New Gener. Comput. (2017) 35:391–416
DOI 10.1007/s00354-017-0018-y

SPECIAL FEATURE

WCDS: A Two-Phase Weightless Neural System for Data Stream Clustering

Douglas O. Cardoso[1] · **Felipe M. G. França**[2] · **João Gama**[3]

Received: 27 October 2016 / Accepted: 26 April 2017 / Published online: 30 May 2017

Abstract Clustering is a powerful and versatile tool for knowledge discovery, able to provide a valuable information for data analysis in various domains. To perform this task based on streaming data is quite challenging: outdated knowledge needs to be disposed while the current knowledge is obtained from fresh data; since data are continuously flowing, strict efficiency constraints have to be met. This paper presents WCDS, an approach to this problem based on the WiSARD artificial neural network model. This model already had useful characteristics as inherent incremental learning capability and patent functioning speed. These were combined with novel features as an adaptive countermeasure to cluster imbalance, a mechanism to discard expired data, and offline clustering based on a pairwise similarity measure for WiSARD discriminators. In an insightful experimental evaluation, the proposed system had an excellent performance according to multiple quality standards. This supports its applicability for the analysis of data streams.

Keywords Clustering · Data streams · Artificial neural networks · WiSARD

✉ Douglas O. Cardoso
douglas.cardoso@cefet-rj.br;
http://docardoso.github.io

Felipe M. G. França
felipe@cos.ufrj.br;
http://www.cos.ufrj.br/∼felipe

João Gama
jgama@fep.up.pt;
http://www.liaad.up.pt/area/jgama/

[1] Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, Campus Petrópolis, Petrópolis, RJ, Brazil

[2] Universidade Federal do Rio de Janeiro, PESC-COPPE, Rio de Janeiro, RJ, Brazil

[3] Universidade do Porto, LIAAD-INESC TEC, Oporto, Portugal

Introduction

Clustering is one of the most basic, important, and oldest tasks related to data mining and knowledge discovery. As long as data sets exist, summarized descriptions in the form of groups will be useful for their interpretation. However, since research on data mining began, data availability and access has changed: the local, static data sets now coexist with data streams, unbounded sources of transient data. Data stream clustering [19] can be accomplished incrementally updating the extracted knowledge as incoming data are processed. This should lead not only the expansion of the information already gathered but also the disposal of outdated knowledge. In context of clustering, this means to track the surge as well as the vanishing of clusters, besides changes of characteristics as positioning and shape. Besides those aspects, it is important to keep in mind that mining data streams impose stricter efficiency constraints compared to those considered while handling regular data sets.

This text presents henceforth an approach to data stream clustering, based on Wilkes, Stonham, and Aleksander Recognition Device (WiSARD) memory-based artificial neural network (ANN) model [2]. This model was chosen as the starting point of this research because: although it was introduced in 1984, recent works in the literature [13, 14, 20] provided a novel perspective of its features, which could favor the accomplishment of the targeted task; compared to other weightless models, its adaptation to feed from data streams appeared to be less cumbersome, assuring the proper functioning and original properties of the model while supporting incremental and decremental learning from temporal data; at last, the relatively low time complexity of WiSARD, one of its patent characteristics, could help to cope with high input rates during data stream processing, a common setting in real applications.

The fundamental idea of this adaptation of WiSARD to data stream clustering is to use discriminators as micro-cluster representatives. One premise of clustering is that there is no information about the true classes of input observations. WiSARD was conceived for classification, with its discriminators being responsible for modelling the classes to be recognized from their respective training examples. To adapt this classifier for clustering, the selection of training examples among unlabeled ones became part of the learning process. This way, any given discriminator would be fed with unlabeled data, but not before checking the compatibility between such data and the pattern represented by the discriminator. Table 1 shows how this modification would affect WiSARD operation in some hypothetical cases.

This idea is supported by some interesting facts: a discriminator is natively an incremental learner; there is no restriction to adding or removing discriminators, since they exist independently; WiSARD provides a richer feedback than just a distance to a decision boundary as discriminative classifiers, what enables decisions beyond class prediction [13]; and data abstraction units of various approaches to data stream clustering depict data samples based on statistics as mean and variance [1, 4, 5, 10], which can be considered less informative than the arbitrarily-shaped summary a discriminator can provide [20].

Table 1 Exemplified description of WiSARD regular and proposed (i.e., for classification and clustering, respectively) learning process

Observation	Class	Best match	Action	
			Classification	Clustering
$\mathbf{x}_a$	A	A, 78%	$x_a \Rightarrow \Delta_A$	$x_a \Rightarrow \Delta_A$
$\mathbf{x}_b$	A	B, 34%	$x_b \Rightarrow \Delta_A$	$x_b \Rightarrow \Delta_B$
$\mathbf{x}_c$	B	A, 02%	$x_c \Rightarrow \Delta_B$	$x_c \Rightarrow \Delta_{new}$

The discriminator which represents class A is denoted by Δ_A. For classification, any given observation always feeds the discriminator representing its true class. For clustering, since the true class is unknown, the best matching discriminator is chosen, unless its compatibility is considered too low

Despite these appealing premises, to provide a complete data stream clusterer, this work successfully overcame limitations and accomplished challenges from the past works on the same subject [11, 12, 15]: a pairwise similarity measure for WiSARD discriminators was defined; based on this measure, it was outlined the offline clustering step which provides high-level clusters; the countermeasure to cluster imbalance was further developed for improved control; and it was performed a richer experimental evaluation of the proposed ideas, with respect to data sets, test conditions, and compared algorithms.

The remainder of this paper is organized as follows: section "Research Background" introduces the concepts and assumptions concerning data stream clustering and WiSARD used through this work; section "WiSARD-based Data Stream Clustering" details the proposed system; an experimental evaluation of the WiSARD-based clusterer comes in "Experimental Evaluation"; and section "Conclusion" brings some final comments on the developed methodology.

Research Background

Data Stream Clustering

Most machine learning tasks rely on the existence of data sets which serve as primary knowledge sources. Alternatively, a data stream can replace a data set, serving as an unlimited provider of observations of a population. Formally, a data stream $S = (s_1, s_2, \ldots)$ can be described as an infinite sequence of elements. Each $s_i = (\mathbf{x}_i, t_i)$ is a pair consisting of a n-dimensional real vector $\mathbf{x}_i \in \mathbb{R}^n$ representing an observation, and the time stamp $t_i \in \mathbb{R}$ of the instant the observation was input to the learning system. To consider $t_i = i$ is a commonly used simplifying assumption [22, 37], also considered during this research to make easier the comparison of the results obtained to those of other works. However, as desirable, it was not established any dependence to such stricter condition.

Data streams clustering requires more than defining reasonable collections of observations. Incremental learning is realized updating the previously defined groups by the addition of novel observations as they arrive. This can prompt not

only the enlargement and density increase of these groups, but also the fusion of groups separated until then. The emergence of new groups also needs to be considered in this sense of incremental learning. The disposal of obsolete information works the opposite way, shrinking and reducing the density of groups as well as prompting splits and vanishings. The continuous combination of information addition and removal results in the displacement of the maintained groups in the feature space. All these should happen still taking into account the temporal information which is attached to each observation.

There are general characteristics of data stream mining tasks, which allow multiple interpretations. An example of this flexibility is the definition of data obsolescence, which is done according to an aging model. A natural alternative, the sliding window model [16] always considers the ω most recent observations equally important to knowledge definition (ω is a model parameter). Another option is the damped window model [10], which covers all stream items previously seen, assigning weights proportionally to how recent the inputs are, according to a decay factor. The landmark window [36] works incrementally adding novel inputs to its range until it is detected a concept drift, a change in the properties of the underlying population, what prompts the disposal of all information maintained until then. Handling concept drift in a smother manner, the adaptive window model [6] expands its range using new observations until change is noticed, what triggers the gradual reduction of window size until covered data are considered stable, drift-free.

With respect to efficiency, handling data streams are more challenging than the same for data sets: each input has to be processed without considering previous inputs individually and unrestrictedly; after all, they can be arbitrarily numerous, even in recent history. Given these settings, the adequate strategy is to use each item to update a data summary maintained by the learning system. Any implementation of this data summary is valid, as long as it properly supports learning under the aforementioned conditions. However, numerous existing alternatives for this matter derive from the same basic idea: to maintain a collection of micro-clusters, structures which represent tiny groups of observations which are similar to each other in the spatio-temporal sense.

This micro-clustering strategy establishes two separate procedures in system operation [31]: the data abstraction (or online) step, regarding the continuous stream processing and consequent update of the micro-clusters, and the clustering (or offline) step, regarding the definition of the high-level clusters. The clustering step is performed on demand, triggered by some external interaction or automatically, for example, when concept drift is detected [34]. Since the offline procedure does not handle stream data, it is not required to be highly efficient as the data abstraction procedure. Consequently, some works in the literature describe this step as regular batch clustering task whose input data are the micro-clusters [1, 24]. On the other hand, the data abstraction step can be briefly described as a loop, wherein it is updated the micro-cluster to which the most recent observation best fits. Algorithm 1 presents the general form of such procedure.

```
1: for all x_i, the observations from the stream do
2:     Discard information obtained from outdated observations
3:     Discard micro-clusters which are no longer useful
4:     Find the micro-cluster mc_j which better encloses x_i
5:     if mc_i is close enough to x_i then
6:         Update mc_j definition using x_i
7:     else
8:         Start a new micro-cluster based on x_i
```

Algorithm 1: A generic data abstraction procedure.

