world applications.

An overview of the data sets with number of samples (second column), number of features (third column) and number of clusters (fourth column) is provided in Table 1.

For comparison purposes of our new method *eVQ-AMS (AutoClust)* with state-of-the-art, we applied conventional *eVQ* (in order to see the improvement of our extensions directly), incremental, evolving clustering methods ESOMs [26] and growing neural gas [35] as well as batch clustering methods *k*-means [37], subtractive clustering [95] and Gustafsson–Kessel [40] for a hard benchmark as these see all data samples at once. The latter is able to even extract ellipsoids in arbitrary position. Apart from *k*-means and Gustafsson–Kessel, for which we set *k* to the real number of clusters contained in the data sets, the learning parameters in the other methods, responsible for the tradeoff between plasticity and stability (i.e. vigilance, Mahalanobis distance radius, distance threshold, range of influence (radius)), were varied in a grid-like manner with small step-sizes and the best cluster partition quality in terms of Xie–Beni index separability criterion reported [93]:

$$cl\_qual = \frac{\sum_{i=1}^{C}\sum_{j=1}^{N}\mu_{ij}^2\|\vec{c}_i-\vec{x}_i\|}{N\min_{i,j=1,\ldots,C,i\neq j}\left(\|\vec{c}_i-\vec{c}_j\|\right)} \tag{36}$$

C denotes the # of final clusters obtained, *N* the number of samples used throughout the whole incremental learning procedure, and μ_{ij} the membership degree of sample *j* in cluster *i*. For hard clustering, μ_{ij} is either always 0 (sample does not fall within cluster *i*) or 1 (sample falls within cluster *i*). The lower Eq. (36) gets, the higher the quality of the cluster partition becomes. A sensitivity analysis on the clustering algorithm parameters will be provided for the two-dimensional data sets. Optimal settings will be then used for the high-dimensional data sets in order to verify the "plug-and-play" capability of the methods for new data stream problems.

6.1. Results on two-dimensional data sets

Tables 2 and 3 present the results for the six low-dimensional streaming clustering set. The first value in each cell denotes the number of extracted clusters (to be compared with the real number in the first column), the second the quality of the cluster partition (the lower the better) and the third the range of optimal parameter settings leading to the same or very similar clustering results (lowest value of Eq. (36) + maximal 10%). Thereby, the first entry contains the number of clusters corresponding to the best (lowest) quality value (second entry) over the variation range of the method's parameter setting. The last two rows represent the results for the batch clustering methods, seeing all the data samples at once. Please note that for *eVQ-AMS* (AutoClust) we tested a very large range of *fac*, namely until $fac = 20$ in Eq. (7) and obtained very similar results, indicating a very low sensitivity and pointing to a plug-and-playable method.

The best method for each data set is highlighted in bold font, whereas the cluster quality is the first choice for ranking, the deviation of the extracted number of clusters to the real number contained in the sets the second choice and the width of the optimal setting interval the third choice – please note that some methods may perform similar in terms of extracting the number of clusters, but the quality may be quite different, which means that the clusters are not ideally placed according to the natural distribution of the data clouds. *eVQ-A*, i.e. the direct extension of *eVQ* to arbitrary ellipsoidal clusters, outperform

[4] http://cs.joensuu.fi/sipu/datasets/.
[5] http://archive.ics.uci.edu/ml/.

Table 2
Quality of clustering partitions achieved by our dynamic clustering variants without merging (eVQ-A), with merging operations (eVQ-AM) and with both, merging and splitting operations (eVQ-AMS – AutoClust) and state-of-the-art methods (incremental and batch); the first sub-column in each column denotes the number of clusters extracted (compare with the real number in the first row (values in braces)), the second one the validation index value (the lower the better) and the third one the optimal parameter setting (a broader range indicates a lower sensitivity). The values in bold font correspond to the best qualities achieved for each data set.

Method	A1 (20)			A2 (35)			S1 (15)		
	# of cl.	Qual.	Range	# of cl.	Qual.	Range	# of cl.	Qual.	Range
eVQ	14	0.47	[0.275]	40	0.63	[0.175,0.2]	14	0.25	[0.225]
eVQ-A (new)	21	0.40	[3.5,4.0]	33	0.39	4.0	15	0.20	[4.5,5.0]
eVQ-AM (new)	**20**	**0.39**	**[2.5,4.0]**	**34**	**0.35**	**[2.5,4.0]**	**15**	**0.20**	**[3.0,5.0]**
eVQ-AMS (AutoClust)	**20**	**0.39**	**[2.5,20.0]**	**35**	**0.34**	**[3.0,20.0]**	**15**	**0.20**	**[3.0,20.0]**
Neural gas	12	0.57	[0.275,0.3]	43	0.82	[0.175]	15	0.26	[0.225,0.25]
ESOMs	12	0.57	[0.275,0.3]	23	0.82	[0.225]	13	0.29	[0.25]
k-means (batch)	20	0.83	NA	35	0.98	NA	15	0.68	NA
Subtr. cl. (batch)	20	0.40	[0.125,0.175]	35	0.37	[0.1,0.15]	15	0.21	[0.05,0.2]

Table 3
Quality of clustering partitions achieved by our dynamic clustering variants without merging (eVQ-A), with merging operations (eVQ-AM) and with both, merging and splitting operations (eVQ-AMS – AutoClust) and state-of-the-art methods (incremental and batch); the first sub-column in each column denotes the number of clusters extracted (compare with the real number in the first row (values in braces)), the second one the validation index value (the lower the better) and the third one the optimal parameter setting (a broader range indicates a lower sensitivity). The values in bold font correspond to the best qualities achieved for each data set.

Method	S2 (15)			S3 (15)			R15 (15)			Avg. qual
	# of cl.	Qual.	Range	# of cl.	Qual.	Range	# of cl.	Qual.	Range	
eVQ	15	0.31	[0.225]	16	0.50	[0.2,0.25]	16	0.41	[0.275,0.3]	0.43
eVQ-A (new)	15	0.33	[4.5]	7	0.68	[4.0]	15	0.23	[3.0]	0.37
eVQ-AM (new)	15	0.28	[2.5,4.0]	12	0.39	[2.5,4.0]	**15**	**0.21**	**[2.5,4.0]**	**0.30**
eVQ-AMS (AutoClust)	15	0.28	[2.5,20.0]	15	0.38	[2.5,20.0]	**15**	**0.22**	**[2.5,20.0]**	**0.30**
Neural gas	14	0.28	[0.225]	12	0.46	[0.25,0.3]	14	0.41	[0.25,0.3]	0.47
ESOMs	14	0.28	[0.225]	8	0.52	[0.3]	13	0.41	[0.3,0.35]	0.48
k-means (batch)	15	0.88	NA	15	0.38	NA	15	1.27	NA	0.84
Subtr. cl. (batch)	**15**	**0.27**	**[0.05,0.15]**	**15**	**0.39**	**[0.125]**	15	0.24	[0.1,0.15]	0.31

eVQ for 4 out of 6 data sets, but in all cases performs weaker than *eVQ-AM* and *eVQ-AMS*. The average qualities of *eVQ-AM* and *eVQ-AMS* are the same, but *eVQ-AMS* allows a much larger variation range of *fac* in the tolerance radius (Eq. (7)), thus having a much lower sensitivity, a more detailed explanation below. Thus, *eVQ-AMS* is superior to the other *eVQ* variants.

