Progress in Artificial Intelligence (2024) 13:51–64
https://doi.org/10.1007/s13748-024-00316-1

REGULAR PAPER

Split incremental clustering algorithm of mixed data stream

Siwar Gorrab[1] · Fahmi Ben Rejab[1] · Kaouther Nouira[1]

Received: 5 April 2022 / Accepted: 21 February 2024 / Published online: 7 March 2024

Abstract
Clustering has been recognized as one of the most prominent functions in data mining. It aims to partition a given set of elements into homogeneous groups without any given knowledge about the distribution of data and according to some (dis)similarity criterion. In this paper, we propose a novel streaming algorithm, based on split technique that was introduced to avoid retaining from the scratch and to ensure the incremental clustering aspect. It intends to cluster continuously arriving chunks of data escorted with new mixed features within memory and time restrictions. Our proposed real-time clustering method clusters mixed data streams using split technique in order to tackle the incremental object, attribute, and class learning spaces at once. So, when necessary, the final distribution of the clusters has to be updated. By dint of split technique, changing the final clusters' distribution has led to a promising clustering model. Experiments performed on real mixed data sets show that the proposal is efficient and outperforms the conventional k-prototypes method based on different evaluation measures.

Keywords K-prototypes · Mixed data stream · Incremental learning · Split · Merge · Dynamic clustering

1 Introduction

As being a subcategory of machine learning, the unsupervised clustering algorithm has always been useful to categorize unlabeled data instances such that according to some particular metrics, similar elements are grouped in the same cluster [1]. Furthermore, when objects are unlabeled and the environment only provides inputs without desired targets, clustering is one of the principal techniques applied for mining data. It is arising from many fields and applied successfully to various domains some of which are banking, e-health, fraud detection, image segmentation and document mining. Indeed, data clustering has been considered as a primary data mining method for knowledge discovery. On a daily basis, large volumes of highly detailed data are continuously forthcoming from sensors, devices, Web applications, and social media in view of data streams and its partitioning into sets of meaningful sub-classes is required for proper mining of intended data [2]. So, big data is not just big because it is coming from a greater variety of sources than ever before. It is also diverse data types and streaming data: sequences of data elements from various sources which further need to be analyzed and investigated. Over and above that, data stream clustering concept is the streaming algorithm that aims to cluster continuously arriving huge volumes of mixed data within memory and time restrictions. On that account, the k-prototypes algorithm is distinguished for its simplicity, speed of convergence and scalability in the respect of clustering mixed data types. Bearing these restrictions in mind, algorithms for clustering data streams should ideally fulfill the following requirements [3]:

- Provide timely results by performing fast and incremental processing of data objects.
- Rapidly adapt to changing dynamics of the data, which means algorithms should detect when new clusters may appear, or others disappear.
- Scale to the number of objects and attributes that are continuously arriving.
- Provide a model representation that is not only compact, but that also does not grow with the number of objects processed.

✉ Siwar Gorrab
siwarg9@gmail.com

Fahmi Ben Rejab
fahmi.benrejab@gmail.com

Kaouther Nouira
kaouther.nouirar@gmail.com

1 ISGT, LR99ES04 BESTMOD, Université de Tunis, Tunis, Tunisia

So, these incoming data streams have to be learned as the learning proceeds and without regenerating the initially constructed model, based on available input data, from scratch. To do so, the most effective and well-known methods in the context of static machine learning offer no alternative to evolution and dynamic adaptation to integrate new data or to restructure problems already partially learned. Therefore, incremental learning turns into an interesting alternative in which data becomes accessible in sequential order and is used to update the most appropriate predictor for future data at each stage, unlike the batch learning techniques that generate the best predictor by learning on the all training data at once [4]. Thereby, additional information from new data streams might be learned in an incremental way. That is to say without having access to previously available data and without requiring the retraining of the system on the old and the new training data. Particularly, most of the clustering algorithms presented in the literature are used for clustering static databases.

Nowadays, research community has shifted focus to the incremental databases for real-world applications and the necessity of handling dynamically updated data points [5]. In incremental clustering method, the learning set is not essentially available from the beginning. Actually, the data streams are introduced as the learning takes place. Still, at the arrival of a new data stream, the traditional clustering methods need to regenerate their clusters from scratch. In contrast, the incremental clustering only needs to cluster new incoming data and then to update the current distribution of clusters in such a way that it optimizes the clustering process and decreases the time-consuming. Generally, the initial number of clusters k is set by users or archives from knowledge of research. However, a bad choice of k can lead to a wrong distribution of observations. Thus, the incremental level remains not only for observations but also for the most appropriate cluster to the joined elements [6]. So the final clusters' distribution remains challenging in the context of high-dimensional clustering algorithms.

Hence, in this paper, we are looking for developing a scalable system that will not only learn their parameters incrementally, but also change their structure. Into the bargain, whenever it is necessary and a domain expert asks to skip from k to k+1 clusters, it is achievable to proceed using our developed SIK-prototypes method with applying the split procedure. It is more flexible than an adaptive system since it allows, in use, the addition of supplementary clusters when necessary. More precisely, our study aims to deal with incremental attribute, object and class learning tasks where the learning environment is steadily changing and the training samples become available one after another over time. So, initial clustering and handling of incremental data points, attributes, and clusters are two important steps of our proposed split incremental clustering approach based on K-prototypes algorithm. This latter is an extension of the primarily proposed IK-prototypes algorithm [18], an increased version of the conventional k-prototypes method, that is skillful in the incremental attribute learning context. Indeed, the study proposed in this paper is capable of handling a bulk of updates owing to the training samples, with new added mixed attributes and with a necessity to the incremental class learning space, which become available one after another over time. In other words, our work addresses the problem of mixed-type categorical and numerical data in incremental unsupervised clustering learning. The goal is building a framework that automatically handles the differences in numerical and categorical features in an emerging data stream environment and groups them into similar clusters while tackling the incremental object, attributes, and class learning spaces. That is to deal with the shrink of clusters and to update the final clusters' distribution without retraining from scratch using a proposed split method, based on statistical and mathematical tools. Actually, the main purpose of our proposed split procedure is to generate a new cluster to the actual clusters' distribution without retraining the whole model from scratch, when dealing with continuously emerging mixed data streams. This would be based on a domain expert as follows: whenever the expert demands to move from k to k+1 clusters, we suggest to apply the split procedure so that to maintain the incremental class learning level. Here, the expert role is to analyze the final clusters' distribution, their variances, densities and shapes so as to eventually decide whenever it is better to split the most appropriate cluster, based on two cluster evaluation criteria: the Index of Dispersion and the Sum of Squared Error.

We have tested our proposed Split Incremental K-prototypes algorithm on a suite of benchmarks, among both simulated and real data sets. This ended up with promising and quite encouraging results. Comparisons have been made with the batch method of similar nature. The quality of the solutions offered by each method is rated in terms of run time value and similarity between and within clusters measures. The acquired outcomes emphasize the performance of our proposal in terms of less memory and time consumption, as being a continual attribute, object and class learning approach. The rest of the paper is organized as follows: literature reviews and related works are introduced in Sect. 2. In Sect. 3, the conventional k-prototypes algorithm is illustrated. We devote Sect. 4 to present and to detail our proposal. Section 5 displays the experimental simulations and results on number of real mixed data sets. Finally, we conclude this paper in Sect. 6.