In the literature [4, 5, 12, 15, 22, 31], micro-clusters were described using different structures (e.g., clustering features and grid cells), with their own attribute sets and organization schemes (e.g., tree, linked list, and graph). Despite such variety, all alternatives have the same summarization purpose. However, they still differ in some aspects: for example, how the best fitting micro-cluster is identified, or the conditions which prompt the creation or elimination of a micro-cluster.

Weightless ANNs and WiSARD

Mainstream ANN models accomplish learning modifying weights associated with edges which interconnect network nodes. Alternatively, weightless ANNs [3] rely on explicit memorization of input patterns to realize the same task. That is, a weightless ANN instance stores small data portions in its nodes. The combination of the content of these nodes represents high-level knowledge. This enables the generalization capabilities of these networks. Because weightless ANNs nodes work as memory units, they are also known as RAM nodes. A mapping of the feature space to a finite set of smaller cardinality provides the addresses to locations in these nodes. For training, numerous weightless ANNs models simply register the occurrence of the addresses provided by observations in the training sample, as a one-shot memorization process. That opposes weighted ANNs training, which is usually is a lengthy, iterative, optimization process, in which training data need to be scanned multiple times.

The biological inspiration of RAM nodes comes from the participation of dendritic trees in real neuron functioning. In the abstraction of McCulloch and Pitts [28], such trees were modeled as a weighted edges, which multiply neuron inputs before activation function is applied on their summation. Although practical, this is a rough simplification of how these trees operate. As a matter of fact, the input signals of biological neurons, which can be of two types (excitatory or inhibitory). These signals are combined in the dendritic tree before reaching the neuron soma, where they may prompt the generation of a new signal. Such action can be naturally compared to the definition of a binary key used to access an index to binary values. This is exactly how the most basic weightless neurons work.

One of these weightless ANN models is the WiSARD [2]. It works recording characteristics of each class, provided by its training examples, as shown in Algorithm 2. For each known class, $\dot{y}$, there is one discriminator, $\Delta_{\dot{y}}$, which is responsible for storing all information regarding that class. Each discriminator is

composed of an array of δ RAM nodes, the jth referred to as $\Delta_{\dot{y},j}$. These nodes work identically to sets, well-known mathematical structures. Like physical RAM modules, their content is retrieved and modified using addresses, which represent characteristics obtained from observations. Because it works explicitly managing information divided into pieces, this model is also known as n-tuple classifier. The variable δ is a model parameter.

1: **for all** $\Delta_{\dot{y},j}$, the network nodes **do**
2: $\quad\Delta_{\dot{y},j} \leftarrow \varnothing$ ▷ Initially, nodes are empty sets
3: **for all** pairs $(\mathbf{x}_i, y_i)$, the training sample **do**
4: $\quad$Let addressing$(\mathbf{x}_i) = (a_1\ a_2\ \cdots\ a_\delta)$ be a vector of outputs of mappings of $\mathbf{x}_i$
5: $\quad$**for all** addresses a_j in addressing$(\mathbf{x}_i)$ **do**
6: $\quad\quad\Delta_{y_i,j} \leftarrow \Delta_{y_i,j} \cup \{a_j\}$

Algorithm 2: A description of WiSARD training procedure.

Storing addresses obtained from observations enables to measure the pertinence of an observation $\mathbf{x}$ to any known class, as shown in expression (1a). Consequently, the classification of such observation is performed simply choosing its best matching class, according to expression (1b):[1]

$$\text{matching}(\dot{y}, \mathbf{x}) = \frac{1}{\delta}\sum_i \left[\text{addressing}_i(\mathbf{x}) \in \Delta_{\dot{y},i}\right]^1; \tag{1a}$$

$$\hat{y} = \underset{\dot{y}}{\text{argmax}}\ \ \text{matching}(\dot{y}, \mathbf{x}). \tag{1b}$$

Considering the feature space $\mathbb{R}^n$, the addressing procedure can be described as a composite function $g \circ f : \mathbb{R}^n \to \{0,1\}^{\delta\times\beta}$, such that $f : \mathbb{R}^n \to \{0,1\}^{n\times\gamma}$ is any encoding function [23, 26] which provides binary representations of the observations, and $g : \{0,1\}^{n\times\gamma} \to \{0,1\}^{\delta\times\beta}$ is a random mapping defined prior to training. Variable γ, which controls encoding resolution, and β, the length of the addresses, are model parameters. If data are originally binary, an identity-like function can be used for encoding: that is the case for black-and-white images, the observations of WiSARD original application. Alternatively, if $[0,1]^n$ is the feature space, the following zero-padded-unary encoding function can be used[2]:

$$f(\mathbf{x}) = (h(x_0), h(x_1), \ldots, h(x_n)),$$

$$h(y) = ([\lfloor\gamma y\rceil \geq 1], [\lfloor\gamma y\rceil \geq 2], \ldots, [\lfloor\gamma y\rceil \geq \gamma])^2.$$

As a walk-through example, consider a WiSARD system with $\delta = 4$, $\beta = 3$ and $\gamma = 6$. Also consider a 2D input space: $\mathbf{x} \in [0,1]^2$. An observation $\mathbf{x}_a = (0.2, 0.7)$, whose true class is known, $y_a = A$, would be used for training as follows: after the

[1] Iverson bracket: $[L] = 1$ if the logical expression L is true; otherwise, $[L] = 0$.

[2] $\lfloor x\rceil$ represents the nearest integer of real number x.

computation of its binary representation, $f(x_a) = ((1,0,0,0,0,0),(1,1,1,1,0,0))$, and its respective addresses$g(f(x_a)) = ((1,0,0),(0,0,0),(1,1,1),(1,0,0))$, the occurrence of each of these addresses would be registered by RAM nodes of discriminator Δ_A, such as $\Delta_{A,1} \leftarrow \Delta_{A,1} \cup (1,0,0)$, $\Delta_{A,2} \leftarrow \Delta_{A,2} \cup (0,0,0)$, and so on.

Still in the same context, the classification of an observation $\mathbf{x}_b$ would happen as follows: assuming that $g(f(\mathbf{x}_b)) = ((1,1,1),(1,0,0),(1,1,0),(0,0,0))$, these addresses would be used to look for matches with RAM nodes, such as $(1,1,0) \in \Delta_{A,3}$; this allows to compute expression (1a) for all known class, such as matching$(A, \mathbf{x}_b) = 25\%$, matching$(B, \mathbf{x}_b) = 75\%$ and matching$(C, \mathbf{x}_b) = 50\%$; and at last, x_b would be ruled as an element of class B, according to expression (1b).

WiSARD-Based Data Stream Clustering

Targeting to use WiSARD discriminators as micro-clusters representatives, some parts of Algorithm 1 would be translated as follows: line 4 is nothing but a search for the best matching discriminator in the same mold of expression (1b); line 5 is similar to compare a matching rate to a threshold; line 6 is a regular absorption of $\mathbf{x}_i$ by the discriminator representing mc_j; and line 8 results in the addition of a new discriminator to WiSARD, which absorbs $\mathbf{x}_i$.

Despite these connections, no straightforward relation could be established regarding information disposal. Mining data streams requires control over learning as well as unlearning, ignoring past data according to some criteria. WiSARD training mechanism supports knowledge expansion whenever it is required. However, discarding outdated information from its knowledge base was explored for the first time in this research. subsection “Unlearning and Knowledge Refreshing” presents such major accomplishment of this work: the realization of an online, continuous learning process.

An issue in WiSARD use is the decrease of its discrimination effectiveness when dealing with unbalanced data collections. This results from how this model works, looking for matches between the addresses (values of binary features) of observations to be recognized and those of examples of the learned patterns. The larger and more varied a data sample is, the larger is the collection of addresses it can provide. Thus, the pattern whose respective data sample is the largest and most varied has the best chance to be considered the one which represents a given observation. Since it would be unrealistic to assume that clusters always result from a balanced partition of all data, an adaptation in this sense was necessary. This achievement is detailed in subsection “Cluster Imbalance and Saturation”.

Subsection “System Overview” provides an overview of the proposed data stream clustering system, aiming to make clear how its components relate to each other and how information is gathered, processed, and stored during data stream processing. A conceptual diagram is used in this regard, portraying the whole solution in a high level of abstraction. After all, considering ‘the big picture’ is indispensable to fully understand how the ideas introduced here work together. An algorithmic description is provided as well, what makes reimplementing the

proposed system and reproducing the obtained results easier. Moreover, it is also explained how offline clustering is realized based on knowledge continuously managed during stream processing.

Unlearning and Knowledge Refreshing

The outlook provided so far is very positive with respect to adapting WiSARD to cluster data streams. However, how lines 2 and 3 of Algorithm 1 were implemented based on this ANN model remains unclear. These two lines regard the disposal of outdated information, in two distinct ways: the first is related to cancelling the influence of observations on aggregated knowledge so to keep it up-to-date and the second one concerns the proper ending of the life cycle of micro-clusters when they cease to exist. A description of the approach for each of these cases follows.

For WiSARD, the influence of an observation on knowledge is materialized by the storage of the addresses obtained from this data point in the nodes of a discriminator. Therefore, the cancellation of such influence comes down to deleting these addresses. In the context of mining data streams, these deletions should happen as past data become obsolete. This can be directly related to the sliding window aging model, which considers data expirations as pinpoint events in the stream timeline. This way, it was assumed that any observation contributes integrally to the current knowledge until it expires, so that its participation is voided by the removal of its respective addresses. This opposes gradual obsolescence of the damping window aging model, which would require the maintenance of addresses weights, resulting in some additional and undesirable computational workload.

