International Journal of Computer Information Systems and Industrial Management Applications. 149
ISSN 2150-7988 Volume 13 (2021) pp. 149-159


Received: 23 Jan 2021; Accepted: 21 May 2021; Published: 5 August 2021

# Incremental-Decremental Attribute Learning Algorithm Based on K-prototypes for Mixed Data Stream Clustering

Siwar Gorrab[1] and Fahmi Ben Rejab[1]

[1]Universit*é* de Tunis, Institut Sup*é*rieur de Gestion de Tunis,
BESTMOD, 41 Avenue de la Liberté, 2000 Le Bardo, Tunisie
*siwarg9@gmail.com; fahmi.benrejab@gmail.com*

*Abstract*: In the view of the high requirement of memory space, the long execution time and the low execution efficiency of the k-prototypes algorithm in large-scale training samples, this paper puts forward a new online incremental and decremental learning algorithm based on k-prototypes in order to deal with mixed data sets. For deep understanding, this study provides a better insight into one particular machine learning algorithm the incremental attribute learning based on unsupervised clustering algorithm and tackles also the decremental attribute learning task while being crucial and essential in data stream context. Indeed, this entering mixed data is either escorted with new attributes, or incorporates less attributes than the old ones. So, it might be processed sequentially over flexible time windows, in less time consuming and with providing a well-defined model with better separation between clusters. The efficiency of our proposed online learning algorithm is proved across the experimental results based on different evaluation criteria and numerous simulations made on real mixed data sets. *Keywords*: Clustering, data stream, mixed attributes, k-prototypes, incremental-decremental attribute learning, merge.

## I. Introduction

With the progress of artificial intelligence and machine learning technologies, systems with acceleration of the decision-making process are accessible in today's companies. This makes it challenging in different machine learning algorithms in terms of analyzing massive data sets faster and more accurately. At present, data accessibility is not such a trivial deal since it is continuously generated by thousands of data sources. Indeed, these sequences of data sample's derived are called data streams, which incorporate a wide variety of elements of both numerical as well as categorical types. In order to collect the most useful hidden information from these mixed data, we have opted to use the k-prototypes clustering algorithm that is, by definition, able to handle mixed data sets.

Indeed, this continuously emerging data streams might occur with evolving features: some of them have disappeared and others are improved in number. More to the point, at any time, and all long the learning task, new instances including new incoming features different from the old set ones or disappeared features might emerge as data stream. Here, two scenarios might be appealed: either to retrain from the scratch at the arrival of these new data or just to ignore the old data and only consider the new one. Still, both of these scenarios are not the best procedures owing to the fact that the former would be a time-consuming since a relearning from the scratch will take place each time new incoming data emerges in terms of features and instances, the latter would lead to lose knowledge due to the focus only on new chunks of data and neglecting the old set. In this area, the incremental learning represents an interesting alternative, another approach to handle with dynamic feature and instance spaces. Online metric learning has been widely exploited for large-scale data classification thanks to its low computational cost. However, amongst online practical scenarios where the features are evolving (e.g., some features are vanished and some new features are augmented), most metric learning models cannot be successfully applied into these scenarios although they can tackle the evolving instances efficiently [2]. Thus, an increasing need to the incremental-decremental feature learning concept is illustrated by the problems' nature in which data are not available at one batch. For instance, you are a marketing consultant, and you worry about your customer's behavior after a marketing campaign, when new incoming clients' feedback is available. You might be a doctor and you handle dynamic space when all the time new biological parameters have to be tracked, new analysis have to be done in order to ensure best patient's follow up. Although learning with such incremental and decremental feature space is tough and crucial, it is a rarely studied task, particularly when handling mixed data streams. Thus, we devote this study to propose an incremental-decremental attribute learning algorithm, based on k-prototypes so that to handle mixed data. In other words, we intend to develop a new version of our firstly proposed IK-prototypes [14] that has been proposed to learn incrementally new joined instances escorted with new features and not decrementally. The goal here is to adapt a pre-trained model to the changes in the training data set, without retraining the model from the scratch, where the changes can include addition and deletion of attributes. To our best knowledge, this is the first attempt to tackle this crucial, but rarely-researched challenge in the evolving feature and instance learning field. The rest of the paper is organized as follow. In Section 2, the k-prototypes algorithm is introduced and we illustrate some related works. The extension of our firstly proposed IK-prototypes is introduced as our proposes approach in Section 3. We devote Section 4 to display the

experimental simulations and results on four real mixed data sets. Finally, we conclude this paper in Section 5.

## II. K-prototypes Algorithm and Related Works

### *A. Theoretical concepts of k-prototypes algorithm*

Data streams are everywhere and are generated by numerous applications. Thus, we can admit that one of the main sources of big data can be the streaming data. In fact, big data usually refers to the following three major characteristics: Volume, Variety and Velocity. Particularly, the Volume refers to the large scale data, Variety indicates the mixed types of data such as numerical, categorical and text data and Velocity refers to the speed at which data is generated and processed [3]. Big data is often characterized by the mixed types of data including numeric and categorical. Into the bargain, k-prototypes algorithm is one of the most prominent clustering methods to deal with such mixed data.

#### *1) Formulation*

Proposed by Huang in [4], the k-prototypes algorithm integrates both k-means [5] and k-modes [6] algorithms. It is more useful and practical for clustering mixed data types where its main objective is to group the data set $X=\{x_1...x_n\}$ of $n$ data points containing $m_r$ numeric attributes and $m_t$ categorical attributes into $k$ different clusters while minimizing the following cost function

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

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 $j$ ; $c_j \in C = \{c_1...c_k\}$ is the center of the cluster j and $d(x_i - c_j)$ is the dissimilarity measure defined as follows:

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

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 in the following way:

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

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 $j$, calculates as follows:

$$c_{jt} = a_{ht} \quad \textbf{where } f(a_t^h) \succeq f(a_t^z),\ \forall z\ ,\ 1 \le z \le m_c \qquad (4)$$

having $a^z{}_t \in \{a^1{}_t...a^{m}{}_t{}_c\}$ is the categorical value $z$ and $m_c$ is the number of categories of categorical attribute $t$; $f(a^z{}_t) = |x_{it} = a^z{}_t | p_{ij} = 1|$ 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 $p \neq q$.

