Neurocomputing 239 (2017) 39–57

Contents lists available at ScienceDirect

Neurocomputing

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

A survey on data preprocessing for data stream mining: Current status and future directions

Sergio Ramírez-Gallego[a,*], Bartosz Krawczyk[b], Salvador García[a], Michał Woźniak[c], Francisco Herrera[a,d]

[a] Department of Computer Science and Artificial Intelligence, CITIC-UGR, University of Granada, Granada 18071, Spain
[b] Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284, USA
[c] Department of Systems and Computer Networks, Wrocław University of Science and Technology, Wyb. Wyspiańskiego 27, Wrocław 50-370, Poland
[d] Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia

ARTICLE INFO

Article history:
Received 28 December 2016
Revised 31 January 2017
Accepted 31 January 2017
Available online 14 February 2017

Communicated by Zidong Wang

Keywords:
Data mining
Data stream
Concept drift
Data preprocessing
Data reduction
Feature selection
Instance selection
Data discretization
Online learning

ABSTRACT

Data preprocessing and reduction have become essential techniques in current knowledge discovery scenarios, dominated by increasingly large datasets. These methods aim at reducing the complexity inherent to real-world datasets, so that they can be easily processed by current data mining solutions. Advantages of such approaches include, among others, a faster and more precise learning process, and more understandable structure of raw data. However, in the context of data preprocessing techniques for data streams have a long road ahead of them, despite online learning is growing in importance thanks to the development of Internet and technologies for massive data collection. Throughout this survey, we summarize, categorize and analyze those contributions on data preprocessing that cope with streaming data. This work also takes into account the existing relationships between the different families of methods (feature and instance selection, and discretization). To enrich our study, we conduct thorough experiments using the most relevant contributions and present an analysis of their predictive performance, reduction rates, computational time, and memory usage. Finally, we offer general advices about existing data stream preprocessing algorithms, as well as discuss emerging future challenges to be faced in the domain of data stream preprocessing.

1. Introduction

Data preprocessing [1,2] is one of the major phases within the knowledge discovery process. Despite being less known than other steps like data mining, data preprocessing actually very often involves more effort and time within the entire data analysis process (> 50% of total effort) [3]. Raw data usually comes with many imperfections such as inconsistencies, missing values, noise and/or redundancies. Performance of subsequent learning algorithms will thus be undermined if they are presented with low-quality data. Thus by conducting proper preprocessing steps we are able to significantly influence the quality and reliability of subsequent automatic discoveries and decisions.

Data preparation, as part of preprocessing [1], is aimed at transforming raw input into high-quality one that properly fits the mining process to follow. Preparation is considered as a mandatory step and it includes techniques such as integration, normalization, cleaning and transformation.

Presently, the amount generated data is growing exponentially following the emergence of Big Data phenomenon [4,5]. Contemporary datasets grow in three dimensions –features, examples and cardinality– making complexity reduction a mandatory step if standard algorithms are to be used. Data reduction techniques perform this simplification by selecting and deleting redundant and noisy features and/or instances, or by discretizing complex continuous feature spaces. This allows to maintain the original structure and meaning of the input, but at the same time obtaining a much more manageable size. Faster training and improved generalization capabilities of learning algorithms, as well as better understandability and interpretability of results, are among the many benefits of data reduction.

With the advent of Big Data comes not only an increase in the volume of data, but also the notion of its velocity. In many emerging real-world problems we cannot assume that we will deal with a static set of instances. Instead, they may arrive continuously,

* Corresponding author.
E-mail addresses: sramirez@decsai.ugr.es (S. Ramírez-Gallego), bkrawczyk@vcu.edu (B. Krawczyk), salvagl@decsai.ugr.es (S. García), michal.wozniak@pwr.edu.pl (M. Woźniak), herrera@decsai.ugr.es (F. Herrera).

http://dx.doi.org/10.1016/j.neucom.2017.01.078
0925-2312/

leading to a potentially unbounded and ever-growing dataset. It will expand itself over time and new instances will arrive continuously in batches or one by one. Such problems are known as data streams [6] and pose many new challenges to data mining methods. One must be able to constantly update the learning algorithm with new data, to work within time-constraints connected with the speed of arrival of instances, and to deal with memory limitations. Additionally, data streams may be non-stationary, leading to occurrences of the phenomenon called *concept drift*, where the statistical characteristics of the incoming data may change over the time. Thus, learning algorithms should take this into consideration and have adaptation skills that allow for online learning from new instances, but also for quick changes of underlying decision mechanisms [7].

Despite the importance of data reduction, not many proposals in this domain may be found in the literature for online learning from data streams [8]. Most of methods are just incremental algorithms, originally designed to manage finite datasets. Direct adaptation of static reduction techniques is not straightforward since most of techniques assume the whole training set is available from the beginning and properties of data do not change over time:

- Most of static instance selectors require multiple passes over data, at the same time being mainly based on time-consuming neighbor searches that makes them useless for handling high-speed data streams [1].
- On the contrary, feature selection techniques are easily adaptable to online scenarios. Yet, they suffer from other problems such as concept evolution or dynamic [9] and drifting [10] feature space.
- Online supervised discretization methods also remain fairly unexplored. Most of standard solutions require several iterations of sharp adjustments before getting a fully operating solution [11].

Therefore, further development of data pre-processing techniques for data stream environments is thus a major concern for practitioners and scientists in data mining areas.

This survey aims at a thorough enumeration, classification, and analysis of existing contributions for data stream preprocessing. Although there exist previous studies that have performed a coarse-grained analysis on some tasks individually (e.g., feature selection or instance selection) [12,13], this work is a first deep overview of advances in this filed, additionally outlining vital future challenges that need to be addressed to ensure meaningful progress and development of novel methods.

In addition to discussing the literature in preprocessing methods for mining data streams, we propose a thorough experimental study to further enrich this survey. We have analyzed predictive, reduction, time and memory performance of selected most relevant algorithms in this field. Additionally, nonparametric statistical tests are used to give support to the final conclusions. The discussed experimental framework involves a total of 20 datasets and 10 reduction methods: three feature selectors, three discretizers, and four instance selectors.

The structure of this work is as follows. First, we present related concepts such as: data streaming and concept drift (Section 2), and data reduction (Section 3). Then online reduction contributions are grouped by task, and described in Section 4. To assess performance and usefulness of methods, a thorough experimental framework is proposed in Section 5, also grouped by task. Section 6 summarizes the lessons learned from this survey and experimental study, and discusses open challenges in data preprocessing for data stream mining, while Section 7 concludes this work.

2. Data streams and concept drift

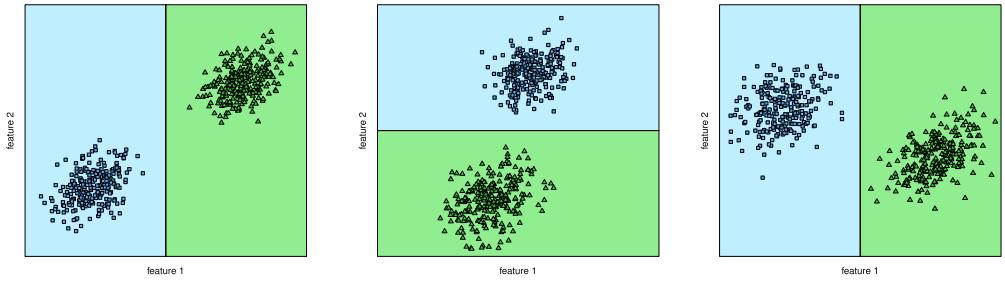
Data stream is a potentially unbounded and ordered sequence of instances that arrive over time [14]. Therefore, it imposes specific constraints on the learning system that cannot be fulfilled by canonical algorithms from this domain. Let us list the main differences between static and streaming scenarios:

- instances are not given beforehand, but become available sequentially (one by one) or in the form of data chunks (block by block) as the stream progresses;
- instances may arrive rapidly and with various time intervals between each other;
- streams are of potentially infinite size, thus it is impossible to store all of incoming data in the memory;
- each instance may be only accessed a limited number of times (in specific cases only once) and then discarded to limit the memory and storage space usage;
- instances must be processed within a limited amount of time to offer real-time responsiveness and avoid data queuing;
- access to true class labels is limited due to high cost of label query for each incoming instance;
- access to the true labels may be delayed as well, in many cases they are available after a long period, i.e., for credit approval could be 2–3 years;
- statistical characteristics of instances arriving from the stream may be subject to changes over time.