Since the original WiSARD model was not intended to deal with temporal aspects, it had to be modified to keep additional information regarding data aging. However, this had to be done without neglecting efficiency, as one of the most basic requirements to work with data streams. Then, to enable the characterization of expired addresses, a time stamp was attached to each entry of the RAM nodes, indicating the last time that it was recorded, what could happen in line 6 or line 8 of Algorithm 1. Alone, such change does not alter the computational complexity of the system: time- or memorywise, it represents a constant, minimum cost increase.

Moreover, a process to continuously detect expirations had to be embedded into WiSARD operation. The most naive strategy for this is to check all entries of every node of every discriminator. This memory full scan is as expensive as learning from scratch the entire knowledge base, what is clearly impractical. As a matter of fact, such operation would be mostly unnecessary: since data are organized sequentially, just a small portion of it becomes obsolete at one time. Thus, it is reasonable to consider least recent data as primary candidates for disposal.

Based on such idea, the following policy for time stamps management was defined: a least recently used (LRU) dictionary is used to keep a reference to each RAM node entry; every time an address is recorded, its reference is updated as the most recently used. An assumed property of an LRU dictionary is that all its entries are always sorted according to their age. Therefore, using this structure makes data processing slightly more expensive, since elements of the dictionary need to be sorted every time an new observation arrives. On the other hand, it expressively

simplifies the identification and removal of expired addresses: now, it is enough to keep checking and deleting the least recently used entry of the dictionary (and its respective RAM node entry) until it is a reference to an up-to-date element.

Summing up what was described so far: targeting to make WiSARD-based sytem capable of handling temporal information, the sliding window aging model showed itself as the best choice, compared to other alternatives, to support the intended functionalities; the characterization of outdated information was performed based on time stamps which indicate the last time that each RAM node entry was recorded; references to these entries were kept sorted in an LRU dictionary, targeting to optimize expired addresses identification and disposal. Thus, it was shown that how outdated information can be efficiently disposed from all discriminators, so that they serve as up-to-date representatives of data micro-clusters. All these operations regard line 2 of Algorithm 1 only.

Considering this first part of the problem solved, to characterize useless discriminators (line 3 of Algorithm 1) becomes trivial, based on the following reasoning. During stream processing, the knowledge a discriminator possess is expanded when new addresses are added to its RAM nodes, and it is contracted by the removal of expired addresses. Suppose that some time after creation, a discriminator Δ_k becomes "empty" (i.e., $\forall j, \Delta_{k,j} = \varnothing$), because all its addresses expired. Consequently, from then on this discriminator will be unable absorb other observations, since it cannot match any of their addresses (i.e., $\forall \mathbf{x}, \text{matching}(k, \mathbf{x}) = 0$). It can also be said that the knowledge stored by this discriminator was reduced to nothing. Thus, it can be discarded as it is no longer useful.

The emergence of micro-clusters is explicitly considered during stream processing, as it prompts the inception of new discriminators. However, their evolution and disappearance is not so evident. A discriminator is fully active as long as there are addresses related to it, so that it can be chosen as the best fit for incoming observations and, consequently, add new elements to its collection. A micro-cluster ceases to exist when there are no addresses associated with its respective discriminator. Changes in the collection of addresses of a discriminator represent micro-cluster evolution, moving inside the feature space as well as expanding and shrinking. Thus, how much of the feature space is covered by a discriminator varies continuously, so that it becomes more or less able to absorb observations over time.

Cluster Imbalance and Saturation

After making this WiSARD-derived system capable of dealing with temporal aspects, it seemed to be ready to cluster data streams. Unfortunately, this was not true. Because of the way WiSARD learns, memorizing data portions, it is strongly susceptible to have its classification effectiveness harmed when dealing with class imbalance: when at least two of all classes have a significantly different number of examples in the training sample. Despite the absence of information about classes for clustering, some sort of imbalance is still possible and harmful: at some point during stream processing, one discriminator could have a noticeably larger number

of observations associated with itself than others; consequently, from then on such discriminator would have the highest chance of being considered the best fit for all incoming observations.

A closer inspection at how imbalance affects WiSARD was the first step for the development of a countermeasure for this problem. Suppose that a discriminator Δ_k absorbed a single observation, $\mathbf{x}$. Hence, each of its nodes contain just one address: $\forall j, |\Delta_{k,j}| = 1$. Now, suppose that it absorbs another observation $\mathbf{x}'$ for which $\text{addressing}(\mathbf{x}) = \text{addressing}(\mathbf{x}')$. Consequently, the set of addresses kept by each node remained unaltered, as well as the chance of this discriminator to match any observation. From this reasoning, it can be noticed that some absorption may have no influence on the chance of matching. Therefore, contrary to what was previously assumed [12], the number of absorptions of a discriminator is not necessarily related to cluster imbalance.

It is not hard to notice how the size of the knowledge base of a discriminator alters its chance of absorbing an observation during stream processing. In a hypothetical situation in which discriminators have a single node (i.e., $\delta = 1$), a discriminator whose node contains the largest number of addresses of all is the most likely to be considered an appropriate match to incoming data. The original matching computation of WiSARD, expression (1a), does not take the size of the nodes into account. Using such expression, it was experimentally verified that the clustering system tends to concentrate all knowledge in a single, saturated discriminator covering most of the feature space.

Still considering $\delta = 1$, $\text{matching}(k, \mathbf{x})/|\Delta_{k,1}|$ seems to be a reasonable "normalized" matching rate: this value grows with the matching rate, but shrinks as $\Delta_{k,1}$ gets larger. To generalize this idea, it is proposed here to define the cardinality of a discriminator according to expression (2a) and to define a normalized matching rate according to expression (2b). The cardinality of a discriminator is asymptotically equivalent to the area of the feature space itself encompasses: $|\Delta_k| \sim \int \text{matching}(k, \mathbf{x})\, d^n\mathbf{x}$. Moreover, it can be noticed that the denominator of $\text{matching}^*(k, \mathbf{x})$ is the geometric mean of the size of the nodes of Δ_k, what is consistent with way the sets of addresses are combined for generalization and pattern recognition:

$$|\Delta_k| = \left(\prod_j |\Delta_{k,j}| \right); \tag{2a}$$

$$\text{matching}^*(k, \mathbf{x}) = \frac{\text{matching}(k, \mathbf{x})}{(|\Delta_k|)^{1/\delta}}. \tag{2b}$$

Expression (2b) influence on WiSARD functioning is evidenced by the fact that, for two hypothetical discriminators Δ_k and $\Delta_{k'}$, it is possible to have $\text{matching}^*(k, \mathbf{x}) > \text{matching}^*(k', \mathbf{x})$, while $\text{matching}(k, \mathbf{x}) < \text{matching}(k', \mathbf{x})$. This can be seen as some kind of penalty for an all-embracing discriminator, whose represented pattern becomes too vague, impossible to define because of how varied

is its data. In other words, the answer provided by a more precise discriminator could be preferred, even if it is not the highest rated one.

When used in place of the original matching rate, its just-introduced normalized version led to better results in experimental tests. This was seen as an evidence in favor of using using such additional information, the cardinality of a discriminator, for refining the matching process for the targeted task. Moreover, modifying WiSARD in this regard added no computational cost: cardinality can be calculated iteratively, alongside expression (1a); such calculation requires nothing besides the length of the RAM nodes, what is readily available during system functioning; and it is possible to cache a cardinality calculation, since its value remains unaltered until the respective discriminator absorbs an observation or discard some of its content.

Despite these arguments supporting the use of the normalized matching rate, to make cardinality influence on matching computation more flexible was considered valid: after all, imbalance occurrence is not binary, but it can take place at various levels. The outcome of this last reflection is expression (3), named adjusted matching rate. Its calculation depends on an additional model parameter μ, the cardinality weight. Setting $\mu = 0$ reduces adjusted matching rate to original matching calculation, while setting $\mu = 1$ leads to the normalized matching rate. As μ grows, the system becomes more prone to choose precise discriminators over generic, all-encompassing ones, despite proximity to data to be absorbed. The adjusted matching rate showed to be the best choice compared to the other two options, allowing the system to be configured to distinct conditions imposed by different data streams:

$$\text{matching}^{*}_{\mu}(k, \mathbf{x}) = \frac{\text{matching}(k, \mathbf{x})}{(|\Delta_k|)^{\mu/\delta}}. \tag{3}$$

System Overview

The ideas described in the previous sections are answers to the major issues raised during the development of the intended data stream clusterer. These parts were presented separately, but their integration with the WiSARD model and with each other as well is important as well. It is interesting to observe how this system processes and manages information, transforming raw data into high-level knowledge. Figure 1 provides such perspective of the whole. It show how data flow from a source, a generic data stream generation process, towards a sink, a report about data clusters.

The system offline module works based on a batch agglomerative clustering algorithm, using as input a collection of micro-clusters representatives, the discriminators, to define top-level clusters. To make such mechanism operative, it was necessary to define an inter-discriminators similarity measure. Hypothetically, such measure could be the distance between micro-clusters centers estimated according to the knowledge stored in the discriminators. This would be aligned with some works in the literature, which consider micro-clusters hyperspherical entities [1, 10, 24].

