Evolving Systems (2012) 3:135–151
DOI 10.1007/s12530-012-9046-5

ORIGINAL PAPER

A dynamic split-and-merge approach for evolving cluster models

Edwin Lughofer

Received: 20 September 2011 / Accepted: 12 January 2012 / Published online: 5 February 2012

Abstract This paper describes new dynamic split-and-merge operations for evolving cluster models, which are learned incrementally and expanded on-the-fly from data streams. These operations are necessary to resolve the effects of *cluster fusion* and *cluster delamination*, which may appear over time in data stream learning. We propose two new criteria for cluster merging: a touching and a homogeneity criterion for two ellipsoidal clusters. The splitting criterion for an updated cluster applies a 2-means algorithm to its sub-samples and compares the quality of the split cluster with that of the original cluster by using a penalized Bayesian information criterion; the cluster partition of higher quality is retained for the next incremental update cycle. This new approach is evaluated using two-dimensional and high-dimensional streaming clustering data sets, where feature ranges are extended and clusters evolve over time—and on two large streams of classification data, each containing around 500K samples. The results show that the new split-and-merge approach (a) produces more reliable cluster partitions than conventional evolving clustering techniques and (b) reduces impurity and entropy of cluster partitions evolved on the classification data sets.

Keywords Evolving cluster models · Cluster fusion and delamination · Dynamic split-and-merge · Touching and homogeneity criteria · Penalized Bayesian information criterion

E. Lughofer (✉)
Department of Knowledge-based Mathematical Systems, Johannes Kepler University of Linz, Linz, Austria
e-mail: edwin.lughofer@jku.at

1 Introduction

1.1 Motivation and state of the art

In current real world-applications, there is an increasing demand for models that can automatically to changing process dynamics in systems with high complexity, fine granularity, and time-varying characteristics. In order to address such dynamics, the methodology of evolving (intelligent) systems (EIS) (Angelov et al. 2010) has been developed over the last decade and plays an important role, as it employs approaches and mechanisms that update model parameters incrementally and evolve new model components as required on the basis of on-line recorded data streams that reflect process changes over time.

Evolving cluster models (ECM), an important sub-field of evolving intelligent systems, are designed to mine the huge amount of on-line data streams (Bifet et al. 2010) or very large data bases (VLDBs)[1] in an unsupervised learning context by clustering, grouping, and compressing data in a fast incremental manner. VLDBs usually contain tera bytes of data, which may even be stored over different spatial locations, for instance, consider the storage of data describing customers buying habits in a supermarket, long-term meteorological data,[2] or spectral data in a dynamic chemical production process (Varmuza and Filzmoser 2009). On-line data streams are characterized by data blocks which are usually not bounded in size and arriving continuously and in a specific order, over which the systems has no control (Gam 2010). Sometimes, ECMs are also referred to incremental or evolving clustering methods (Bouchachia 2011), because they can process data step-wise and update and evolve cluster partitions in

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

incremental learning steps. Deciding between parameter updating and evolution of new clusters is usually done on the basis on the current dynamics of the process. Several incremental clustering techniques have been proposed in the past, for example, objective-function-based approaches such as single-pass k-means (Farnstrom et al. 2000), a recursive Gustafson–Kessel approach (Dovzan and Skrjanc 2011), evolving neural-type models based on neural gas (Vachko 2010) and vector quantization (*eVQ*; Lughofer 2008) and the approach in Tabata and Kudo (2010), and approaches based on density criteria such as on-line Gaussian mixture models (Bouchachia and Vanaret 2011) and a recursive variant of subtractive clustering called *eClustering* (Angelo 2004).

A specific problem in ECMs arises whenever two or more local data clouds (each one modeled by a cluster) are moving together or are delaminate within one cluster (refer to Sect. 2 for a detailed problem description). This may results from changes in the characteristics of the underlying data distributions or as a result of refining and intensifying existing clusters which were roughly estimated/evolved at a preliminary stage based on a small snapshot of data samples. In such cases, clusters should be dynamically merged or split in order to maintain cluster partitions of high quality that follow the natural distribution of the data clouds. In Beringer and Hüllermeier (2006, 2007), an approach was presented which tackles this issue by integrating a block-wise split-and-merge concept into a k-means algorithm, where merged and split partitions are compared with the original partitions based on an extended version of the Xie and Beni (1991) validation index; however, for each complete data block only one cluster is merged or split, i.e., the number of clusters k is increased or decreased by 1. In Song and Wang (2005), clusters in the form of Gaussian mixture models (GMMs) are trained on-line by incremental learning mechanisms, integrating merging concepts that (1) are only applicable in chunk mode (new Gaussians are generated from each arriving data block) and (2) require significant computation time, as each new Gaussian is evaluated for merging with any of the existing Gaussians. Splitting was not considered in Song and Wang (2005). The on-line incremental GMMs in Hall and Hicks (2005) apply both, a merge and a split concept in incremental mode; however, this method is too slow for fast on-line learning because of a grouping of Gaussians using the Chernoff bound. A speed up is achieved in Declercq and Piater (2008), where suitability for merging is assessed based on the fidelity of a Gaussian using the Kolmogorov–Smirnoff test. However, this method causes an unnecessary increase in complexity because each new sample creates a new Gaussian, which often requires back-merging with existing Gaussians that are more significant.

1.2 Our approach

In this paper, we want to go a step further and bring in more dynamics in the evolving clustering process by integrating a joint incremental *on-line split-and-merge scenario*. This helps to overcome under- and over-clustered partitions which may arise due to specific changes in a data stream over time that affect data cloud shapes and distribution within regions of the feature space. This problem is considered in Sect. 2. The novel split-and-merge technique is characterized by the following aspects:

- Two merging criteria—a touching and a homogeneity condition—decide whether two clusters should be merged or not: clusters which touch each other or slightly overlap can be merged, if they fulfill the homogeneity criterion, i.e. if they form a homogeneous data region when joined together. This is in contrast to the approach in Lughofer et al. (2011), which merges only clusters that overlap significantly (according to fuzzy rules) and requires no homogeneity criterion. Thus, our approach can also be employed as a further complexity reduction step in fuzzy rule bases, using a touching criterion in place of a significant overlap.
- Upon fulfillment of the criteria, clusters are merged by weighted averaging of their centers (weights determined according to their support) and updating their spreads (ranges of influence) based on a recursive variance concept and including a coverage term.
- A split criterion based on a *penalized Bayesian information criterion* (*BIC*), evaluated for the actual clusters and hypothetically split clusters, updated in the previous incremental learning step. The lower the criterion gets, the higher the quality of the cluster partition becomes. Updated clusters are applying 2-means to cluster sub-samples.
- The whole split-and-merge algorithm can be connected with any incremental and evolving prototype-based clustering, extracting ellipsoidal clusters in main position and providing centers, spreads and support of the clusters: these three components will be used in the split and merging conditions and operations—thus they technically work and can be tried with any approach, to improve, refine the evolved cluster partitions, such as: *eClustering* (Angelov 2004) (using the fixed radius of clusters as spread) and its extended version in Angelov and Zhou (2006) (using the permanently component-wise updated radii of the clusters), *ECM* (Kasabov and Song 2002) (using the range of influence as spread), *eVQ* (evolving vector quantization) (Lughofer 2008), its extension in Tabata and Kudo (2010), *ePL* (Lima et al. 2010) (evolving Participatory Learning) (using the fixed radius r as spread), single pass k-means (Farnstrom et al. 2000) (when using the sufficient statistics over the discard sets to compute the cluster variances as spreads) and others.

The paper is structured as follows: the subsequent section described the need for a dynamic split-and-merge strategy, illustrated by examples favoring cluster merges

and splits; Sect. 3.1 defines the basic generic algorithm for dynamic cluster split-and-merge, and Sects. 3.2 and 3.3 introduce the merging criteria and the split criterion, respectively, including their implementation/realization. Section 4 discusses the additional computational costs caused by the split-and-merge techniques. Section 5 evaluates the whole approach and compares it with conventional ECMs using (a) two-dimensional streaming data sets (for which the clusters are visualized), (b) high-dimensional streaming clustering data sets (inspecting the number of found versus extracted clusters and the quality of the cluster partitions) and (c) high-dimensional classification data sets comprising 500K samples and multiple classes (for which entropy and purity of cluster partitions are evaluated with respect to the supervised class label information).

2 Problem statement

In ECMs, two types of problems may manifest, both of which point to dynamic changes in the characteristics of local data clouds in feature space over time (as the data stream progresses).

The first one arises whenever two clusters seem to be distinct at the beginning of the data stream, however may move together due to data samples that fill the intervening gap. This effect is called *cluster fusion*. Figure 1 shows such an occurrence, (a) demonstrating the partition (three distinct clusters) after loading the initial data block, (b) demonstrating the partition (two clusters moved together) due to the new data block.

Obviously, the two clusters can be merged into one joint data cloud in feature space. Preventing such cases is hardly possible, because an ECM always receives a snapshot of the data and tries to find the most suitable clustering. Note that the situation shown in Fig. 1 differs from a drift case (Klinkenber 2004), in which usually a complete local data cloud (cluster) moves from one position in feature space to another and the vacated position becomes obsolete, hence should be outdated, forgotten.

In extreme cases, the clusters may overlap significantly, causing a redundancy in a part of the feature space that can be handled by similarity measures (see Lughofer et al. 2011, where overlapping clusters were considered to be redundant rules in an evolving fuzzy system). In this paper, we go a step further by resolving occurrences as visualized in Fig. 1 within an unsupervised learning context. Therefore, we propose two criteria for cluster merging, a touching and a homogeneity condition: clusters which are touching or slightly overlapping can be merged, if they fulfill the homogeneity criterion, i.e., if they form a homogeneous data region when joined together. From a mathematical point of view, we can define such a situation by

$$\exists_{C_i,C_j,i\neq j}\left(C_i \cap C_j \neq \emptyset \wedge hom(C_i,C_j) \geq thresh\right) \quad (1)$$

with C_i representing cluster i.

The second case arises whenever one cluster (rule) in a local region of the data space seems appropriate at the beginning, but subsequently turns out to contain two distinct data clouds. This effect is called *cluster delamination*. An example is shown in Fig. 2b.