Let us assume that our stream consists of a set of states $\mathbf{S} = \{S_1, S_2, \ldots, S_n\}$, where S_i is generated by a distribution D_i. By a stationary data stream we will consider a sequence of instances characterized by a transition $S_j \rightarrow S_{j+1}$, where $D_j = D_{j+1}$. However, in most modern real-life problems the nature of data may evolve over time due to various conditions. This phenomenon is known as concept drift [7,15] and may be defined as changes in distributions and definitions of learned concepts over time. Presence of drift can affect the underlying properties of classes that the learning system aims to discover, thus reducing the relevance of used classifier as the change progresses. At some point the deterioration of the quality of used model may be too significant to further consider it as a meaningful component. Therefore, methods for handling drifts in data streams are of crucial importance to this area of research.

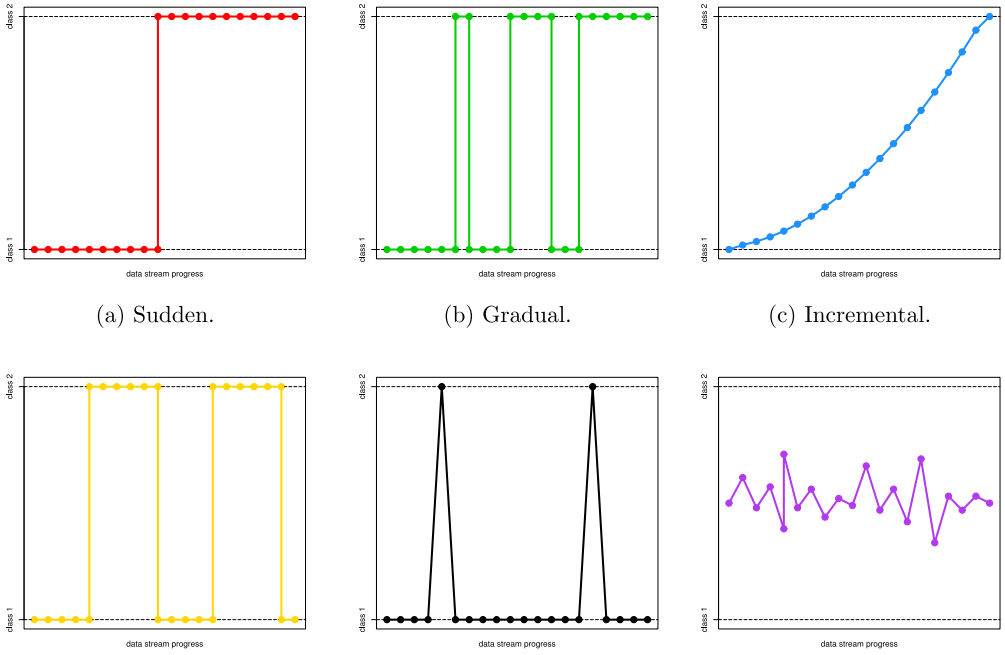
Let us now present shortly a taxonomy of concept drift. There are two main aspects that must be taken under consideration when analyzing the nature of changes taking place in the current state of any data stream:

- **Influence on the learned classification boundaries** - here we distinguish two types of concept drift. A **real** concept drift affects the decision boundaries (posterior probabilities) and may impact unconditional probability density function, thus poses a threat to the learning system. A **virtual** concept drift does not impact the decision boundaries (posterior probabilities), but affect the conditional probability density functions, thus not influencing the currently used learning models. However, it should still be detected. Visualization of these drift types is presented in Fig. 1.
- **Types of change** - here we may distinguish three main types of concept drift taking into consideration its rapidness. **Sudden** concept drift is characterized by S_j being rapidly replaced by S_{j+1}, where $D_j \neq D_{j+1}$. **Gradual** concept drift can be considered as a transition phase where examples in S_{j+1} are generated by a mixture of D_j and D_{j+1} with their varying proportions. **Incremental** concept drift has a much slower ratio of changes, where the difference between D_j and D_{j+1} is not so significant, usually not statistically significant.
- We may also face with so-called **Recurring** concept drift, what means that a concept from kth previous iteration may



(a) Initial distribution. (b) Real concept drift. (c) Virtual concept drift.

Fig. 1. Two main types of concept drift with respect to their influence over decision boundaries.



(a) Sudden. (b) Gradual. (c) Incremental.

(d) Recurring. (e) Blips. (f) Noise.

Fig. 2. Six types of drifts with respect to the ratio of changes. Graphs show transitions between the concepts along during the data stream progress.

reappear $D_{j+1} = D_{j-k}$ and it may happen once or periodically. **Blips**, also known as outliers which should be ignored as the change it represents is random [16]. **Noise**, which represents insignificant fluctuations of the concept and should be filtered out [17]. **Mixed** concept drift is a hybrid phenomenon, where more than a single type of concept drift may appear during the stream mining process. One should note that in real-life scenarios types of changes to appear are unknown beforehand and must be determined during the stream processing. Visualization of these types of drifts are presented in Fig. 2.

- Minku et al. [18] proposed *severity* criterion which allows to distinguish between **local** and **global** drift. The local drifts mean that changes affects only the small region of the feature space, while global drift affects the overall feature space, what cause that it is easier detected than the local one [19]. Additionally, we may also face with so-called "feature drift" [10], where the changes affect only selected attributes.
- Unfortunately, in real classification tasks concept drift may appear as a mixture of mentioned above changes.

As mentioned before, managing concept drift is a crucial issue in learning from data streams. Here we may use on of three solutions: (a) retrain classification system from scratch every time a new instance or chunk becomes available; (b) detecting changes and retraining classifier only when the degree of changes has been considered as significant enough; and (c) using adaptive learning method that can follow the shifts and drifts in stream on its own. Obviously, the first approach is characterized by an unacceptable computational cost and therefore two remaining solutions are used in this field.

Let us now discuss four main approaches to efficiently tackling drifting data streams:

- **Concept drift detectors** are external tools used together with the classification module. They measure various properties of data stream, such as standard deviation [20], predictive error [21], instance distribution [22], or stability [23]. Any changes in these properties are attributed to the potential presence of drift and thus allow to monitor the continuous progress of data stream. Most of drift detectors work in a two-stage setting. A warning signal is emitted when the changes start to occur, being a single to the learning system that a new classifier should be trained on the most recent instances. A detection signal informs the learning system that current degree of changes is severe and the old classifier should be replaced by a new one. This solution is also known as explicit drift handling. One should notice that ensembles of detectors start to attract the

attention of research community, although there is still much work needed to be done in this area [24,25].