#### *2) Algorithm*

The process of the conventional k-prototypes algorithm is detailed as follows:

**Algorithm 1** K-prototypes clustering algorithm

1: Input: X: data set, k: number of clusters
2: Output: Cluster centers
3: Begin
4: Select randomly *k* initial cluster centres 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 centres after each allocation using Equations (3) and (4).
7: If the updated cluster centers are identical to the previous ones then terminate, otherwise, return to step 5.
8: End.

Nonetheless, this conventional clustering method is not suitable when dealing with large scale data (streaming data). This is due to its high computational cost and high requirement of memory space when storing and processing all input data in the memory. Besides, it is unable to handle incremental and/or decremental attribute learning task.

### *B. Related works*

#### *[1] ) Incremental Attribute Learning Algorithms*

As being a feasible machine learning strategy for solving high-dimensional pattern classification problems, incremental attribute learning (IAL) algorithms train gradually input attributes one by one [7]. This critical task has been proposed to avoid retraining from the scratch when handling dynamic feature space. Heretofore, IAL algorithms have been applied for solving multiple supervised learning algorithms in terms in terms of new entering attributes, arriving when new data stream takes place. For instance, support vector machines [8] where this proposal keeps learning incrementally new structural parameters from the newly added features and takes advantage of the use of a historic structural parameters trained primarily. Also, IAL was applied in neural networks where a new sub-network is constructed and trained incrementally while retraining the old sub-network and finally. Same for decision trees [10] with a primary offline construction of a decision tree then an incremental online revision of this constructed tree and genetic algorithms [11] with copying the old chromosomes into a new recent one for which new attributes are added then combine both of them in reference to a class matching mechanism. Additionally, bayesian classifier [12]: when features are progressively entering, the probability density is updated during the training step, and so on. Add it to that, a multichannel one-dimensional convolutional neural network-based feature learning for fault diagnosis of industrial processes has been proposed in [13] to investigate feature learning from high-dimensional process signals. In particular, Wavelet transform is employed to decompose the one-dimensional process signal into multi-scale components according to time and frequency information [13]. Last but not least, a novel feature incremental learning method for sensor-based activity recognition Despite that, IAL has not been

widely applicable for numerous clustering algorithms. Actually, we have recently proposed and incremental k-prototypes algorithm that tackles both incremental attribute and object learning at the same deal [14].

*2) Incremental and Decremental Feature Learning Algorithms*

Learning with such incremental and decremental features and evolving instances simultaneously might be considerate as a tough and crucial task but it is rarely studied [15]. In point of fact, few are the literature that discusses incrementa/decremental algorithms which treat dynamic feature and object spaces all at once. Some related works have been proposed to solve part of this problem which consists in addressing the instance evolving problem. From these proposed online instance learning algorithms, we might cite [16]-[17][18]-[19]. On the other hand, there are some researches concerning missing and corrupted features, proposed to solve feature incremental and decremental problem such as [20][21]. In [15], a one-pass learning method with incremental and decremental features has been proposed and proved that it can tackle the mentioned problem effectively. Furthermore, the study in [2] proposes an online evolving metric learning (EML) model for incremental and decremental features to tackle both instance and feature evolution's simultaneously. Mainly, they present two stages for both feature and instance evolutions, i.e., the Transforming stage (T-stage) and the Inheriting stage (I-stage), which can not only make full use of the vanished features in the T-stage, but also take advantage of streaming data with new augmented features in the I-stage [2].

Nevertheless, these methods can not achieve the requirements arisen from the above mentioned problem despite the notable achieved performances in their learning settings because they do focus only on one aspect of attribute and object evolution problem. Also, these problems are suitable only for supervised classification problems. So, here is a requirement to promote the unsupervised learning algorithms with such incremental and decremental attribute and object spaces for mixed data stream clustering.

## III. Proposed Online Learning Approach

This section introduces our proposed online incremental and decremental attribute learning algorithm to solve effectively the conventional k-prototypes problem with evolving mixed attributes, joined with data streams.

### *A. Online incremental-decremental k-prototypes approach*

Once new instances become available over time, particularly when these new input patterns include, in addition to the old set of features, new ones and/ or eliminates some of them, our proposed incremental and decremental k-prototypes (IDKprototypes) approach is dedicated. In such cases, either retraining from the scratch or ignoring the old data on which an initial model has been built, and just consider the new chunk of data could take place, when based on conventional kprototypes algorithm. The former, means retraining from the scratch that could be considered as a time consuming task, since we are going to retrain one part of the data on which we have already learned. The latter could lead to loss of information, since we would ignore a part of data and focus only on the latest added one. Accordingly, finding a way to include the old knowledge which has already been extracted from the initial model based on the first available input data is in our focus of interest. So the main question here is how to include knowledge which has been already extracted from initial available chunk of data, once new other data stream which includes new and/ or less features appears.

Figure. 1 illustrates the different steps of the procedure for our proposed online incremental and decremental attribute learning algorithm based on IDK-prototypes approach. To start with, once a primarily model has been built based on some initially available input mixed data, and at the arrival of new and/ or less features included in the newly joined instances as data stream, we keep learning on the new samples. Afterwards, we inject the extracted knowledge from the initial model in our modified k-prototypes namely the IDK-prototypes so that it could deal with incremental and decremental attribute learning task without relearning from the scratch. Considering Figure. 1, we can notice that our procedure is decomposed in three steps. Firstly, an initial model would be built based on the initial input pattern. Consequently, the clustering result would be reached and the procedure could be finished at this level. Secondly, each time new data stream emerges with new instances which include new and/ or less added features, the information from the earliest built model would be incrementally injected in the IDK-prototypes. This latter would learn only on the newly joined data stream which include new and/ or less instances, escorted with evolving features (new or less ones), and inject each time the old information which has already been generated from the previous amount of data. To do so, the main difficulty was how to inject this knowledge and how to make our model aware about the previous data on which an initial model has already been constructed.