It is also easy to realize that *eVQ-AMS* can outperform the other incremental clustering methods as well as the two batch clustering methods, whereas *subtractive clustering* performs almost similar (0.31 versus 0.30 average quality of the final cluster partition – see the last column Table 3). Surprisingly, *k*-means produces almost useless results with much higher validation index values than the other methods, despite it is parameterized with the real number of clusters in advance.

An interesting and useful observation is the sensitivity of the method with respect to the parameter settings of its learning engine, reflected by the last entries in each cell. This gives rise how stable one method is with respect to different parameterizations; in fact, in an on-line incremental learning context, the best settings are not known a priori. Therefore, a large band range of optimal settings is highly recommended, allowing the user a large variation while still achieving a similar clustering result. The variation range is calculated as

$$range = \frac{b-a}{a} \tag{37}$$

where $[a,b]$ the interval of optimal parameter settings. Best values (largest intervals) are again obtained for *eVQ-AM(S)* and *subtractive clustering*, whereas the cut-set of optimal intervals over all data sets in case of *eVQ-AM* is $[3.0,4.0]$, i.e. allowing an expected variation range of 33.3%, whereas for all other methods the cut-set is empty, thus the optimal setting varies from data set to data set. This finally brings an enormous increase in useability of *eVQ-AM* over the original *eVQ* as well as over all other methods. The reason for this less sensitivity (more stability) is mainly because of the integrated merging operation which compensates over-clustering effects due to an inappropriate parametrization of the Mahalanobis distance radius for the actual data stream under investigation. The sensitivity is even more improved in the case of *eVQ-AMS*, as we tested a very large range of *fac*, namely until $fac = 20$ in Eq. (7) and obtained very similar results in most of the data sets. With such a high *fac*, *eVQ-A* as well as *eVQ-AM* extracted only 1 cluster for the whole sets as joining all samples together. In this sense, the splitting procedure almost works perfectly as still being able to dynamically realize the real underlying data distribution (without any manual parameter tuning), successively splitting big clusters, as soon as they contain two density areas.

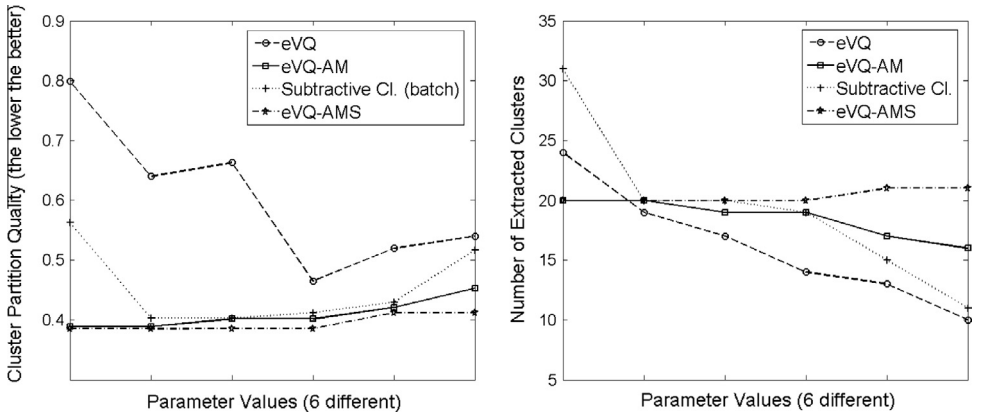

Fig. 13. Left: sensitivity of *eVQ* (dashed line), *eVQ-AM* (solid line), *eVQ-AMS* (dashed dotted line) and *subtractive clustering* (dotted line) according to 6 variations of their most influential parameter (steering plasticity versus stability) with respect to cluster partition quality on the A1 data set; right: the same with respect to the extracted number of clusters; note the more constant (more stable = less sensitive) curve produced by *eVQ-AMS*, slightly outperforming *eVQ-AM*.

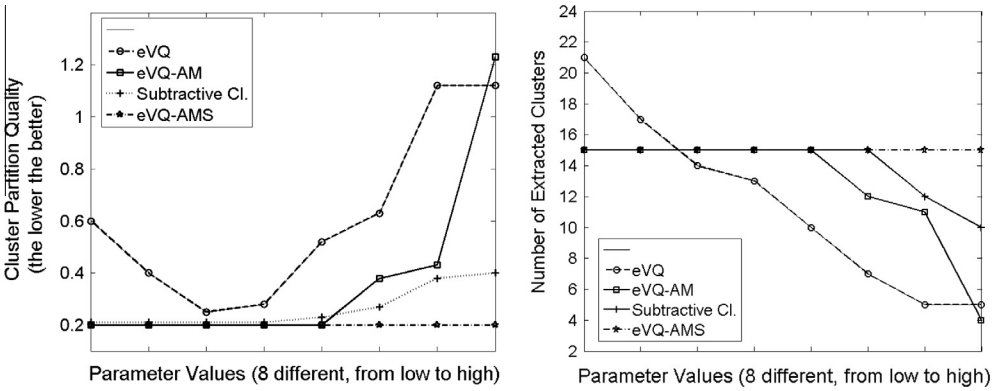

Fig. 14. The same as in Fig. 13 for the S1 data set; here, *eVQ-AMS* is significantly more stable than *eVQ-AM* towards higher values of the *fac* parameter (6, 7, 8 in value), controlling the Mahalanobis tolerance radius in Eq. (7); in this sense, splitting overcomes a too highly set learning parameter, which affects an adjoining of separate data clouds to one cluster (in the right image, it can be clearly seen that only 12, 10 and 4 clusters are extracted with *eVQ-AM* for $fac = 6, 7, 8$).