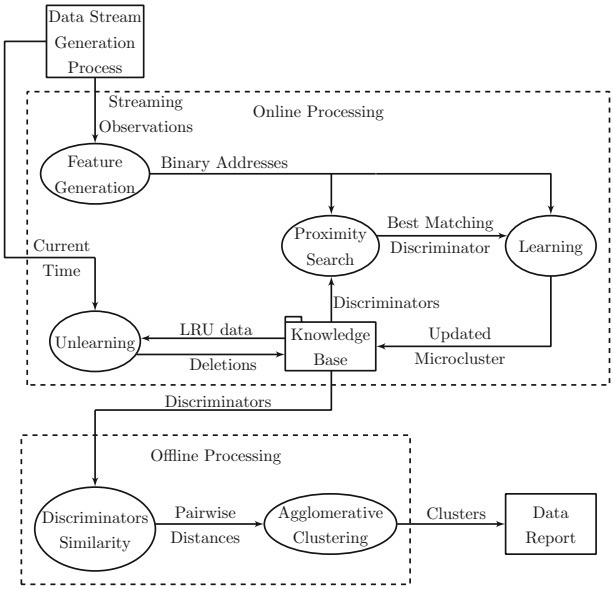


Fig. 1 Data flow diagram of WiSARD for clustering data streams. Some remarks: the data stream is continuously generated by some external entity, what contrasts with full prior availability of a regular data set; such external entity also acts as a global time reference

However, discriminators are more flexible, allowing micro-clusters to have arbitrary shapes. Consequently, the distance between estimated centers may not provide a true notion of proximity. To better represent this, expression (4) was used to evaluate discriminators similarity. It was inspired in the Jaccard Index [25], but contrasts with it since it is strictly positive: $0 < s(\Delta_a, \Delta_b) \leq 1$. This way, no pair of discriminators is considered completely dissimilar. However, their similarity diminishes as the amount of content not shared between them grows:

$$s(\Delta_a, \Delta_b) = \frac{1 + \sum_i |\Delta_{a,i} \cap \Delta_{b,i}|}{1 + \sum_i |\Delta_{a,i} \cup \Delta_{b,i}|}. \tag{4}$$

To conclude the description of the proposed WiSARD-based system to cluster data streams, its online functioning is detailed in Algorithm 3, in the same format of Algorithm 1. The least recently used dictionary, denoted by LRU, is indexed by triples (k, j, a_j), whose elements regard a discriminator, a RAM node and a RAM address, respectively. Each iteration of the algorithm has a complexity of $O(\delta\beta d)$: the loops starting at lines 2 and 11 require $O(\delta\beta)$ steps; line 6 and the block starting at line 8 can be realized in $O(1)$ steps; and line 7 takes $O(\delta\beta d)$, where d is the number of discriminators which match an observation $\mathbf{x}$ in any degree. The expected value of d varies according to model parameters β and μ.

Targeting a better understanding of Algorithm 3, a step-by-step example of its functioning is now provided. Let $(x = (0.2, 0.7), t = 66)$ be a incoming observation which was just received, as shown in line 1 of the algorithm. In addition, let $\omega = 30$. Before the absorption of this observation, outdated knowledge has to be discarded, it

there is any of it. Assuming that $t = 66$ represents the current time, $t - \omega = 36$ is the oldest possible time stamp of an unexpired RAM node entry. If the condition shown in line 2, min LRU $\leq t - \omega$, is true, it means that the oldest entry in the LRU dictionary, whose time stamp is represented by min LRU, is expired and needs to be deleted.

This way, lines 3, 4, and 5 regard the retrieval of this oldest entry of the LRU dictionary, the removal of its respective RAM node entry, and also the removal of the LRU dictionary entry itself, respectively. Supposing that min LRU $= 30$, such executing could happen as follows: after line 3 is executed, $k = 22, j = 1$ and $a_j = (1, 1, 1)$. Then, in line 4, the binary address a_j is removed from the jth node of discriminator k: $\Delta_{k,j} \leftarrow \Delta_{k,j} \backslash \{a_j\}$. The same goes for the LRU dictionary entry (k, j, a_j) in line 5. At last, line 6 regards the complete removal of any reference to all vanished micro-clusters.

1: **for all** $(\mathbf{x}, t)$, the incoming observations **do**
2: **while** min LRU $< t - \omega$ **do**
3: $k, j, a_j = \underset{k,j,a_j}{\text{argmin}}\ \text{LRU}_{k,j,a_j}$
4: $\Delta_{k,j} \leftarrow \Delta_{k,j} \setminus \{a_j\}$
5: Delete LRU_{k,j,a_j}
6: Delete all Δ_k for which $|\Delta_k| = 0$
7: $k \leftarrow \text{argmax}_k\ \text{matching}^*_\mu(k, \mathbf{x})$
8: **if** $\text{matching}^*_\mu(k, \mathbf{x}) < \epsilon$ **then**
9: Let Δ_{new} be a new discriminator
10: Let k be the index of Δ_{new}
11: **for all** addresses a_j in addressing$(\mathbf{x}_i)$ **do**
12: $\Delta_{k,j} \leftarrow \Delta_{k,j} \cup \{a_j\}$
13: $\text{LRU}_{k,j,a_j} \leftarrow t$

Algorithm 3: WiSARD-based data abstraction procedure.

Still in the same regard, after disposing all outdated knowledge, it is now possible to carry on the observation absorption. After finding the best matching discriminator k in line 7, let $\text{matching}^*_\mu(k, \mathbf{x}) = 10^{-11}$. Supposing that $\epsilon = 10^{-100}$, then a new discriminator would be added to the collection maintained in a WiSARD system. Moreover, k would be now a reference to such new discriminator. Then, lines 11 and 12 would be realized just like lines 5 and 6 of Algorithm 2. Finally, line 13 represents updating the LRU dictionary as a consequence of feeding the knowledge base.

Experimental Evaluation

The effectiveness of the proposed system, named WiSARD for Clustering Data Streams (WCDS), was assessed through various tests. This clusterer is intended to be flexible enough to work with distinct goals and conditions: for example, anomaly detection, concept drift, and merging and splitting of clusters. With such goal in mind, the experimental evaluation reported next was designed aiming to assess this

versatility, featuring data sets from multiple domains which pose varied challenges to the tested algorithms. The experiments were developed in Python, and used the scikit-learn module [29] which provides implementations of various clustering algorithms and performance measures. Moreover, implementations of stream-oriented clustering algorithms were used as provided by the MOA software bundle [7] and its extensions.

Numerous existing approaches to data stream clustering rely on aging models different of the sliding window model employed in this research [1, 4, 5, 10, 24]. Comparing the results of these approaches to those obtained by the proposed method would be questionable in some sense: if two clusterings are produced using different criteria to decide which are the current observations, they are based on distinct data and, consequently, cannot be naively compared; the same goes for results obtained considering the sliding window model but with different values for window length, ω.

WCDS performance was compared to those of clustering methods organized in two groups. The first group was composed of algorithms for the conventional clustering: K-means [27] and average-linkage agglomerative clustering [17], two classical clustering algorithms; and HDBSCAN [9], a state-of-art batch clusterer. The second group included data stream clusterers only: three well-known methods, ClusTree [24], DenStream [10], and Clustream [1]; and also two recently introduced methods, CNDenStream [4] and SNCStream+ [5]. Methods of this last group were adjusted targeting to establish a criteria to weight stream elements similar to that of WCDS: the larger the sliding window length was set, the smaller was the decay factor considered for the damping window model. The methods in the first group provided baseline results for the batch clustering tests, while the second was used for data stream clustering evaluation.

All experiments were performed using data for which label information was available. Such information was not exposed to clustering algorithms during their functioning, but it was used to evaluate the provided data partitions. This same experimental setting was used in numerous recent works on data stream clustering [4, 5, 12, 15, 22, 34]. This shows how data stream clustering validation is generally performed using external evaluation measures, which are based on ground truth knowledge regarding data labels. On the other hand, internal evaluation measures, computed according to clusters shape, topology, and other geometric properties, are used less frequently [21].

The following clustering validation indexes were employed: V-measure [30], Adjusted Rand Index (ARI) [33], and Adjusted Mutual Information (AMI) [32]. Except ARI, whose lower bound is -1, the values of all these measures range from 0 to 1, with greater values associated with clusterings similar to ground truth labeling. The use of measures based on combinatorics (ARI) and information theory (the remainder) aims to present different perspectives of the results. Furthermore, the data stream clustering tasks were evaluated comparing the ground truth labeling and clustering of every ω observations. Finally, the superiority of some method when compared to another was attested using the Wilcoxon signed-rank test with a significance level (α) of 0.05.

Batch Clustering of Synthetic Data

The considered methods were compared in two different settings. The first of these regards synthetic, bidimensional data sets, whose number of observations are in the range from over three hundreds to just over three thousands. In this setting, no temporal information was considered during data processing. Such conditions combined with the use of prepared data enable to obtain proper initial impressions of the examined alternatives: for example, the use of low-dimensional data allows to visualize the resulting partitions in simple scatter plots. In this setting, WCDS worked as a two-pass algorithm, defining the micro-clusters in the first pass, and the final clusters in the second one.

The five data sets used in this first set of experiments come from two publicly available data sets repositories: Jain, a 'two-moons'-like data set, D31, which features 31 circle-shaped clusters and Aggregation, with 7 clusters in varied shapes, come from a collection found in the Web;[3] Complex8 and Complex9, which feature, respectively, 8 and 9 clusters in varied shapes can also be found in the Internet.[4] Such variety of characteristics poses an interesting test of learning adaptability.