- **Sliding windows** assume that we keep a buffer of fixed size containing most recent examples [26]. They are used for the classification purposes and then discarded when new instances become available. This allows us to keep a track on the progress of data stream by storing its current state in the memory [27]. This is realized either by cutting-off oldest instances or weighting them dynamically according to their relevance [28]. However, the size of the window has a crucial impact on its performance. A small window will be able to adjust to small and rapid changes, but may lose the general context of the analyzed problem and be prone to overfitting. A large window can efficiently store more information, but may contain instances originating from different concepts. To solve this issue recent studies focus on dynamically adapting size [29] or using multiple windows at the same time [30]. One should notice that a properly set sliding window will be able to adjust to changes in the stream. This is known as implicit drift handling.
- **Online learners** are updated instance by instance, thus accommodating changes in stream as soon as they occur. Such models must fulfill a set of requirements [31]: each object must be processed only once in the course of training, computational complexity of handling each instance must be as small as possible, and its accuracy should not be lower than that of a classifier trained on batch data collected up to the given time. One must notice that a set of standard classification algorithms may work in online mode, e.g., Neural Networks [32] or Naïve Bayes. However, there exist a plethora of methods modified to provide efficient online mode of operation [33,34]. These methods also offer implicit drift handling.
- **Ensemble learners** are a popular family of methods for data stream mining [35,36]. Due to their compound structure they can easily accommodate changes in the stream, offering gains in both flexibility and predictive power. Two main approaches here assume a changing line-up of the ensemble [37–39] or updating base classifiers [40,41]. In the former solution a new classifier is being trained on recently arrived data (usually collected in a form of chunk) and added to the ensemble. Pruning is used to control the size of the committee and remove irrelevant or oldest models. A weighting scheme allows to assign highest importance to newest ensemble components, although more sophisticated solutions allow to increase weights of classifiers that are recently best-performing. Here one can use static classifier, as the dynamic line-up keeps a track of stream progress. Latter solutions assume that a fixed-size ensemble is kept, but update each component when new data become available. Here managing the diversity of the ensemble is crucial for achieving good predictive power [42]. Additionally, ensembles must consist of classifiers working in incremental or online modes. There also exist hybrid approaches that combine both of these solutions within the ensemble structure [43,44].

Proper experiment design and evaluation of the examined algorithms is a key issue in machine learning domain. One need an unbiased, fair and repeatable way of comparing tested algorithms that will allow to shed lights on their strength and weaknesses, at the same time leading to valuable conclusions towards better understanding of used methods. We may evaluate certain method to assess some of our hypothesis about it, or to check its usability for a particular real-life application. Before starting any computations one must reasonably state goals of the experiment to be undertaken, choose relevant datasets, select proper metrics that will reflect the nature of examined data and establish a correct procedure for learning and comparing different models. This issue has been well-discussed in static scenarios and there exist a number of generally accepted procedures to be undertaken [45]. In the context of data stream mining, especially in non-stationary environments, canonical metrics and procedures become no longer applicable. We deal with massive, continuously incoming and evolving data that requires updating the learning model and adjusting to shifts and drifts. New classes may appear, feature space change and decision rules loose relevance over time. Additionally, canonical metrics for measuring the quality of learning process are not sufficient to perform a meaningful evaluation of models [46]. Let us discuss the correct metrics to be used for algorithms applied to data stream mining. One must understand that good algorithm must aim to strike a balance among all of these criteria.

- **Predictive power** is an obvious criterion measured in all learning systems. However, in data stream mining we must accommodate the fact that the relevance of instances diminishes over time. Therefore, simply using any averaged measure does not reflect how the learning system was able to adapt and react to changes in the stream and constant increase in the number of processed instances. Therefore, one needs to use prequential metrics that are calculated only over the most recent examples with a forgetting mechanism embedded. Prequential accuracy [47] and prequential area under the Receiver Operating Characteristics curve (AUC) [48] are the two most widely used ones.
- **Memory consumption** is a necessary criterion due to the hardware limitations during processing potentially unbounded data stream [49]. Not only the average memory usage should be taken under consideration, but also how it changes over time and with specific actions made by each algorithm.
- **Recovery time** informs us how much time an algorithm needs to accommodate new instances and update its structure. This is a crucial measure that can be a bottleneck of many methods. Assuming that new instances arrive rapidly, a good stream mining algorithm should be able to process instances before new ones will arrive to avoid queuing [50].
- **Decision time** is another time-complexity measure used. Here we are interested how long certain algorithms need to make a prediction for each new instance. As recognition phase usually precedes the update phase, it may be another bottleneck for the system. Additionally, in many applications we require a real-time response and cannot allow for a delay when speed is decision speed is vital [51].
- **Requirement for true class labels** can strongly limit the real-life applicability of many data stream mining algorithms. Many works on supervised learning in streaming scenarios assume that class labels become available soon after the instance was being classified by the system, or arrive with some delay. However, the costs of labeling the entire data stream are far from realistic and thus we must deal with limited availability of true class labels. It is useful to examine the influence of available budget (number of labeled samples) on the effectiveness of algorithms. Active learning strategies allow to select only the most relevant samples for labeling [52,53]. Semi-supervised and unsupervised methods for both classification [54,55] and drift detection [56,57] are also of interest in order to cope with this issue.

3. Data reduction

Data reduction [2] is an important preprocessing step in data mining, as we aim at obtaining accurate, fast and adaptable model that at the same time is characterized by low computational complexity in order to quickly respond to incoming objects and changes. Therefore, dynamically reducing the complexity of the incoming data is crucial to obtain such models. Additionally, due to the presence of concept drift the number and relevance of

instances and features may change over time. This must also be taken into consideration while maintaining and updating an online model. Let us now discuss the main areas in data preprocessing for reducing the complexity of data.

- **Dimensionality reduction**: There exist a wide range of techniques in the literature that aim at reducing the number of features, among others: Feature Selection (FS), Feature Extraction (FE) or locality preserving projection [58–60]. In this paper, we focus on FS and FE techniques. FS [61] eliminates irrelevant or redundant features/columns, whereas Feature Extraction (FE) generates a simpler feature space through transformations of the original one. The aim here is to yield a minimum set of features so that the subsequent distribution probability of classes remains as unchanged as possible. As FS maintains the original features, it is more convenient for model interpretation. Depending on the relationship between the selector and the predictive algorithms, we can classify FS algorithms into three categories: filters, which act before the learning process, being independent from it; wrappers, which use the specified learning algorithm to evaluate subgroups of features; and embedded, where the search is a part of the learning process itself. Wrappers methods tend to be more accurate than filters, but more complex. Embedded methods are less costly than wrappers, but require direct modifications of the learning procedure.
- **Instance reduction**: Instance Selection (IS) or Instance Generation (IG) [62]. IS is aimed at reducing the number of training instances by selecting the most representative examples. IG methods can generate new instances to fill the gaps in concept definitions. IS differs from data sampling in that the former categorizes instances depending on the problem, whereas sampling is more stochastic. Based upon the kind of search implemented by the IS algorithms, they can be classified into three categories: condensation, which removes redundant points far from the borders; edition, which removes noisy points close to the class boundaries; or hybrid, which combines both noise and redundancy removal.
- **Feature space simplification**: Normalization, Discretization, and etc. Discretization [63] summarizes a set of continuous values into a finite set of discrete intervals. This process returns nominal features that can be used by any mining process. Although most of mining algorithms work with continuous data, many of them can only cope with nominal features, specially those based on statistical and information measures (e.g.: Naïve Bayes (NB)) [64]. Other algorithms, like tree-based classifiers [65], generate more accurate and compact results when using discrete values. Good discretizers try to achieve the best predictive performance derived from discrete data, while reducing the number intervals as much as possible [66,67]. We can distinguish two main categories, based upon how intervals are generated by discretizers: splitting methods, which split the most promising interval in each iteration into two partitions; and merging methods, which merge the best two adjacent intervals in each iteration.

4. Data reduction on data streams

In streaming scenarios reduction techniques are demanded to preferably process elements online or in batch-mode as quick as possible and without making any assumptions about data distribution in advance. In the next sections, we describe those reduction proposals that were tailored for mining data streams. These methods are grouped by family/task: dimensionality reduction (Section 4.1), instance reduction (Section 4.2), and feature space simplification (Section 4.3).

4.1. Dimensionality reduction

Many FS algorithms for data streams have been proposed in the literature. Most of them are naturally incremental algorithms designed for offline processing [1], whereas others are specifically thought to cope with flowing streams [12]. All FS methods can be divided into three groups: filters, wrappers, and hybrid; according to when selection is performed: before and independently to the learning step, or tightly coupled with it.

Most of online selectors proposed in the literature are incremental adaptations of offline filters. As these filters rely on cumulative functions (mainly based on information or statistical measures), these are easily adaptable to the online environment. Despite being simple, online filters seems to adapt well to drifts, and do not need to ingest all data at once like their offline counterparts. Furthermore, online methods usually face problems derived from streams that cannot be addressed by offline methods, like the arrival of new features or classes.