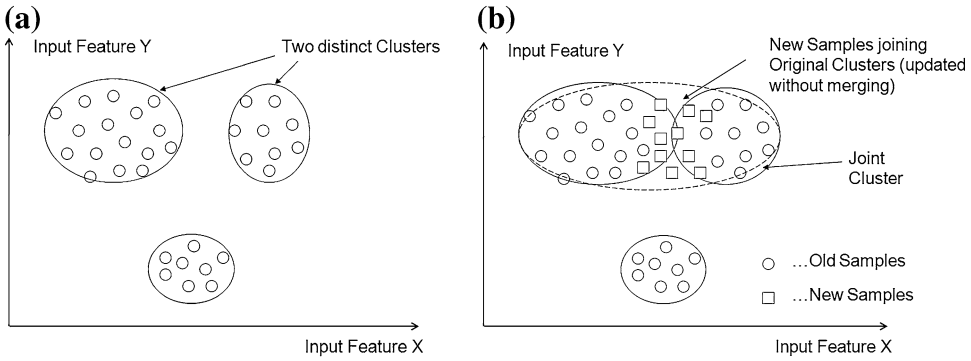

Fig. 1 **a** Three distinct clusters after loading the initial data block, **b** two distinct clusters moved together due to new data samples filling up the gap → cluster fusion effect (merge preferred as indicated by the *dotted ellipsis*)

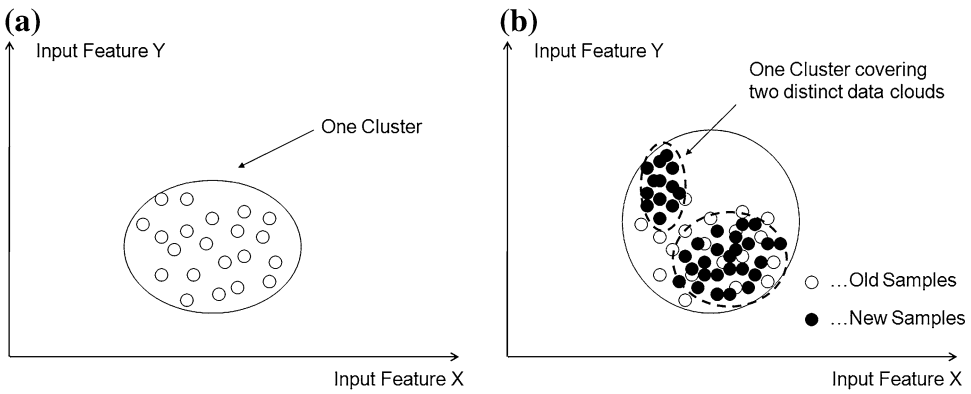

Fig. 2 **a** One data cloud, one cluster at the beginning, **b** two distinct data clouds in close proximity contained in one cluster → cluster delamination effect (split preferred as indicated by *dotted ellipses*)

Clearly, in such a case it would be preferable if the large cluster could be split in two to represent the disjoint regions more appropriately. From the mathematical point of view, we can define such a situation by:

$$\exists_{C_i}(C_{i_1} \subseteq C_i \wedge C_{i_2} \subseteq C_i \wedge Qual(\mathrm{C} \backslash C_i \cup \{C_{i_1}, C_{i_2}\}) \geq Qual(\mathrm{C})) \quad (2)$$

where C denotes the whole cluster partition (set of clusters), C_{i_1} and C_{i_2} are two sub-clusters within cluster C_i, and *Qual* stands for an arbitrary quality measure to calculate the quality of a cluster partition (we use a penalized version of the BIC, see Sect. 3.3). Note that such a case may also occur whenever a learning parameter (e.g., a threshold) is not chosen appropriately for the actual data stream or when it does not adapt to the actual data characteristics automatically. For instance, consider a distance parameter that decides whether a new cluster should be evolved or older clusters updated. If an excessively high value is chosen, distinct local data clouds may be joined together into one cluster. This tendency can be ameliorated within incremental learning steps by integrating a dynamic split mechanism.

Split and merge criteria need to be evaluated only for the updated clusters after each incremental learning step. We keep them independent from the selected/chosen cluster update algorithm, because the criteria evaluate only the nature and quality of the current (updated) cluster partition, independently of how clusters were moved, reset, evolved, expanded, etc.

3 On-line split-and-merge in ECMs

First, we define the basic split-and-merge algorithm, to bring more dynamics in the clustering process and to resolve cluster fusion and delamination. We consider only a prototype-based clustering nature in which cluster centers = prototypes and their ranges of influence (spread in each dimension) are extracted from the data. Then, we describe the merging and splitting criteria (when to merge or split clusters) and operations (how to merge and split clusters).

3.1 Basic algorithm

The basic steps for split-and-merge operations during incremental training after updating the cluster partition with each new incoming sample are summarized in Algorithm 1.

1. **Input:** Current cluster partition C, new input sample $\mathbf{x}$.
2. Update current cluster partition with $\mathbf{x}$ (using an incremental, evolving prototype-based clustering algorithm).
3. Store clusters affected by the update in a list $\hat{\mathrm{C}} = \{C_{win1}, C_{win2}, ..., C_{winK}\} \subseteq \mathrm{C}$; in some clustering algorithms the list may contain only one cluster (the winner one according to the highest potential, the largest distance, the highest compatibility index, etc.).
4. For each $C_{win} \in \hat{\mathrm{C}}$, do the following steps (5 and 6)
5. Merging operation:
   (a) Calculate *merge condition* for C_{win} and any other cluster in the partition.
   (b) Find the cluster C_{near} for which the merge condition is best fulfilled; if no such cluster exists, goto Step 6.
   (c) Merge the two clusters into $C_{merge} = Merge(C_{win}, C_{near})$; $\mathrm{C}_{upd} = \mathrm{C} \setminus \{C_{win}, C_{near}\} \cup C_{merge}$; $|\mathrm{C}_{upd}| = |\mathrm{C}| - 1$; goto 7.
6. Splitting operation:
   (a) Apply the quality criterion to the current cluster partition C_1: $clqual_1$.
   (b) Perform split of C_{win} into two parts: C_{win1}, C_{win2}.
   (c) Apply the quality criterion for the cluster partition including the split cluster C_2: $clqual_2$.
   (d) Obtain the cluster partition of the highest quality $\mathrm{C}_{upd} = \mathrm{C}_{k=argmax_{i=1,2}(clqual_i)}$; if $k = 1$, $|\mathrm{C}_{upd}| = |\mathrm{C}| + 1$.
7. **Output:** New cluster partition C_{upd} including new cluster centers and spreads of the clusters (if merging or splitting was performed) or the original C.

The algorithm is optimal in the sense that it operates on a sample per sample basis rather than performing block-wise split-and-merge and decreasing or increasing the number of clusters by only 1 for a complete new batch (with a high likelihood of being trapped in local optima according to a cluster validation criterion). Thus, after each sample-wise incremental learning step, a homogeneity criterion drives the decision whether the updated cluster should be merged with an existing one in order to achieve a more homogenous partition (see Sect. 3.2). Further checks using a quality criterion determine whether splitting this cluster will increase partition quality (see Sect. 3.3). Thus, it is checked whether the updated cluster fulfills the conditions (1) and (2), and if so, merge and split operations are performed, respectively, to resolve these situations. Splitting introduces an additional cluster evolution step (in addition to the cluster evolution steps of the incremental clustering method used), whereas merging constitutes a cluster pruning and complexity reduction step (reduction of unnecessary complexity, in fact).

For classification problems, Algorithm 1 can be applied in the same manner, but with the difference that only clusters with a majority vote for the same class can be merged. For instance, if 80% of the samples in one cluster belong to class #1 and 80% of the samples in another cluster belong to class #2, then merging of the two clusters is not recommended, because important parts of the decision boundaries between the overlapping classes may be lost. Merging would result in one larger cluster that represents both classes to a similar extent, increasing both the uncertainty of decisions and the degree of conflict for query points falling into or near this cluster (Hühn 2009; Lughofe 2011).

In the subsequent subsections, we explain the core components of Algorithm 1 in detail. Thereby, for all split and merge operations, we assume normalized data in order to achieve unbiased calculations and conditions when being applied to features with very different ranges.

3.2 Merging clusters

The merge condition between an updated cluster C_{win} and any other cluster C_{near} in the partition can be deduced from the consideration that the two clusters are sufficiently close to each other, so that their joint contour forms a homogeneous uni-modal structure. Whenever the two clusters are spherical, the condition whether two clusters overlap or touch is obviously given by

$$d(\mathbf{c}_{win}, \mathbf{c}_{near}) \leq r_{win} + r_{near} \tag{3}$$

with $d(x, y)$ denoting the Euclidean distance between x and y, and r_{win} denoting the radius of the sphere described by the updated cluster. Touching is the case when equality holds. In the case of ellipsoids in main position, the above condition can be extended to account for the different axis lengths along different dimensions, approximated by the spread σ of the clusters in a data-driven context:

$$d(c_{win}, c_{near}) \leq \frac{\sum_{k=1}^{p} |c_{win;k} - c_{near;k}|(fac \times \sigma_{win;k} + fac \times \sigma_{near;k})}{\sum_{k=1}^{p} |c_{win;k} - c_{near;k}|} + \epsilon. \tag{4}$$

Usually, the spreads along each dimension are estimated by the standard deviations in the corresponding dimension contained in the data samples, and can be updated through Eq. 19 and applying the square root function. This means that the condition on the right-hand side is not simply the sum of the spreads along all dimensions in both clusters, but a weighted average according to the degree of center shift along each direction. Consider two ellipsoids with identical center coordinates $x_1, \ldots, x_{p-1}$, then obviously the overlap/touching condition should be reduced to the sum of the axes along coordinate x_p (due to which the ellipsoids deviate). This is accomplished by Eq. 4, because all summands are 0 and therefore reduced to $(\sigma_{win;p} + \sigma_{near;p})$ due to the normalization term (denominator). If one of these clusters is displaced in some direction $i \in \{1, \ldots, p-1\}$, then accordingly the other summands contribute a bit to the final weighted average. The symbol ϵ denotes a near-zero constant (either positive or negative) and accounts for the degree of overlap *that is required* to allow merging (when $\epsilon < 0$), resp., the degree of deviation between the two clusters *that is allowed* to to trigger merging (when $\epsilon > 0$). If ϵ is set to 0, the two ellipsoids need only touch to fulfill the condition, and the clusters overlap in accordance with Eq. 1. Thus, we use 0 as default value. The factor *fac* denotes the spread of the cluster, which is used for the touching/overlapping criterion. The default setting is 2, giving a convenient representation of the contours of a Gaussian type-like data cloud which is a represented by a high-dimensional ellipsoid; and in case of a lower dimensionality (2, 3, 4), a 2σ range covers approximately 90–95% of the data.

For $\sigma_{win;1} = \sigma_{win;2} = \cdots = \sigma_{win;p} = r_{win}/2$, we then obtain the formula in Eq. 3.