2 Retaled works

All long this Sect. 2, several related works would be illustrated with details. Starting from the more general topic (data stream clustering methods) to the specific one (incremental clustering techniques for handling mixed data) and passing by number of split and merge algorithms into the bargain.

2.1 Data stream clustering methods

This subsection discusses previous works on data stream clustering problems and highlights the most pertinent partitioning stream algorithms proposed in the literature to deal with this problem. In [10], the *CluStream* method has been proposed to generate approximate clusters in any user-specified length of history from the current moment. Its main purpose was to divide the clustering process into an online component which periodically stores detailed summary statistics and an offline component which uses only this summary statistics [11]. The final clusters are determined by using a modification of a k-means algorithm. In [12], a seeding procedure for the k-means algorithm that gives good practical results is proposed: the StreamKM++. It is a two-phase (online–offline) algorithm, which maintains a short outline of the input data based on the merge and reduce technique. The merge step is performed via a data structure, named the bucket set (buffer), whereas the reduce step is performed by a notably different summary data structure that is suitable for high-dimensional data: the coreset-tree [11]. When a new data point arrives, it is stored in the first bucket. If the first bucket is full, all of its data are moved to the second bucket. If the second bucket is full, the two buckets are merged resulting in 2 m data points, which are then reduced to m data points, by the construction of a coreset tree, as previously detailed. The resulting m data points are stored in the third bucket, unless it is also full, and then again a new merge and reduce step is needed [12]. In [13], the High-dimensional Projected Stream clustering method (HPStream) was introduced where its focal objective was to put forward the concept of projected clustering to data streams. This algorithm is a projected clustering for high-dimensional streaming data with higher clustering quality compared to CluStream [10]. Overall, most of these existing algorithms (e.g. CluStream [10], StreamKM++ [12]) partition the clustering process in two phases: (a) online, the data will be summarized; (b) offline, the final clusters will be generated. Nevertheless, these mentioned data stream clustering methods are not able to handle mixed data sets of both numerical and categorical attributes at the same deal. Also, they do only tackle the incremental object learning space. Therefore, here is a requirement to promote the unsupervised clustering algorithms with a new method, capable of learning continuously joined mixed data streams in an incremental manner and with handling all object, attribute and class spaces.

2.2 Split and merge algorithms

A large area of research in clustering looks toward one way of improving the clustering process such that the clusters are not dependent from the initial identification of cluster representation. In other words, changing the final clusters' distribution may lead to a promising clustering model. In this issue, most of clustering algorithms become ineffective when provided with unsuitable parameters or applied with data sets, composed of clusters with different shapes, sizes and densities [14]. There are versions of adaptive clustering algorithms that allow the regeneration of all the clustering procedure from scratch to response to the change of data patterns which is a time-consuming task [15]. In addition, those techniques result in producing large difference in terms of the size, the shape and the clusters' density. So, when necessary, the final distribution of the clusters has to be updated. Into the bargain, the split technique was introduced in clustering to avoid retaining from the scratch and to ensure the incremental clustering aspect. The real challenge here is to choose the appropriate cluster that should be split. In fact, the following several approaches have been used for the selection of the cluster to split [16]:

- Complete partition of the clusters which results in a complete binary tree.
- Split the cluster owing the highest size (number of elements).
- Split the cluster owing the highest variance.
- According to the clusters' shapes.

However, the first split criteria completely ignores the issue of the clusters' quality. The second one has the advantage of yielding a balanced tree, where the leaves have approximately the same size. The variance and the shape of the clusters are the most sophisticated criteria since they are based on a meaningful statistical property of a cluster. This is the reason why highest variance criteria is the most commonly used criterion for cluster selection [16].

On the other hand, the merge technique was introduced in clustering to avoid the retraining from the scratch. Indeed, merging techniques, that consists in joining the most likely elements of two sets of data together, has become more and more relevant since data generated at real time have increased which makes the retraining of the result from the scratch difficult and a time-consuming task [17]. Particularly, after providing a new set of clusters on a new portion of data collected over a defined time period, the merge technique provides the possibility to update the existing clustering solution by the new added ones. As a result, some clusters will be updated by merging them with ones from the newly constructed clusters.

2.3 Incremental clustering techniques for handling mixed data

In recent times, data mining community turned their focus on incremental clustering of dynamically updated data points that contains numerical as well as categorical attributes [7]. Accordingly, few are the literatures that highlight the incremental clustering techniques, capable to handle mixed data sets. The Clustering Algorithm based on the methods of Variance and Entropy (CAVE) [8] calculates the similarity measure for categorical attributes by entropy and numerical attributes by variance and where the number of clusters is predefined. At first, the dissimilarity of two objects is computed and grouped into two different clusters and the dissimilarity of remaining records is calculated and associated into the appropriate clusters. Indeed, this current process is repeated until the records are empty which reflects the incremental aspect. Besides, the CAVE algorithm can stop processing at any time and produces the output and it incrementally gets the updated clusters whenever any new data pattern arrives at the cluster. In [9], a Mixed Self-Organizing Incremental Neural Network algorithm was proposed. Initially, the clusters are applied to the neural network and, in every iteration, it will be deleted if it does not win through the learning process. A new distance measure has been proposed based on the frequencies of two categorical features; instances are not clustered in the similar cluster, if the dissimilarity measure of categorical attributes is larger. Another incremental clustering algorithm for mixed data sets is the Cluster-Feature-Based incremental clustering method [7] that proposes a new distance measure based on the weight-age which is automatically generated. Using this algorithm, the incremental data objects are handled in the following two phases (a) Firstly, the k-means clustering algorithm is employed for initial clustering of the static database; (b) Secondly, the designed distance measure is used to generate the appropriate cluster for the incremental data points. Accordingly, their combination has proved to be more efficient in covering both numerical and categorical attributes. Nonetheless, all these cited methods do not deal neither with incremental attribute learning, nor with incremental class learning tasks. Expressing differently, the final distribution of the clusters remains unchanged and the added attributes, arriving with the joined data point are neglected.

3 Preliminaries

This section first presents the theoretical concepts of the k-prototypes clustering algorithm, then presents the Incremental-K-prototypes method (IK-prototypes) proposed in [18], on which this work is based on.

3.1 Theoretical concepts of k-prototypes algorithm

In [19], the k-prototypes clustering algorithm integrates both k-means [20] and k-modes [21] algorithms in order to deal with mixed-type data. It is widespread thanks to its simplicity, adaptivity and speed of convergence.