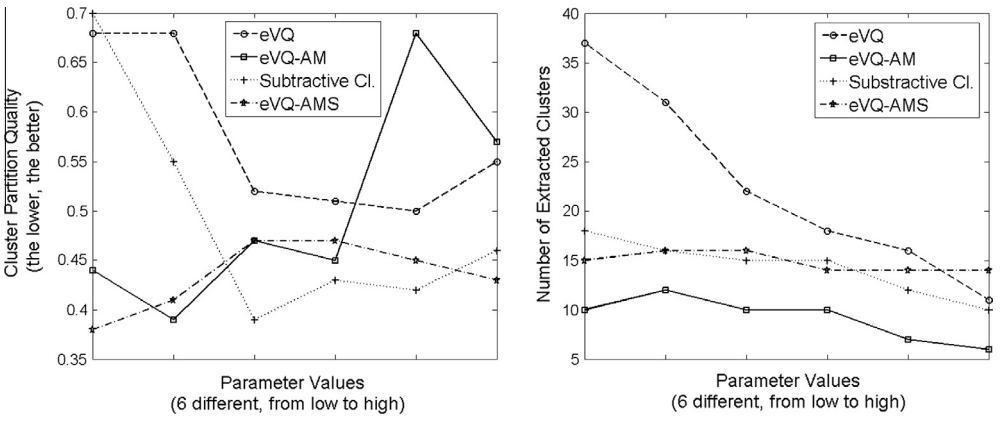

Fig. 15. The same as in Fig. 13 for the S3 data set, including a high noise (overlap) level in the data clouds, still *eVQ-AMS* is more stable in terms of extracted # of clusters and also partition quality, but the gap to the performance of *substractive clustering* is lower than in case of noise free data S1.

Table 4
Variance in the quality of the finally obtained cluster partitions (after stream ended) and in the extracted number of clusters according to different parameterizations of the methods.

Method	A1 (20)	A2 (35)	S1 (15)	S2 (15)	S3 (15)	R15 (15)	Avg. var.
eVQ	0.18	0.11	0.24	0.16	0.15	0.24	0.18
eVQ-A (new)	0.59	0.15	0.52	0.15	0.15	0.24	0.3
eVQ-AM (new)	0.03	0.10	0.36	0.04	0.04	0.23	0.13
eVQ-AMS (new)	0.02	0.03	0.03	0.02	0.03	0.004	0.02
Neural gas	0.17	0.33	0.22	0.23	0.12	0.23	0.21
ESOMs	0.51	0.40	2.00	0.23	2.70	0.20	1.01
k-means (batch)	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Subtr. cl. (batch)	0.08	0.12	0	0.04	0.16	0.16	0.09

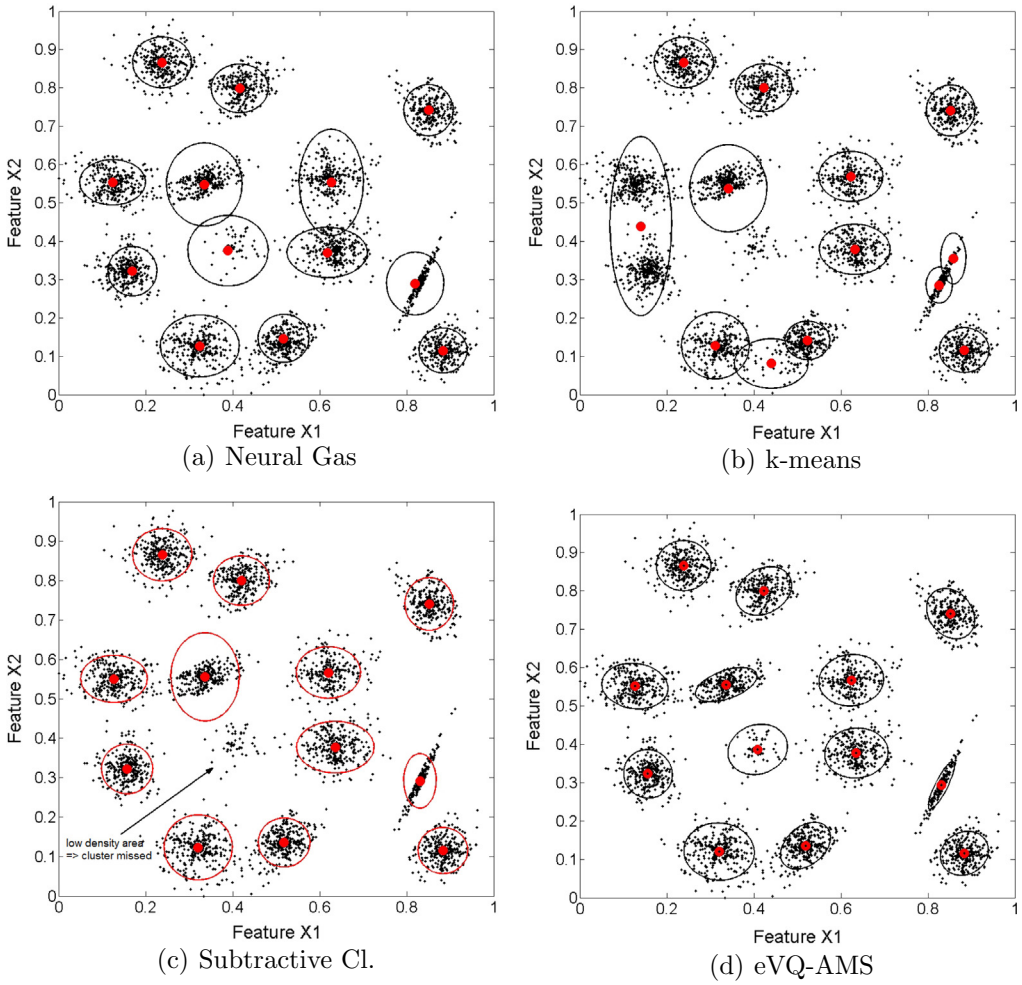

Fig. 16. Cluster partitions for first 4000 samples in S1 set obtained by (a): neural gas (incremental sample-wise update internally), (b): *k*-means (parameterizing # of real clusters), (c): subtractive clustering (optimal radius according to Table 2) and (d): *eVQ-AMS* (optimal radius according to Table 2 – when increasing the radius to 10, 15 or 20, still the same cluster partition is achieved).

Fig. 13 shows an example of the sensitivity of quality values (left image) and number of extracted clusters (right image) for the A1 data set, comparing *eVQ* (dashed lines), *eVQ-AM* (solid lines) and *eVQ-AMS* (dashed-dotted) with *subtractive clustering* (dotted lines). Obviously, *eVQ-AMS* produces the most constant curves for both cases, thus showing a lower sensitivity than the other methods. Subtractive clustering is a bit behind *eVQ-AM*, whereas *eVQ* is clearly the worst, depending heavily on the concrete setting of the vigilance parameter. A similar picture is drawn for S1 and S3 data sets as shown in Figs. 14 and 15.