Merging the clusters does not only require two ellipsoids to overlap, touch, or be in close proximity, but also that the two clusters form a joint homogeneous region. Examples of merged clusters with different homogeneity levels are shown in Fig. 3.

Basically, we can say, that the larger the increase of the merged cluster compared to the sum of the single clusters, the lower the homogeneity which has to be handled with care. In fact, although they have the same degree of overlap (i.e., they touch each other), it is desirable to merge the

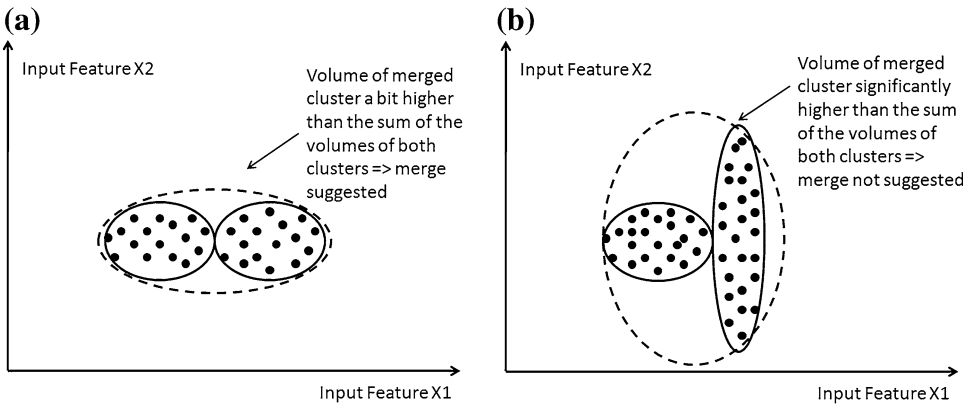

Fig. 3 **a** Two clusters (*solid ellipsoids*) which touch each other and are homogeneous in the sense that the (volume, orientation of the) merged cluster (*dashed ellipsoid*) conforms with the original two clusters → merge recommended, **b** two clusters (*solid ellipsoids*) which touch each other and are not homogeneous → merge not recommended

clusters shown in Fig. 3a, but not those shown in Fig. 3b. The reason for this is that in (a) the clusters have the same tendency and orientation, whereas in (b) the two clusters represent data clouds with different orientation and shape characteristics. We thus define an additional condition for cluster merging:

$$V_{merged} \leq p(V_{win} + V_{near}) \tag{5}$$

with p denoting the dimensionality of the feature space and V the volume of an ellipsoid in main position, which is defined by Jimenez and Landgrebe (1998):

$$V = \frac{2 \times \prod_{j=1}^{p} \sigma_j \times \pi^{p/2}}{p \times \Gamma(p/2)} \tag{6}$$

with Γ denoting the gamma function:

$$Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt. \tag{7}$$

Multiplication by p is introduced because of the curse of dimensionality effect in volume calculation (the volume increases exponentially with the number of features). Criterion (5) can be rewritten as

$$\frac{p(V_{win} + V_{near})}{V_{merged}} \geq 1 \tag{8}$$

which means that $hom(C_i, C_j) = \frac{p(V_{win}+V_{near})}{V_{merged}}$ and $thresh = 1$, according to the definition in Eq. 1.

We expect the cluster center of the merged cluster to be somewhere in-between the two original clusters, but not necessarily exactly in the middle, as the center should be closer to the more significant cluster. Therefore, it is reasonable to merge two clusters centers by taking their weighted average, the weight for each cluster being defined by its significance, i.e. the number of samples (k_{win} and k_{near}, respectively) that form it and on the basis of which the clusters were updated significantly. Merging the ranges of influence (spreads) of the two clusters is based on the idea that the less significant cluster can be represented by a compressed information about a mass of data, which is used to update the more significant cluster. Thus, the spreads of the two clusters can be merged by using recursive variance formula, including rank-one modification for better stability and convergence (Qin et al. 2000); also see Eq. 19. There, the last term includes the distance of the current sample $\mathbf{x}$ from the center of the cluster to be updated; in our case, the current sample is replaced with the center of the less significant cluster (as standing for the most informative representative of the data cloud the cluster models). In order to guarantee good coverage of the original data cloud by the merged cluster, a number of samples from the less significant cluster equal to a fraction of the variance is added; the fraction is determined by the ratio of samples belonging to the less significant cluster to the samples belonging to both clusters. Hence, two clusters, *win* and *near*, are merged in the following way ($\forall j = 1, \ldots, p$ separately, and obtaining a new cluster indicated by the index *new*):

$$\begin{aligned} c_{new,j} &= \frac{c_{win,j}k_{win} + c_{near,j}k_{near}}{k_{win} + k_{near}} \\ \sigma_{new,j} &= \sqrt{\frac{k_{cl_1}\sigma_{cl_1,j}}{k_{cl_1} + k_{cl_2}} + (c_{cl_1,j} - c_{new,j})^2 + \frac{(c_{new,j} - c_{cl_2,j})^2}{k_{cl_1} + k_{cl_2}}} \\ &\quad + \frac{k_{cl_2}}{k_{cl_1} + k_{cl_2}}\sigma_{cl_2,j} \\ k_{new} &= k_{win} + k_{near} \end{aligned} \tag{9}$$

where $cl_1 = argmax(k_{win},k_{near})$ denotes the (index of the) more significant cluster, and $cl_2 = argmin(k_{win},k_{near})$ denotes the (index of the) less significant cluster.

Figure 4 shows cluster merging examples without (a) and with integration (b) of the last term in Eq. 9 for merging the spreads. Integration of the term yields better coverage of the data distribution over both overlapping clusters.

Figure 5 illustrates the effect of merging nearby lying clusters generated from a sequence of data blocks, each containing 1,000 samples: upper and lower rows show the update progress of the incremental clustering procedure

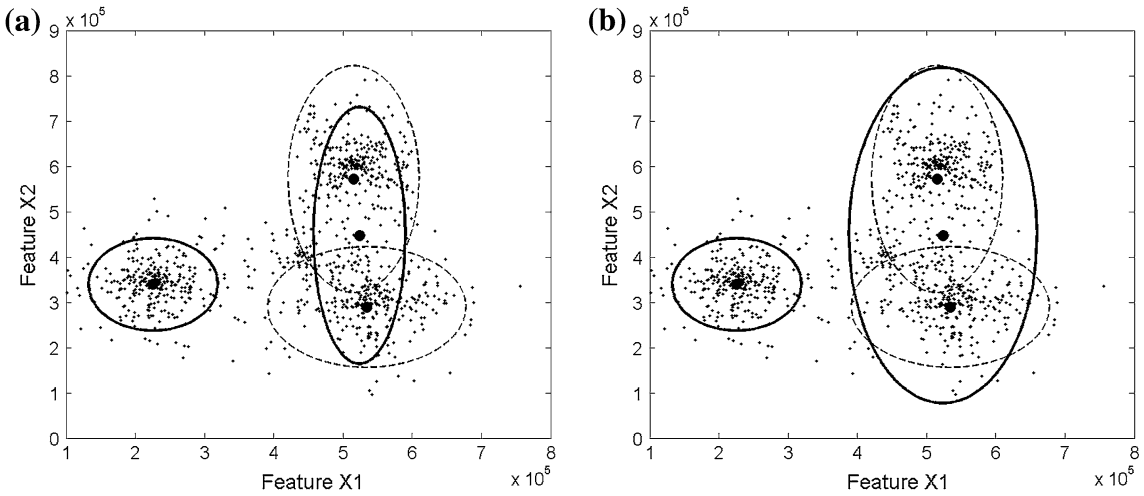

Fig. 4 **a** Merged cluster visualized as a solid ellipsoid without integration of the 'coverage' term (last term in Eq. 9), **b** integration of the coverage term achieves better coverage of the joint data distribution of the two original clusters; the ranges of influence of the two clusters are indicated by a 2σ area

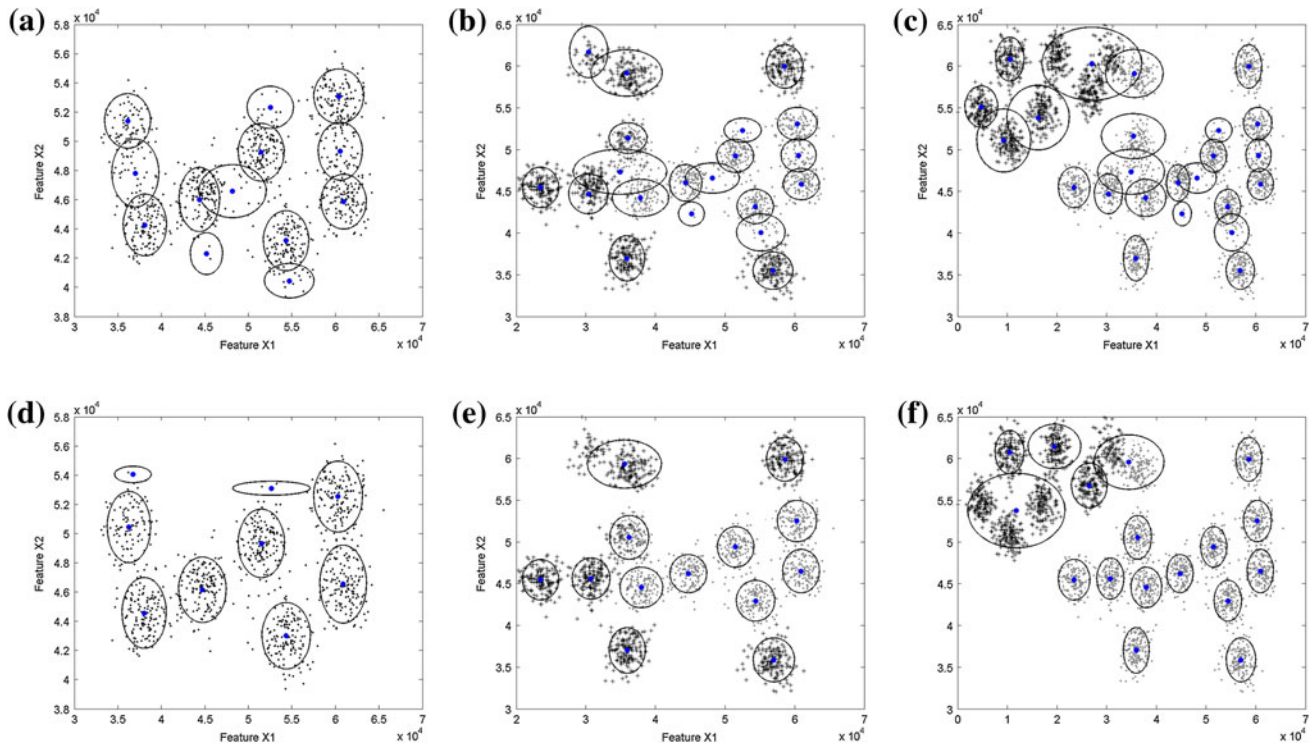

Fig. 5 *Upper row* Incremental cluster learning and evolution progress using two-dimensional streaming clustering data (loaded in blocks of 1,000 samples) without integrated merging. Note the over-clustering in many cluster partitions; *lower row* the same example with the merging concept incorporated (criteria + merging formulas); *grey dots* older data samples, *dark crosses* the samples contained in the current data block

without and with integrated merging, respectively. Clearly, superfluous clusters are merged back: for instance, the two smaller clusters extracted from the first 1,000 samples are erased by merging with nearby clusters which become touching to this two smaller ones over the next batch of 1,000 samples [compare (d) with (e)], and the resulting cluster partition appears more feasible [clusters are representing the natural data clouds more perfectly compared to (b)].