*B. Algorithm*

In the first place, once an initial input data is 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. Afterwards, new data objects with new and/ or features penetrate as clustering proceeds as well be a data stream. At this level, we move on to the second step of our proposed IDK-prototypes method that consists in applying the k-prototypes algorithm only on the newly joined data stream. Hence, we aim to create $k^0$ clusters using the same standard k-prototypes algorithm and then save the obtained results as a second knowledge from the data stream model. Finally, to make our point, we have proposed to apply a king of data cleaning which intends to fill the missing data values (the missing incoming attributes and objects joined with arriving data streams) and to correct inconsistent data. Otherwise stated, these transformers allow to clean our resulted data set by eliminating or replacing the missing values present in it. To do so, we have opted to use the SimpleImputer transformer by virtue of the fact that it is a simple, straightforward and effective baseline for imputing the data. Besides, it executes the mentioned task without requiring complicated operations. In particular, this transformer is capable of replacing these resulted missing values with statistic values namely the mean, the median or the most-frequent value of the corresponding containing column. As well, we can even replace them with a constant value. *Impute*

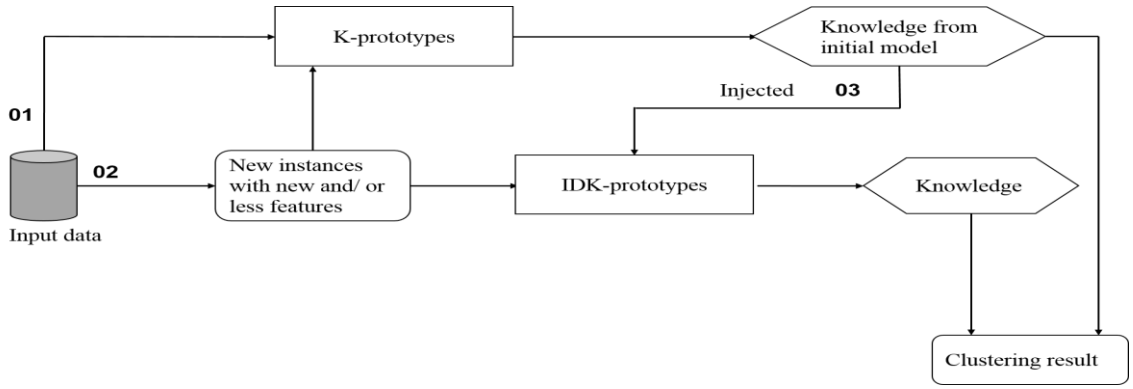



Figure. 1: An overview of the proposed procedure for online incremental and decremental attribute learning using IDK-prototypes

model aware about the primarily gained knowledge, we suggest to merge and fuse both knowledge from both initial and data stream models. Our proposed IDK-prototypes algorithm (Algorithm 2) works as follows:

**Algorithm 2** Incremental-decremental 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 centres) randomly from each emerging data stream *X*
5: Attribute each data point in *X* to its closest cluster center according to Equation (2).
6: Update the cluster centres after each allocation using Equation (3) and Equations (4).
7: If the updated cluster centers are identical to the previous ones then terminate, otherwise, return to step 5.
8: If a new data stream arrives having missing attributes and/ or escorted with new ones then return to step 5,
Merge the resulting clusters from initial and data stream models Impute missing values for all resulted merged clusters, otherwise, terminate.
9: End.

In fact, we have supplied our previously proposed IK-prototypes algorithm [14] with a novel property, that is to manage the decremental attribute and instance learning tasks in addition to its incremental attribute and object learning capability. Notably, we have proposed to impute the resulted merged clusters in such a way to avoid getting data with missing values as a result of the merge process. More to the *(merged-clusters):*

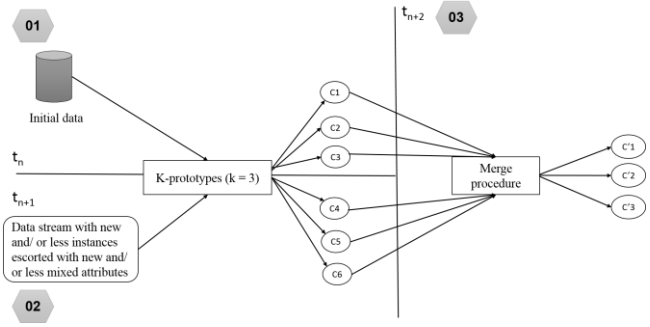
- Input: clusters with missed values, representing the evolving attributes and objects, joined with new data streams.
- Output: clusters without missed values.

Now and after having clarified the focal idea behind our proposed IDK-prototypes algorithm, we go forward to detail the main process which has been adopted to inject the initially gained knowledge from initially built model and to make our model aware about it.

*C. Merge procedure*

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, the merge technique was induced in clustering so that to avoid retraining from the scratch where some clusters will be updated by merging them with ones from the newly constructed clusters [22]. Additionally, merging techniques, that consists in joining likely elements of two sets of data together, has become more and more relevant since data generated at real-time has increased which makes the retraining of the result from the scratch difficult and a time consuming task [23]. Thus, we have opted to use this technique to create an incremental clustering procedure which leads to combine a set of clusters based on some specific criteria in order to end up with a resultant clusters that group the most likely starting ones. In

particular, our the merge algorithm takes as input $K = k+k'$ clusters and delivers as output the initial number of clusters $k$ including all similar elements of the input ones since we are dealing only with evolving object and attribute spaces and not class space. Put it differently, to ensure the




unsupervised incremental-decremental attribute and object learning tasks, the most two similar clusters resulting from initial and data stream models would be fused. For the sake of clarity, Figure. 2 below describes in details the workflow of the merge procedure.

By way of illustration, we take the example of a data set D1, used in our study in order to give a better insight to the workflow of the merge procedure. Subsequently, Table 1 shows the number of instances that contains each cluster of the 690 objects data set D1 with number $K = k+k' = 6$ of clusters.

Figure. 2: Workflow of the merge procedure

| Clusters | Number of instances (690) |
|---|---|
| Cluster 1 | 171 |
| Cluster 2 | 126 |
| Cluster 3 | 103 |
| Cluster 4 | 63 |
| Cluster 5 | 77 |
| Cluster 6 | 150 |

*Table 1*: Number of instances for each cluster of data set D1.