Focusing on online FS, further distinctions can be made depending on the properties of streams. Some FS methods suppose that features arrive one-by-one (*streaming features*) while feature vectors are initially available [68,69]; whereas others assume that the instances always arrive sequentially, and the feature set may be subject to potential changes [70] (*online FS*). New classes can also emerge from streams without previous knowledge (concept evolution), requiring a complete redefinition of the used model. In data stream mining, feature space can also be affected by changes in data distribution. *Feature drifts* occur whenever the relevance of a given attribute changes over time when new instances arrive to the system [71]. As in other concept drifts, changes in relevance enforce algorithms to discard or adapt the model already learned by removing the most irrelevant features in the new scenario [72], as well as including the most relevant ones (*dynamic FS*). As changes in relevance directly affect the decision boundaries, feature drift can be seen as a specific type of real concept drift.

As the set of selected features evolves over time, it is likely that the feature space in test instances differs from the current selection. Therefore, when a new instance is being classified, we need to perform a conversion between feature spaces for homogenization purposes [9]. The types of conversion to consider are the following:

- Lossy Fixed (Lossy-F): the same feature set is used for the whole stream. It is generated from the first batch. All the following instances (training and test) will be mapped to this set, resulting in a clear loss in future information.
- Lossy Local (Lossy-L): a different feature space is used for each new training batch. Test instances are thus mapped to the training space in each iteration. This conversion is also troublesome because relevant features in test may be omitted.
- Lossless Homogenizing (Lossless): Lossless is similar to the previous conversion, except that the feature space in the test set is being considered here. There exist a homogenization between spaces, for example, by unifying both spaces and padding with zeros any missing feature in the other set. This conversion results in using all current and previous information, so it can be seen as the best option.

In this paper, we will focus on **online** techniques that allow the arrival of new instances and features at the same time, because they represent a scenario present in real-world problems. Let us now present a list formed by the most relevant algorithms on this topic:

- Katakis et al. [70] was among the first to introduce the problem of dynamic feature space in data streams. They proposed a technique that includes a feature ranking (filter) method to select relevant features. As the importance score of each feature

can be measured using many cumulative functions like Information Gain (IG), χ^2 or mutual information, it can be seen as a versatile solution for online feature ranking.

- Carvalho et al. [73] proposed Extremal Feature Selection (EFS), an online FS method that uses the weights computed by an online classifier (Modified Balanced Winnow) to measure the relevance of features. The score is computed as the absolute difference between the positive and negative weights for each feature.
- Masud et al. [9] proposed a streaming classification technique (DXMiner), which uses the deviation weight measure to rank features during the classification phase. Furthermore, DXMiner naturally address the problem of novel classes (concept-evolution) by building a decision boundary around the training data. In contrast to previous methods, DXMiner uses lossless conversion, which is useful for novelty detection. To rank features in the test space, DXMiner uses a unsupervised technique (e.g., the highest frequency in the batch) that selects features more representative for incoming concepts. Note that this requires a batch-mode setting to compute such statistics.
- Nguyen et al. [72] designed an ensemble technique based on windowing to detect feature drifts. The algorithm is based on a ensemble of classifiers, where each classifier has its own feature set. If a drift is detected, the ensemble is updated with a new classifier together with a new feature subset; otherwise, each classifier is updated accordingly. Fast Correlation-Based Filter (FCBF) based on Symmetrical Uncertainty is being used here. FCBF heuristically applies a backward technique with a sequential search strategy to remove irrelevant and redundant features.
- In [74], authors propose an algorithm to mine recurring concepts (called MReC-DFS). Here, they adopt the same selection solution proposed in [70]. Hover, instead of selecting a fixed number of features, they propose to use either a fixed threshold or an adaptive one based on percentiles. They also compare the effects of using different space conversions [9] (like Lossy-F, Lossy-L or Lossless).
- Wu et al. [75] proposed two approaches for handling streams with growth of feature volumes over time, named Online Streaming Feature Selection (OSFS) and Fast Online Streaming Feature Selection (Fast-OSFS). They are based on a two-phase optimal subset discovery scheme: online analysis of relevance and then redundancy. Class-based relevance is used to select or discard a new feature. Then a new and extended feature set is analyzed to detect if there exist a subset of features that may make one of the used features and class variable conditionally independent. If yes, then such a feature is discarded. This allows to to control the expansion of the feature space. In Fast-OSFS the redundancy analysis is divided into two parts. Firstly a redundancy of new feature is being checked, in order to decide if this feature should be selected. Only if new feature was included, the redundancy of previous features is being analyzed. This leads to a significant computational speed-up of this method.
- Wang et al. [76,77] proposed a greedy online FS method (called OFS) based on a classical technique that makes a trade-off between exploration and exploitation of features. The algorithm spends ε iterations on exploration by randomly choosing N attributes from the whole set of attributes, and the remaining steps on exploitation by choosing the N attributes for which the linear classifier has nonzero values. In this work, no feature drift is addressed explicitly, and no comparison with previous works is performed.
- An online feature selection method based on group structure analysis was proposed in [78]. This work was based on assumption that features may arrive in specific groups, like textures, colors etc. Authors proposed Online Group Feature Selection (OFGS) algorithm that utilized intra-group and inter-group criteria. The former criterion used spectral analysis to select discriminative features in each group. The latter one applied linear regression model to chose an optimal subset of from all preselected features. It is worth noticing that a similar problem was discussed by Li et al. [79].

Table 1 details the type of selection and space conversion performed by each algorithm. Two remarkable selection strategies emerges from this summary: one based on information filtering and another based on the use of classifier weights (wrapper).

Apart from the previously mentioned most relevant algorithms there exist a number of other online and streaming feature selection proposals in the literature. Let us now discuss them shortly. Yan et al. [80] proposed simultaneous feature extraction and selection using orthogonal centroid algorithm. Tadeuchi et al. [81] proposed a quick online feature selection that used filters to generate several potential subsets and a wrapper to chose the best one from them. Authors speculated that this solution should be able to handle concept drift appearance. Cai et al. [82] proposed to use $l1$-norm regularization for continuous variable selection. Similar approach was used by Ooi and Ninomiya, however they had employed a regularized regression for this task [83]. Fan and Bouguila [84,85] presented a combination of clustering based on a Dirichlet process mixture of generalized Dirichlet distributions and unsupervised feature selection in incremental learning scenarios. Amayri and Bouguila [86] discussed similar combination of group discovery and feature reduction using finite mixtures of von Mises distributions, while Yao and Liu [87] combined online selection with density estimation. A problem of online feature selection for multi-task learning was discussed in [88]. The issue of scalability of the discussed family of models for big data mining was addressed in [89]. Roy [90] discussed how to use ensemble of Kohonen neurons for choosing features from high-dimensional streams. Recently, Yang et al. [91] introduced a parallel method using limited memory, while Hammoodi et al. [92] discussed a concept drift detection approach using only selected features. Extension of OSFS

Table 1
Summary description of streaming FS methods. Information about the type of selector (wrapper or filter), the feature conversion accomplished (if appropriate), and whether concept-evolution appears, is presented below.

Method	Selection type (measure)	Streaming features (conversion)	Concept-evolution
Katakis' method [70]	Filter (IG, χ^2, etc.)	no (Lossy-F)	no
EFS [73]	Wrapper (online classifier's weights)	no (Lossy-F)	no
DXMiner [9]	Filter (deviation weight) + unsupervised	yes (Lossless)	yes
HEFT-Stream [72]	Filter (SU)	no (Lossy-F)	no
MReC-DFS [74]	Filter (IG, χ^2, etc.)	yes (all)	no
OSFS / Fast-OSF [75]	Filter (relevance and redundancy)	no (Lossy-F)	no
OFS [77]	Wrapper (online classifier's weights)	no (Lossy-F)	no
OFGS [78]	Filter (spectral clustering and regression)	no (Lossy-F)	no

method using rough set approach for data streams was analyzed in [93], while a combination of online discretization with feature selection for neural networks was depicted in [94].