Interesting to see in (f) that two cluster pairs in the upper left corner that touch or overlap are (correctly) not merged, due to violation of the homogeneity condition: the volume would increase too much if these were merged, because the orientations of the local distributions are too different,

which also means that, intuitively, the clusters should not be merged.

In classification problems using a clustering-based technique and labeled data, two clusters fulfilling above conditions are merged only when they have the same majority class in both clusters (which can be updated over time by counting). Otherwise, some important parts of the decision boundaries between classes might be erased: consider two overlapping clusters where one cluster contains only samples from class '0' and the other only samples from class '1'. Despite the possible overlap of the two classes, a large proportion of the samples could still be well discriminated; if the clusters were merged, this discrimination would be lost and the classification rate on new samples would decrease in this part of the feature space.

3.3 Splitting clusters

In order to determine whether a cluster contains two disjoint data clouds (due to a delamination effect, cf. Fig. 2), samples, which characterize the inner distribution of a cluster, are required. As we are aim for an algorithm, both with minimal memory usage and computational complexity, we use sub-samples of clusters, obtained by the reservoir sampling technique (Vitte 1985). The whole cluster splitting strategy is performed using a trial-and-error procedure [refer to Algorithm 1, Steps 6 (a)–(d)], where an updated cluster is split hypothetically performed and the quality of the cluster partition (containing $C + 1$ clusters) assessed.

3.3.1 Cluster split technique

An obvious way to split a cluster would be to halve it, i.e. to halve the spreads in the original cluster along each dimension (as, for instance done in the LOLIMOT algorithm for splitting fuzzy rules; Nelle 2001). However, this does not necessarily represent the real data distribution in the two parts and would always result in the same type of split. It is thus necessary to track the inner structure of each cluster over time in an incremental manner. We do this with a sub-sampling approach as part of Step 6b in Algorithm 1, in which we keep each other sample belonging to a cluster (for which the cluster was the nearest one) and never more than N_{sub} samples in total. Let therefore S_{win} denote a representative sub-sample of the updated cluster *win* with $|S_{win}| \leq N_{sub}$ (in all our experiments we used $N_{sub} = 300$). Then, we divide this further into two sets according to the inner structure of the winning cluster. To this end, we apply a 2-means clustering algorithm (Jain and Dubes 1988) to S_{win} and split the samples into two subsets according to

$$\begin{aligned} S_{1;win} &= \{\mathbf{x} | \mathbf{x} \in S_{win} \wedge \|\mathbf{x} - \mathbf{c}_{win1}\| \leq \|\mathbf{x} - \mathbf{c}_{win2}\|\} \\ S_{2;win} &= \{\mathbf{x} | \mathbf{x} \in S_{win} \wedge \|\mathbf{x} - \mathbf{c}_{win1}\| > \|\mathbf{x} - \mathbf{c}_{win2}\|\} \end{aligned} \tag{10}$$

where $\mathbf{c}_{win1}$ and $\mathbf{c}_{win2}$ correspond to the two cluster centers obtained by 2-means clustering. Then, we calculate the spreads $\sigma_{win1}, \sigma_{win2}$ based on the samples falling into the sets in Eq. 10 and use these to evaluate the quality criterion (see subsequent paragraph).

3.3.2 Cluster quality criterion

The counterpart of the merging process would be to invert the geometric criteria in Eqs. 4 and 5:

$$\begin{aligned} & d(\mathbf{c}_{win1}, \mathbf{c}_{win2}) \\ & > \frac{\sum_{k=1}^{p} |c_{win1;k} - c_{win2;k}| (fac \times \sigma_{win1;k} + fac \times \sigma_{win2;k})}{\sum_{k=1}^{p} |c_{win1;k} - c_{win2;k}|} + \epsilon \end{aligned} \tag{11}$$

and

$$V_{merged} > p(V_{win1} + V_{win2}) \tag{12}$$

and use a significantly higher ϵ in Eq. 11 than in Eq. 4 to avoid alternating merge and split operations in consecutive incremental learning steps. However, since sub-samples and a 2-means clustering are used, the contours of the resulting clusters are usually not the exact contours of the real data clouds. Furthermore, a split may also be useful whenever two clusters overlap partially: for instance, consider two overlapping components in a Gaussian mixture model, which is often the data model used of choice in prototype-based clustering algorithms (Sun et al. 2004). Therefore, we replace the strict geometric criteria with a quality comparison of split and un-split cluster models based on the BIC (Schwar 1978), which relies on a log-likelihood estimation, penalizes more complex models, and—most importantly—is easily applicable in an incremental learning context, because it does not rely on criteria that use all the (past) training samples (such as the famous Xie and Beni (1991) validation index or its enhanced version presented in Beringer and Hüllermeier (2006). The BIC in its original form is defined by:

$$BIC(C) = -2 \times loglik + k \log(p) \tag{13}$$

with *loglik* denoting the log-likelihood of the data according to the model C (the model with lower value BIC value is to be preferred) and k the complexity of the model, measured in terms of the number of independent parameters. Our cluster models deal with ellipsoidal clusters, which can be associated with components of a GMM (Hastie et al. 2009). The support of each cluster is given by the number of samples that form it. Thus, we can model the likelihood that a data sample $\mathbf{x}$ belongs to a

certain cluster C_j as the product of the likelihood of observing C_j (defined by its relative support among all samples seen so far) with the multivariate (Gaussian) density function (=component of a Gaussian mixture model):

$$P_j(\mathbf{x}) = \frac{n_j}{N} \frac{1}{\sqrt{2\pi \times det(\Sigma_j)}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mathbf{c}_j)^T \Sigma_j^{-1}(\mathbf{x}-\mathbf{c}_j)\right) \tag{14}$$

with n_j denoting the number of samples forming cluster C_j, N the total number of data samples seen so far, and Σ_j the covariance matrix spanned by the cluster. For axis parallel ellipsoids, the covariance matrix is a p-dimensional diagonal matrix, in which each entry represents the variance in each dimension. We yield:

$$\Sigma_j = diag(\boldsymbol{\sigma}_j^2) = diag\left(\frac{1}{n_j - 1}\sum_{i=1}^{n_j}(\mathbf{x}-\mathbf{c}_j)^2\right). \tag{15}$$

Thus, the maximum likelihood of the data samples in cluster $\mathbf{c}_j$ can be calculated as:

$$\begin{aligned} loglik_j &= \log \prod_{x_i \in c_j} P_j(\mathbf{x}_i) \\ &= \sum_{x_i \in c_j} \left(\log\left(\frac{n_j}{N}\right) + \log\left(\frac{1}{\sqrt{2\pi \times \prod_{k=1}^{p} \sigma_{j;p}^2}}\right) \right. \\ &\quad \left. -\frac{1}{2}(\mathbf{x}_i - \mathbf{c}_j)^T diag(1/\boldsymbol{\sigma}_j^2)(\mathbf{x}_i - \mathbf{c}_j) \right) \\ &= n_j \log(n_j) - n_j \log(N) - \frac{n_j}{2}\log(2\pi) \\ &\quad - \frac{n_j}{2}\sum_{k=1}^{p}\log(\sigma_{j;p}^2) - \frac{n_j - 1}{2} \end{aligned} \tag{16}$$

The overall BIC representing the current cluster partition containing K clusters is obtained by:

$$\begin{aligned} BIC(C) &= -2 \times loglik + k \log(p) \\ &= -2 \times \sum_{j=1}^{K} loglik_j + d \log(N) \end{aligned} \tag{17}$$

with d denoting the number of parameters to be learned. In our case, this is equal to the number of clusters K times 2, as each cluster contains two degrees of freedom: center and spread. At first glance, it seems that the BIC increases with the number of clusters (more negative summands multiplied by -2 and a higher k), and thus always favors a lower number of clusters. On the other hand, the spreads σ of the clusters also decrease when more clusters are extracted, which increases the fourth term in Eq. 16 and decreases the first term in Eq. 17.

In order to ensure that the BIC is able to distinguish between two truly disjoint clusters and two clusters which are very close to each other or which even overlap within one cluster, we integrate a penalty term, which penalizes overlapping clusters more than clearly disjoint ones. The extended BIC for cluster model j with splits $j1$ and $j2$ thus becomes:

$$\begin{aligned} BIC_{ext}(C_{j1}, C_{j2}) &= -2(loglik_{j1} + loglik_{j2}) + d(\log(N_{j1}) \\ &\quad + \log(N_{j2})) + \frac{N_j}{2}\sum_{i=1}^{N_j}\log\left(\prod_{k=1}^{2}\mu_{ij_k}\right) \end{aligned} \tag{18}$$

with d equal to 4 (two parameters for each cluster in the split), N_j denoting the number of data samples belonging to cluster j, and μ_{ij_k} the membership degree of sample i (from the subset) in cluster j_k (note that we store a subset of samples). Obviously, the more disjoint the clusters, the closer to 0 the product and therefore the lower the value of the logarithm, penalizing more disjoint clusters less (note that the lowest BIC value corresponds to the best model). Figure 6 shows two cluster splits according and the inner distribution of the original cluster. Both splits result in clusters of the same size and would therefore lead to the same value of the original BIC according to Eq. 17. The BIC defined in Eq. 18 yields a much lower value for the more disjoint option.