WCDS was tested using the following parameter setup: $\delta = 200$ and unary enconding with $\gamma = 200$; since no aging was considered, $\omega = \infty$ and $\mu = 0$; the β parameter was adjusted to each data set, and its value is indicated together with the obtained results. The following alternatives, with respective configurations, were also tested: K-means and average-linkage agglomerative clustering, targeting the true number of clusters of each data set, and HDBSCAN, with a minimum cluster size of ten elements.

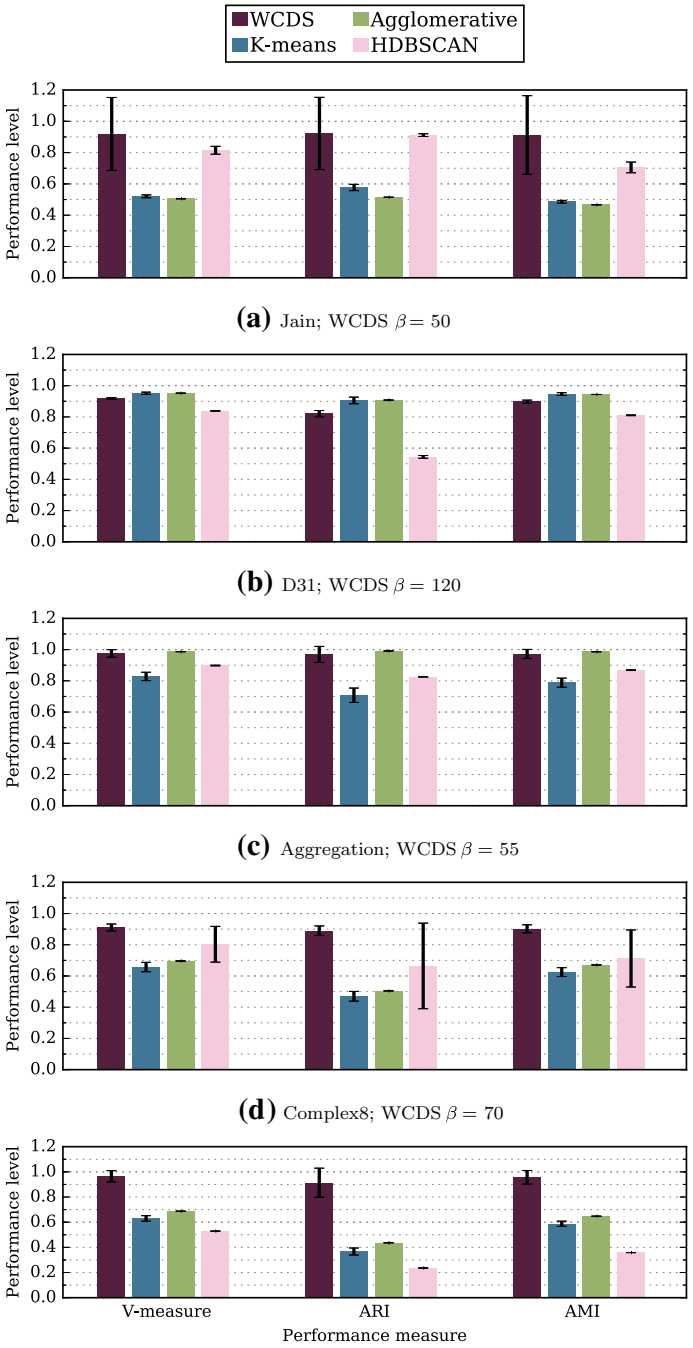
The results of this first task are depicted in Fig. 2. The average performance of the tested options in 30 rounds is presented together with error bars, whose endpoints are mean $\pm$ standard deviation. In each round, the order in which the observations were presented was randomly set. This aims to verify if WCDS can still produce reasonable results when observations, in a relatively small amount, are not favorably sequenced.

For almost all data sets and quality standards, it was statistically confirmed that WCDS was the most effective alternative or exhibits a performance not worse than the best one. Relatively, its worst results regarded data set D31, whose circle-shaped clusters were more properly identified by K-means and agglomerative clustering. HDBSCAN, which tries to detect the number of clusters automatically, performed well considering the other methods knew the true number of clusters a priori.

As a different perspective, the visualization of the clusters defined for some data set could provide an interesting feedback of the algorithms. Figure 3 presents such point of view for data set Complex8, which highlights differences between the tested approaches: only WCDS and HDBSCAN correctly identified the five horizontally-spread clusters and the other three clusters were mistaken in different degrees by all methods, with HDBSCAN followed by WCDS being the top

[3] http://cs.joensuu.fi/sipu/datasets/ (accessed 2017/1/11).

[4] https://github.com/deric/clustering-benchmark (accessed 2017/1/11).



(a) Jain; WCDS $\beta = 50$

(b) D31; WCDS $\beta = 120$

(c) Aggregation; WCDS $\beta = 55$

(d) Complex8; WCDS $\beta = 70$

(e) Complex9; WCDS $\beta = 80$

Fig. 2 Results for the task of batch clustering of synthetic data

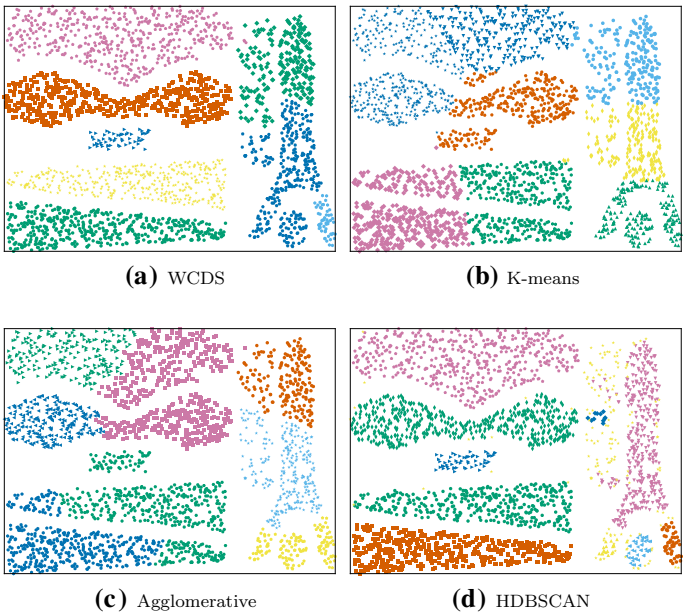


Fig. 3 Clusters for data set Complex8

performers in this regard. This is an evidence of how handling non-convex groups can still be seen as one of the basic and most challenging targets of clustering.

Data Stream Clustering

After analyzing WCDS performance while clustering toy 2D data sets, which is a more steady and favorable condition, its behavior when dealing with temporal data was examined. Such gradual exploration of system features enables a clearer understanding of its functioning as a whole, highlighting the contribution of each of its parts. This way, as an initial step, WCDS successfully found simple clusters from relatively small data sets. Next, its clustering and forgetting capabilities were combined for the accomplishment of two popular data stream clustering tasks. At last, its cardinality weighting feature was used to ensure system proper functioning with larger values of ω.

While the true number of clusters was assumed to be known in the first test setting, to do the same for this last case would be unreasonable: it is impossible to know at every instant during stream processing the true number of partitions to be defined from current data. On the other hand, depending on the application or domain, the total number of labels can be known a priori. This way, the number of labels was used to define the target number of clusters for the Clustream algorithm. DenStream, CNDenStream, and SNCStream+ defined the number of clusters according the their configuration. The same goes for ClusTree, which used DBSCAN to produce high-level clusters from its micro-clusters. WCDS also defined the number of clusters automatically: its offline operation still came down to