Given a data set X={$x_1 \dots x_n$} of n data points containing m_r numeric attributes and m_t categorical attributes, the focal objective of the k-prototype algorithm is to group the data set X into k different clusters while minimizing the following cost function in Eq. (1):

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij} d(x_i - c_j), \tag{1}$$

where $u_{ij} \in \{0, 1\}$ is an element of the partition matrix U_{n*k}, indicating the membership of data point i in cluster j; $c_j \in C = \{c_1 \dots c_k\}$ is the center of the cluster j and $d(x_i - c_j)$ is the dissimilarity measure defined in Eq. (2):

$$d(x_i - c_j) = \sum_{r=1}^{m_r} \sqrt{(x_{ir} - c_{jr})^2} + \sum_{t=1}^{m_t} \delta(x_{it}, c_{jt}), \tag{2}$$

where x_{ir} and x_{it} represent, respectively, the values of the numeric attribute r and the categorical attribute t for a data point i; c_{jr} represents the mean of the numeric attribute r and cluster j, calculated using Eq. (3):

$$c_{jr} = \frac{\sum_{i=1}^{|c_j|} x_{ir}}{|c_j|} \tag{3}$$

where $|c_j|$ is the number of data points assigned to a cluster j; c_{jt} is the most common value (mode) for categorical attributes t and cluster j, calculated using Eq. (4):

$$c_{jt} = a_t^h \tag{4}$$

where $f(a_t^h) \succeq f(a_t^z)$, $\forall z$, $1 \leq z \leq m_c$ **having** $a_t^z \in \{a_t^1 \dots a_t^{m_c}\}$ is the categorical value z and m_c is the number of categories of categorical attribute t; $f(a_t^z) =| x_{it} = a_t^z \mid p_{ij} = 1 \mid$ is the frequency count of the attribute value a_t^z; for categorical features, $\delta(p, q) = 0$ when p=q and $\delta(p, q) = 1$ when $p \neq q$.

Algorithm 1 K-prototypes algorithm

```
1: Input: X: data set, k: number of clusters
2: Output: Cluster centers
3: Begin
4:     Choose randomly $k$ initial cluster centers from the data set $X$.
5:     Attribute each data point in $X$ to its nearest cluster center according to Equation (2).
6:     Update the cluster centers after each allocation using Equation (3) and Equation (4).
7:     If the updated cluster centers are identical to the previous ones then terminate, otherwise, go back to step 5.
8: End.
```

Algorithm 2 Incremental K-prototypes algorithm

```
1: Input: $X$: data set, $k$: number of clusters
2: Output: $k$ Cluster centers
3: Begin
4:     Select $k$ initial prototypes (cluster centers) randomly from each arriving data stream $X$.
5:     Attribute each data point in $X$ to its closest cluster center according to Equation (2).
6:     Update the cluster centers after each allocation using Equation (3) and Equation (4).
7:     If the updated cluster centers are identical to the previous ones then terminate, otherwise, go back to step 5.
8:     If a new data stream arrives with new attributes then go back to step 5, Merge the resulting clusters from both models, otherwise, terminate.
9: End.
```

3.2 Incremental k-prototypes method

The conventional k-prototypes algorithm is unable to deal with incremental data sets, emerging as streaming data. Each time a new chunk of data arrives, two procedures might be appealed: either to retrain from the scratch that is to build a new model and forget about the initial one which is a time-consuming task, or just to ignore the old data and only consider the new one which would lead to the waste of knowledge. Into the bargain, a new incremental k-prototypes method has been proposed in [18], namely IK-prototypes so that to deal with both incremental object and attributes spaces. In other words, once new instances become available over time, particularly when these new input data streams include in addition to the old set of features new ones, the IK-prototypes algorithm is dedicated. Mainly, this novel approach is an extension of the k-prototypes algorithm, that is able to handle such dynamic object and attribute spaces, namely the skillful incremental attribute learning task. As a matter of fact, a primary model is built relying on the current data available at first. Then, the learning process proceeds as long as new instances escorted with new set of attributes, in addition to the old ones, take place for the sake of building a complementary model that incorporates knowledge from the old extracted and the newly added data stream as well. The main process of the IK-prototypes algorithm is detailed as follows:

At the outset, assuming that we have learned on some primarily accessible data using the k-prototypes algorithm, then at the arrival of new instances escorted with new features as streaming data, we suggest to keep in background a standard k-prototypes algorithm which would, in this step, learn only these incoming data. Subsequently, knowledge from both initial and data stream models would be merged to further acquire knowledge from the joined models and to achieve the incremental attribute and object learning tasks. In details, once an initial input data are available, we start with applying the k-prototypes algorithm. The result here is k different clusters as knowledge from the initial model in which each similar instances are joined together in one cluster. Afterward, new data objects with new features penetrate as the learning proceeds as well be a data stream. At this level, we move on to the second phase of our proposed IK-prototypes method that consists in applying the k-prototypes algorithm only on the newly joined data stream. Hence, we aim to create k' clusters and then save the obtained results as a second knowledge from the data stream model. Consequently, the proposed IK-prototypes will pursue with integrating the acquired results derived from both models in order to get a coherent model based on the following merge algorithm:

In fact, to ensure an incremental attribute learning system, the focal idea was to merge the knowledge coming from both initial and data stream models in such a way that each two similar clusters are combined together while guaranteeing the go back to the initial number of clusters k because this work deals only with incremental object and attribute learning tasks and not with incremental class learning. For the sake of clarity, the similarity between clusters is based on two main indexes:

1. The Davies–Bouldin Index [22] where the similarity measure here is based on a comparison between the distance between clusters and the size of the clusters themselves (values closer to zero relate to a better partition of clusters). This index is calculated using Eq. (5):

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max R_{ij};\ i \neq j \tag{5}$$

where R_{ij} is the similarity measure, calculated as follows in Eq. (6)

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \tag{6}$$

With s_i is the cluster i diameter and d_{ij} is the distance between both cluster centers i and j.

Algorithm 3 Merge algorithm

```
1: Input: clusters = $\{c_1, c_2, ..c_k\}$
2: Output: clusters = clusters = $\{c'_{i..k}\}$
3: Begin
4:   For each cluster $A_i$ in $\{c_1, c_2, c_k\}$ do
5:     For each cluster $B_j$ in $\{c_4, c_5, c_{k'}\}$ do
6:       $DB \leftarrow computeDaviesBouldinIndex(A_i, B_j)$
7:       $CH \leftarrow computeCalinskiHarabaszIndex(A_i, B_j)$
8:     end For
9:   end For
10:    For each cluster $A_i$ in $\{c_1, c_2, c_k\}$ do
11:     For each cluster $B_j$ in $\{c_4, c_5, c_{k'}\}$ do
12:       If $DB(A_i, B_j)$ = Max(DB) then
13:         $Cluster1 \leftarrow A_i$
14:         $Cluster2 \leftarrow B_j$
15:       end If
16:       If $CH(A_i, B_j)$ = Min(CH) then
17:         $Cluster3 \leftarrow A_i$
18:         $Cluster4 \leftarrow B_j$
19:       end If
20:     end For
21:    end For
22:    For $i$ in $[1..k]$ do
23:     If (Cluster1=Cluster3) and (Cluster2=Cluster4) then
24:       $c'_i \leftarrow Merge(Cluster1, Cluster2)$
25:       Delete (Cluster1)
26:       Delete (Cluster2)
27:       i=i+1
28:     end If
29:       $SSE1 \leftarrow SSE(Merge(Cluster1, Cluster2))$
30:       $SSE2 \leftarrow SSE(Merge(Cluster3, Cluster4))$
31:     If SSE1 < SSE2 then
32:       $c'_i \leftarrow Merge(Cluster1, Cluster2)$
33:       Delete (Cluster1)
34:       Delete (Cluster2)
35:     Else
36:       $c'_i \leftarrow Merge(Cluster3, Cluster4)$
37:       Delete (Cluster1)
38:       Delete (Cluster2)
39:     end If
40:    end For
41: Return clusters = $\{c'_{i..k}\}$
42: End.
```

2. The Calinski–Harabasz score [23]: it determines the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters where the dispersion here is definitely the sum of the squared distances between each point to its nearest cluster center. A higher value refers to a model with better defined clusters.
   The Calinski–Harabasz score s is calculated as shown in Eq. (7):

$$s = \frac{tr(B_k)}{tr(W_k)} * \frac{n_E - k}{k - 1} \tag{7}$$

   where n_E is the size of the set of data **E**; k is the number of clusters; $tr(B_k)$ is trace of the between group dispersion matrix; and $tr(W_k)$ is the trace of the within-cluster dispersion matrix.
   defined by the subsequent Eqs. (8) and (9):

$$B_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^{\mathrm{T}} \tag{8}$$

$$W_k = \sum_{q=1}^{k} n_q (C_q - c_E)(C_q - c_E)^{\mathrm{T}} \tag{9}$$

   knowing that n_q is the number of points in cluster q and c_q and c_E are, respectively, the centers of the cluster q and of E.