In an incremental learning context, the BIC criterion is updated simply by counting the number of data samples n_j belonging to each cluster, counting the total number of data samples N, and updating the variance in each direction (used in the fourth term). The latter is also necessary for a fully updated representation of the ECMs (centers plus spread) and can be achieved by using the recursive variance formula, in the most stable form including rank-one modification (for better convergence to batch variance) according to Qin et al. (2000):

$$\begin{aligned} &(n_i + 1)\sigma_{i,j}^2 \leftarrow n_i\sigma_{i,j}^2 + (n_i + 1)\Delta c_{i,j}^2 + (c_{i,j} - x_j)^2 \\ &\forall j = 1, \ldots, p \end{aligned} \tag{19}$$

with Δc denoting the difference between the updated and the original position of the cluster center. In each incremental learning cycle, only those parts of the *loglik* function in the BIC criterion (Eq. 17) need to be updated which belong to the updated clusters, as the components of the other clusters remained unchanged. If L new clusters were evolved, the sum in (Eq. 17) is extended by the terms $loglik_{K+1}, \ldots, loglik_{K+L}$ and d incremented by 2^*L

Figure 7 a cluster splitting example using a two-dimensional streaming data set (called S2) and the eVQ incremental clustering technique without (Fig. 7a, b) and with integration of the dynamic splitting method described in this section (Fig. 7c, d). Both approaches perform equally well and at learning from the initial 1,000 samples provide the same cluster partition (Fig. 7a, c), hence (correctly) none of the compact clusters are split by our

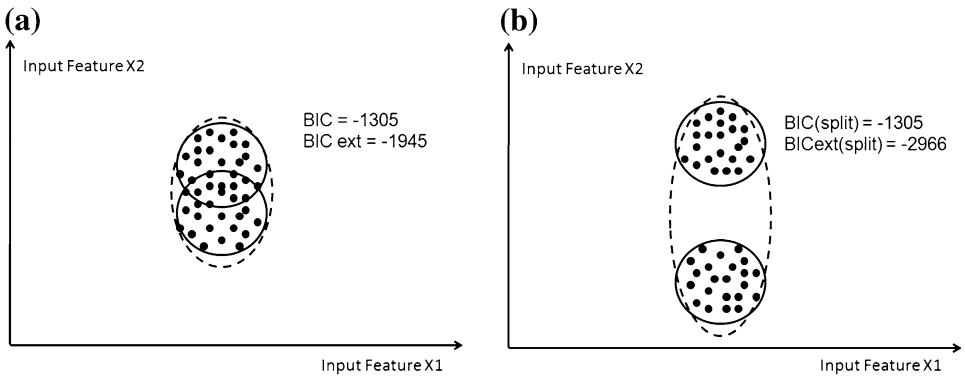

Fig. 6 **a** Splitting along the second feature leads to two extensively overlapping clusters, **b** splitting results in clearly disjoint clusters, but the conventional BIC is the same as in **a**; the extended BIC yields a significantly lower value due to the last term in Eq. 18, indicating a cluster partition of high quality

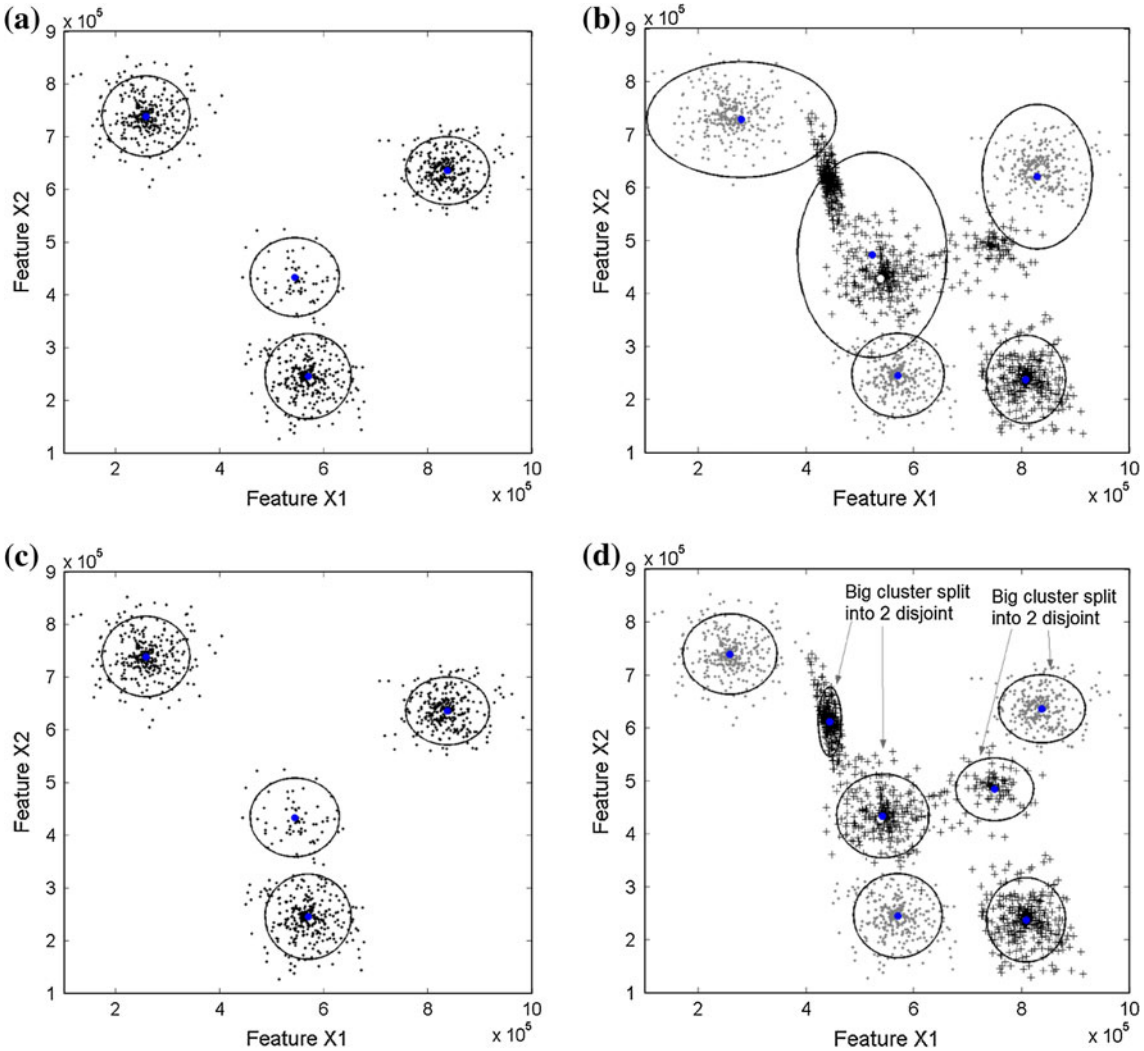

Fig. 7 *Upper row* The learning (*left*) and update (*right*) when not integrating any automatic splitting technique (see the enlarged cluster covering three data clouds), *lower row* the better performance as the two bigger clusters are split, finally achieving seven clusters which are more appropriate according to the data cloud positions in the actual data stream snapshot

technique. Updating the cluster partitions with another 1,000 samples, which appear within the original feature range (marked as crosses) leads to excessively large clusters in the first case (the middle and upper right clusters in Fig. 7b are significantly enlarged). The dynamic splitting technique, however, yields a better partition by splitting these two clusters dynamically into two. Note that reason for the cluster enlargement in the first case is that the

distance between existing clusters and new samples is not sufficiently large for new clusters to evolve. Enlargement of the upper left cluster is prevented by the integrated technique, since early in processing the large cluster in the middle is split; thus no sample from the small dense cloud influences this cluster.

For classification problems using labeled data, a split suggested by the quality criterion is executed only if the majority class in both split clusters is different.

4 Comments on computational costs

In this section, we examine the additional costs caused by the split-and-merge operations for one incremental learning step, assuming that one cluster is updated in each step.

4.1 Merging

First, Eq. 4 is used to determine whether any of the existing clusters touch the updated cluster. Assuming that C clusters are available, this requires $(C-1)(5p+3)$ floating point operations for the right hand side: a sum over the dimensionality of p, where for each summand 5 floating point operations are conducted, plus additional 3 operations for the division, the addition of ϵ, and the comparison; and $(C-1)(2p+1)$ operations for the distance calculation on the left hand side, resulting in a total quadratic complexity of $O(Cp)$. The additional homogeneity criterion is only evaluated for clusters fulfilling Eq. 4. In sample-wise incremental learning, where the cluster shifts are small and in one specific direction, we can assume that only one other clusters fulfills (4), thus yielding three volume calculations: for the updated cluster, for the nearest cluster and for the hypothetically merged cluster. Taking into account that the gamma function interpolates the factorial function (Andrews et al. 2001), thus the complex calculation of the denominator in Eq. 6 requires $\frac{p}{2}+1$ operations; hence, the volume calculation has a complexity of $O(p)$. Merging two clusters fulfilling both criteria requires 18p floating point operations. The complete merging step has thus a complexity of $O(Cp+p) \approx O(Cp)$.

4.2 For splitting

The k-means clustering algorithm has a complexity of $O(CnpI)$ (Jain and Dubes 1988), so it is linear in all of its factors: the number of clusters C, the number of samples n, the dimensionality of the learning problem p, and the number of iterations I. In our case, n denotes the buffer-size for the sub-sample, which is small, usually $n \ll N$ with N the number of samples seen so far. The number of iterations is usually low, as k-means usually converges quickly (Gan et al. 2007). The index list that describes which sample belongs to which cluster according to its distance from the center is an additional result of k-means, adding complexity to the calculation of $S_{1;win}$ and $S_{2;win}$. Calculating the new means for the two split cluster requires $O(n)$, calculating the new spreads along all dimensions requires $O(np)$ floating point operations. To evaluate the BIC, only $loglik_{win}$ for the updated cluster must be re-calculated in each incremental learning step (assuming that the log-likelihood of the other cluster has already been calculated). This requires $p+14$ operations according to the last line in Eq. 16; the BIC of the (hypothetically) split clusters must also be calculated, increasing the complexity to $3p+42$. The penalty criterion for overlaps in Eq. 18 requires $3n+5$ operations. Therefore, the overall complexity of calculating the penalized BIC for split and unsplit clusters is $O(p+n)$. Summing all contributions, we yield a complexity of $O(Cnp+np+(n+p)) \approx O(Cnp)$, i.e. still linear in all of its factors. The most crucial factor is n, the size of the sub-sample buffer used, as usually $C \ll n$ and $p \ll n$.