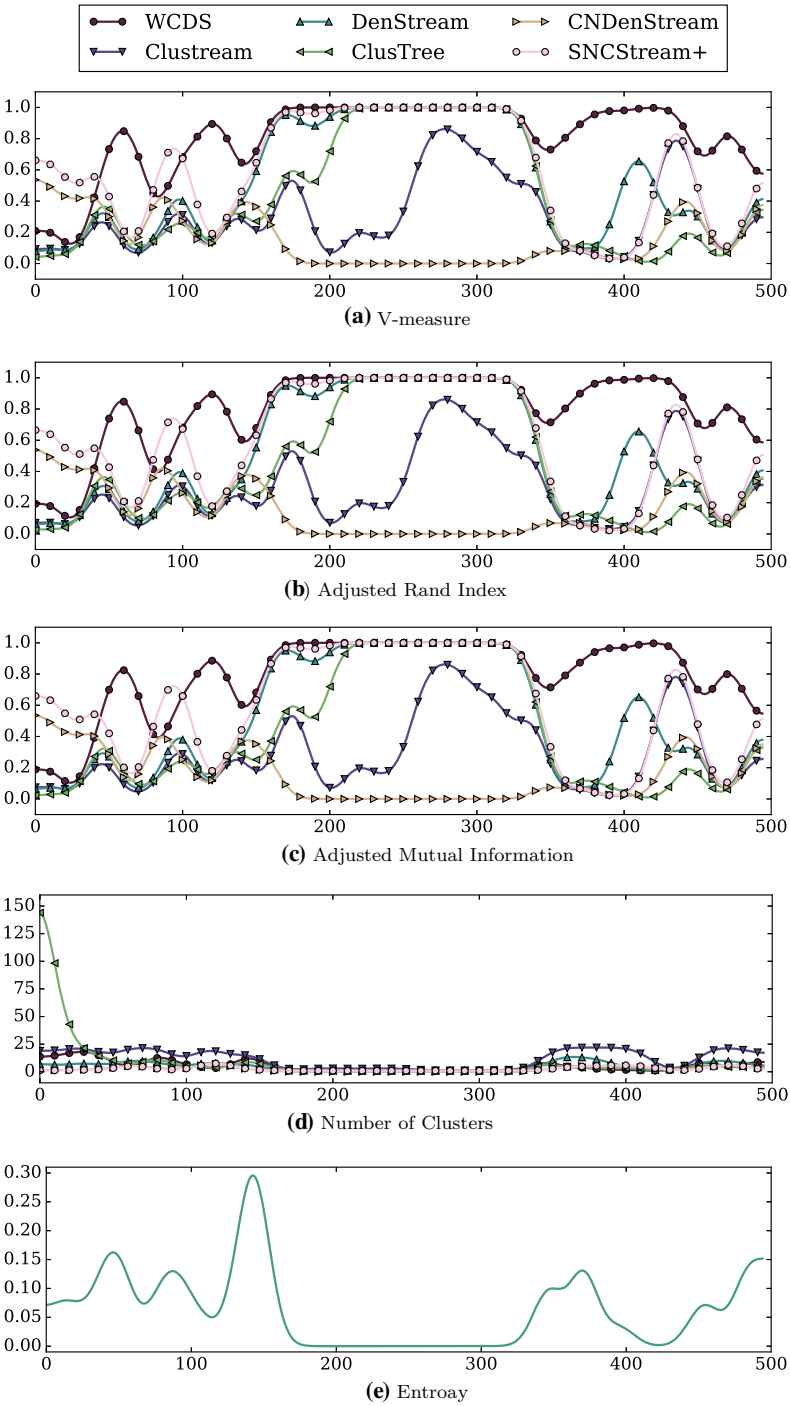


Fig. 4 Results for the experiment with the NID data set

performing agglomerative clustering of its micro-clusters; however, instead of targeting a given number of clusters, the stopping criterion for this procedure was defined based on the inconsistency coefficients of the links of its hierarchical cluster tree [35].

This first test with stream data uses the Network Intrusion Detection (NID) data set.[5] This data set was originally used in the KDD Cup 1999, whose task was to classify network connections as good or malicious. Here, however, these connections should be clustered as they continuously happen. Just the numeric attributes of this data set were used, totalling 34 variables. Moreover, the original 10% data split was preferred over the full data set: the former is more challenging, having fewer stream intervals with observations of just one class than the latter. The 494,021 observations are divided into 23 classes, and were processed in their original order. WCDS used the following setup: $\beta = 70, \delta = 50, \gamma = 50, \mu = 1$ and $\omega = 1000$. WCDS rival methods were configured as described by Barddal et al. [5], and also based on MOA default settings.

Figure 4 presents the results of the experiment with the NID data set. Observing the entropy plot, it can be noticed that during the intermediate section of the data stream, there is a single class, what explains the null entropy in such period. Although clustering data from a single class could be considered easy, just two of the six tested algorithms perfectly accomplished this: WCDS and SNCStream+. In the periods before and after this intermediate section, WCDS had the best performance for most of the time. Such superiority over all alternatives for each of the clustering validation indexes was statistically confirmed. Another interesting fact is that although this data set features 23 classes, there is a severe imbalance between them, what hurts Clustream performance: since it always tries to define the same number of clusters, its performance in the single-class period was far from ideal.

Another real-world data set, the Forest Cover-Type (FCT) data set [8] is also popularly used for data stream clustering benchmarking. It is composed of 581,012 observations described by 54 attributes of various types. The non-numeric attributes were discarded, so that only ten attributes remained. The observations are elements of seven classes. A normalized version of the original data set was used.[6] The observations of this data set were also processed according to their original order. WCDS as well as the other clusterers used the same settings of the NID experiment. The sole modification was to consider $k = 1$ for CNDenStream and SNCStream+, what led to better results than the default setting $k = 4$.

The results of this experiment are depicted in Fig. 5. Compared to the experiment with the NID data set, to identify clusters from the FCT data set could be considered a harder task, since the performance of all methods got worse. The greater entropy levels of the FCT data set compared to those of the NID data set are another evidence of such higher difficulty. Consequently, it also became harder to visually determine which method had the best performance: with respect to V-measure, WCDS dominated the other methods for most of the stream duration, but the same

[5] http://kdd.ics.uci.edu/databases/kddcup99/ (accessed 2017/02/07).

[6] http://moa.cms.waikato.ac.nz/datasets/ (accessed 2017/02/07).

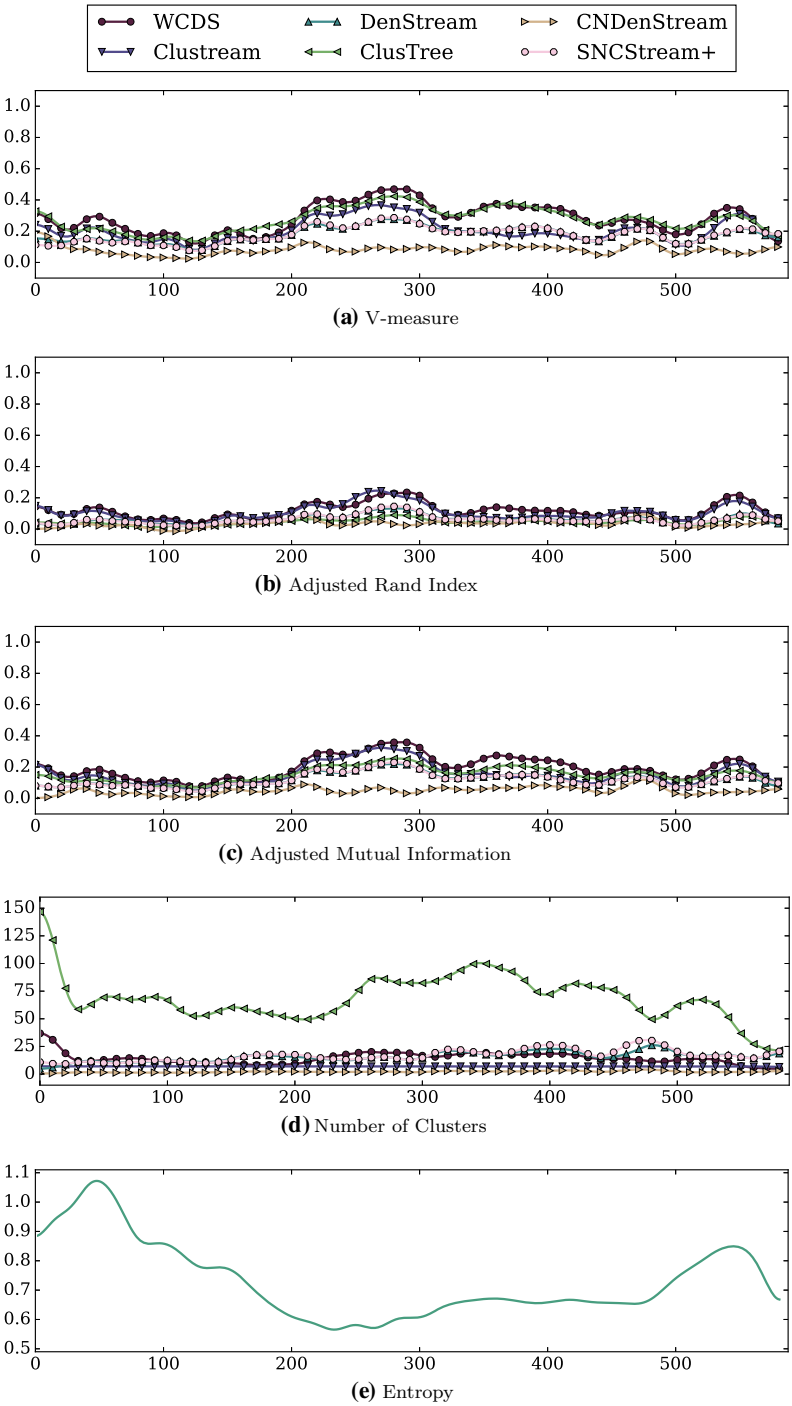


Fig. 5 Results for the forest cover-type data stream

cannot be said about ARI and AMI. Fortunately, this doubt does not exist from a statistical point of view: once again, the performance of WCDS was significantly superior than that of the other methods, considering each of the three clustering validation indexes.

For the two final tests, it was used a data collection provided by Fonollosa et al. [18] which contains paraphrasing its description in a data sets repository,[7] acquired time series from 16 chemical sensors exposed to two gas mixtures (ethylene and methane in the air, and ethylene and CO in the air) at varying concentration levels. Thus, each of these two sets of stream data have 16 input attributes, which should be used to predict the concentration level of the gases which compose the mixture being considered. Moreover, each data set has over 4 million observations, and features more than 70 classes.

Compared to NID and FCT, these chemical data streams are much denser, so that with $\omega = 1000$, the resulting entropy level is even lower than that of the NID data stream, what suggests that this would be a trivial clustering task. For a more interesting use of these data sets, a larger window was used, setting $\omega = 20{,}000$. Consequently, the clustering results were evaluated considering larger observation sequences, what also led to a higher entropy level. The remaining parameters of WCDS were set as: $\beta = 100, \delta = 50, \gamma = 50$ and $\mu = 150$. Because of the definition of a larger window, not only WCDS was reconfigured but also its rival alternatives. This way, Clustream considered horizon = 20,000. DenStream, CNDenStream, and SNCStream+ considered $\lambda = 0.0125$ and $\epsilon = 0.001$. Moreover, CNDenStream and SNCStream+ considered $k = 4$. ClusTree was configured with horizon = 20,000 and $\epsilon = 0.001$.