Nevertheless, once in a while the highest Davies–Bouldin index and the lowest Calinski–Harabasz score may not be the best choices for the merge procedure if they result in different combination of clusters. In such a case, the algorithm will carry on merging both clusters resulting from the two calculated indexes and ends up with maintaining the cluster with the lowest sum of squared error (SSE) [24]. This SSE determines the dispersion of elements of a cluster in relation with their centroids, that is the sum-squared distances of samples to their closest cluster center.

4 Split incremental k-prototypes clustering method for mixed data streams

The proposed method toward handling mixed large-scale data, as well be emerging data streams, consists of an increase and an enlargement method to the previously proposed IK-prototypes method [18] through incremental attribute learning (IAL) context. Here, we cannot forget to mention that the proposed IK-prototypes [18] are, at first, an extension of the K-prototypes algorithm [19] that does not perform well in such a streaming mixed data environment. Accordingly, in this paper we are stepping from the K-prototypes algorithm, passing through the IK-prototypes method until reaching our new proposed Split Incremental K-prototypes clustering algorithm (SIK-prototypes). The major novelty of this latter remains in suggesting a split function that aims to achieve the third incremental space: incremental class learning task.

4.1 Definition and approach presentation

In order to improve the k-prototypes algorithm and to append the incremental object, attribute, and class learning tasks, the split incremental k-prototypes clustering algorithm (SIK-prototypes) is proposed in this paper. Our proposal is an incremental mixed data stream clustering method based on IK-prototypes [18] method, which well performs when new input pattern appears in addition to the old set. It is a novel split approach, capable of handling such dynamic object, attribute and class spaces. In other words, the conventional k-prototypes method does not perform well in such context. Consequently, it has been extended to an IK-prototypes method [18], mentioned in Sect. 3.2, that basically deals with mixed attribute and object learning tasks based on the merge technique, detailed in Algorithm 3. In our work, we are supplying the IK-prototypes algorithm with a recent and crucial property: the ability to learn new mixed features in a streaming data environment with fulfilling the incremental class learning task using a split technique. This would be in reference to an expert domain that better decides whenever it is necessary to adjust and change the final clusters' distribution according some metrics.

This novel incremental class learning algorithm takes place when continuously adding features with newly added instances as data streams is a question. Furthermore, the IK-prototypes [18] are an adaptive incremental learning algorithm, which will learn its parameters incrementally as time proceeds but whose structure is initialized at the beginning of the learning procedure and remains unchanged during the training step. Conversely to our new proposed SIK-prototypes which is a scalable system that will not only learn its parameters incrementally, but also change its structure. In fact, its innovation is depicted in the split function.

The main idea is to split one cluster into two clusters for the purpose of gaining a well-defined model with better separation between clusters and more similarity within clusters. As a matter of fact, at anytime and all long the clustering task, new instances escorted with new incoming features might emerge continuously as data streams. One may appeal to retrain the model from the scratch at the arrival of these new data or just ignore the old data and only consider the new ones. Nevertheless, both of these procedures are not the best choices when dealing with these emerging data streams for the following reasons: the former would be time- and memory-consuming task since a relearning from the scratch will take place each time new mixed data stream comes out, the latter would result in losing old knowledge from initial model. Thus, our proposal is more flexible than the IK-prototypes since it allows, in use, the addition of supplementary clusters when necessary and without retraining from scratch and without forgetting initial/ old data, as they become obsolete.

The focal concept behind our proposed SIK-prototypes method, established in four steps, is shown in Fig. 1.

At the outset, we start by applying the conventional k-prototypes algorithm on the initially available input data. The result here is k different clusters such as similar instances are grouped together in one cluster. Afterward, new data objects with new mixed features penetrate as time proceeds. At this level, we move on to the second step of our proposed SIK-prototypes method that consists in applying the k-prototypes algorithm only on the newly joined mixed data stream. Hence, we aim to create k' clusters using the same k-prototypes algorithm and then save the obtained results as a second knowledge from the data stream model. Accordingly, a second model have just been created once learning the new data stream. Overall, we have gained knowledge from both models that would be merged according to some similarity measures using the merge process (see Sect. 3.2) and results in the initial number k of clusters because at this step, we have not yet dealing with incremental class learning task. The process could end up at this level with a coherent clustering result and while ensuring both incremental object and attribute learning tasks. So, the merge procedure is applied before starting the split function.

Thereafter, whenever the expert demands to move from k to k+1 clusters, we suggest to apply the split procedure so that to maintain the third incremental level, namely incremental class learning task. This is with the objective to change the last clusters' distribution in order to gain a more consistent model. Besides, data streams are continuously emerging with new numeric and categorical features over time and while the learning process. Consequently, similarity within clusters may decrease and it would be better to elevate the number of clusters so that to ensure a promising clustering result. Here comes the role of an expert domain, who will resolve such problem with applying our proposed split procedure as shown in Fig. 1. At this step, our proposed SIK-prototypes is supposed to apply the split function that intends to divide one chosen cluster into two. As a result, the number of clusters increases to k+1 clusters. Here is our proposed SIK-prototypes algorithm:

Algorithm 4 Split Incremental K-prototypes algorithm
```
1: Input: $X$: data set, $k$: number of clusters
2: Output: $k$+1 Cluster centers
3: Begin
4:     Choose randomly $k$ initial prototypes from each arriving data $X$
5:     Attribute each data point in $X$ to its nearest cluster center according to Equation (2).
6:     Update the cluster centers after each allocation using Equation (3) and Equation (4).
7:     If the updated cluster centers are identical to the previous ones then terminate, otherwise, go back to step 5.
8:     If a new data stream arrives with new attributes then go back to step 5, Merge the resulting clusters from initial and data stream models, otherwise, terminate.
9:     If the expert asks to go from $k$ to $k$+1 clusters then, Split the clusters resulting from merge procedure, otherwise, terminate.
10: End.
```

First, the input data set is randomly divided into m chunks of data, namely data streams. Then, each chunk is processed independently in the learning phase using the conventional k-prototypes method. The resulting k cluster centers are extracted from each chunk and the number of clusters is pre-