On that account, the similarity between clusters has been determined based on the following two measures:

1. Davies-Bouldin Index (DB) [24] It is the first merge criteria that calculates the average similarity between clusters. Particularly, the similarity measure here is based on a comparison between the distance between clusters and the size of the clusters themselves. The lower DB is, the better partition between clusters is. So, values closer to zero relate to a better partition of clusters. The DB similarity is calculated using Equation (5):

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

DB is defined as the average similarity between each cluster $C_i$ for $i = 1,...,k$ and its most similar one is $C_j$; $R_{ij}$ is the similarity measure. In the context of the DB index, a simple choice to construct $R_{ij}$ so that it is non-negative and symmetric is as follows in Equation (6):

$$R_{ij} = \frac{s_i + s_j}{d_{ij}} \quad (6)$$

Where $s_i$ is the cluster diameter, it represents the average distance between the point i/j and it's cluster centroid; $d_{ij}$ represents the distance between both cluster centroids i and j.

| | **C1** | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|
| C1 | 0.0 | **** | **** | **** | **** | **** |
| C2 | 1.431 | 0. 0 | **** | **** | **** | **** |
| C3 | 0.718 | 1.051 | 0.0 | **** | **** | **** |
| **C4** | **2.196** | 1.382 | 0.796 | 0.0 | **** | **** |
| C5 | 0.938 | 1.178 | 1.187 | 1.144 | 0.0 | **** |
| C6 | 0.689 | 0.953 | 1.597 | 0.776 | 1.464 | 0.0 |

The matrix below in Figure. 3 displays the degree of similarity between all the $K$ clusters, where the closest clusters coincide with the higher Davies-Bouldin index value. The first line and the first column of this matrix are cluster centres of the different six clusters.

2. Calinski-Harabasz Index (CH) [25] CH is the second merge criteria, also known as the variance ratio criterion, used to determine the ratio of the sum of between-clusters dispersion and of inter-cluster dispersion for all clusters. The dispersion here is certainly the Figure. 3: Davies-Bouldin index matrix for each two clusters of data set D1.

sum of the squared distances of each point to its closest cluster center. Contrary to the DB index, a higher CH score refers to a model with better defined clusters. The CH score is calculated as follows in Equation (7):

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

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; $tr(W_k)$ is the trace of the within-cluster dispersion matrix defined by:

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

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

knowing that $n_q$ is the number of points in cluster q; $c_q$ and $c_E$ are respectively the centers of the cluster q and of E. The Calinski-Harabasz score is high when clusters are dense and well separated which leads to the basic clustering concept. Given that, we will select the two clusters owing the lowest CH score as shown in Figure. 4, where the same six clusters used with the previous merge criteria are present.

| | **C1** | C2 | C3 | C4 | C5 | C6 |
|---|---|---|---|---|---|---|
| C1 | 10000 | **** | **** | **** | **** | **** |
| C2 | 102.047 | 10000 | **** | **** | **** | **** |
| C3 | 450.566 | 241.2 | 10000 | **** | **** | **** |
| **C4** | **33.759** | 95.732 | 316.75 | 10000 | **** | **** |
| C5 | 249.744 | 180.158 | 204.4 | 146.258 | 10000 | **** |
| C6 | 302.609 | 176.655 | 67.716 | 210.596 | 77.58 | 10000 |

Figure. 4: Calinski-Harabasz index matrix for each two clusters

Between those scores of the similarity measures that have been calculated, our algorithm will choose the clusters corresponding to the highest DB index and coinciding with the lowest CH score. Actually, indexes represented in bold in Figure. 3 as in Figure. 4 refer to the same couple of clusters that will be merged: cluster1 and cluster4. So, we can admit that we have successfully extracted the most two similar clusters where the two latter scores coincide, and then, merge these two selected clusters so that the incremental/decremental attribute learning tasks will be established. Proceeding in the same context until reaching $k$= 3 clusters, we have effectively avoided retraining from the scratch when new data stream joins with new and/ or less instances, escorted with newly added attributes and/ or some eliminated ones. In spite of that, once in a while the highest DB index and the lowest CH score may not be the best choices for the merge procedure if they result in different combination of clusters. In such a case, the merge algorithm will carry on merging both clusters resulting from the two calculated indexes and ends up with maintaining the resulting cluster, owing the lowest Sum of Squared Error (SSE) value. This latter measure computes the dispersion of elements in relation with their of a cluster centres, that is the sum of squared distances of samples to their nearest cluster center. By the same token, it is used as an evaluation criteria for clustering algorithms and its calculation is as follows in Equation (10) :

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

Where $c_j$ is the centroid of the cluster j.

The merge procedure, scheduled in our proposed IDK-prototypes is detailed in Algorithm 3. Now, that the merge algorithm has been presented, we move on to clarify some functions which have been used.

- *Compute (index):*
  - Input: Clusters.
  - Output: Davies-Bouldin index and CalinskiHarabasz score matrices.

  Clearly, the Compute (index) function is the basic and the key function for the merge algorithm owing to the fact that it determines the best couple of clusters to further be merged as long as it calculates the DaviesBouldin index and the Calinski-Harabasz scores, provided ultimately in the two symmetric matrices as dhown in Figure. 3 and Figure. 4. Each box of these matrices shows the measured DB index or the CH score between each two clusters. So, this is in sum the utility of the Compute (index) function.

- *Max (DB / CH), Min (DB / CH) :*
  - Input: Matrix of calculated indexes.
  - Output: Highest or lowest index.

  Max (DB/CH) and Min (DB/CH) functions are used in our proposed merge algorithm to search respectively about the highest and the lowest indexes from the input matrices. Accurately, the Max (DB/CH) function searches into Davies-Bouldin matrix for the largest score between two clusters. Unlike the Min (DB/CH) function that aims to retrieve the smallest CalinskiHarabasz score between each two clusters.

- *Merge (clusters):*
  - Input: cluster i, cluster j.
  - Output: cluster ij.