Figure 6 presents the results of the experiment with the ethylene–CO data set. Despite the entropy variation during the entire stream length, the performance measurements of all methods were quite stable. Comparing the entropy plot of this data set with that of the FCT and NID data sets, clustering the ethylene–CO data set appears to be harder than NID data set, but still easier than FCT data set. Again, WCDS obtained the best results, which are statistically superior than those of SNCStream+, the second best option. Based on the ARI plot, which features a smaller overlap, a ranking of all methods in this task would be, from the best to the worst: WCDS, SNCStream+, ClusTree, CNDenStream, Clustream, and DenStream.

The results regarding the ethylene–methane data set are similar to those of the ethylene–CO data set, as shown in Fig. 7: during the entire stream length, there is a minimal variation of the performance indexes of all methods, despite the entropy variation graphically evidenced; based on the entropy plot, the task of data stream clustering feeding from this data set is as hard as the same task for the ethylene–CO data set. It is also harder to graphically rank all the methods, although the V-measure plot shows WCDS as the best alternative, while ARI and AMI plots indicate ClusTree superiority. Most of such impressions were confirmed statistically: WCDS and ClusTree were significantly better than the other methods while considering V-measure and ARI measures, respectively; however, it was impossible to break the tie about AMI.

[7] http://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+dynamic+gas+mixtures (accessed 2016/10/17).

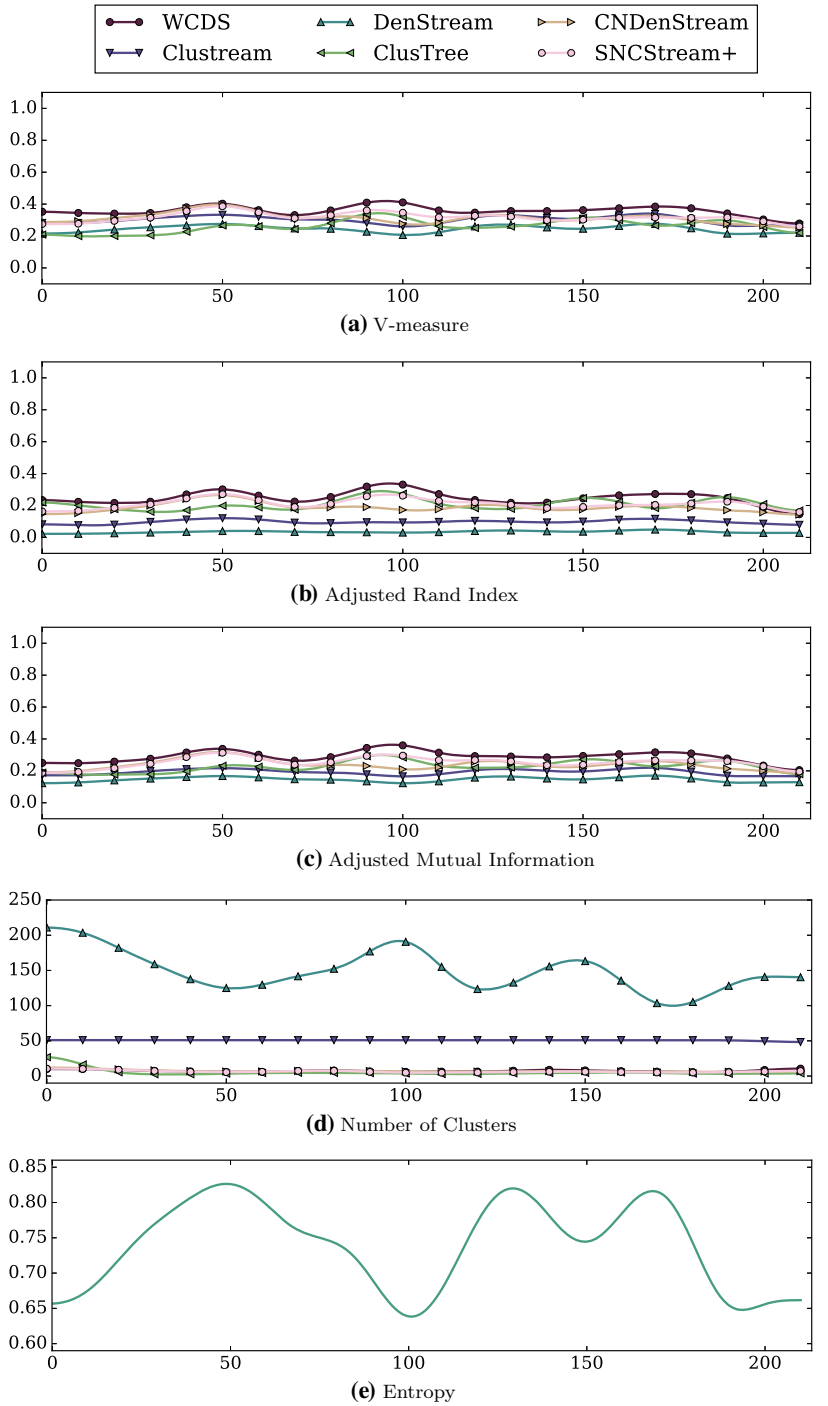


Fig. 6 Results for the ethylene–CO data stream

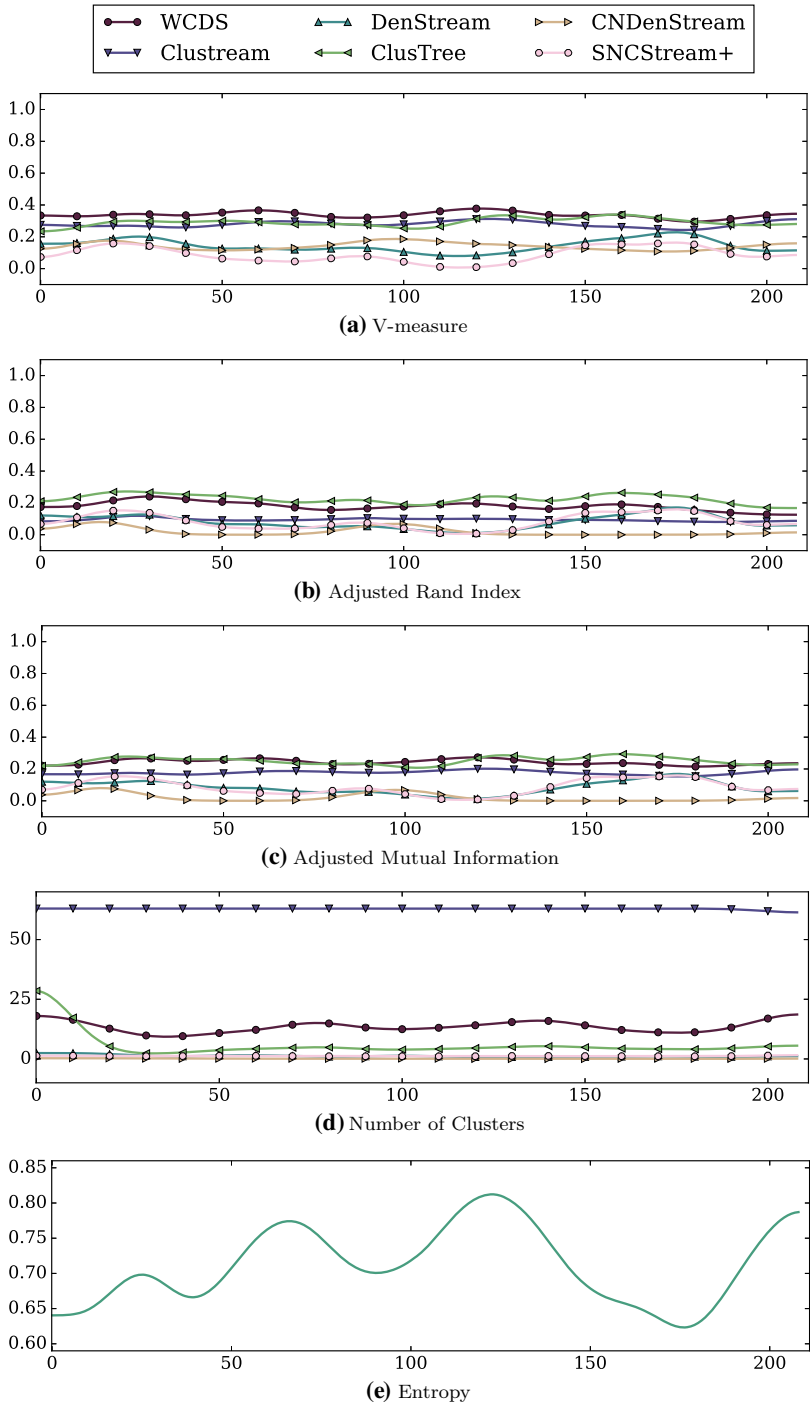


Fig. 7 Results for the ethylene–methane data stream

Conclusion

Data stream clustering is a challenging task of major relevance nowadays. It combines one of the most basic and important knowledge discovery tools with an ubiquitous data organization scheme. Although there is a rich variety of works about data stream clustering, there is still room for improvements. The definition of the concept of cardinality of a discriminator, as well as its usage to establish WiSARD matching functions which counter cluster imbalance are some of the accomplishments of this work. The WiSARD model in its original form was prone to saturation, an ill condition in which its discriminative power is severely compromised. Despite being a well-known problem of such model, effective strategies against it still need to be developed. The ideas presented here are intended to be a valuable contribution in this regard.