One may view a video sequence as a stream of images and in this domain online feature selection has also been explored in order to handle dynamic object detection. Yeh et al. [95] introduced an online Boosting-based feature selection, where new features were selected one at a time to compensate for changes in the background. Yang et al. [96] described an online Fisher discrimination boosting feature selection mechanism for real-time visual tracking.

Finally, it is worthwhile to mention work by Yu et al. [97], where authors implemented several popular online feature selection methods and created an open software package for Matlab.

Besides FS, dimensionality reduction can be accomplished through an artificial mapping between the original space of features and a new space of fewer dimensions. Feature extraction techniques, although less popular than FS ones, have shown their ability in many predictive problems. One of the most important contributions here is Principal Component Analysis (PCA) [98]. In [99], two online gradient-based versions of PCA are studied in depth. The aim of previous work is to obtain an online model with the lowest difference in cumulative losses with respect to the best offline alternative. A novel analysis of theorethical properties of Oja's streaming PCA was discussed in [100]. Although optimal, online PCA is not able to update projections in less than $O(n^3)$ per iteration [101]. Thus more efficient techniques needs to be developed in the future if we want a real streaming solution in feature extraction. So far it is worth mentioning streaming versions of kernel PCA proposed by Joseph et al. [102] and by Ghashami et al. [103]. Additionally, PCA was successfully applied for concept drift detection in non-stationary data streams by Kuncheva and Faithfull [104], as well as by Qahtan et al. [105]. One must notice that feature extraction from data streams is not only limited to PCA and other works, although few in numbers, exist. Allahyar and Yazdi [106] described Online Discriminative Component Analysis for continuous computation of Linear Discriminant Analysis. Sheikholeslami et al. [107] proposed a kernel-based feature extraction for mining streams with limited computational resources. Li et al. [108] introduced canonical correlation analysis with uncertainty suitable for multi-view classification of data streams.

4.2. Instance reduction

Lazy learning has been broadly used in predictive analytics [109]. Yet, case-bases naturally deteriorate and grow in size over time. In data stream scenario, past preserved cases that belong to a previous concept may degrade the performance of the learner if a new concept appears. Likewise, new instances that represent a new concept may be classified as noise and removed by a misbehavior of the IS mechanism, because they disagree with past concepts [13].

Some enhancement (**edition**) and maintenance (**condensation**) [1] should be thus performed on case-bases in form of sophisticated IS processes, which select those cases that best represent the current state of the data stream. However, most of current techniques are designed for stationary environments and ignore the concept drift phenomenon. Firstly, we present a subset of IS techniques that incrementally or in a batch way select instances from a case-base [110]:

- Instance-Based learning Algorithm 3 (IB3) [111] is one of the first attempts to deal with non-stationary nature of data. It is based on accuracy and retrieval frequency measures. By means of a confidence interval test, IB3 decides whether a case should be added to the case-base or it needs to wait until its insertion is marked as appropriate. Removal of cases is performed whenever the accuracy of a case is below (in a certain degree) its class frequency. Due to IB3 defers the inclusion of examples, it is only suitable for gradual concept drift.
- The Locally Weighted Forgetting (LWF) algorithm [112] is an instance weighting technique based on k-nearest neighbors (kNN). In LWF, those cases with a weight below a threshold are removed. LWF algorithm has been criticized by its lower asymptotic classification in static environments and by its tendency to overfitting [113]. This method has shown good performance for both gradual and sudden concept drifts.
- Salganicoff [114] designed the Prediction Error Context Switching (PECS) algorithm, which is designed to work in both dynamic and static environments. PECS algorithm is based on the same measures used by IB3, also adopting the same confidence test. In order to introduce time dimension in its decisions, PECS only consider the newest predictions in its computations. Furthermore PECS immediately add new cases to the base to expedite the slow adaptation process. PECS disables cases instead of permanently deleting them. Those cases can be re-introduced if their may once again contribute towards improved accuracy. It is argued in [115] that PECS holds high memory requirements and a slow removal process, as new instances are retained right after they arrive.
- Iterative Case Filtering Algorithm (ICF) [116] is a redundancy removal technique that discards those instances with a coverage set size smaller than its reachability set. Authors included Repeated Edited-NN [117] to remove the noise around the borders.

Although there exist more complex proposals in the literature [110], the previous list includes those methods that have served as a keystone for further developments in IS for concept drift [13]. The next list deal with those techniques that explicitly address concept drift:

- Delany et al. [118] proposed a drift control mechanism with two levels, called Competence-Based Editing (CBE). In the first level, an hybrid of two competence-based editing methods[1]: Blame Based Noise Removal (BBNR) and Conservative Redundancy Reduction (CRR), is launched. BBNR is aimed at deleting those cases whose removal do not imply coverage loss, whereas CRR selects misclassified cases with the smallest coverage. Note that both methods are designed for stationary environments, which can cause some problems like the removal of novel concepts when gradual drift appears, or forgetting of small groups of cases where examples covers each other but misclassifies all the surrounding neighbors. BBNR do not keep the competence model up-to-date, it only rebuild the model in the second level. An outdated competence model may yield inconsistencies during the evaluation phase as the model does not accurately reflect the current concept.
- Instance-Based Learning on Data Streams (IBL-DS) [115], and IBLStreams [120] are presented as the first solutions that deem both time and space factors to control the shape and size of the case-base. In both algorithms, every neighbor in a test range is removed if the class of new instance is dominant in this range. IBL-DS also introduces an explicit drift detection method developed by Gama [20], which determines when to remove a fixed number of instances considering space and time. Number of removals is computed considering the minimum error rate and the aggregated error of last predictions. Both algorithms control the size of the case-base by removing the oldest instances. However the time-based removal strategy

[1] Basic concepts about competence models can be reviewed in [119]

Table 2
Summary description of streaming IS methods. Information about the selection measure, whether drift detection is used or not, and the type of selection is shown here.

Method	Selection type	Drift detection	Edition/Condensation
IB3 [111]	Case accuracy	no	yes/no
LWF [112]	Instance weighting	no	yes/no
PECS [114]	Case accuracy	no	yes/no
ICF [116]	Competence	no	yes/yes
CBE [118]	Competence	no	yes/yes
IBL-DS [115]	Time-space distance	yes	yes/yes
FISH [121]	Time-space distance	no	yes/yes
AES [122]	Bio-inspired	yes	no/yes
COMPOSE [123]	Geometry	no	no/yes
SimC [124]	Time-space distance/case accuracy	no	yes/yes
NEFCS-SRR [13]	Competence & case accuracy	yes	yes/yes

implemented by them has been criticized because some old, yet still relevant instances may be eliminated in this process.

- FISH algorithms [121] are also based on a combination of time and space, in this case, computed as distances. The idea behind these algorithms is to dynamically select the most relevant examples, which will serve as training for next model. Three different versions of FISH were proposed. In FISH1, the training size is fixed at the start. FISH2 selects the best training size according to the accuracy (through leave-one-out cross validation). FISH3 also weights time and space by using a different loop of cross validation. FISH2 is considered as the leader of the family. FISH represents a time-consuming option since it stores all seen examples in order to compute space/time distances.
- Zhao et al. [122] present a new nearest neighbor algorithm for data streaming, based on an artificial endocrine system; called AES. This system removes the necessity of a complete case-base as in previous models, replacing case-base by representative cells. A condensation-based process is also a key feature in AES. The algorithm maintains only K boundary prototypes or cells. These prototypes keep moving during the whole process in order to adapt concept boundaries to incoming drifts.
- COMPOSE [123] is a geometry-based framework for semi-supervised learning and active learning. The idea behind COMPOSE is to label incoming instances through a semi-supervised approach, and then to create and select those α-shapes that better model the current state. This selection is, in fact, a compaction process that maintains only those shapes/prototypes more representatives for the current state. COMPOSE is mainly designed to address gradual drifts.
- SimC [124] aims at creating groups of instances for each class so that each one represents a different region of the space. Noisy and old examples are removed by selecting and discarding the least relevant example in the oldest group. As concept drift appears, the algorithm creates new groups to allocate examples that represents new concepts. Relevance in groups is measured by using space distances and their ages. For individual instances, the precision using the nearest rule is employed.
- Lu et al. [13] propose a case-base editing technique based on competence preservation and enhancement [119]. Their solution consists of three stages: the first one compares the distribution between two windows in order to detect if there is a drift or not. Apart from detecting the drift, this method also limits the area where the distribution changes most. After that the Noise-Enhanced Fast Context Switch (NEFCS) method is applied. NEFCS examines all new cases and determines whether there is noise or not (enhancement). However only the noisy cases that lie outside the detected competence areas are removed, because they may be part of novel concepts. Stepwise Redundancy Removal (SRR) method is aimed at controlling the size of the case-base (preservation). SRR removes redundant examples recursively until the case-bases' coverage starts to deteriorate.