A broader picture of this aspect is presented in Table 4 (same rows and columns as in Table 2), including the variance of cluster partition qualities (w.r.t. to the optimal values from Table 2) over the complete parametrization ranges of the various

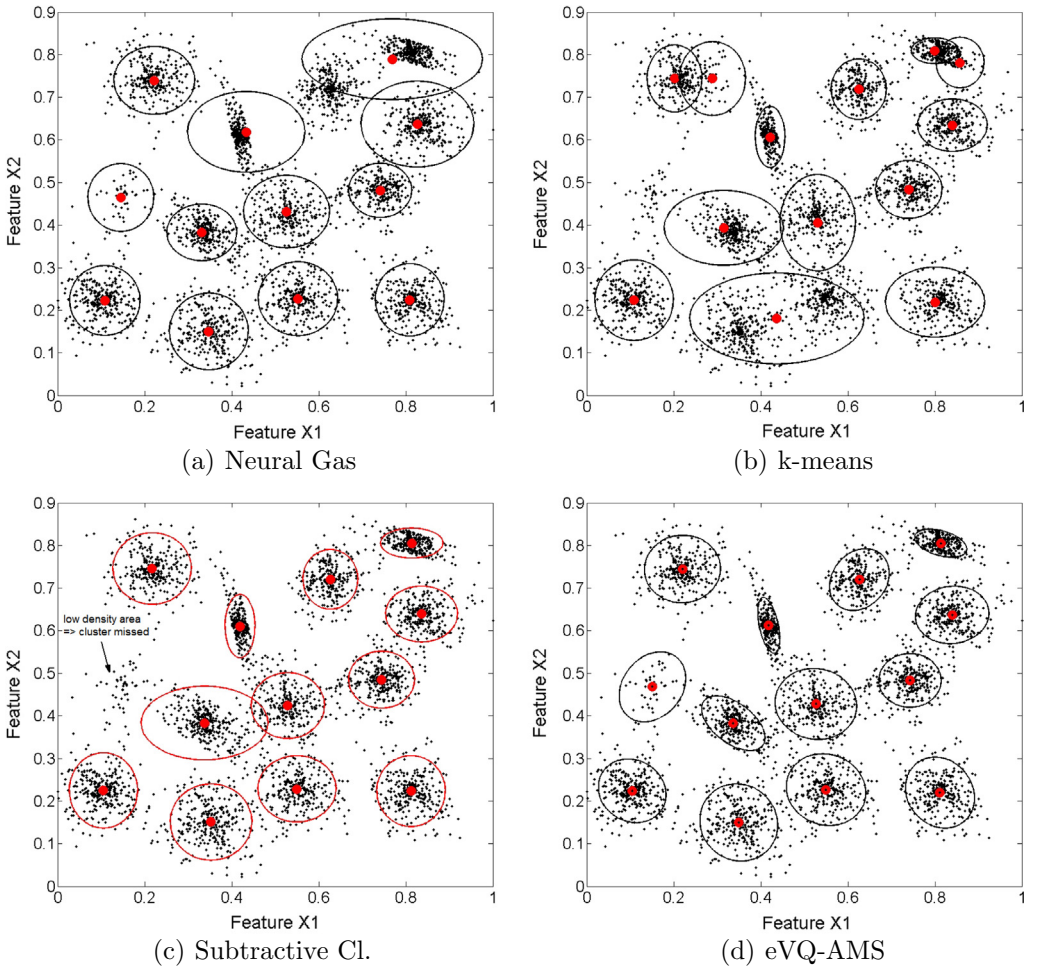

Fig. 17. Cluster partitions for first 4000 samples in S2 set obtained by (a): by neural gas (incremental sample-wise update internally), (b): *k*-means (parameterizing # of real clusters), (c): subtractive clustering (optimal radius according to Table 2) and (d): *eVQ-AMS* (optimal radius according to Table 2 – when increasing the radius to 10, 15 or 20, still the same cluster partition is achieved).

methods as described above (the lower the less sensitive). For the sake of clarity, the variance is here presented in terms of the standard deviation to the optimal values. From this table it gets clear that *eVQ-AMS* is able to provide the lowest variances and thus the lowest sensitivity over the complete parameter variation range, and this with the most qualitative cluster partitions (as underlined in Table 2). *eVQ-AM* and *subtractive clustering* are behind, performing approximately equally. Neural gas provides good variance values, however the optimal values are clearly worse than those of *eVQ-A*, *eVQ-AM eVQ-AMS* and *subtractive clustering*.

Finally, we exemplarily show some obtained cluster partitions by four methods on the first 4000 samples of S1 and S2 data sets (containing 13 clusters each) and on the whole R15 data set (containing 15 clusters) in Figs. 16–18, including neural gas in (a), *k*-means in (b), subtractive clustering in (c) and our method in (d). It can be clearly seen that in all cases *k*-means, despite it is a batch method, has some problems to extract appropriate clusters, as sometimes compact data clouds are split into two clusters and sometimes too large clusters are extracted covering two data clouds. Subtractive clustering does the job better, but has problems when clusters with different densities occur in different regions of the feature space as is the case in S1 and S2. The lower dense data regions are usually ignored as having a lower potentials than the other ones – as is indicated in Figs. 16 and 17 by arrows. Furthermore, subtractive clustering looses a bit in accuracy in terms of the trends and shapes of the clusters, as favoring ellipsoids in main position. Neural gas is suffering a bit from the same problem as conventional *eVQ* (similar to as shown in Fig. 4), as not being able to represent clusters in arbitrary rotation.

6.2. Results on high-dimensional data sets

For the tests on the high-dimensional data sets, we tried a "plug-and-play" option mimicking the real on-line data stream mining case, not being able to tune the parameters as no various trial-and-error runs are allowed (due to the single-pass