**Algorithm 3** Merge algorithm

```
1: Input: clusters = {c1,c2,c3,c4,c5,c6} 2:
Output: clusters = {c1', c2', c3'}
3: Begin
4:        For each cluster Ai in {c1,c2,c3} do
5:         For each cluster Bj in {c4,c5,c6} do
6:              DB ← computeDaviesBouldinIndex(Ai,Bj)
7:              CH ← computeCalinskiHarabaszIndex(Ai,Bj)
8:       end For
9:      end For
10:       For each cluster Ai in {c1,c2,c3} do
11:        For each cluster Bj in {c4,c5,c6} do
12:                 If computeDaviesBouldinIndex(Ai,Bj) = Max(DB) then
13:            Cluster1 ← Ai
14:            Cluster2 ← Bj
15:         end If
16: If computeCalinskiHarabaszIndex(Ai,Bj) = Min(CH) then
17:            Cluster3 ← Ai
18:            Cluster4 ← Bj
19:         end If
20:      end For
21:     end For
22:     If (Cluster1=Cluster3) and (Cluster2=Cluster4) then
23:         NewCluster ← Merge (Cluster1,Cluster2)
24:         Delete (Cluster1)
25:         Delete (Cluster2)
26:     end If
27:         SSE1 ← SSE(Merge(Cluster1,Cluster2)
28:         SSE2 ← SSE(Merge(Cluster3,Cluster4)
29:     If SSE1 < SSE2 then
30:         NewCluster ← Merge (Cluster1,Cluster2)
31:         Delete (Cluster1, Cluster2)
32:     Else
33:         NewCluster ← Merge (Cluster3,Cluster4)
34:         Delete (Cluster1, Cluster2)
35:     end If
36: Return clusters = {c1', c2', c3'}
37: End.
```

The Merge (clusters) function allows to combine and to fuse the most similar couple of clusters in such a way to result in one single cluster, which incorporates elements from both clusters. In other words, this major function consists in joining the likely elements of two sets of data together. Through the merge function, the extended number of attributes and instances as well as the eliminated objects and features, coming in the newly joined data stream are well treated in the learning process.

- *Delete(cluster):*
  - Input: clusters.
  - Output: removed clusters.

  After merging the most appropriate clusters, we need to remove each single one of them so that to avoid redundancy, given that the new created cluster contains the most likely elements from both of them.

- *NewCluster:*
  As indicated by its name, NewCluster is the cluster result from the merge procedure which incorporates instances from the most two similar clusters. Add it to that, the NewCluster here respects the focal objective of our approach: attributes and objects are incrementally and decrementally learned using our proposed IDKprototypes.

## IV. Experimentation Results and Discussion

In this section, we perform numerous experiments to evaluate the efficiency of our proposed IDK-prototypes, to test its performance and conclusively to validate it.

### *A. Experimental protocol*

With the purpose of evaluating the performance of our approach, comparisons between the conventional k-prototypes algorithm, our previously proposed IK-prototypes [14] and our actually proposed Incremental Decremental K-prototypes (IDK-prototypes) algorithm, were done based on real mixed data sets. Thereupon, we have developed this IDK-prototypes method in addition to those in which our program will be compared to using Python 3.7 a 64-bit computer with 6 GB memory, and through the powerful scientific environment Spyder as being an open source cross-platform integrated development environment (IDE). On top of that, we have implemented the proposed incremental-decremental clustering approach using four real mixed data sets for which we define its number of instances, number of attributes and acronym in Table 2. Likewise, the Chess and Credit Approval data sets are derived from the U.C.I machine learning repository [26]. Whereas, the German Credit and the sfpolice-incidents data sets are imported from openML [27].

| Data set | Instances | Attributes | Acronym |
|---|---|---|---|
| Credit Approval | 690 | 5 numeric, 10 categorical | Ca |
| German Credit | 1000 | 7 numeric, 13 categorical | Cg |
| Chess | 28056 | 2 numeric, 4 categorical | krk |
| sf-police-incidents | 538638 | 4 numeric, 3 categorical | SFPI |

*Table 2*: Description of used data sets.

- Credit Approval (Ca): This data It contains 690 objects described by 15 mixed attributes. It 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.
- German Credit (Cg): It is a 1000 instances data set that integrates 20 mixed attributes, which intend to classify German people according to their credits. People could be arranged as owing good or bad credit risks.
- Chess (krk): It is a Chess end-game data set for white king and rook against black king. Krk is composed of 28056 instances distributed within 6 mixed attributes. It was generated by Michael Bain and Arthur van Hoff at the Turing Institute, Glasgow, UK.
- sf-police-incidents (SFPI): This last used data set is about incident reports from the San Francisco police department between January 2003 and May 2018. It holds 538638 objects described by 7 mixed attributes.

### *B. Performance evaluation criteria*

The quality of our proposed method was measured using number of evaluation criteria as follow. Actually, they has been used to figure out to what extent our new clustering approach is relevant in unsupervised incremental and decremental attribute/ object learning context. Most of these indices 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) [28]. We have attributed a down arrow to each abbreviation since we are looking for lower scores which reflect a better partition between clusters.

1. The Davies-Bouldin Index (DB ↓): this index is usually one of the most used criteria to evaluate clustering results as it calculates the average similarity between clusters. So, a lower DB value relates to a model with better defined clusters. The DB index is calculated using Equation (5).
2. The Sum of Squared Error (SSE ↓): also called cluster inertia, the SSE counts the dispersion of a cluster elements' in relation with their cluster centre. It is well known as a fundamental criteria which provides a better insight about the inter-cluster (between clusters) and the intra-cluster (within clusters) similarity. The SSE is calculated as cited earlier in Equation (10).
3. The run time (RT ↓): it is the time demanded to attend the terminal results, starting from the beginning of the learning process until achieving the final clustering result: instances are distributed within *k* clusters.