Table 2 lists the most relevant instance selectors for drifting streams. We can draw three major types of selection from this table: competence-based, weighting-based, and accuracy-based. Competence-based methods (like CBE or ICF) tend to be more accurate but time-consuming, because they require a constant update of the competence model. Distance-based selection strategies can require even more time than competence-based models, when the number of distances and/or the features involved are high. Accuracy-based methods have difficulties in identifying noisy examples coming during drifts. Finally, feature weighting techniques tends to over-fit data and to perform worse than instance selectors according to [113].

Another relevant topic to be considered when electing instance selectors is whether enhancement and/or maintenance tasks are applied or not. Competence-based methods usually consists of two techniques, one for noise removal and another for redundancy. Redundancy is mainly ignored in accuracy-based techniques since most of them select instances according to the number incorrect predictions committed by each one. Distance-based algorithms implicitly removes redundancy through the space factor in the distance formula.

4.3. Feature space simplification

Discretization algorithms for data stream scenarios must also be able to handle the appearance of concept drifts. Definition and number of discretization intervals may change over time, following shifts in data characteristics. Therefore, it is desirable that discretization intervals are able to smoothly adapt to concept drift, without imposing increased computational cost when being recalculated.

Equal-frequency discretization (based on histograms) can be considered as one of the first techniques in dealing with incremental discretization. By using quantiles as cut points, the feature space can be partitioned in equal-frequency intervals. Estimation of quantiles in streams have been studied in depth in the literature, in approximate [11,125] and exact [126,127] forms. One of the agilest and most effective discretization alternatives is Incremental Discretization Algorithm (IDA) [11]. IDA approximates quantiles through the maintenance of a reservoir sample of the input stream. Intervals here are structured using interval heaps, an efficient data structure that allows to insert and delete elements in $O(log(n))$, and to retrieve the maximum and minimum (the interval boundaries) in constant time. As in most of cases it is not feasible to maintain a complete record of all data, approximative solutions have shown much more suitable for processing high-throughput streams than exact solutions.

Other techniques based on frequency has relied on establishing size thresholds assigned bins to cope with evolving discretization. Lu et al. [128] presented the Incremental Flexible Frequency Discretization (IFFD) algorithm. IFFD defines a range instead of a strict number of quantiles. If the updated intervals' frequency reaches the maximum and the resulting frequencies are not below the minimum (in order to prevent a high classification variance), IFFD splits the interval into two partitions.

Equal-width discretizer is another unsupervised approach that only requires as input the range of features and the number of splitting intervals. However, the main drawback here is that both approaches require streamed records arriving in random order, which is impossible in many learning problems.

Another important requirement to be considered is that some incremental algorithms require to maintain the same set of cut points (number, structure and meaning) over time [11]. That is the case of the most discriminative learning algorithms. Here usage of either an equal-width or an equal-frequency discretizer is suggested, as both of them define the number of bins in advance. Other static algorithms (e.g.: NB) does not require the preservation of intervals during subsequent predictive phases, but only to save some statistics for the current discretization step. However, generalization capabilities of such classifiers are still affected by such displacements in definitions, specially if they are sharp.

According to [129], one of the main problems of unsupervised discretizers is the necessity of defining the number of intervals in advance. Such decision can be assisted by some pre-defined rules (e.g., Sturges' rule) or by an exploratory analysis process. However, exploratory analysis is no longer possible in the present days where the number of instances is too large and pre-defined rules have shown to work only with small-sized datasets. However, unsupervised discretizers are naturally designed for streaming environments since the number of intervals remains invariant.

Most of supervised approaches tend to perform several merges and splits before obtaining a functional final scheme. Abrupt changes in intervals' definition may negatively influence the online learning process. Therefore, methods should strive for a smoother transitions. We present a short list of supervised discretization approaches:

- Gama et al. [129] presented the Partition Incremental Discretization algorithm (PiD), consisting of two layers. The first one summarizes data and creates the preliminary intervals, which will be optimized in the next layer. An equal-width strategy can be used to initialize this step. Then the first layer is updated through a splitting process whenever the number of elements in an interval is above a pre-defined threshold. The second layer performs a merging process over the previous phase in order to yield the final discretization scheme. Any discretizer can be used in the second layer, since the intervals generated in the previous phase are used as inputs. Minimum Description Length Discretizer is used as reference in the original paper. However, there are three main reasons for criticism of PiD approach. Firstly, there is no exact correspondence between the first layer and the second one, which produces inaccuracies that will chain and increase over time. Secondly, if the distribution of data is highly skewed, the number of intervals generated will dramatically increase, due to frequency overflowing. Finally, the splitting process may become even more inaccurate if many repetitions of a single value appear. In this case such a cut point might be generated that divides instances with the same feature values into two different bins, leading to inconsistencies.
- In [130] an online version of ChiMerge (OC), which maintains the O($nlog(n)$) time complexity held by the original algorithm, is proposed. In order to guarantee equal discretization results, authors implement an online approach based on sliding windows. Several data structures are being used to emulate the same behavior held by the original version. Despite of the great effectiveness claimed by the authors, a high increase in the memory usage derived from the set of data structures is displayed by this online version. This fact may prevent from its usage in some data stream scenarios with limited computational resources.

Table 3
Summary description of streaming discretization methods. Information about the name and type of discretization strategy is shown here.

Method	Discretization strategy
PiD [129]	Binning & information (split & merge)
OC [130]	Statistical (merge)

A brief classification about streaming discretizers is given in Table 3. Two alternatives representing different discretization types [1] are shown here. Classification is performed according to two factors: evaluation measures (statistical/binning/information/others) and the type of interval generation (merging/splitting intervals). The most important lesson here is that there is no wrapper online discretization solution. An approach that generate intervals by means of an online classifier weights, as proposed before by some feature selectors, would be highly suitable for this task. A wrapper approach could even solve the problem of displacements in intervals' definitions due to the closer relationship between the classifier and discretizer.

5. Experiments

In this section, we evaluate the usefulness and performance of the data preprocessing algorithms for mining data streams from different perspectives:

- Effectiveness: measured as the number of correctly classified instances divided by the total number of instances in the training set (accuracy). It can be considered as the most relevant factor in measuring usefulness of proposals.
- Time and memory performance: measured as the total time spent by the algorithm in the reduction/discretization phase. Usually performed before the learning phase, although sometimes it runs simultaneously to the prediction phase. Additionally, memory usage for the preprocessing step is being measured to show the resource consumption displayed by each tested method.
- Reduction rate: measured as the amount of reduction accomplished with respect to the original set (in percentage). For selection methods, it is related to the number of rows/columns removed, whereas for discretization, it is related to the degree of simplification of the feature space.

The experimental framework is defined in Section 5.1. Here, the list of datasets and methods, and other considerations are presented. The results and discussion of examined algorithms are presented with respect to the type of task being performed. Each task requires different settings due to its specific characteristics, which will be explained in each section. The order is as follows: FS (Section 4.1), IS (Section 4.2), and discretization (Section 4.3).