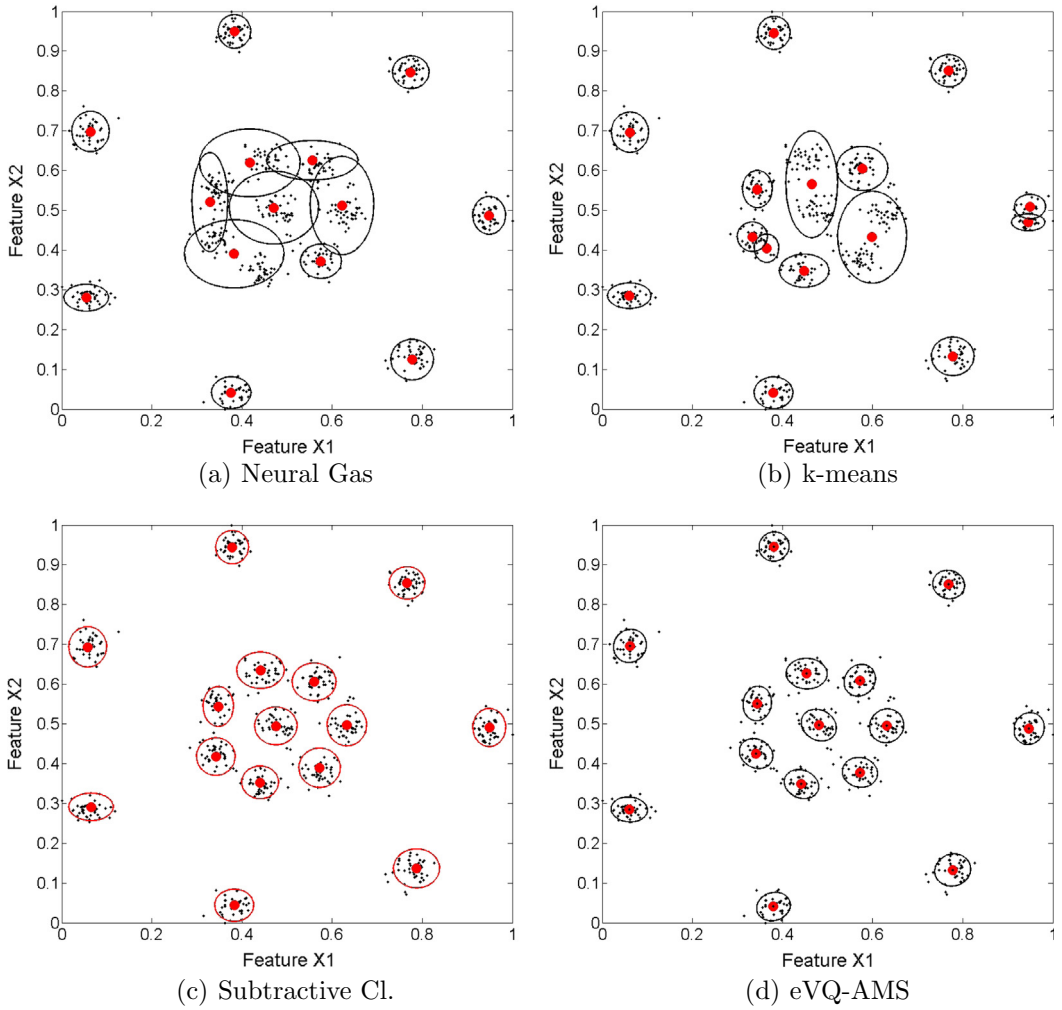

Fig. 18. Cluster partitions for R15 data set obtained by (a): neural gas (incremental sample-wise update internally), (b): *k*-means (parameterizing # of real clusters), (c): subtractive clustering (optimal radius according to Table 2) and (d): *eVQ-AMS* (optimal radius according to Table 2 – when increasing the radius to 10, 15 or 20, still the same cluster partition is achieved).

nature of streams). Thus, we followed the parametrization suggestions from the results on the two-dimensional data sets by taking the average of the optimal setting intervals from Tables 2 and 3. For the incremental learning methods, this means that we used $0.24 * \frac{\sqrt{dim}}{\sqrt{2}}$ for *eVQ*, $0.25 * \frac{\sqrt{dim}}{\sqrt{2}}$ for neural gas, $0.28 * \frac{\sqrt{dim}}{\sqrt{2}}$ for ESOMs, $fac = 4.1$ for *eVQ-A*, $fac = 3.3$ for *eVQ-AM* and a very high value of $fac = 10.0$ for *eVQ-AMS* (equals approx. 3 times the expected tolerance region when using a significance level of $\alpha = 0.05$), with *dim* the dimensionality of the learning problem (in case of $dim = 2$ they reduce to the values shown in Tables 2 and 6). For *subtractive clustering*, we applied a value of 0.15 for the range of influence of a cluster in each single dimension. The results in terms of number of clusters extracted and final obtained cluster partition qualities are reported in Table 5. In bold font, the best method for each data set in terms of the latter measure is shown.

Obviously, in all cases *eVQ-AMS* is the best option in terms of cluster partition quality and also in average (last column), only for KDDCup data set the state-of-the-art method *ESOMs* performs better than *eVQ-AM*. Also remarkable is the fact that, despite a very high setting of *fac* in *eVQ-AMS*, still the implicit cluster structure could be resolved, even with a slight out-performance of the others in terms of partition quality. Furthermore, it is interesting to see that the batch approaches *k*-means, subtractive clustering and Gustafson–Kessel (GK) could not produce any results for the two big streams (including more than 100 K samples). In fact, GK and *k*-means caused an out of memory crash in MATLAB for KDDcup data, while subtractive clustering did not stop for several days on Animals and KDDcup data. Furthermore, GK produced four clusters for Animals data set which were all exactly equal (same prototypes and covariance matrices). Thus, we can conclude that for large streams, well-known and renowned batch clustering methods are hardly being able to produce any reliable results.

The computation times for incremental updating the cluster partition with one new incoming sample in case of KDDCup data set (large stream with high dimensionality) are as follows: $3.37 * 10^{-4}$ seconds for *eVQ*, $6.23 * 10^{-4}$ seconds for *eVQ-A*,

Table 5
Quality of high-dimensional clustering partitions achieved by our dynamic clustering variants without merging (eVQ-A), with merging operations (eVQ-AM) and with full merging and splitting functionality (eVQ-AMS) proposed in this paper, as well as state-of-the-art methods (incremental and batch); the first value denotes the number of clusters extracted (compare with the real number in the first row (values in braces)), the second the validation index value (the lower the better). The values in bold font correspond to the best qualities achieved for each data set.

Method	Gaussian 6-dim (9)	Gaussian 10-dim (9)	Yeast (10)	DIM032 (16)
eVQ	40/1.46	36/1.33	24/0.78	16/0.04
eVQ-A (new)	**9/0.061**	**9/0.067**	5/0.76	**16/0.039**
eVQ-AM (new)	**9/0.061**	**9/0.067**	**8/0.45**	17/0.05
eVQ-AMS (new)	**9/0.061**	**9/0.067**	**8/0.45**	**16/0.039**
Neural gas	31/1.41	77/1.72	25/0.94	16/0.04
ESOMs	25/1.49	40/1.79	15/0.90	16/0.04
k-means (batch)	9/3.3	9/4.4	10/0.88	16/5.2
Subtr. cl. (batch)	9/0.064	9/0.071	23/1.45	16/0.04
GK (batch)	**9/0.061**	9/0.69	N/A/N/A	16/2.6

Table 6
Quality of high-dimensional clustering partitions achieved by our dynamic clustering variants without merging (eVQ-A), with merging operations (eVQ-AM) and with full merging and splitting functionality (eVQ-AMS) proposed in this paper, as well as state-of-the-art methods (incremental and batch); the first value denotes the number of clusters extracted (compare with the real number in the first row (values in braces)), the second the validation index value (the lower the better). The values in bold font correspond to the best qualities achieved for each data set.