5 Experimental setup

To evaluate the proposed split-and-merge operations, we used the evolving vector quantization (eVQ) approach as learning engine. Its basic algorithmic steps can be summarized as follows (refer to Lughofer 2008 for the details):

- Load a new sample, normalize it to [0, 1], and determine whether it fits into the current cluster partition—the decision is driven by vigilance, the only sensitive learning parameter: this is pre-parameterized to a default value of $0.3 \times \frac{\sqrt{dim}}{\sqrt{2}}$ with dim denoting the dimensionality of the feature space, to avoid a tuning phase, which is usually not possible in on-line learning from data streams.
- If the sample fits into the current cluster, update the center by moving it a small distance towards the current data sample; the distance is determined by a decreasing learning gain along the support of the cluster (ensuring convergence); update the spread of the cluster in each dimension using a stable recursive variance approach including rank-one modification (Qin et al. 2000).
- If the sample does not fit into the current cluster, create a new cluster by setting its center to the coordinates of the current sample and its spread to 0.

The default parameter setting may lead to over- or under-clustered situations (depending on the characteristics of the data), which, with the exception of natural cluster

fusion and delamination problems, can be resolved with the dynamic split-and-merge approach apart from the natural cluster fusion and delamination problems. As alternative clustering engine for the classification streaming data sets we applied the EC (evolving clustering method) as integrated in DENFIS (Kasabov and Song 2002). This provides us an impression how much dependent our split-and-merge algorithm is on the underlying clustering engine.

We used the following three streaming data sets to evaluate our approach:

- Two data sets from the Internet[3]—S1 and S2: these are two-dimensional streaming data sets for clustering, containing a few thousand samples and Gaussian clusters with different degrees of cluster overlaps (in S2 the clusters overlap more than in S1).
- Synthetic clustering data with clusters in multi-dimensional space: from 2-dimensional to 15-dimensional, we used all those with an even number of dimensions.
- Cover-type data set (UCI machine learning repository): contains the forest cover type from cartographic variables only (no remotely sensed data), including 7 classes: Lodge-pole Pine, Ponderosa Pine, Cottonwood/Willow, Aspen, Douglas-fir, Krummholz and over 580000 samples. The forest cover type for a given observation (30 × 30 meter cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data.
- Animals data set: a set of synthetic data with 72 dimensions and 500,000 samples; produced by the "animals.c" program obtained from the UCI ML Repository, containing 4 classes of animals. We removed all constant features (all values equal to 0 or equal to 1), yielding 46 dimensions.

The data sets and their characteristics are summarized in Table 1.

The evaluation scheme is based on an incremental learning procedure, which loads a single data sample and sends it to the update algorithms for the cluster partitions. In order to see the effect of merge and split-and-merge on conventional incremental clustering (eVQ), three variants of incremental are compared, denoted as ECM, ECM with merge and ECM-SAM (ECM with split-and-merge). In case of ECM and ECM with merge single-pass incremental learning is achieved, where after the update of the cluster partition each single sample is immediately discarded (so no prior data is used). In case of ECM-SAM, a (small) reservoir of samples is required for representing the inner structure of a cluster and updated in incremental manner. The cluster partitions of conventional ECM and ECM-SAM were visualized for the two-dimensional clustering data sets and compared with the true number of clusters in the high-dimensional streaming clustering sets. For the latter, a comparison using the well-known Xie and Beni (1991) index is conducted, which measures the quality of cluster partitions. For the classification data (cover-type and animals), we used the measures *entropy* and *purity* of the final cluster partitions of the whole data set. In addition, we compared our method with the two batch clustering methods CLUTOR and GARDEN. These show how 'clean' the clusters are, i.e. to what degree the classes are mixed up in each cluster. Ideally, clusters are clean, i.e., they contain only one class; this ensures a minimal 'conflict' when classifying new samples with the extracted clusters.

Table 1 Characteristics of the clustering data sets used

	No. of samples	No. of feat.	No. of Cl.
S1	2,000	2	6
S2	4,000	2	13
Synth 2-dim	1,350	2	9
Synth 4-dim	2,701	4	9
Synth 6-dim	4,051	6	9
Synth 8-dim	5,401	8	9
Synth 10-dim	6,750	10	9
Synth 12-dim	8,100	12	9
Synth 14-dim	9,450	14	9
Cov-type	581,012	54	7
Animals	500K	46	4

No. of Cl. denotes the number of clusters for unsupervised data and the number of classes for classification data (cov-type and animals)

6 Results

6.1 Two-dimensional streaming clustering data sets

For the first 2,000 $(1,000 + 1,000)$ samples of the S2 streaming data sets, the split-and-merge-technique exhibited similar performance (yielded the same cluster partitioning) as the split option alone (as shown in Fig. 7c, d).

Further updating of the cluster partition with streaming samples leads to the results shown in Fig. 8, (again upper row corresponding to conventional incremental evolving clustering, the lower row includes the new split-and-merge techniques). Figure 8b, d (the final results over all batches of the streaming data set) show that dynamic split-and-merge operations improve the performance remarkably: the final cluster partition obtained by our method reflects the real data distribution (represented by local data clouds) much better than that obtained by conventional clustering, where the default setting of the vigilance parameter (which

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

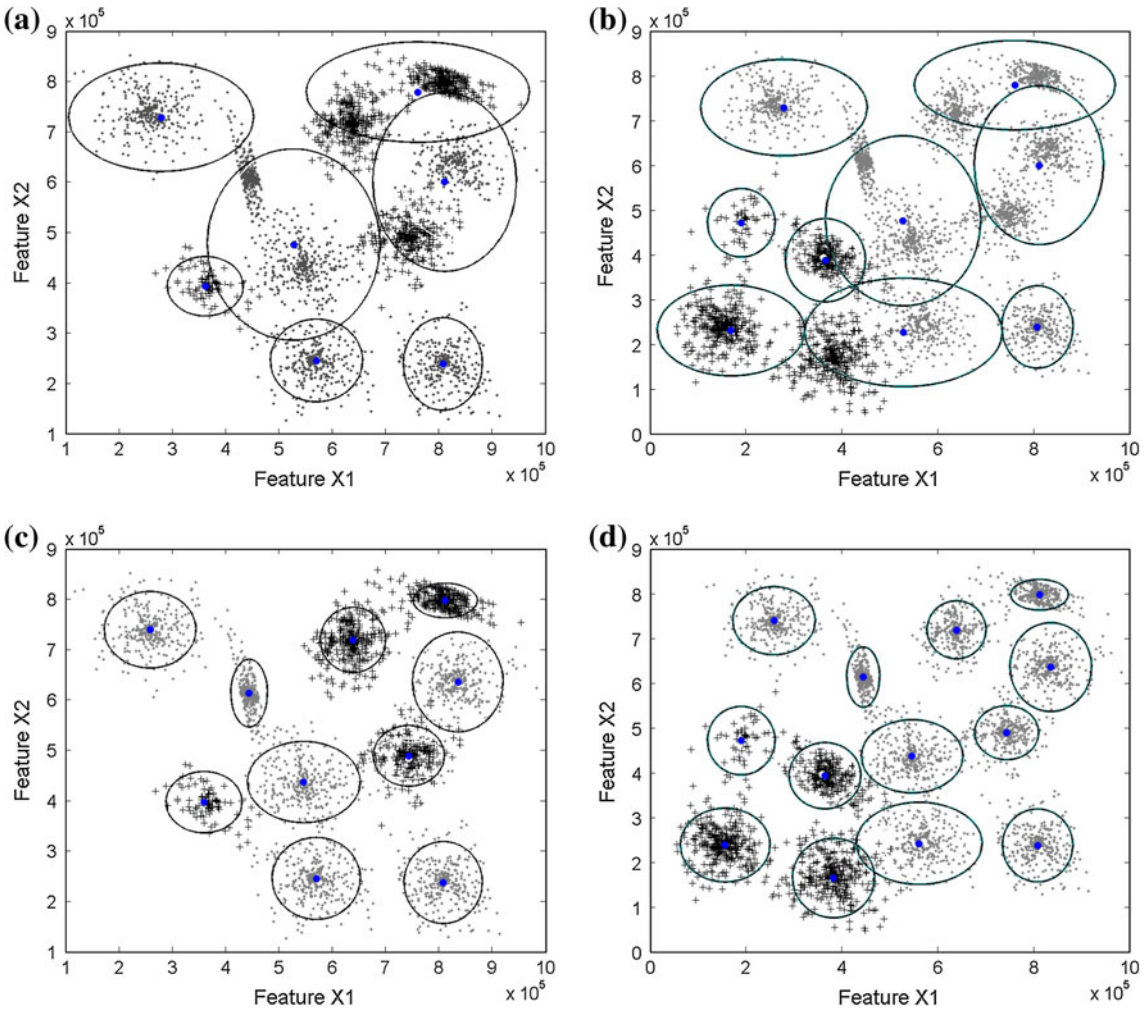

Fig. 8 *Upper row* Conventional incremental clustering updated with 1,000 (*left*) plus another 1,000 (*right*) samples without integration of an automatic splitting technique; the final cluster partition does not reflect the natural distribution of the data cloud (too few and too large clusters); the *lower row* illustrates the superior performance of the dynamic split-and-merge concept, which automatically splits excessively large clusters and merges touching and over-lapping ones on demand; in both cases, the learning parameter is set to a fixed value

performed well for the initial data block) leads to too few, large clusters due to compression of the data cloud spreads over time. Thus, dynamic split operations also compensate for inappropriately chosen fixed learning parameters, improving automatization in an on-line data streaming context, where manual parameter tuning is hardly possible.

In order to gain a more detailed insight into the dynamic split technique, Fig. 9c shows the values of the quality information criterion over time for the cluster partition shown in (a) (using the S1 data set) with the number of clusters marked as such—note that no split is actually performed in this period, just the criterion indicated. Whenever the solid line (=normalized cluster partition quality criterion with one cluster split in two) is below the dashed line (=normalized cluster partition quality criterion without any split), a split is suggested. According to Fig. 9b showing which clusters are updated when [*x*-axis represents the data sample number, y-axis the cluster number according to (a)], this is the case for clusters #6, #4, #5 and also #1, the last one being updated by just three samples At the beginning of the data stream (samples 1–200), cluster #6 is updated, which appears as compact joint data cloud and for which no split is correctly suggested (dashed line clearly below the solid line). When the update switches to Cluster #4 (samples 200–approx. 600), our indicator suggests a split, but not uniquely (solid line very slightly below the dashed line): because this cluster also contains a few samples from the lower region of the nearby long diagonal cluster. The large cluster #5, which contains two clearly distinct clouds, is updated with sample 600–1,000 and a split is correctly suggested (solid line significantly below the dashed one). The only exception is when the update switches to the compact cluster #1, for which correctly no split is suggested in three cases.