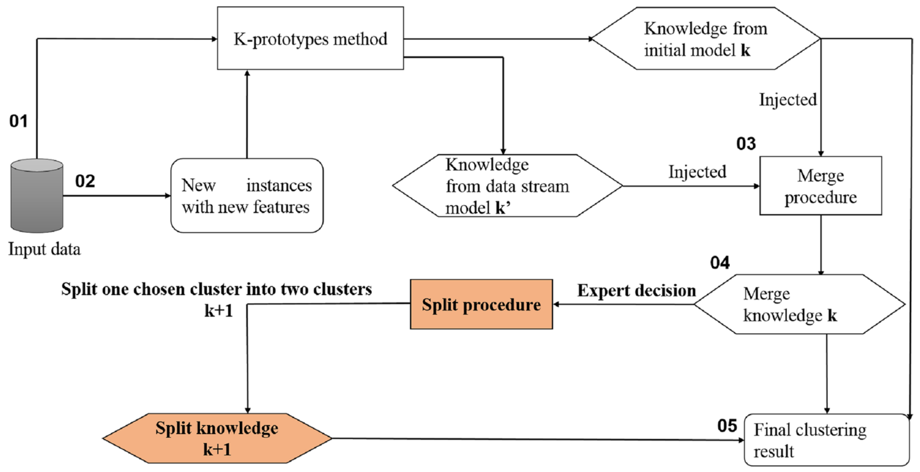

Fig. 1 The proposed Split Incremental K-prototypes through IAL context

defined. Thereafter, the merge procedure processes the set of gained clusters from both initial and data stream models in order to generate the final cluster centers. This step allows to ensure the IAL task, which represents the focal objective of the firstly proposed IK-prototypes [18] method. Consequently, we first gained knowledge from the merge procedure, namely k cluster centers resulting from all accessible data at that time. The learning process could end up at this level with a coherent clustering result. In the following, we aim to fulfill the final incremental level: the incremental class learning step. To do so, whenever it is necessary and the expert asks to move from k to k+1 clusters, the split procedure takes place on the resulting knowledge of the merge process. Notably, we have to choose the most appropriate cluster to split according to some specific criteria. One may ask here: how can we split the integrated results derived from both models in order to get a coherent model? We are going to deeply explain the split process in the next subsection which represents the fourth step of our proposal.

4.2 Split procedure

The main novelty in this proposed SIK-prototypes method is the development of a split procedure that effectively performs the class-incremental learning task in a streaming data context. Accordingly, it avoids retraining from scratch when learning continuously emerging mixed data streams. This is accomplished instead of regenerating all the clustering procedure from scratch to response to the change of data patterns, which is a time- and memory-consuming task. More precisely, if the final clusters' distribution is composed of clusters with different densities, shapes and sizes, it is better to be changed. In our proposal, this problem is solved based on our proposed split function. Indeed, to ensure an incremental class learning system, we have proposed to split the most appropriate cluster from knowledge, coming from both initial and data stream models after merge, using the split algorithm. More to the point, once a new data stream emerges with more added mixed features, it is possible through our proposed SIK-prototypes, to learn it without retraining from the scratch and without forgetting the old data which have been already learnt.

How is that performed? Firstly, we have suggested to merge clusters resulting from both initial and data stream models such that every couple of similar clusters are combined together and with ensuring to return to the initial number of clusters k. Afterward, a split procedure takes place. Its purpose remains in changing the final clusters' distribution whenever needed, with reference to an expert, since data generated at real-time has increased. This makes the retraining of the clustering result from the scratch difficult and a time-consuming task. Hence, in such situations it is necessary to split a chosen cluster in two in order to accomplish the incremental class learning task. Whenever the expert demands to move from k to k+1 clusters, we suggest to apply the split procedure so that to maintain the incremental class learning level. Here, the expert role is to analyze the final distribution of clusters, their variances, densities and shapes to eventually decide whenever it is better to split the most appropriate cluster.

In other words, the focal objective of our proposed split procedure is the generation of a new cluster to the actual clusters' distribution without retraining from scratch. Aiming to showcase the split algorithm, the main challenge remains in selecting the most suitable cluster to be split from the present distribution. In the literature, there are multiple criteria that can be the base of a split procedure. The variance, the density and the shape of the clusters are the most sophisticated criteria since they are based on a meaningful statistical property of a cluster. Between those, we have opted for the most significant ones in our unsupervised clustering context in terms of dispersion of elements between and within clusters. They are presented as follows:

- **Sum of Squared Error (SSE)**: the SSE value is the cluster inertia that computes the sum of squared distances of samples to their closest cluster center. By the same token, it is used as an evaluation criteria for clustering algorithms where the more it is closer to zero, the more clusters are well defined and its calculation is as follows in Eq. (10):

$$SSE_{x\in C} = \sum_{i=1}^{n}\sum_{j=1}^{m}(x_{kj} - c_j) \tag{10}$$

  where c_j is the center of the cluster j.
- **Index of Dispersion (ID)**: as indicated by its name, the it quantifies either if data objects are close to their cluster center or if they are broadly dispersed [25]. ID is also known as variance to mean value, a common used index in statistics and probability theory, that calculates the square of the standard deviation σ divided by the mean of the observation μ. Accordingly, a high ID value indicates that data points are spread out over a wider range of values. Hence, we are looking for a lowest ID score. It is calculated using Eq. (11):

$$ID_{x\in C} = \frac{\sigma^2}{\mu} \tag{11}$$

  where the standard deviation σ is calculated in Eq. (12):

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{n}(x_j - \mu)^2} \tag{12}$$

Specifically, in our study the split process is based on three consecutive steps as shown in Fig. 2, organized as follows:

1. Calculate the SSE and the ID of each present cluster.
2. Select the cluster with the highest SSE and ID scores at the same deal.
3. Split the selected cluster using the k-prototypes algorithm.

However, the highest SSE and ID scores may result in different combination of clusters. It means that one cluster may not have the maximum SSE and ID values at once. In this case, our proposed split algorithm is supposed to carry on with comparing the size of the highest SSE score cluster and the highest ID score one. Consequently, the chosen cluster to be split is the one with the highest size.

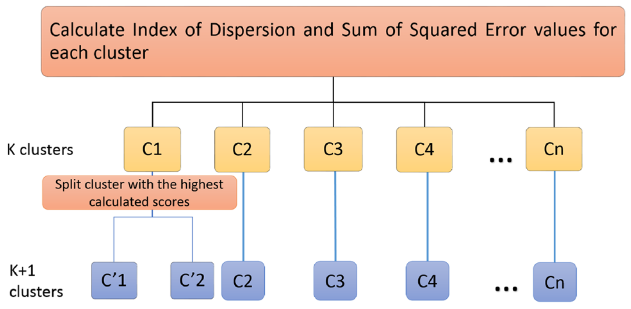
Calculate Index of Dispersion and Sum of Squared Error values for each cluster
K clusters
C1
C2
C3
C4
...
Cn
Split cluster with the highest calculated scores
K+1 clusters
C'1
C'2
C2
C3
C4
...
Cn

Fig. 2 Workflow of the split procedure

Algorithm 5 Split algorithm

```
1: Input: clusters = $\{c_1, c_2, c_3\}$
2: Output: clusters = $\{c'_1, c'_2, c'_3, c'_4\}$
3: Begin
4: $c' = c$
5:   For each cluster $c'_i$ in $\{c'_1, c'_2, c'_3\}$ do
6:       $SSE \leftarrow computeSSE(c'_i)$
7:       $ID \leftarrow computeID(c'_i)$
8:   end For
9:   For ($j$=1, $j <= k$) do
10:      If $SSE(cl_i)$ = Max( SSE($c'_1$), SSE($c'_2$), SSE($c'_3$) ) then
11:          $Indice_{SSE} \leftarrow j$
12:      end If
13:      If $ID(cl_i)$ = Max( ID($c'_1$), ID($c'_2$), ID($c'_3$) ) then
14:          $Indice_{ID} \leftarrow j$
15:      end If
16:  end For
17:  If ($Indice_{SSE} = Indice_{ID}$) then Split ($cl_{Indice_{SSE}}$)
18:      Else IF (cl.Size($cl_{Indice_{SSE}}$) > cl.Size($cl_{Indice_{ID}}$)) then Split ($cl_{Indice_{SSE}}$)
19:  Else
20:      Split ($cl_{Indice_{ID}}$)
21:  end If
22: Return clusters = $\{c'_1, c'_2, c'_3, c'_4\}$
23: End.
```

Now, that the split algorithm has been presented as shown in Algorithm 5, we move on to clarify some functions which have been used there.