### *C. Results analysis and discussion*

Aiming to investigate the capability of our proposed approach in incremental-decremental attribute learning context and to

show its proficiency, we proceed by studying the results obtained from performing different simulations experiments on our four real mixed data sets. Our study leads to expert the scalability of our proposed approach by not only exposing the experimental results but also by measuring the evaluation metrics devoted to evaluate our proposal. Actively, we highly prioritize engaging our research to show how it capes with the weaknesses of the batch k-prototypes clustering algorithm. Above all, the incremental/decremental aspect is detailed as follows: admitting that an initial model has been built based on the first available input data pattern (an initial sample of instances with a specific number of features) and once new instances escorted with new/ less added features take place as time proceeds, the new and the old knowledge from both constructed models will be merged using the IDK-prototypes. For the sake of clarity, our contributions is based on the merge algorithm where clusters, resulting from applying separately the k-prototypes algorithm on the input data as well as on the continuously joined data streams with more/ less attributes than the old set ones, are merged in such a way that leads to conserve the initial number of clusters k. It is explained with the fact that we are dealing with incremental and decremental attribute and objects spaces and not with evolving class learning space. At this level, we start by fixing the initial number of clusters *k* (*k* is equal to three in the following example) for both k-prototypes applications that will end up with six resulting clusters as input for the merge algorithm. By way of illustration, we cite the example of the Cg data set in which an initial data arrives at first with 15 attributes. Then, as time proceeds and while the learning process pursues, an arriving new mixed data stream emerges with new and less 17 joined attributes. In fact, 5 of these 17 attributes are newly added, 2 old ones (from the first available 15 ones) are eliminated and the old 13 attributes (arriving as input data) are maintained. All in all, this global data set arrives with 20 attributes. As mentioned in Section 3, the merge procedure follows up the subsequent steps:

1. Calculate the Davies-Bouldin index and the Calinski-Harabasz score for each cluster as shown in Figure. 3 and Figure. 4.
2. Identify clusters to be merged corresponding to the highest DB index and coinciding with the lowest CH score (see Figure. 3 and Figure. 4).
3. Merge the two selected clusters (the most similar ones).

After merging the most likely clusters, we end up our IDK-prototypes algorithm with imputing the missing values, present in the resulted merged clusters, using the SimpleImputer transformer that provides constant values or uses statistic values (median, mean,...). Specifically, we have chosen to replace these missing values with the calculated mean of each column in which the missing values are located. The following subsubsections are consecrated to expose the numerical results of our simulations in order to analyse and discuss them and to make comparisons between the batch k-prototypes, the Incremental k-prototypes (IK-prototypes) and the Incremental Decremental K-prototypes (IDK-prototypes) methods based on the multiple mentioned evaluation criteria.

*1) Comparison of Sum of Squared Error Results*

The following Table 3 exhibits the comparison between the k-prototypes, IK-prototypes and IDK-prototypes algorithms when fixing the number of clusters *k* to 3 and based on the sum of squared error (SSE) evaluation criteria. For further details, rows are dedicated to the SSE values obtained using respectively the three compared methods. For a better comprehension, Figure 5 that is reproduced from Table 3 and which illustrates the final SSE results, underlines the fact that our proposed IDK-prototypes outperforms the conventional k-prototypes algorithm as well as our previously proposed IK-prototypes method in terms of stability of clusters for all the used data sets since it has procured the lowest SSE values.

| Data sets | Cg | Ca | krk | SFPI |
|---|---|---|---|---|
| K-prototypes | 2.95 | 4.93 | 3.19 | 4.18 |
| IK-prototypes | 2.90 | 4.62 | 3.02 | 3.51 |
| IDK-prototypes | 1.82 | 3.49 | 2.82 | 3.25 |

*Table 3*: The SSE results of k-prototypes vs. Incremental k-prototypes vs. Incremental Decremental k-prototypes per data set.

For instance, according to the Cg data set, we observe that the IDK-prototypes method outperforms the k-prototypes as well as the IK-prototypes where SSE values are respectively 2.95, 2.90 and 1.82 (see Figure.5). Admitting that the lower SSE value is, the maximum coherence within clusters and the minimum similarity between clusters are, our proposed contribution, namely IDK-prototypes, exceeds both k-prototypes and IK-prototypes in terms of stability of clusters.

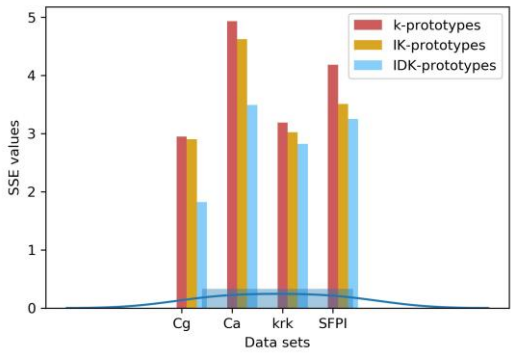



Figure. 5: The SSE results of k-prototypes vs. Incremental k-prototypes vs. Incremental Decremental k-prototypes per data set.

*2) Comparison of Davies-Bouldin Index Results*

In the next table (Table 4), we detail the calculated Davies-Bouldin index vales according to the diverse used data sets for the different three mentioned methods.

| Data sets | Cg | Ca | krk | SFPI |
|---|---|---|---|---|
| K-prototypes | 3.42 | 2.56 | 2.69 | 2.66 |
| IK-prototypes | 3.05 | 2.54 | 2.64 | 2.53 |
| IDK-prototypes | 2.44 | 2.19 | 2.43 | 2.32 |

*Table 4*: The DB index results of k-prototypes vs. Incremental k-prototypes vs. Incremental Decremental k-prototypes per data set.

We transcribe this summary Table 4 in the Figure. 6 for a better insight of the obtained DB results. Looking at this table (Table 4), we can confirm that our proposed IDK-prototypes method outperforms the k-prototypes method as long as it has also gained the smallest DB index values for the numerous used data sets. Thus, we can conclude that our proposed contribution represents the optimum feature learning approach since it generates a well-defined model which respects better basics of clustering in terms of elements' dispersion within and between clusters while owing the lowest DB index values (see Figure. 6). Over the exposed results in
Figure. 6, we take for instance the example of the Cg data

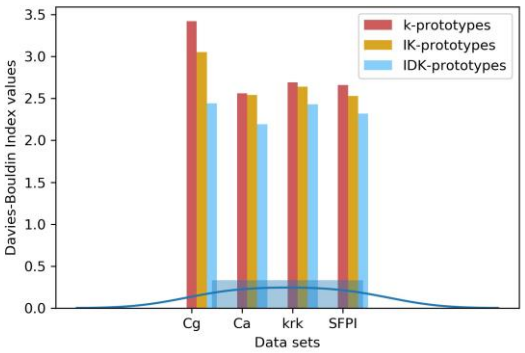



Figure. 6: The DB index results of k-prototypes vs. Incremental k-prototypes vs. Incremental Decremental k-prototypes per data set.

set in which the DB scores decelerate from 3.42 for the k-prototypes to 3.05 for the IK-prototypes and lastly to 2.44 for our proposed IDK-prototypes algorithm. Figure 6 does also confirm the prosperous results obtained by applying both IK-prototypes and IDK-prototypes and further highlights their relevance in incremental and decremental attribute learning context. As a matter, we notice that the DB index emphasizes our proposed method efficiency.