Method	DIM064 (16)	Animals (4)	KDDCup (2000)	Avg. qual.
eVQ	16/0.024	20/1.85	62/2.51	0.43
eVQ-A (new)	17/0.08	2/0.29	13/2.55	0.37
eVQ-AM (new)	16/0.024	3/0.17	54/2.03	**0.30**
eVQ-AMS (new)	**16/0.023**	**4/0.09**	**149/1.33**	**0.29**
Neural gas	16/0.024	11/2.07	45/1.98	0.47
ESOMs	16/0.024	4/0.78	23/1.84	0.48
k-means (batch)	16/3.1	4/0.78	N/A (out of mem)	0.84
Subtr. Cl. (batch)	16/0.025	N/A (not stopping)	N/A (not stopping)	0.31
GK (batch)	16/3.1	4/N/A	N/A (out of mem)	0.31

0.005 s for *eVQ-AM*, $1.86 * 10^{-4}$ seconds for *neural gas* and $3.06 * 10^{-4}$ seconds for eSOMs, where *eVQ-AMS* with splitting integrated slows a bit down to 0.012 s. However, still it is able to proceed single samples within (almost) milliseconds (as is also the case for all other methods), i.e. to cope with a data stream processing frequency of up to 100 Hz, while most of the other methods can even cope with a frequency of up to 1000 Hz.

7. Conclusion

In this paper, we presented a new algorithm for dynamically extracting *prototype-based clusters* employing convex shapes from data which is sample-wise fed into the memory (e.g. in case of on-line data streams). The approach extends conventional evolving vector quantization by allowing clusters to be in arbitrary positions (rather than main position), thus being able to model density areas with different orientations. It integrates two options, namely dynamic merging and splitting, for compensating homogenous cluster overlaps as well as heterogenous density areas within clusters. These may arise due to the nature of the upcoming data samples or by an inappropriately chosen tolerance radius. The achieved sensitivity of the tolerance radius is much lower than that of corresponding parameters in benchmark (state-of-the-art) algorithms – in fact, a very wide range of *fac* (> 10) could still resolve the implicitly contained cluster structure on all data sets. In this sense, our method can be seen as a step towards a parameter-free (or at least towards a parameter-sensitive-low) evolving clustering methods, operating in incremental sample-wise manner. All the state-of-the-art methods could not follow this low sensitivity resp. this wide allowed variation of the learning parameter's range in order to obtain the same (optimal) cluster partitions. The plug-and-play capability (an even stronger criterion as a low sensitivity) was further underlined by applying optimal parameter settings from the tuning phase on two-dimensional sets to high-dimensional streaming clustering data: our method was able to extract the real structures (except in one case). Furthermore, our method integrating merging and especially splitting could outperform state-of-the-art benchmark methods on all high-dimensional problems (when tested in a plug-and-play manner).

Limitations of our method include (1) the primary focus on prototype-based clusters supporting convex shapes, leading to over-clustering effects when more complex clusters with arbitrary shapes are contained in the stream and (2) a fixed essential (most sensitive) learning parameter *fac* whose unlucky parameterization can be indeed compensated by the split-and-merge techniques, but still this is a rich workaround on the price of computational speed, see the last paragraph of Section 6.2 (although it has some interesting pure methodological aspects regarding the characterization and

conceptualization of cluster homogeneity verses heterogeneity). Finally, another limitation of our approach is the lack of an appropriate drift handling, which requires a more intense update of cluster prototypes and ellipsoids in order to move clusters more flexibly and quickly to new density areas (represented by shifted/drifted data clouds). Currently, all operations are performed in a life-long learning context, i.e. by equally weighting all new incoming samples.

Future work thus includes (1) handling upcoming drifts in data streams, which requires an out-dating of older samples/clusters over time whenever appropriate to increase flexibility of cluster movements, (2) the usage of more complex shapes for prototype-based cluster representation in a closed analytical/mathematical form, (3) the extension and application of our method in the context of incremental, on-line micro-clustering for extracting clusters with arbitrary (non-predefined) shapes, and (4) the possibility to adapt the most essential learning parameter (*fac*) properly according to the current characteristics of the stream, reducing back-merge and -split operations.

Acknowledgements

The first author acknowledges the support of the Austrian COMET-K2 programme of the Linz Center of Mechatronics (LCM), funded by the Austrian Federal Government and the Federal State of Upper Austria. This publication reflects only the authors' views.