- ***Compute (cluster):***
  - **Input:** cluster i.
  - **Output:** SSE and ID scores of corresponding cluster.

  Notably, the function Compute (cluster) is the fundamental and the key function of the split procedure since it has a major impact on the choice of the cluster to be split. In fact, it computes the SSE and the ID scores of each cluster. Suitably, it determines the cluster corresponding to the highest SSE and ID values, which further will be split.
- ***Max (SSE / ID):***
  - **Input:** clusters of calculated indexes.
  - **Output:** highest cluster index.

The function Max (SSE/ID) is used in our proposed split algorithm to search appropriately into the clusters' SSE and ID calculated values. Accurately, it retrieves the largest SSE and ID scores.

- ***Split (cluster):***
  - **Input:** cluster i.
  - **Output:** two sub-clusters i′ and j′.

  After defining the most suitable cluster to be split, we pursue with applying the function Split (cluster) that consists in running the simple k-prototypes algorithm with number of clusters k=2.
- ***cl.size(cluster):***
  - **Input:** cluster i.
  - **Output:** size of the corresponding cluster.

  This function returns the size of a given cluster, means its number of elements.
- $Indice_{SSE}$, $Indice_{ID}$ are two variables in which we save, respectively, the highest SSE and ID scores.

5 Simulation experiment and analysis

In this section, several experiments are carried out to compare SIK-prototypes with firstly proposed IK-prototypes [18] as well as with conventional k-prototypes method. Indeed, there does not exist prior work for comparison that tackles the triplet incremental object, attribute and class learning spaces when dealing with mixed data streams using the k-prototypes algorithm.

5.1 Simulation environment

We have implemented our proposed Split Incremental K-prototypes algorithm using python 3.7 language and executed on an Intel i5 processor (2.5 GHz) with 4 GB memory running on windows 7 operating system. In order to evaluate the effectiveness and the potency of our algorithm, we used the proposed SIK-prototypes algorithm and additionally the conventional k-prototypes algorithm to cluster the following six real-world mixed data sets: Credit Approval, Abalone and Chess (King-Rook vs. King) data sets derived from U.C.I repository [26], German Credit and sf-police-incidents data sets imported from openML [27] and finally Used Cars data set derived from Kaggle.[1] Table 1 illustrates a summary details of the used real mixed data sets.

- **German Credit (GC):** intends to classify German people according to their credits. People could be arranged as owing good or bad credit risks.
- **Credit Approval (CA):** concerns credit card applications in which all attribute names and values have been changed to meaningless symbols in order to protect the confidentiality of the data.
- **Chess (KRK):** is a chess end-game data set for white king and rook against black king. It was generated by Michael Bain and Arthur van Hoff at the Turing Institute, Glasgow, UK.
- **sf-police-incidents (SFPI):** is about incident reports from the San Francisco police department between January 2003 and May 2018.
- **Abalone (Ab):** is a real mixed data set which concerns the prediction of the age of abalone from physical measurements.
- **Used Cars (UC):** is a data set which includes every used vehicle entry within the United States on Craigslist, the world's largest collection of used vehicles for sale.

Taking as an example the GC data set, composed of 690 instances and 15 attributes. To start with, we introduce an input data composed of 400 objects and 10 attributes. Then, a data stream came to join the learning process with 290 instances and 15 attributes (10 old attributes and 5 new mixed ones). Conceptually, our proposed SIK-prototypes method keeps learning these new added samples. In fact, it is supposed to apply the conventional k-prototypes algorithm only on this recently joined data stream, escorted with new mixed features. Then, knowledge from both models will be merged until reaching the initial number of clusters so that to achieve both incremental object and attributes learning tasks. Finally, if the expert demands, the split procedure will be applied in order to establish the incremental class learning task. Indeed, according to the mentioned criteria in Sect. 4.2, one chosen cluster will be split in two clusters such that each similar points will be joined together in one cluster. Therefore, more coherent clustering results are made up.

5.2 Performance evaluation criteria

While assessing the performance of an unsupervised clustering algorithm is not as trivial as counting the accuracy, the precision and the recall of a supervised classification algorithm, several metrics were disposed to be used in this context [28]. Thus, we evaluate the efficiency of our proposal using indices that estimate the cluster cohesion (within or intra-variance) and the cluster separation (between or inter-variance) and combine them to compute a quality measure, in which the combination is performed by a division (ratio-type indices) or a sum (summation-type indices). More precisely, we have chosen to examine our final clustering model based on indices that are responsible to estimate the effectiveness of our proposal in an unsupervised clustering context where data streams are forthcoming with unlabeled data objects.

[1] https://www.kaggle.com/austinreese/craigslist-carstrucks-data.

Table 1 Summary details of used real mixed data sets

Data set	Number of object	Number of attributes	Acronym
Used cars	150	3 numeric, 3 categorical	UC
Credit approval	690	5 numeric, 10 categorical	CA
German credit	1000	7 numeric, 13 categorical	GC
Abalone	4177	7 numeric, 1 categorical	Ab
Chess	28056	2 numeric, 4 categorical	KRK
sf-police-incidents	538638	4 numeric, 3 categorical	SFPI

Also, we have attributed an up or down arrow to each abbreviation in which the down arrow denotes that a lower value of that index reflects a better partition between clusters and/or the lower index is better.

1. The Sum of Squared Error (SSE $\downarrow$): it gives a feedback about the inter-cluster and the intra-cluster similarity while measuring the squared errors between each instance and its closest cluster center. It is defined by Eq. (10).
2. The Davies–Bouldin Index (DB $\downarrow$): it calculates the average similarity between clusters. Conceptually, a lower DB value relates to a model with better defined clusters. DB is calculated using Eq. (5)
3. The run time (RT $\downarrow$): Starting from the beginning of the learning procedure, the run time is the time required to achieve the final clustering result: objects are distributed between k clusters.

In fact, while the SSE measure highlights the better distribution between clusters with calculating both similarities between and within clusters, the DB comes to emphasize and accentuate the SSE results. This is explained by the fact that the DB also computes the average similarity between clusters. Accordingly, these two evaluation measures are mutually complementary to asses the quality of the final clustering result. On the other hand, the run time is essential to estimate the time needed to incrementally learn these streaming data.