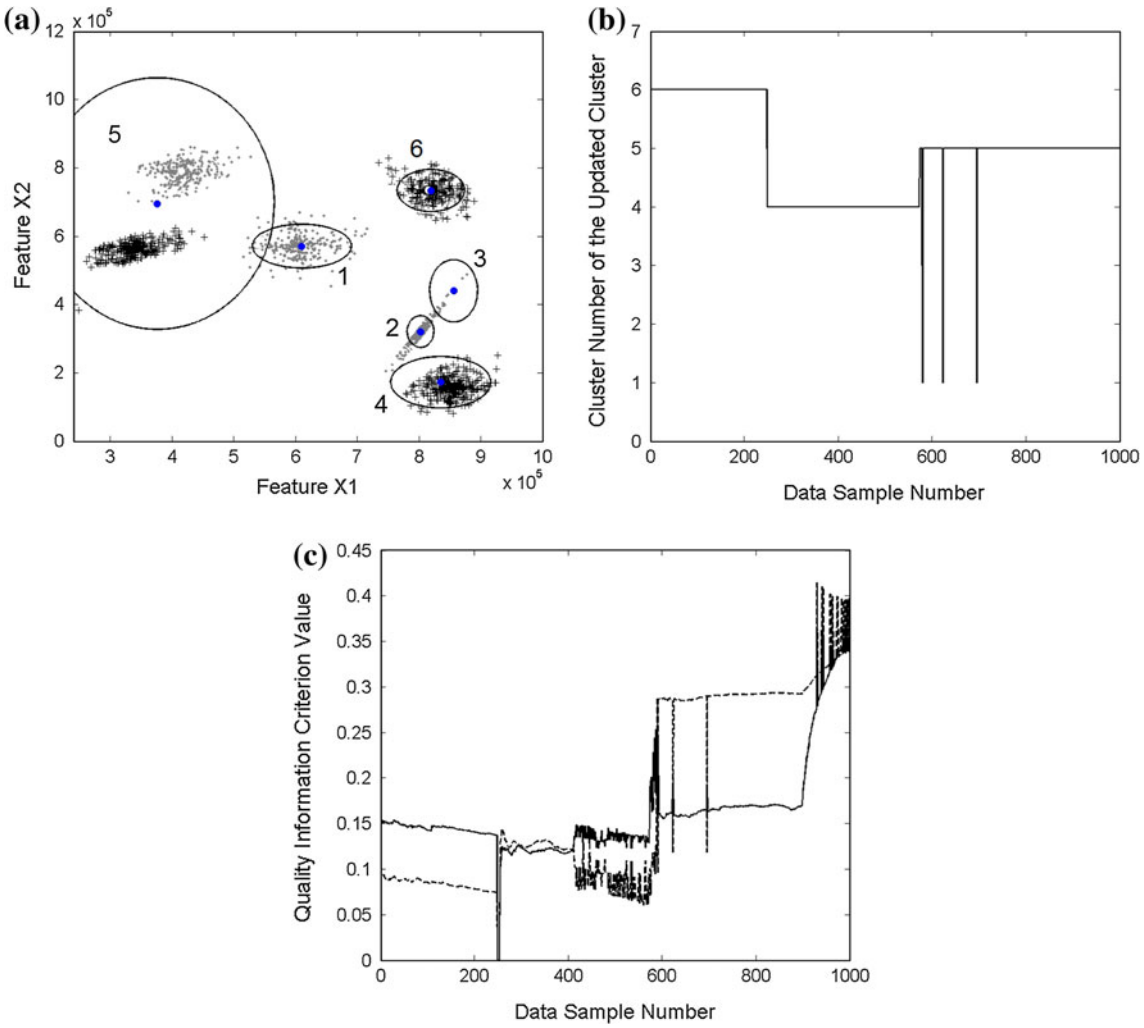

Fig. 9 *Upper left* Final cluster partition when no split is conducted, the numbers of the clusters indicated; *upper right* carries the information when which cluster according to the cluster number is updated (*x*-axis, sample number; *y*-axis, cluster number); *lower* cluster partition quality criteria over the number of samples. *Solid lines* the criterion for split clusters, *dotted lines* one without any splits (the lower criterion denotes the better quality); note that for each sample only a split of the updated cluster is considered

6.2 Gaussian streaming data sets (2-dim to 14-dim)

Table 2 shows the resulting numbers of clusters for seven data sets, ranging from 2-dimensional (low) to 8-dimensional (medium) and 14-dimensional (high) feature spaces; in all data sets, the true number of clusters is 9. Clearly, ECM-SAM comes closer to the true numbers of clusters in the various data sets than conventional ECM, and with increasing dimensionality the difference becomes greater. A fundamental reason why conventional ECM performs so weak for six-dimensional and ten-dimensional data sets is that in these sets the first feature increases its range continuously. Thus, at the beginning of the data stream, its range is small, producing a zoom effect on the data set, in which the samples seem to be more spread out than they truly are, making the likelihood of over-clustering high. After a while, when the range is extended, the real data clouds turn out to be more compact, and the originally extracted clusters are compressed. The over-clustered situation (several clusters for one data cloud) can be resolved (in large parts) by the merge operation in ECM-SAM, as can be seen in Table 2. The right part of the table shows the quality values obtained by the Xie and Beni (1991) index, which measures the ratio between separation quality and compactness of the obtained cluster partitions:

$$cl\_qual = \frac{\sum_{i=1}^{C}\sum_{j=1}^{N}\mu_{ij}^2\|\mathbf{c}_i - \mathbf{x}_i\|}{N\min_{i,j=1,\ldots,C,i\neq j}(\|\mathbf{c}_i - \mathbf{c}_j\|)} \tag{20}$$

with C denoting the 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 the value obtained for Eq. 20, the

Table 2 Number of clusters identified by the evolving cluster model (ECM) variants: without split-and-merge operations, with merge operation only and using both, split-and-merge (SAM); in all cases, the true number of clusters contained in the data stream is 9, and the learning parameter was set to the default value; the right part of the table shows the quality measures obtained by the Xie–Beni index (the lower the better)

	ECM	ECM with merge	ECM-SAM	ECM	ECM with merge	ECM-SAM
2-dim	11	8	10	5.343	0.205	0.439
4-dim	7	7	9	0.268	0.268	0.175
6-dim	22	12	12	1.347	0.474	0.421
8-dim	11	11	10	1.089	0.734	0.502
10-dim	36	9	9	1.334	0.068	0.068
12-dim	9	9	9	0.114	0.074	0.074
14-dim	12	9	9	0.254	0.071	0.071
Av. dev./ qual.	7.0	1.1	0.7	1.398	0.271	0.250

better the cluster partition represents compact and separable clusters. From Table 2, it can be seen that ECM-SAM performs better than ECM with merge (average deviation from the true number of clusters in the complete stream is lower), and also achieves slightly higher average quality in terms of the Xie–Beni validation index (Table 2).

Figure 10 compares the computation times required for clustering tasks with various dimensionalities. Note, that the number of data samples differs across tasks as indicated in the figure. While ECM with merge require computation times comparable to those of conventional ECM, ECM-SAM requires significant additional time. However, its speed is still in the range of milliseconds for updating single samples

6.3 High-dimensional classification data (purity, entropy)

Table 3 shows the results obtained for the classification data sets *animals* and *cov-type*. Both contain a large number of samples and are therefore ideal test data sets for data-stream clustering methods. In order to judge how well the clusters follow the characteristics of the class distribution in feature space, we used two well-known measures from the literature (also used in Zhao and Karypis 2004; Orlandic et al. 2005 for the same purpose): entropy and purity. While entropy measures how the various sample classes are distributed within each cluster, purity measures the extent to which each cluster contains samples from primarily one class. Thus, the entropy of a cluster i is defined by Zhao and Karypis (2004):

$$E(C_i) = -\frac{1}{log(K)} \sum_{k=1}^{K} \frac{n_{ik}}{n_i} log\left(\frac{n_{ik}}{n_i}\right) \quad (21)$$

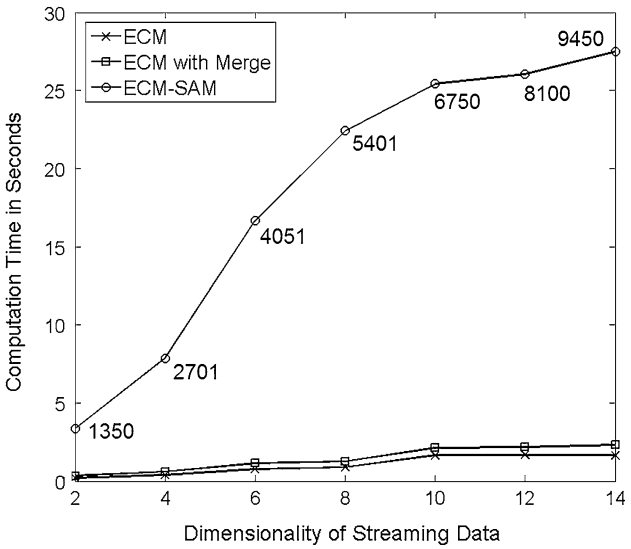

Fig. 10 Computation times for clustering the whole data stream versus dimensionality of the streaming data in the clustering task, the number of data samples for each dimensionality are indicated

Table 3 Unsupervised clustering of high-dimensional streaming classification data, using evolving cluster model (ECM) variants: without split-and-merge operations, with merge operations only, using both split and merge (SAM); the measures of entropy (before slashes) and purity (after slashes) indicate the quality of the clusters with respect to their class content (ideally, each cluster represents one class → purity equal to 1 and entropy equal to 0)

	Animals	Cov-type
ECM w/o SAM (using eVQ)	0.248/0.751	0.383/0.610
ECM with merge (using eVQ)	0.248/0.751	0.383 /0.610
ECM–SAM (using eVQ)	**0/1**	**0.105/0.891**
ECM w/o SAM (using EC)	0.248/0.751	0.453/0.542
ECM with merge (using EC)	0.248/0.751	0.421/0.580
ECM–SAM (using EC)	0.134/0.864	0.223/0.775
CLUTOR	0.212/0.818	0.548/0.518
GARDEN	0.239/0.767	0.389/0.652

Bold values indicate the best performing method in terms of entropy/purity (best entropy = 0, best purity = 1)

with n_i the number of samples falling into cluster i (=the support of cluster i) and n_{ik} the number of samples falling into cluster i and belonging to class k. Then, the entropy of a whole cluster partition becomes:

$$E(\mathrm{C}) = \sum_{i=1}^{|\mathrm{C}|} \frac{n_i}{N} E(C_i) \quad (22)$$

with N the number of data samples in the whole data set. The purity of a cluster i is defined by Zhao and Karypis (2004):

$$P(C_i) = \frac{1}{n_i} max_{k=1,\ldots,K}(n_{ik}). \quad (23)$$

Then the purity of a whole cluster partition becomes:

$$P(\mathrm{C}) = \sum_{i=1}^{|\mathrm{C}|} \frac{n_i}{N} P(C_i) \tag{24}$$

The optimum entropy and purity values are 0 and 1, respectively, which means, that all clusters are 'clean', i.e. they contain only samples from one class, thus achieving a non-ambiguous class representation in the whole cluster partition. In fact, new samples falling in a clean cluster can be safely classified to the class it contains (no conflict between classes occurs); furthermore the training samples can be classified with 0 error rate when all clusters in a partition are clean.