References

[1] W.C. Abraham, A. Robins, Memory retention the synaptic stability versus plasticity dilemma, Trends Neurosci. 28 (2) (2005) 73–78.
[2] P. Angelov, D. Filev, N. Kasabov, Evolving Intelligent Systems – Methodology and Applications, John Wiley & Sons, New York, 2010.
[3] P.P. Angelov, An approach for fuzzy rule-base adaptation using on-line clustering, Int. J. Approx. Reason. 35 (3) (2004) 275–289.
[4] P.P. Angelov, D. Filev, An approach to online identification of Takagi–Sugeno fuzzy models, IEEE Trans. Syst. Man Cybern., Part B: Cybern. 34 (1) (2004) 484–498.
[5] B. Ari, H.A. Guevenir, Clustered linear regression, Knowl.-Based Syst. 15 (2002) 169–175.
[6] T.R. Babu, M.N. Murty, S.V. Subrahmanya, Compression Schemes for Mining Large Datasets, Springer, London, Heidelberg, 2013.
[7] R. Babuska, Fuzzy Modeling for Control, Kluwer Academic Publishers, Norwell, Massachusetts, 1998.
[8] S. De Backer, P. Scheunders, Texture segmentation by frequency-sensitive elliptical competitive learning, Image Vis. Comput. 19 (9–10) (2001) 639–648.
[9] J. Beringer, E. Hüllermeier, Online clustering of parallel data streams, Data Knowl. Eng. 58 (2) (2006) 180–204.
[10] A. Bhattacharyya, On a measure of divergence between two statistical populations defined by their probability distributions, Bull. Calcutta Math. Soc. 35 (1943) 99–109.
[11] P.J. Bickel, E. Levina, Regularized estimation of large covariance matrices, Ann. Stat. 36 (1) (1943) 199–227.
[12] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: Massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[13] A. Bifet, R. Kirkby, Data Stream Mining – A Practical Approach, Technical report, Department of Computer Sciences, University of Waikato, Japan, 2011.
[14] C. Borgelt, Prototype-based Classification and Clustering, PhD thesis, Otto-von-Guericke-Universität Magdeburg, Magdeburg, Germany, 2005.
[15] A. Bouchachia, Evolving clustering: an asset for evolving systems, IEEE SMC Newslett. 36 (2011).
[16] A. Bouchachia, R. Mittermeir, Towards incremental fuzzy classifiers, Soft. Comput. 11 (2) (2006) 193–207.
[17] A. Bouchachia, M. Prossegger, Incremental spectral clustering, in: M. Sayed-Mouchaweh, E. Lughofer (Eds.), Learning in Non-Stationary Environments: Methods and Applications, Springer, New York, 2012, pp. 77–99.
[18] A. Bouchachia, C. Vanaret, Incremental learning based on growing gaussian mixture models, in: Proceedings of 10th International Conference on Machine Learning and Applications (ICMLA 2011), Honululu, Haweii, 2011, pp. 47–52.
[19] J. Fisher Box, Guinness, gosset, fisher, and small samples, Stat. Sci. 2 (1) (1987) 45–52.
[20] K. Chen, Z. Jin, Partial linear regression models for clustered data, J. Am. Stat. Assoc. 101 (473) (2006) 195–204.
[21] F. Chu, C. Zaniolo, Fast and light boosting for adaptive mining of data streams, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2004, pp. 282–292.
[22] J. Dean, Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners, John Wiley and Sons, Hoboken, New Jersey, 2012.
[23] A. Declercq, J.H. Piater, Online learning of gaussian mixture models – a two-level approach, in: Proceedings of the 3rd International Conference on Computer Vision Theory and Applications VISAPP, Funchal, Portugal, 2008, pp. 605–611.
[24] J. Demmal, I. Dumitriu, O. Holtz, Fast linear algebra is stable, Numer. Math. 108 (1) (2007) 59–91.
[25] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. Roy. Stat. Soc., Ser. B 39 (1) (1977) 1–38.
[26] D. Deng, N. Kasabov, On-line pattern analysis by evolving self-organizing maps, Neurocomputing 51 (2003) 87–103.
[27] J.A. Dickerson, B. Kosko, Fuzzy function approximation with ellipsoidal rules, IEEE Trans. Syst. Man Cybern. – Part B: Cybern. 26 (4) (1996) 542–560.
[28] A. Djouadi, O. Snorrason, F. Garber, The quality of training-sample estimates of the bhattacharyya coefficient, IEEE Trans. Pattern Anal. Mach. Intell. 12 (1) (1990) 92–97.
[29] D. Dovzan, I. Skrjanc, Recursive clustering based on a Gustafson–Kessel algorithm, Evol. Syst. 2 (1) (2011) 15–24.
[30] K.L. Du, Clustering: a neural network approach, Neural Networks 23 (2010) 89–107.
[31] C. Eitzinger, W. Heidl, E. Lughofer, S. Raiser, J.E. Smith, M.A. Tahir, D. Sannen, H. van Brussel, Assessment of the influence of adaptive components in trainable surface inspection systems, Mach. Vis. Appl. 21 (5) (2010) 13–626.
[32] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, 1996, pp. 226–231.
[33] F. Farnstrom, J. Lewis, C. Elkan, Scalability for clustering algorithms revisited, in: SIGKDD Explorations, vol. 2(1), London, 2000, pp. 51–57.
[34] D. Filev, O. Georgieva, An extended version of the Gustafson–Kessel algorithm for evolving data stream clustering, in: P. Angelov, D. Filev, N. Kasabov (Eds.), Evolving Intelligent Systems: Methodology and Applications, John Wiley & Sons, New York, 2010, pp. 273–299.
[35] B. Fritzke, A growing neural gas network learns topologies, Adv. Neural Inform. Process. Syst. 7 (1995) 625–632.
[36] J. Gama, Knowledge Discovery from Data Streams, Chapman & Hall/CRC, Boca Raton, Florida, 2010.
[37] G. Gan, C. Ma, J. Wu, Data Clustering: Theory, Data Clustering: Theory, Algorithms, and Applications (Asa-Siam Series on Statistics and Applied Probability), Society for Industrial & Applied Mathematics, USA, 2007.
[38] R.M. Gray, Vector quantization, IEEE ASSP Mag. 1 (2) (1984) 4–29.
[39] S. Guha, R. Rastogi, K. Shim, ROCK: a robust clustering algorithm for categorical attributes, Inf. Syst. 25 (5) (2000) 345–366.
[40] D. Gustafson, W. Kessel, Fuzzy clustering with a fuzzy covariance matrix, in: Proceedings of the IEEE CDC Conference 1979, San Diego, CA, USA, 1979, pp. 761–766.
[41] S.K. Halgamuge, L.P. Wang, Classification and Clustering for Knowledge Discovery, Springer Verlag, Berlin Heidelberg, 2005.