*3) Comparison of Run Time Results*

As being a crucial evaluation criteria for all cited methods, the run time (RT) in seconds of each used data set is provided in the next Table 5.

| Data sets | Cg | Ca | krk | SFPI |
|---|---|---|---|---|
| K-prototypes | 20.68 | 10.05 | 354 | 14094.60 |
| IK-prototypes | 11.24 | 5.23 | 267 | 10807.20 |
| IDK-prototypes | 10.09 | 4.82 | 227,10 | 9766,81 |

*Table 5*: The run time results of k-prototypes vs. Incremental k-prototypes vs. Incremental Decremental k-prototypes per data set.

Looking at Table 5 that outlines the run time of the three following algorithms k-prototypes, IK-prototypes and IDK-prototypes for all used data sets, we can notice the decreased results starting from the first row to the last one. Analyzing these outcomes, we can observe the progress of our proposed IDK-prototypes in terms of time processing regarding all compared methods, while presenting more promising results. For a better insight, we have chosen to present in the same figure (Figure. 7) the run time of the data sets with convergent number of instances, namely the Ca and Cg data sets. While the run time of the SFPI and the krk data sets are presented in Figure. 8.
With the reproduced results from Table 5, both Figure. 7 and Figure. 8 show that the time required for the IDK-prototypes algorithm execution is almost less by half than the time needed for the two other methods' execution. We can take as an example, the Cg data set where the time needed for the ordered three compared methods are respectively 20.68s, 11.24s and 10.09s. So, this is the reason why we can admit that the run time criterion emphasizes our

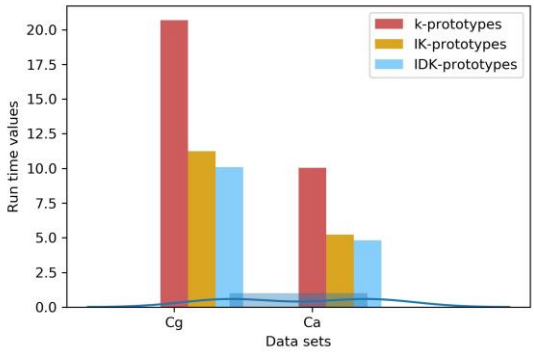



Figure. 7: The run time results of k-prototypes vs. Incremental k-prototypes vs. Incremental Decremental k-prototypes for the Cg and Ca data sets.

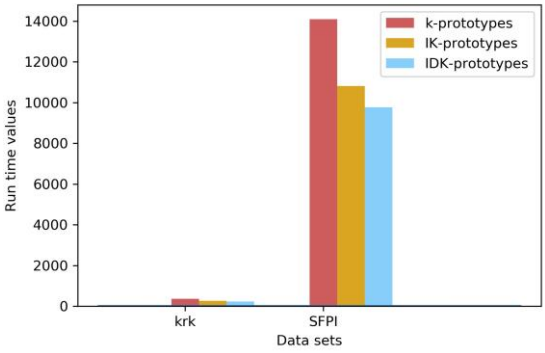



Figure. 8: The run time results of k-prototypes vs. Incremental k-prototypes vs. Incremental Decremental k-prototypes for the krk and SFPI data sets..

Method's performance. For all the rest of the RT results, the offset is proportional the size of the data set. Thus, we can conclude that our proposed IDK-prototypes method is the most appropriate one since it requires the lowest run time for its execution.

In summary, the IDK-prototypes does not only outperform our previously proposed IK-prototypes method, but also does generally outperform the conventional k-prototypes algorithm by providing a well-defined model with better separation between clusters and higher similarity within these resulted clusters. In fact, the IDK-prototypes needs less run time to get final results and according to the SSE values and DB scores, it leads to a model with better-defined clusters and which

respects more clusters notions where elements of the same cluster are the most similar and elements from two different clusters are as dissimilar as possible. In few words, the achieved results below confirm that the incremental and decremental attribute learning is an interesting alternative and constitutes one of the major concerns of the machine learning algorithms. Furthermore, performing data cleaning before modeling data reduces over-fitting since it minimizes the possibility to make decisions based on noisy data or missing data. Besides, it eases the learning process and speeds it up.

## V. Conclusion

In this paper, we have proposed the online incremental and decremental attribute learning algorithm based on the conventional k-prototypes algorithm for mixed data streams clustering. The proposed IDK-prototypes has taken full advantage of our firstly proposed IK-prototypes, suggested to learn incrementally new entering chunks of data escorted with newly joined features, and append it with the ability to deal with evolving attributes, means incremental and decremental attribute learning at the same deal. Our proposal does not need to retrain the entire data sets from the scratch after emergence of streaming data, provides a well-defined model which respects better basics of clustering and speeds up the learning process. The effectiveness of our approach has been confirmed through experiments. Our approach is specifically useful when both the data and features are evolving. Experimental results show that the merge algorithm is very effective to solve the problem with both mixed feature and instance evolution. Lastly, applying our proposed algorithm in multiple mixed data clustering fields is an interesting future work. Also, applying some king of data preprocessing techniques, namely feature selection, in such unsupervised incremental and decremental feature learning space and checking its effects would be attractive since it has not been evaluated so far.

## References


[1] HOU, Chenping et ZHOU, Zhi-Hua. One-pass learning with incremental and decremental features. *IEEE transactions on pattern analysis and machine intelligence*. IEEE, 2017, vol. 40, no 11, p. 2776-2792.

[2] DONG, Jiahua, CONG, Yang, SUN, Gan, et al. Evolving Metric Learning for Incremental and Decremental Features. *arXiv preprint arXiv:2006.15334*, 2020, vol. 14, no 8.