This research also led to the definition of a continuous learning system, capable of incrementally storing and discarding streaming information. Most online learning systems discard expired data on an estimate basis, based on ideas like the damping window model. To develop an alternative approach to this matter, it was necessary to rethink of one of the most fundamental building blocks of this area: the micro-cluster. Using WiSARD discriminators to represent micro-clusters supported the development of an alternative mechanism for the disposal of outdated knowledge. That enabled WCDS to provide results comparable to those of a state-of-art batch algorithm while maintaining the processing granularity of a stream-oriented algorithm.

Practical use of WCDS is supported by its top performance while working under varied conditions, as shown in the reported experiments. Using non-synthetic data in this regard is specially interesting, reassuring the claimed model usefulness out of its test bed. At last, to judge the found clusterings according to a variety of intrinsically different quality standards contributes to a fair comparison of all alternatives. Fortunately, WCDS excelled among all its competitors.

Acknowledgements Douglas O. Cardoso thanks CAPES (process 99999.005992/2014-01) and CNPq for financial support. João Gama thanks the support of the European Commission through the project MAESTRA (Grant Number ICT-750 2013-612944). Felipe M. G. França thanks the support of FAPERJ, FINEP, and INOVAX.

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: VLDB, pp. 81–92 (2003)
2. Aleksander, I., Thomas, W., Bowden, P.: WiSARD, a radical step forward in image recognition. Sens. Rev. **4**(3), 120–124 (1984)
3. Aleksander, I., Gregorio, M.D., França, F.M.G., Lima, P.M.V., Morton, H.: A brief introduction to weightless neural systems. In: ESANN 2009, 17th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 22–24, 2009, Proceedings (2009)
4. Barddal, J.P., Gomes, H.M., Enembreck, F.: A complex network-based anytime data stream clustering algorithm. In: Arik, S., Huang, T., Lai, W.K., Liu, Q. (eds) Neural Information Processing—22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9–12, 2015, Proceedings, Part I, Springer. Lecture Notes in Computer Science, vol. 9489, pp. 615–622 (2015)

5. Barddal, J.P., Gomes, H.M., Enembreck, F., Barthès, J.A.: Sncstream^{+}: extending a high quality true anytime data stream clustering algorithm. Inf Syst **62**, 60–73 (2016)
6. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: Proceedings of the 7th SIAM International Conference on Data Mining, April 26–28, 2007, Minneapolis, Minnesota, USA, SIAM, pp. 443–448 (2007)
7. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: massive online analysis. J. Mach. Learn. Res. **11**, 1601–1604 (2010)
8. Blackard, J.A., Dean, D.J.: Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. Comput. Electron. Agric. **24**(3), 131–151 (1999)
9. Campello, R.J.G.B., Moulavi, D., Zimek, A., Sander, J.: Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM Trans. Knowl. Discov. Data **10**(1), 5:1–5:51 (2015)
10. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: SDM, SIAM, pp. 328–339 (2006)
11. Cardoso, D.O., Lima, P.M.V., Gregorio, M.D., Gama, J., França, F.M.G.: Clustering data streams with weightless neural networks. In: ESANN 2011, 19th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 27–29, 2011, Proceedings (2011)
12. Cardoso, D.O., Gregorio, M.D., Lima, P.M.V., Gama, J., França, F.M.G.: A weightless neural network-based approach for stream data clustering. In: Intelligent Data Engineering and Automated Learning—IDEAL 2012—13th International Conference, Natal, Brazil, August 29–31, 2012, pp. 328–335. Proceedings, Springer (2012)
13. Cardoso, D.O., França, F.M.G., Gama, J.: A bounded neural network for open set recognition. In: 2015 International Joint Conference on Neural Networks , IJCNN 2015, Killarney, Ireland, July 12–17, 2015, IEEE, pp. 1–7 (2015)
14. Cardoso, D.O., Carvalho, D.S., Alves, D.S.F., de Souza, D.F.P., Carneiro, H.C.C., Pedreira, C.E., Lima, P.M.V., França, F.M.G.: Financial credit analysis via a clustering weightless neural classifier. Neurocomputing **183**, 70–78 (2016)
15. Cardoso, D.O., França, F.M.G., Gama, J.: Clustering data streams using a forgetful neural model. In: Ossowski, S. (ed.) Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4–8, 2016, ACM, pp. 949–951 (2016)
16. Datar, M., Motwani, R.: Data Stream Management—Processing High-Speed Data Streams, Data-Centric Systems and Applications. In: Garofalakis, M.N., Gehrke, J., Rastogi, R. (eds.) The sliding-window computation model and results. Springer, Berlin, Heidelberg (2016)
17. Day, W.H., Edelsbrunner, H.: Efficient algorithms for agglomerative hierarchical clustering methods. J. Classif. **1**(1), 7–24 (1984)
18. Fonollosa, J., Sheik, S., Huerta, R., Marco, S.: Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. Sens. Actuators B Chem. **215**, 618–629 (2015)
19. Gama, J.: Knowledge Discovery from Data Streams. CRC data mining and knowledge discovery series. CRC Press, Chapman and Hall, Boca Raton (2010)
20. Grieco, B.P.A., Lima, P.M.V., Gregorio, M.D., França, F.M.G.: Producing pattern examples from "mental" images. Neurocomputing **73**(7–9), 1057–1064 (2010)
21. Hassani M, Seidl T (2016) Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. Vietnam J. Comput. Sci. 1–13. doi:10.1007/s40595-016-0086-9
22. Jin, C., Yu, J.X., Zhou, A., Cao, F.: Efficient clustering of uncertain data streams. Knowl. Inf. Syst. **40**(3), 509–539 (2014)
23. Kolcz, A., Allinson, N.: Application of the cmac input encoding scheme in the n-tuple approximation network. IEE Proc. Comput. Digit. Tech. **141**(3), 177–183 (1994)
24. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: The clustree: indexing micro-clusters for anytime stream mining. Knowl. Inf. Syst. **29**(2), 249–272 (2011)
25. Levandowsky, M., Winter, D.: Distance between sets. Nature **234**(5323), 34–35 (1971)
26. Linneberg, C., Jorgensen, T.: Discretization methods for encoding of continuous input variables for boolean neural networks. In: International Joint Conference on Neural Networks, 1999. IJCNN '99, vol. 2, pp. 1219–1224 (1999)
27. Lloyd, S.: Least squares quantization in pcm. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982). doi:10.1109/TIT.1982.1056489

28. McCulloch, W.S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. **5**(4), 115–133 (1943)
29. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., VanderPlas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
30. Rosenberg, A., Hirschberg, J.: V-measure: a conditional entropy-based external cluster evaluation measure. In: Eisner, J. (ed.) EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28–30, 2007, Prague, Czech Republic, ACL, pp. 410–420 (2007)
31. Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., Carvalho, A.C.P.L.F., Gama, J.: Data stream clustering: a survey. ACM Comput. Surv. **46**(1), 13:1–13:31 (2013)
32. Vinh, N.X., Epps, J., Bailey, J.: Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J. Mach. Learn. Res. **11**, 2837–2854 (2010)
33. Wagner, S., Wagner, D.: Comparing clusterings—an overview (2007). http://staff.ustc.edu.cn/∼zwp/teach/MVA/cluster_validation.pdf. Accessed 7 Feb 2017
34. Wan, L., Ng, W.K., Dang, X.H., Yu, P.S., Zhang, K.: Density-based clustering of data streams at multiple resolutions. ACM Trans. Knowl. Discov. Data **3**(3), 14:1–14:28 (2009)
35. Zahn, C.T.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Trans. Comput. **20**(1), 68–86 (1971)
36. Zhu, Y., Shasha, D.E.: Statstream: statistical monitoring of thousands of data streams in real time. In: VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20–23, 2002, Hong Kong, China, Morgan Kaufmann, pp. 358–369 (2002)
37. Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Trans. Neural Netw. Learn. Syst. **25**(1), 27–39 (2014)

Douglas O. Cardoso is an assistant professor at the Department of Computer Engineering of CEFET-RJ Petrópolis, Brazil. He received his B.Sc. in Computer Science, M.Sc. and Ph.D. in Computer and Systems Engineering in 2009, 2012 and 2017, respectively, all from Federal University of Rio de Janeiro (UFRJ). His research interests include clustering, open set recognition, data streams mining and weightless neural networks.

Felipe M. G. França is Professor of Computer Science and Engineering, COPPE, Federal University of Rio de Janeiro (UFRJ), Brazil. He received his Electronics Engineer degree from UFRJ (1982), the M.Sc. in Computer Science from COPPE/UFRJ (1987), and his Ph.D. from the Department of Electrical and Electronics Engineering of the Imperial College London, U.K. (1994). He has research and teaching interests in computational intelligence, distributed algorithms, sensor networks, dataflow computing, and other aspects of parallel and distributed computing.

João Gama received his Ph.D. in Computer Science in 2000. He is a senior researcher at INESC TEC. He has worked in several National and European projects on Incremental and Adaptive learning systems, Ubiquitous Knowledge Discovery, Learning from Massive, and Structured Data, etc. He served as Program chair at several Machine Learning and Data Mining conferences. He is author of a monography on Knowledge Discovery from Data Streams and more than 200 peer-reviewed papers in areas related to machine learning, data mining, and data streams.