5.3 Results analysis and discussion

In this subsection, we go forward to detail our experimental results and further discuss and analyze them. So, we compare the performance of the proposed SIK-prototypes method versus the conventional k-prototypes method and also versus the earlier proposed IK-prototypes method in [18]. Actually, in the present literature, there does not exist proposed methods to compare with that tackle the triplet incremental object, attribute and class learning spaces at the same deal. In greater detail, our proposed SIK-prototypes algorithm is able to deal with continuously emerging mixed data streams incrementally without retraining from the scratch and by accomplishing the incremental attribute and instance tasks. Furthermore, whenever it is necessary and the expert asks to skip from k to k+1 clusters (as detailed in Sect. 4.2), it is achievable to proceed using the SIK-prototypes method with applying the split procedure. The advantage of our proposal is the ability to learn incrementally new mixed features with assuring the incremental attributes, object and class learning tasks without retraining all the model from the scratch and without forgetting the old data. Contrary to the conventional k-prototypes method that requires to retrain from the scratch once new chunks of data become accessible over time, with new added attributes than the old set ones. As a consequence, this latter necessitates to firstly stop the program, then to fuse the initial input data and the new emerging data stream with its new added attributes and finally to retrain from scratch with re-applying the k-prototypes algorithm with number $k + 1$ of clusters. Therefore, it is crucial and essential to provide the memory and time restrictions to do so because the conventional k-prototypes method needs the complete input data being loaded into the memory. Thus, it highly requires memory space and time-consuming. We used the real mixed data sets (illustrated in Sect. 5.1) in this experiment and applied the SIK-prototypes, the IK-prototypes [18] and the conventional k-prototypes methods on them. The results are reported in Table 2

5.3.1 Run-time results analysis

Table 2 exhibits the results of comparison between the conventional k-prototypes, the IK-prototypes and the SIK-prototypes algorithms based on the previously mentioned evaluation criteria. From above, the run-time experiment results mainly shows the duration rounds from the beginning of the learning process until reaching the final clustering results. Looking at Table 2, we can observe the progression of our proposed SIK-prototypes in terms of time processing compared to the conventional k-prototypes and to the IK-prototypes algorithms since the time required for its execution is almost less by half than the time needed for the

Table 2 The SSE, DB index and run time (in seconds) values of Conventional K-prototypes vs. Incremental K-prototypes vs. Split Incremental K-prototypes methods per data set

Data sets	Conventional k-prototypes			Incremental K-prototypes			SIK-prototypes		
	SSE	DB	RT	SSE	DB	RT	SSE	DB	RT
UC	2.673	2.437	1.621	1.981	2.032	1.131	1.750	1.834	0.970
CA	4.620	3.006	10.127	4.498	2.541	5.237	3.959	2.716	6.777
GC	2.902	2.860	20.765	2.477	2.701	11.248	1.887	2.450	12.595
Ab	2.611	0.818	60.034	1.980	0.791	13.061	2.232	0.745	41.912
KRK	3.281	2.982	450.119	3.022	2.646	2.672	3.020	2.300	327.010
SFPI	4.560	3.640	15728.130	3.514	2.534	10807	3.620	2.561	9954.520

k-prototypes execution. The offset here is proportional to the size of the data set. Likewise, the IK-prototypes outperforms the conventional k-prototypes in terms of time needed for its implementation. We mention as an example, the GC data set where the time needed for the k-prototypes, the IK-prototypes and the SIK-prototypes are respectively 20.765 s, 11.248 s and 12.595 s. Besides, for the CA data set, the SIK-prototypes method needs time execution 6.777 s. However, the conventional k-prototypes and the IK-prototypes methods end up their learning process through 10.127 s and 5.237 s. What is important to mention here is that the firstly proposed IK-prototypes deals only with attribute and object learning tasks. While our proposed SIK-prototypes tackles the triplet incremental attribute, object and class learning tasks at the same deal based on the proposed split function. Hence, we can admit that the run time criterion emphasizes our proposal's scalability and speed of convergence. Add it to that, as this is a continual learning approach, we can observe the progression on the performance measures as well as memory consumption through time of our new proposed SIK-prototypes.

5.3.2 Sum of squared error and Davies–Bouldin index results analysis

Actively, we highly prioritize engaging our research to show how it capes with the weaknesses of the conventional k-prototypes clustering algorithm as well as with the IK-prototypes method. Therefore, we detailed the experimental results of the SSE and DB index criteria for all used data sets in Table 2. More to the point, we are looking for the lowest SSE and DB index scores, which lead to better defined models. For better clarification, we cite for instance the UC data set where the SSE values move from 2.673 to 1.981 to 1.750 and the DB values go from 2.437 to 2.530 to 1.834 using respectively the conventional k-prototypes, the IK-prototypes method and our proposed SIK-prototypes method. Subsequently, let us acknowledge that the SSE and DB index values also enhance our proposed SIK-prototypes effectiveness and capability of handling the three dimensional incremental spaces. As well, the different

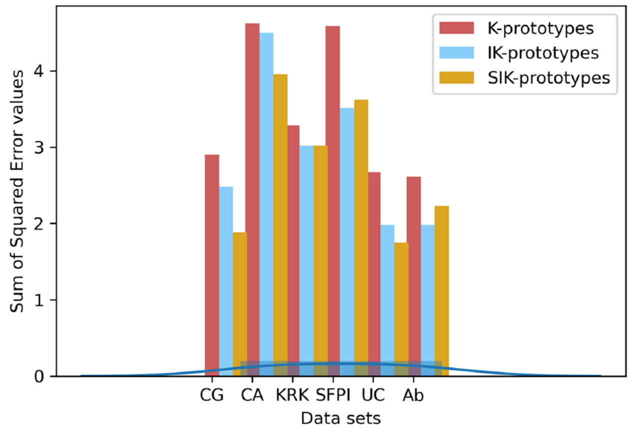

Fig. 3 The SSE results of Conventional K-prototypes vs. IK-prototypes vs. SIK-prototypes per data set

ranged results of correspondingly SIK-prototypes and IK-prototypes are explained by the fact that they do not tackle the same deal. Thus, we can not prioritize one instead of another since the IK-prototypes tackles only the incremental instance and feature spaces. Whereas our new proposed SIK-prototypes tackles the incremental instance and feature and class spaces at once.

From the results exposed in this Table 2, we can notice that the SIK-prototypes outperforms the conventional k-prototypes method in terms of stability of clusters while providing the lowest SSE values for all used data sets. Figure 3, that is reproduced from Table 2, underlines the fact that our proposed SIK-prototypes outperforms the conventional k-prototypes as well as the IK-prototypes with establishing a model with better defined clusters. In fact, it the gained lowest SSE scores, knowing that the lower SSE value is, the maximum coherence within clusters and the minimum similarity between clusters are.

Moreover, the DB index results, detailed in Table 2, confirm that our proposed SIK-prototypes method outperforms the conventional k-prototypes method since it gained the smallest DB index scores and further highlight our proposal's relevance in incremental attribute, object and class learning contexts. Likewise, as depicted in Fig. 4, SIK-prototypes approach represents the most appropriate feature, object, and

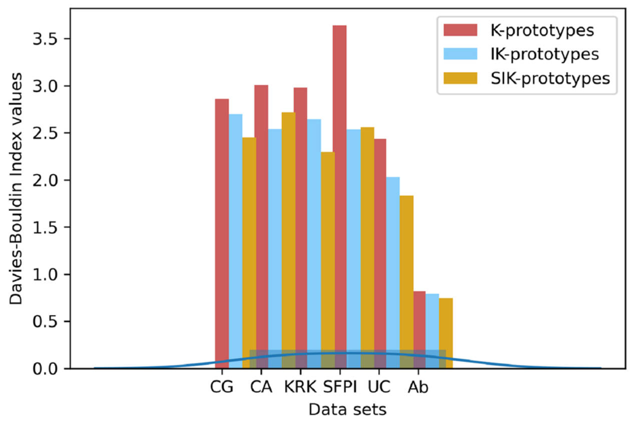

Fig. 4 The DB index results of conventional K-prototypes vs. IK-prototypes vs. SIK-prototypes per data set

class learning approach in terms of defining a more coherent model with higher similarity within clusters while owing the smallest DB index values. As well, this latter relates to a model with better separation between the clusters and with more similarity within clusters.