[42] P. Hall, Y.A. Hicks, A Method to Add Gaussian Mixture Models, Technical report, University of Bath, 2005.
[43] F.H. Hamker, RBF learning in a non-stationary environment: the stability–plasticity dilemma, in: R.J. Howlett, L.C. Jain (Eds.), Radial Basis Function Networks 1: Recent Developments in Theory and Applications, Physica Verlag, Heidelberg, New York, 2001, pp. 219–251.
[44] L. Hartert, M. Sayed-Mouchaweh, P. Billaudel, A semi-supervised dynamic version of fuzzy *k*-nearest neighbors to monitor evolving systems, Evol. Syst. 1 (1) (2010) 3–15.
[45] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, Second Edition., Springer, New York, Berlin, Heidelberg, 2009.
[46] S. Haykin, Neural Networks: A Comprehensive Foundation, 2nd Edition., Prentice Hall Inc., Upper Saddle River, New Jersey, 1999.
[47] A. Hinneburg, D. Keim, An efficient approach to clustering in large multimedia databases with noise, in: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD98), 1998, pp. 58–65.
[48] L.O. Jimenez, D.A. Landgrebe, Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data, IEEE Trans. Syst. Man Cybern., Part C: Rev. Appl. 28 (1) (1998) 39–54.
[49] G. Karypis, E.H. Han, V. Kumar, Chameleon: hierarchical clustering using dynamic modeling, IEEE Comput. 32 (8) (1999) 68–75.
[50] N. Kasabov, Evolving Connectionist Systems: The Knowledge Engineering Approach, second ed., Springer Verlag, London, 2007.
[51] T. Kohonen, Se5lf-Organizing Maps, second ed.,., Springer, Berlin Heidelberg, 1995.
[52] K. Krishnamoorthy, T. Mathew, Statistical Tolerance Regions: Theory, Applications, and Computation, John Wiley & Sons, Hoboken, New Jersey, 2009.
[53] S. Kullback, R.A. Leibler, On information and sufficiency, Ann. Math. Stat. 22 (1) (1951) 79–86.
[54] A.A. Kurzhanskiy, P. Varaiya, Ellipsoidal Toolbox, Technical report UCB/EECS-2006-46, EECS Department, University of California, Berkeley, May 2006.
[55] A. Lemos, W. Caminhas, F. Gomide, Adaptive fault detection and diagnosis using an evolving fuzzy classifier, Inf. Sci. 220 (2013) 64–85.
[56] E. Lughofer, Extensions of vector quantization for incremental clustering, Pattern Recogn. 41 (3) (2008) 995–1011.
[57] E. Lughofer, FLEXFIS: a robust incremental learning approach for evolving TS fuzzy models, IEEE Trans. Fuzzy Syst. 16 (6) (2008) 1393–1410.
[58] E. Lughofer, Evolving Fuzzy Systems – Methodologies, Advanced Concepts and Applications, Springer, Berlin Heidelberg, 2011.
[59] E. Lughofer, On-line incremental feature weighting in evolving fuzzy classifiers, Fuzzy Sets Syst. 163 (1) (2011) 1–23.
[60] E. Lughofer, A dynamic split-and-merge approach for evolving cluster models, Evol. Syst. 3 (3) (2012) 135–151.
[61] E. Lughofer, Flexible evolving fuzzy inference systems from data streams (FLEXFIS++), in: M. Sayed-Mouchaweh, E. Lughofer (Eds.), Learning in Non-Stationary Environments: Methods and Applications, Springer, New York, 2012, pp. 205–246.
[62] E. Lughofer, On-line assurance of interpretability criteria in evolving fuzzy systems – achievements, new concepts and open issues, Inf. Sci. 251 (2013) 22–46.
[63] E. Lughofer, P. Angelov, Handling drifts and shifts in on-line data streams with evolving fuzzy systems, Appl. Soft Comput. 11 (2) (2011) 2057–2068.
[64] E. Lughofer, J.-L. Bouchot, A. Shaker, On-line elimination of local redundancies in evolving fuzzy systems, Evol. Syst. 2 (3) (2011) 165–187.
[65] E. Lughofer, O. Buchtala, Reliable all-pairs evolving fuzzy classifiers, IEEE Trans. Fuzzy Syst. 21 (4) (2013) 625–641.
[66] E. Lughofer, V. Macian, C. Guardiola, E.P. Klement, Identifying static and dynamic prediction models for NOx emissions with evolving fuzzy systems, Appl. Soft Comput. 11 (2) (2011) 2487–2500.
[67] R. De Maesschalck, D. Jouan-Rimbaud, D.L. Massart, The mahalanobis distance, Chemom. Intell. Lab. Syst. 50 (2000) 1–18.
[68] P.C. Mahalanobis, On the generalised distance in statistics, Proc. Nat. Inst. Sci. India 2 (1) (1936) 49–55.
[69] L.M. Manevitz, M. Yousef, One-class SVMs for document classification, J. Mach. Learn. Res. 2 (2001) 139–154.
[70] J. Valente De Oliveira, W. Pedrycz, Advances in Fuzzy Clustering and its Applications, John Wiley & Sons, Hoboken, New Jersy, 2007.
[71] S. Pang, S. Ozawa, N. Kasabov, Incremental linear discriminant analysis for classification of data streams, IEEE Trans. Syst. Men Cybern., Part B: Cybern. 35 (5) (2005) 905–914.
[72] W. Pedrycz, A dynamic data granulation through adjustable clustering, Pattern Recogn. Lett. 29 (16) (2008) 2059–2066.
[73] W. Pedrycz, F. Gomide, Fuzzy Systems Engineering: Toward Human-Centric Computing, John Wiley & Sons, Hoboken, New Jersey, 2007.
[74] W. Pedrycz, P. Rai, Collaborative clustering with the use of fuzzy *c*-means and its quantification, Fuzzy Sets Syst. 159 (18) (2008) 2399–2427.
[75] W. Pedrycz, R. Weber, Special issue on soft computing for dynamic data mining, Appl. Soft Comput. 8 (4) (2008) 1281–1282.
[76] M. Pratama, S.G. Anavatti, E. Lughofer, GENEFIS: towards an effective localist network, IEEE Trans. Fuzzy Syst. 22 (3) (2014) 547–562.
[77] S.J. Qin, W. Li, H.H. Yue, Recursive PCA for adaptive process monitoring, J. Process Control 10 (5) (2000) 471–486.
[78] S. Roweis, EM algorithms for PCA and SPCA, Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems, 10, MIT Press, Cambridge, MA, USA, 1997, pp. 626–632.
[79] M. Sayed-Mouchaweh, E. Lughofer, Learning in Non-Stationary Environments: Methods and Applications, Springer, New York, 2012.
[80] B. Scholkopf, Estimating the support of a high-dimensional distribution, Neural Comput. 13 (7) (2001).
[81] G.E. Schwarz, Estimating the dimension of a model, Ann. Stat. 6 (2) (1978) 461–464.
[82] F. Schwenker, H.A. Kestler, G. Palm, Three learning phases for radial-basis-function networks, Neural Networks 14 (2001) 439–458.
[83] A. Shaker, E. Lughofer, Self-adaptive and local strategies for a smooth treatment of drifts in data streams, Evol. Syst. 5 (4) (2014) 239–257.
[84] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[85] M. Song, H. Wang, Highly efficient incremental estimation of gaussian mixture models for online data stream clustering, in: Kevin L. Priddy (Ed.), Intelligent Computing: Theory and Applications III, Proceedings of the SPIE, vol. 5803, 2005, pp. 174–183.
[86] H. Sun, S. Wang, Measuring the component overlapping in the gaussian mixture model, Data Min. Knowl. Disc. 23 (2011) 479–502.
[87] K. Tabata, M. Sato, M. Kudo, Data compression by volume prototypes for streaming data, Pattern Recogn. 43 (9) (2010) 3162–3176.
[88] G. Vachkov, Similarity analysis and knowledge acquisition by use of evolving neural models and fuzzy decision, in: P. Angelov, D. Filev, N. Kasabov (Eds.), Evolving Intelligent Systems: Methodology and Applications, John Wiley & Sons, Hoboken, New Jersey, 2010, pp. 247–272.
[89] Jeffrey Scott Vitter, Random sampling with a reservoir, ACM Trans. Math. Softw. 11 (1985) 37–57.
[90] H. Wang, W. Fan, P.S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, USA, 2003, pp. 226–235.
[91] B.L. Welch, The generalization of 'students' problem when several different population variances are involved, Biometrika 34 (1–2) (1947) 28–35.
[92] H. Widiputra, R. Pears, N. Kasabov, Dynamic learning of multiple time series in a non-stationary environment, in: M. Sayed-Mouchaweh, E. Lughofer (Eds.), Learning in Non-Stationary Environments: Methods and Applications, Springer, New York, 2012, pp. 303–348.
[93] X.L. Xie, G. Beni, A validity measure for fuzzy clustering, IEEE Trans. Pattern Anal. Mach. Intell. 13 (48) (1991) 841–847.
[94] R. Xu, D. Wunsch, Survey of clustering algorithms, IEEE Trans. Neural Networks 16 (3) (2005) 645–678.
[95] R. Yager, D. Filev, Approximate clustering via the mountain method, IEEE Trans. Syst. Man Cybern. 24 (8) (1994) 1279–1284.
[96] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: and efficient data clustering method for very large databases, in: SIGMOD 96, Montreal, Canada, 1996, pp. 103–114.