[3] HAJKACEM, Mohamed Aymen Ben, N'CIR, Chiheb-Eddine Ben, et ESSOUSSI, Nadia. One-pass MapReduce-based clustering method for mixed large scale data. *Journal of Intelligent Information Systems*. Springer, 2019, vol. 52, no 3, p. 619-636.

[4] HUANG, Zhexue. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data mining and knowledge discovery*. Springer, 1998, vol. 2, no 3, p. 283-304.

[5] MACQUEEN, James, et al. Some methods for classification and analysis of multivariate observations. *In : Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. 1967. p. 281-297.

[6] HUANG, Zhexue. A fast clustering algorithm to cluster very large categorical data sets in data mining. *DMKD*. Citeseer, 1997, vol. 3, no 8, p. 34-39.

[7] WANG, Ting, ZHOU, Wei, ZHU, Xiaoyan, et al. Integrated feature preprocessing for classification based on neural incremental attribute learning. *In : 2016 19th International Conference on Information Fusion (FUSION)*. IEEE, 2016. p. 386-393.

[8] LIU, Xinwang, ZHANG, Guomin, ZHAN, Yubin, et al. An incremental feature learning algorithm based on least square support vector machine. *In : International Workshop on Frontiers in Algorithmics*. Springer, 2008. p. 330-338.

[9] GUAN, Sheng-Uei et LI, Shanchun. Incremental learning with respect to new incoming input attributes. *Neural Processing Letters*. Springer, 2001, vol. 14, no 3, p. 241-260.

[10] CHAO, Sam, WONG, Fai, et LI, Yiping. An incremental and interactive decision tree learning algorithm for a practical diagnostic supporting workbench. *In : 2008 Fourth International Conference on Networked Computing and Advanced Information Management*. IEEE, 2008. p. 202-207.

[11] GUAN, Sheng-Uei et ZHU, Fangming. An incremental approach to genetic-algorithms-based classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. IEEE, 2005, vol. 35, no 2, p. 227239.

[12] AGRAWAL, R. K. et BALA, Rajni. Incremental Bayesian classification for multivariate normal distribution data. *Pattern Recognition Letters*. Elsevier, 2008, vol. 29, no 13, p. 1873-1876.

[13] YU, Jianbo, ZHANG, Chengyi, et WANG, Shijin. Multichannel one-dimensional convolutional neural network-based feature learning for fault diagnosis of industrial processes. *Neural Computing and Applications*. Springer, 2021, vol. 33, no 8, p. 3085-3104.

[14] Siwar Gorrab and Fahmi Ben Rejab: IK-prototypes: Incremental Mixed Attribute Learning based on Kprototypes Algorithm, a new method. *International Conference on Intelligent Systems Design and Applications (ISDA)*. Springer, 2020.

[15] HOU, Chenping et ZHOU, Zhi-Hua. One-pass learning with incremental and decremental features. *IEEE transactions on pattern analysis and machine intelligence*. IEEE, 2017, vol. 40, no 11, p. 2776-2792.

[16] ZHANG, Lijun, YANG, Tianbao, JIN, Rong, et al. Online bandit learning for a special class of non-convex losses. *In : Proceedings of the AAAI Conference on Artificial Intelligence*. 2015, vol. 29, no 1.

[17] HAZAN, Elad, AGARWAL, Amit, et KALE, Satyen. Logarithmic regret algorithms for online convex optimization. *Machine Learning*. Springer, 2007, vol. 69, no 2-3, p. 169-192.

[18] CHEN, Yuantao, XIONG, Jie, XU, Weihong, et al. A novel online incremental and decremental learning algorithm based on variable support vector machine. *Cluster Computing*. Springer, 2019, vol. 22, no 3, p. 7435-7445.

[19] LEE, Wei-Han, KO, Bong Jun, WANG, Shiqiang, et al. Exact Incremental and Decremental Learning for LSSVM. *In : 2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019. p. 2334-2338.

[20] HAZAN, Elad, LIVNI, Roi, et MANSOUR, Yishay. Classification with low rank and missing data. *In : International conference on machine learning*. PMLR, 2015. p. 257-266.

[21] DEKEL, Ofer, SHAMIR, Ohad, et XIAO, Lin. Learning to classify with missing and corrupted features. *Machine learning*, 2010, vol. 81, no 2, p. 149-178.

[22] MARSZAŁEK, Zbigniew. Performance tests on merge sort and recursive merge sort for big data processing. *Technical Sciences/University of Warmia and Mazury in Olsztyn*, 2018.

[23] YUAN, Man et SHI, Yong. Text clustering based on a divide and merge strategy. *Procedia Computer Science*. Elsevier, 2015, vol. 55, p. 825-832.

[24] DAVIES, David L. et BOULDIN, Donald W. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*. IEEE, 1979, no 2, p. 224-227.

[25] CALINSKI, Tadeusz et HARABASZ, Jerzy. A den-´ drite method for cluster analysis. *Communications in Statistics-theory and Methods*. Taylor & Francis, 1974, vol. 3, no 1, p. 1-27.

[26] LICHMAN,Moshe, et al. UCI machine learning repository, 2013.[Online]. Available: http://archive.ics.uci.edu/ml

[27] J.V. Rijn, 2014. [Online]. Available: https://www.openml.org/d/268

[28] ARBELAITZ, Olatz, GURRUTXAGA, Ibai, MUGUERZA, Javier, et al. An extensive comparative study of cluster validity indices. *Pattern Recognition*. Elsevier, 2013, vol. 46, no 1, p. 243-256.

## Author Biographies

Siwar Gorrab is a doctoral student at the Institut Superieur´ de Gestion de Tunis, university of Tunisia and a member of BESTMOD laboratory. She received her master degree in business computing. Her current research interests include real-time learning, big data, machine learning, data mining, and artificial intelligence. Contact her at: siwarg9@gmail.com

Fahmi Ben Rejab received the Ph.D degree in artificial intelligence and computer science from the Institut Superieur´ de Gestion de Tunis, university of Tunisia. He is an assistant professor at the ISGT and a member of BESTMOD Laboratory. In his researches, he is interested in data mining, artificial intelligence, health-care, real-time machine learning algorithms. Contact him at: fahmi.benrejab@gmail.com