All in all, our proposed Split Incremental k-prototypes method emits clusters which respect better basics of clustering with regard to the dispersion of objects between and within clusters than the conventional k-prototypes clustering algorithm and sometimes than the IK-prototypes method. Besides, the measured evaluation criteria accentuate the efficiency and the performance of the new proposed SIK-prototypes method when applied to mixed large scale data compared to conventional k-prototypes method. In fact, SIK-prototypes needs less run time as well as less memory consumption to get final results because it is a continual and incremental learning approach. Means that SIK-prototypes does not exhibit to store the whole input data from the beginning of the learning process. Also, according to the SSE and DB index results, SIK-prototypes leads to a model with increased similarity within clusters in such dynamic object, attribute and class learning space. In the same way, our proposal exceeds the k-prototypes algorithm with generating a better partition of clusters where elements of the same cluster are the most similar and elements from two different clusters are as dissimilar as possible.

6 Conclusion

In this paper, we have developed a new incremental clustering method for large-scale mixed data, arriving continuously as data streams, using the k-prototypes algorithm based on the split technique. Our proposal provides a simple, transparent and efficient environment to ensure the dynamic incremental object, mixed attribute and class learning spaces at the same deal. The experiment results confirm that our method is scalable and able to upgrade the efficiency of existing k-prototypes methods when dealing with mixed streaming data.

Encouraged by such promising results, we leave for future work to tackle the decremental class learning task and further to define when it is necessary to do so. Another direction for future work is to extend the application of the presented SIK-prototypes method to the medicine and healthcare fields.

Data availability The data used in the experimentation section are open source and derived from U.C.I repository [26], openML [27] and Kaggle data sets https://www.kaggle.com/austinreese/craigslist-carstrucks-data.

Declaration

Conflict of interest Authors have no conflict of interest to declare.

Ethical approval: This article does not contain any studies with human participants performed by any of the authors.

Informed consent Informed consent was obtained from all individual participants included in the study. This manuscript is the authors' original work and has not been submitted simultaneously elsewhere. All authors have checked the manuscript and agreed to the submission.

References

1. Anderlucci, L., Fortunato, F., Montanari, A.: High-dimensional clustering via random projections. J. Classif. 1–26 (2021)
2. Bhagat, A., Kshirsagar, N., Khodke, P., Dongre, K., Ali, S.: Penalty parameter selection for hierarchical data stream clustering. Proc. Comput. Sci. **79**, 24–31 (2016)
3. Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., Carvalho, A.C.D., Gama, J.: Data stream clustering: a survey. ACM Comput. Surv. **46**(1), 1–31 (2013)
4. Chefrour, A.: Incremental supervised learning: algorithms and applications in pattern recognition. Evol. Intel. **12**(2), 97–112 (2019)
5. Sowjanya, A.M., Shashi, M.: Cluster feature-based incremental clustering approach (CFICA) for numerical data. Int. J. Comput. Sci. Netw. Sec. **10**(9), 73–79 (2010)
6. Lamirel, J. C., Mall, R., Ahmad, M.: Comportement comparatif des méthodes de clustering incrémentales et non incrémentales sur les données textuelles hétérogènes. In: 11th International Francophone Conference on Knowledge Extraction and Management (EGC 2011) (2011)
7. Sowjanya, A.M., Shashi, M.: A cluster feature-based incremental clustering approach to mixed data. J. Comput. Sci. **7**(12), 1875 (2011)
8. Noorbehbahani, F., Mousavi, S.R., Mirzaei, A.: An incremental mixed data clustering method using a new distance measure. Soft. Comput. **19**(3), 731–743 (2015)
9. Shen, F., Hasegawa, O.: A fast nearest neighbor classifier based on self-organizing incremental neural network. Neural Netw. **21**(10), 1537–1547 (2008)
10. Aggarwal, C.C., Philip, S.Y., Han, J., Wang, J.: A framework for clustering evolving data streams. In: Proceedings 2003 VLDB Conference. Morgan Kaufmann, pp. 81–92 (2003)

11. Ghesmoune, M., Lebbah, M., Azzag, H.: State-of-the-art on clustering data streams. Big Data Anal. **1**(1), 1–27 (2016)
12. Ackermann, M.R., Märtens, M., Raupach, C., Swierkot, K., Lammersen, C., Sohler, C.: Streamkm++ a clustering algorithm for data streams. J. Exp. Algorithmics **17**, 2–1 (2012)
13. Amini, A., Wah, T.Y., Saboohi, H.: On density-based data streams clustering algorithms: a survey. J. Comput. Sci. Technol. **29**(1), 116–141 (2014)
14. Ounali, C., Ben Rejab, F., & Nouira Ferchichi, K.: Incremental algorithm based on split technique. In: International Conference on Intelligent Systems Design and Applications. Springer, Cham, pp. 567–576 (2018)
15. Bao, J., Wang, W., Yang, T., Wu, G.: An incremental clustering method based on the boundary profile. PLoS ONE **13**(4), e0196108 (2018)
16. Savaresi, S.M., Boley, D.L., Bittanti, S., Gazzaniga, G.: Cluster selection in divisive clustering algorithms. In: Proceedings of the 2002 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, pp. 299–314 (2002)
17. Marszałek, Z.: Performance tests on merge sort and recursive merge sort for big data processing. Technical Sciences/University of Warmia and Mazury in Olsztyn (2018)
18. Gorrab, S., Rejab, F.B.: IK-prototypes: incremental mixed attribute learning based on K-prototypes algorithm, a new method. In: International Conference on Intelligent Systems Design and Applications. Springer, Cham, pp. 880–890 (2020)
19. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Disc. **2**(3), 283–304 (1998)
20. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, no. 14, pp. 281–297 (1967)
21. Huang, Z.: A fast clustering algorithm to cluster very large categorical data sets in data mining. Dmkd **3**(8), 34–39 (1997)
22. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE Trans. Pattern Anal. Mach. Intell. **2**, 224–227 (1979)
23. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Commun. Stat. Theory Methods **3**(1), 1–27 (1974)
24. Xu, R., Wunsch, D.C.: Clustering algorithms in biomedical research: a review. IEEE Rev. Biomed. Eng. **3**, 120–154 (2010)
25. Clarke, K.R., Chapman, M.G., Somerfield, P.J., Needham, H.R.: Dispersion-based weighting of species counts in assemblage analyses. Mar. Ecol. Prog. Ser. **320**, 11–27 (2006)
26. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
27. Vanschoren, J., Van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. ACM SIGKDD Explor. Newsl. **15**(2), 49–60 (2014)
28. Guo, S., Dong, X.L., Srivastava, D., Zajac, R.: Record linkage with uniqueness constraints and erroneous values. Proc. VLDB Endow. **3**(1–2), 417–428 (2010)