5.1. Experimental framework: datasets, methods and parameters

Table 4 shows the complete list of artificial and real datasets used in our experiments to evaluate the reduction techniques. Artificial datasets have been generated using Massive Online Analysis (MOA) benchmark [131], providing a wide range of drifting environments (blips, sudden and gradual, among others described in

Table 4
Relevant information about classification datasets. For each row, the number of instances evaluated (#Inst.), the number of attributes (#Atts.) (which ones are numerical (#Num.) and which ones nominal (#Nom.)), the number of classes (#Cl), and whether the dataset is artificially generated or not (artificial) are shown.

Data set	#Inst.	#Atts.	#Num.	#Nom.	#Cl.	Artificial
blips	500,000	20	20	0	4	yes
gradual_drift	500,000	3	3	0	2	yes
gradual_recurring_drift	500,000	20	20	0	4	yes
incremental_fast	500,000	10	10	0	4	yes
incremental_slow	500,000	10	10	0	4	yes
no_drift	500,000	24	0	24	10	yes
sudden_drift	500,000	3	3	0	2	yes
airlines	539,383	6	3	3	2	no
covtypeNorm	581,011	54	10	44	7	no
elecNormNew	45,311	8	7	1	2	no
kddcup_10	494,020	41	39	2	2	no
poker-lsn	829,201	10	5	5	10	no
spambase	4601	57	57	0	2	no
spam_nominal	9324	40,000	0	40,000	2	no
usenet_recurrent	5931	659	0	659	2	no
spam_data	9324	499	0	499	2	no
usenet1	1500	100	0	100	2	no
usenet2	1500	100	0	100	2	no
usenet3	5997	27,893	0	27,893	2	no
power_supply	29,928	2	2	0	24	no

Section 2). Each artificial dataset has been created using different combinations of generators and different parameter values. For a complete description of datasets, and source code, please refer to our GitHub repository[2].

Real datasets come from different sources:

- *airlines, elecNormNew, poker-lsn*, and *covtypeNorm* can be found in MOA's streams repository.
- *spam_data, usenet1, usenet2*, and *usenet3* are e-mail datasets affected by concept drift, collected by The Machine Learning and Knowledge Discovery (MLKD) group (http://mlkd.csd.auth.gr/concept_drift.html).
- *spambase* is a collection of e-mails classified as spam [132].
- *kddcup_10, spam_nominal* (SpamAssasin), and *usenet_recurrent* were collected by Dr. Gama and his research group KDUS (http://www.liaad.up.pt/kdus/products/datasets-for-concept-drift).
- Last dataset (*power_supply*) comes from Stream Data Mining Repository (http://www.cse.fau.edu/~xqzhu/stream.html), and contains power supply registers collected hourly from an electricity company.

Not all datasets described above have been used for every experiment. Some algorithms are designed to deal with a particular data types. For instance, most of feature selectors require discrete features, especially if they utilize information-based measures. Because MOA generators [131] only generate datasets with continuous attributes, these datasets will not be considered for FS. The final choice of datasets and any detail concerned to their features will be described in further sections.

No previous fixed partitioning has been performed on datasets, instead an online evaluation approach has been elected to asses the quality of methods, known as **interleaved test-then-train**. This technique, proposed by Bifet et al. in [133], defines a model in which each example/batch (arriving at time t) is evaluated against $t-1$-model, and then it serves as input to update that model and forms the subsequent t-model.

Reduction techniques used in experiments are listed and grouped by task in Table 5. The default parameter values has been

Table 5
Parameters of methods. Default values for each block of methods are detailed in first rows. Unless specified, these values are common to every method in block.

Method	Parameters
Feature selection	window size = 1 (default)
NB	–
IG [70]	–
SU [72]	–
OFS [77]	$\eta = 0.2$, $\lambda = 0.01$
Instance selection	$k = 3$, window size = 100 (default)
kNN	window size = 1
NEFCS-SRR [13]	l = 10, pmax = 0.5, size limit = 1000
CBE [118]	–
ICF [116]	–
FISH [121]	learner = kNN, distance proportion (time/space) = 0.5, window size = 1
Discretization	initial elements = 100, window size = 1 (default)
NB	–
OC [130]	–
PiD [129]	$\alpha = 0.75$, initial bins = 500, instances to update layer #2 = 10,000, min/max = 0/1

established according to the authors' criteria. Common parameters, like window size or the number of initial elements to consider before starting the reduction process, tends to have common values within the same group. A window size equal to one means that the algorithms work in an online manner, whereas a value higher than one implies a batch-based processing. For instance, FS and discretization methods are suitable for online scenarios, whereas most of instance selectors process elements in batches (except FISH and kNN).

As most of feature selectors and discretizers are focused on NB, it has been elected as a base classifier for these groups. Likewise, kNN serves as reference for instance selectors. Training and testing processes are performed differently for each task.

In FS contingency tables in NB are updated whenever an example arrives. During the classification phase NB only makes predictions by considering the most relevant features.

Training in discretization is also accomplished following the previous scheme, with the particularity that the structure of contingency tables may change whenever new intervals are generated. A new discretization scheme means old model will be outdated and the amount of errors will sharply increase.

As to IS, those methods with best results according to [13] have been selected for our experiments. Different update schemes have been adopted depending on the original design held by each selector. For kNN and FISH, an instant-update scheme has been adopted. In this scheme new instances are immediately added to the case-base. Note that this approach gives kNN a clear advantage over the rest of methods since an ever-updated case-base tends to adapt well to changes. However, it also introduces a lot of redundancy which does not affect accuracy.

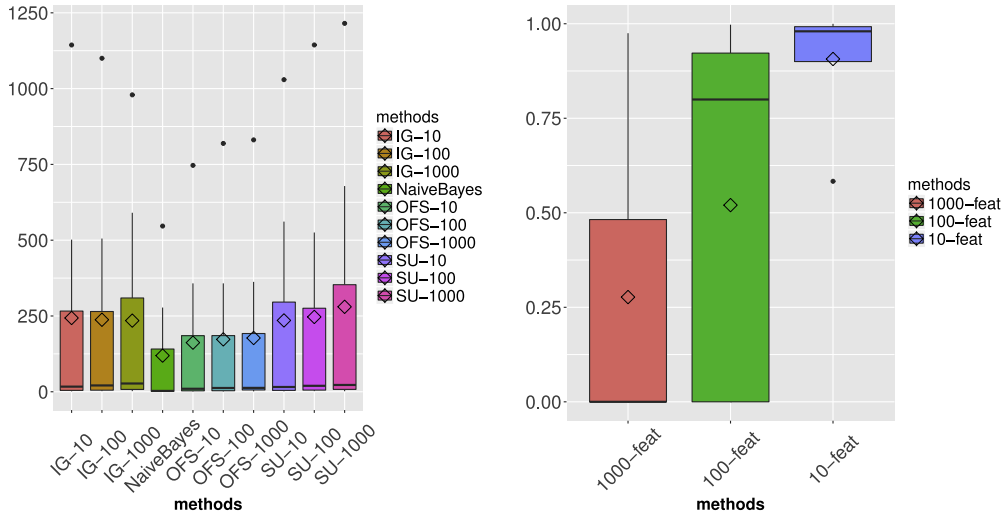
FISH selects a different training set whenever a new example arrives, thus acting in an online way. In counterpart, NEFCS shows a batch-like behavior which requires two windows for drift detection. Here, the updating of the case-base is deferred until a complete batch of examples is available. For a fair comparison between competence models (CBE, ICF, NEFCS-SRR), we have adopted a model based on batches for all these algorithms. New instances are immediately added to the case-base in CBE and ICF, but reduction is only performed when the batch size condition is met.

The whole experimental environment has been executed in a single standard machine, with the following features: 2 processors Intel Core i7 CPU 930 (4 cores/8 threads, 2.8 GHz, 8 MB cache), 24 GB of DDR2 RAM, 1 TB SATA HDD (3 Gb/s), Ethernet network connection, CentOS 6.4 (Linux). Examined algorithms have been

[2] https://github.com/sramirez/MOAReduction

Table 6
Final test accuracy by method (FS). The best outcome for each dataset is highlighted in bold. The second row in header represents the number of feature selected. No selection is performed for NB.