Table 3 compares the performance of the new ECM-SAM technique with conventional ECM, ECM including merge operations only (first two parts of the table) and with the batch clustering algorithms GARDEN and CLUTOR (as used in Orlandic et al. 2005) (third part of the table). The split-and-merge concepts are here connected with two different incremental prototype-based clustering approaches, namely eVQ (as used in the prior results) and evolving clustering method (EC) as implemented in DENFIS approach (Kasabov and Song 2002) for constructing rules in an evolving neuro-fuzzy inference system. Also in the latter approach touching or overlapping clusters may arise over time due to the cluster expansion concept. The interesting thing is that (1) this clustering method performs similar as eVQ (compare second with fifth row) and (2) our split-and-merge operations have a similar effect on the performance (achieving a higher purity and a lower entropy).

Furthermore, the ECM-SAM (using eVQ or EC) performs clearly better than the conventional ECMs and also outperforms GARDEN and CLUTOR for cov-type data set and CLUTOR for animals data set. For the animals data set and using eVQ, such a good performance is even achieved with the number of extracted clusters equal to the number of classes (4). Since entropy is equal to 0, a perfect cluster-class match can be obtained; using EC as clustering method performs slightly worse as achieving five clusters. ECM w/o SAM and ECM with merge both extracted three clusters in case when using eVQ and EC; in fact, since ECM w/o SAM only extracts three clusters, achieving an under-clustered partition, the merge operation is more or less obsolete: hence, it is remarkable that it does not perform an unncessary merge of three to two clusters, which shows that the merging criteria work in this case. Moreove, the split operation is able to split the correct cluster with mixed classes (on an unsupervised basis) into two, yielding four clusters in total and optimal entropy/purity values (0/1). CLUTOR also extracts four clusters, but these contain mixed classes, as the entropy level is significantly higher than 0 (0.212). GARDEN performs similarly to CLUTOR, but extracts 1,480 clusters. In Orlandic et al. (2005), it is reported that GARDEN can achieve an entropy of 0.999 with a different parameter setting, but then it extracts 6,208 clusters, which is remarkably high and hard to handle in an on-line classification setting. On the cov-type data set, ECM-SAM clearly out-performs the other methods, but extracts 240 clusters when using eVQ and 322 clusters when using EC, while ECM w/o SAM and ECM with merge both extract 28 clusters when using eVQ and more than 40 clusters when using EC; CLUTOR extracts 84 clusters and GARDEN 112 clusters, while entropy and purity values were similar to those of ECM w/o SAM (however, with 3–4 times more clusters).

7 Conclusion

In order to make data stream clustering more dynamic, we have introduced a new split-and-merge approach for evolving cluster models. Specifically, our approach is able to adapt to changes in the local data clouds, overcomes cluster fusion and delamination effects, and eliminates trial-and-error tuning phases for achieving (an) optimal learning parameter(s). It operates on the basis of sample-wise incremental learning and is able to split and merge clusters as required by the characteristics of local data distributions. It is generally applicable to all kinds of prototype-based incremental, evolving clustering algorithms that extract ellipsoidal clusters in main position and providing centers, spreads and support of all evolved clusters (these three variables are required in all merging and split operations). The merging criteria are based on touching and homogeneity conditions for two clusters, the splitting criterion employs a penalized Bayesian information criterion evaluated for the current versus a sub-clustered partition. All criteria operate in fully sample-wise incremental (on-line) learning mode, single-pass capability is achieved for merging operations, but not for splitting technique, which may go at the expense of computation time (see e.g. Fig. 10), however still operates within milliseconds on each sample, usually fast enough for on-line processes.

The results show that ECMs connected with the proposed split-and-merge operations can extract better cluster partitions than conventional ECMs as coming closer to the real number of clusters contained in high-dimensional streaming data sets together with a higher cluster validation index and producing more reliable partitions visualized by two-dimensional plots. Furthermore, the results for classification data show that split-and-merge is able to extract clusters with lower entropy and higher purity, which indicates a more unique, non-ambiguous representation of the

class distribution in feature space than when applying conventional incremental as well as batch clustering methods. Future work includes (1) the extension and application of the proposed techniques to clustering-based classifiers (classifiers that consist of prototype clusters and use supervised label information) or to clustering/rule-based regression models; and (2) an extension of all the proposed techniques to ellipsoidal clusters in arbitrary position.

Acknowledgments This work was funded by the Austrian fund for promoting scientific research (FWF, contract number I328-N23, acronym IREFS). This publication reflects only the authors' views. The author also acknowledges Eyke Hüllermeier for providing valuable comments on the paper.

References

Andrews GE, Askey R, Roy R (2001) Special functions. Cambridge University Press, Cambridge

Angelov P (2004) An approach for fuzzy rule-base adaptation using on-line clustering. Int J Approx Reason 35(3):275–289

Angelov P, Filev D, Kasabov N (2010) Evolving intelligent systems—methodology and applications. Wiley, New York

Angelov P, Zhou XW (2006) Evolving fuzzy systems from data streams in real-time. In: 2006 International symposium on evolving fuzzy systems (EFS'06). Ambleside, Lake District, pp 29–35

Beringer J, Hüllermeier E (2006) Online clustering of parallel data streams. Data Knowl Eng 58(2):180–204

Beringer J, Hüllermeier E (2007) Adaptive optimization of the number of clusters in fuzzy clustering. In: Proceedings of the FUZZ-IEEE 2007, London, pp 1–6

Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: massive online analysis. J Mach Learn Res 11:1601–1604

Bouchachia A (2011) Evolving clustering: an asset for evolving systems. IEEE SMC Newslett 36

Bouchachia A, Vanaret C (2011) Incremental learning based on growing gaussian mixture models. In: Proceedings of 10th international conference on machine learning and applications (ICMLA 2011), Honululu, Haweii (to appear)

Declercq A, Piater J (2008) 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, pp 605–611

Dovzan D, Skrjanc I (2011) Recursive clustering based on a Gustafson-Kessel algorithm. Evol Syst 2(1):15–24

Farnstrom F, Lewis J, Elkan C (2000) Scalability for clustering algorithms revisited. In: SIGKDD explorations, London 2(1): 51–57

Gama J (2010) Knowledge discovery from data streams. Chapman & Hall/CRC, Boca Raton

Gan G, Ma C, Wu J (2007) Data clustering: theory, algorithms, and applications (Asa-Siam Series on Statistics and Applied Probability). Society for Industrial & Applied Mathematics, USA

Hall P, Hicks Y (2005) A method to add gaussian mixture models. Tech. rep., University of Bath

Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning: data mining, inference and prediction, 2nd edn. Springer, New York

Hühn J, Hüllermeier E (2009) FR3: a fuzzy rule learner for inducing reliable classifiers. IEEE Trans Fuzzy Syst 17(1):138–149

Jain A, Dubes R (1988) Algorithms for clustering data. Prentice Hall, Upper Saddle River

Jimenez L, Landgrebe D (1998) 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):39–54

Kasabov NK, Song Q (2002) DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Trans Fuzzy Syst 10(2):144–154

Klinkenberg R (2004) Learning drifting concepts: example selection vs. example weighting. Intell Data Anal 8(3):281–300

Lima E, Hell M, Ballini R, Gomide F (2010) Evolving fuzzy modeling using participatory learning. In: Angelov P, Filev D, Kasabov N (eds) Evolving intelligent systems: methodology and applications. Wiley, New York, pp 67–86

Lughofer E (2008) Extensions of vector quantization for incremental clustering. Pattern Recogn 41(3):995–1011

Lughofer E (2011) All-pairs evolving fuzzy classifiers for on-line multi-class classification problems. In: Proceedings of the EUSFLAT 2011 conference. Elsevier, Aix-Les-Bains, pp 372–379

Lughofer E, Bouchot JL, Shaker A (2011) On-line elimination of local redundancies in evolving fuzzy systems. Evol Syst 2(3):165–187

Nelles O (2001) Nonlinear system identification. Springer, Berlin

Orlandic R, Lai Y, Yee W (2005) Clustering high-dimensional data using an efficient and effective data space reduction. In: Proceedings of ACM conference on information and knowledge management CIKM05, pp 201–208

Qin S, Li W, Yue H (2000) Recursive PCA for adaptive process monitoring. J Process Control 10(5):471–486

Schwarz G (1978) Estimating the dimension of a model. Ann Stat 6(2):461–464

Song M, Wang H (2005) Highly efficient incremental estimation of gaussian mixture models for online data stream clustering. In: Priddy KL (ed) Intelligent computing: theory and applications III. In: Proceedings of the SPIE, vol 5803, pp 174–183

Sun H, Wang S, Jiang Q (2004) FCM-based model selection algorithm for determining the number of clusters. Pattern Recogn 37(10):2027–2037

Tabata K, Kudo MSM (2010) Data compression by volume prototypes for streaming data. Pattern Recogn 43(9):3162–3176

Vachkov G (2010) Similarity analysis and knowledge acquisition by use of evolving neural models and fuzzy decision. In: Angelov P, Filev D, Kasabov N(eds) Evolving intelligent systems: methodology and applications. Wiley, Hoboken, pp 247–272

Varmuza K, Filzmoser P (2009) Introduction to multivariate statistical analysis in chemometrics. CRC Press, Boca Raton

Vitter JS (1985) Random sampling with a reservoir. ACM Trans Math Softw 11:37–57

Xie XL, Beni G (1991) A validity measure for fuzzy clustering. IEEE Trans Pattern Anal Mach Intell 13(48):841–847

Zhao Y, Karypis G (2004) Empirical and theoretical comparisons of selected criterion functions for document clustering. Mach Learn 55(3):311–331