	Naïve Bayes	InfoGain			SU			OFS		
		10	100	1000	10	100	1000	10	100	1000
spam_data	90.6692	89.2750	90.8516	90.6692	88.9103	90.4333	90.6692	90.0579	**91.7417**	90.6692
spam_nominal	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**
usenet1	**63.3333**	53.6667	**63.3333**	**63.3333**	53.2667	**63.3333**	**63.3333**	58.3333	**63.3333**	**63.3333**
usenet2	**72.1333**	66.9333	**72.1333**	**72.1333**	66.6667	**72.1333**	**72.1333**	68.2000	**72.1333**	**72.1333**
usenet3	**84.6038**	68.8073	78.2319	82.9024	69.0242	77.8816	82.8691	54.0951	57.4646	70.5922
usenet_recurrent	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**
no_drift	51.4120	**51.4240**	51.4120	51.4120	**51.4240**	51.4120	51.4120	32.5830	51.4120	51.4120
MEAN	**80.3074**	75.7295	79.4232	80.0643	75.6131	79.3134	80.0596	71.8956	76.5836	78.3057



(a) Selection time (in seconds) (b) Reduction rate (in percentage)

Fig. 3. Box-plot representation for selection time and reduction (FS).

integrated in MOA software (16.04v) as an extension library [3]. MOA has also served as benchmark for our experiments.

5.2. *Feature selection*

Here, we evaluate how well selection of relevant features is performed by the streaming methods. As most of these approaches assume features are discrete, we have only selected from Table 4 those benchmarks with no numerical attributes. Please note that all these datasets comes from the text mining field, in which each attribute represents the presence or the absence of a given word. These datasets fits well for FS as the corpus of words/features is normally quite large.

Firstly, in Table 6 we measure the classification accuracy held by the three feature selectors considered in the experimental framework: IG, SU, and OFS; plus native NB using all features. From these results, we can conclude that:

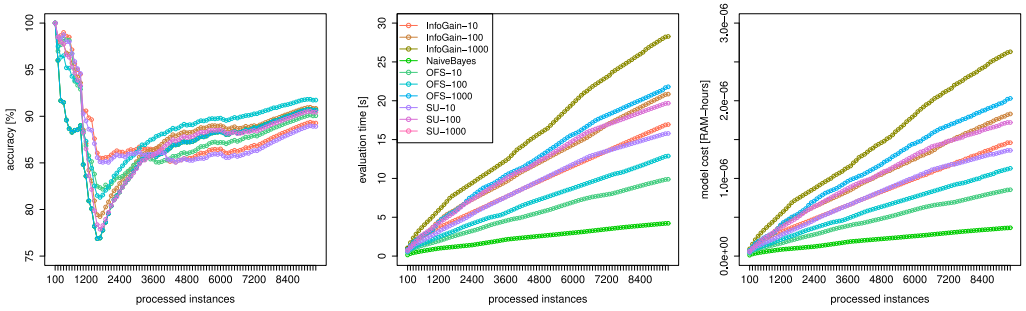
- NB yields better accuracy when all features are available during prediction. None of the selection schemes show better effectiveness than NB.
- However, IG and SU generate results pretty close to NB, with the advantage of generating much simpler solutions (as can be seen in Fig. 3b).
- Information-based methods are more accurate than OFS (based on feature weighting). Specially remarkable are the *spam_data* and *usenet_recurrent* cases, where even with only ten words the classifier is able to predict perfectly all examples.

To assert that no method is better than NB, we convey an statistical analysis on classification accuracy results through two non-parametric tests: Wilcoxon Signed-Ranks Test (one vs. one) and Friedman–Holm Test (one vs. all) [134,135]. Wilcoxon Test conducts pairwise comparisons between the reference method and the rest. A level of significance $\alpha = 0.05$ has been chosen for this experiment. The first two columns in Table 7 show Wilcoxon's results for accuracy, where '+' symbol indicates the number of methods outperformed by each algorithm in row. Symbol '±' represents the number of wins and ties yielded by each method. The best value by column is highlighted by a shaded background. The remaining columns show the results for the Friedman test. The first one shows effectiveness ranking of methods, ordered from the best mark (top row) to the worst. Note that the best method is established as the control algorithm. The second column contains the adjusted p-values for each method according to the post hoc Holm's test. The same level of significance ($\alpha = 0.05$) has been established for this test.

According to the results shown in Table 7, we can assert no method is significantly better than NB without discretization when using 10 features. As to 100 and 1000 features, the new outperforming method is SU although without showing statistically significance with respect to most of the alternatives.

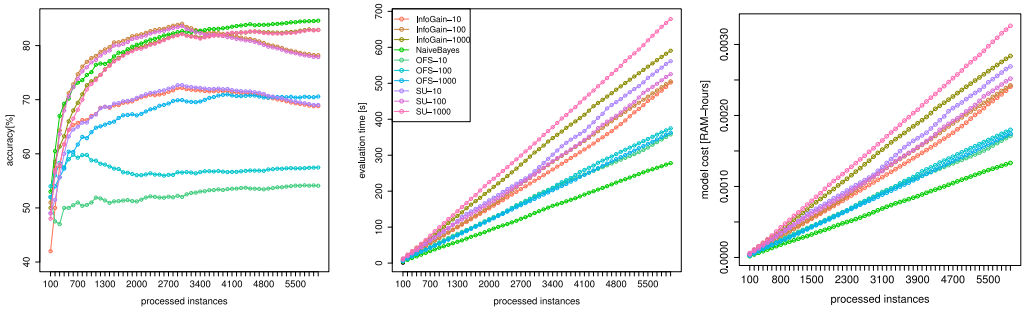
Fig. 3 depicts selection time spent by each algorithm, as well as the amount of reduction performed by each selection scheme, ranging from ten to one thousand features. No selection stands

[3] http://moa.cms.waikato.ac.nz/moa-extensions/



(a) Prequential accuracy. (b) CPU usage. (c) Memory usage.

Fig. 4. Plots of prequential accuracy (in %), CPU processing time (in seconds) and memory usage (in RAM-hours) over the data stream progress (processed instances) for feature selection methods on *spam_data* benchmark.



(a) Prequential accuracy. (b) CPU usage. (c) Memory usage.

Fig. 5. Plots of prequential accuracy (in %), CPU processing time (in seconds) and memory usage (in RAM-hours) over the data stream progress (processed instances) for feature selection methods on *usenet3* benchmark.

Table 7
Wilcoxon test results and average rankings of feature selectors (Friedman Procedure & Adjusted p-value with Holm's Test) for accuracy.

Algorithms	Accuracy +	Accuracy ±	Ranking	p_{Holm}
NB	1	3	7	–
OFS-10	0	2	16.7143	0.063201
InfoGain-10	0	3	17.1429	0.063201
SU-10	0	3	17.1429	0.063201

(a) 10 features selected

Algorithms	Accuracy +	Accuracy ±	Ranking	p_{Holm}
SU-100	0	3	11.6429	–
OFS-100	0	3	11.9286	0.017455
NB	0	3	17.2143	0.615351
InfoGain-100	0	3	17.2143	0.615351

(b) 100 feature selected

Algorithms	Accuracy +	Accuracy ±	Ranking	p_{Holm}
SU-1000	0	3	10.5714	–
OFS-1000	0	3	14.1429	0.487181
NB	0	3	16.5714	0.487181
InfoGain-1000	0	3	16.7143	0.487181

(c) 1,000 feature selected

as the fastest alternative. Despite the complete set of feature is used for predictions, this alternative offers better results due to the avoidance of feature relevance computations. Among the selection alternatives, OFS performs faster than the information-based selectors. However, OFS has shown in Table 6 to obtain less accurate schemes than its competitors.

Although a better reduction rate is achieved in *10-features* scheme (close to 100% in mean), by selecting 1000 features we can yield better accuracy, while the obtained reduction rate is still acceptable (> 25%). Please note that no selection is conducted on 4/7 problems when we choose the *1000-features* scheme since there are not enough attributes to select.

In conclusion, *SU-1000* can be elected as the best choice because of its competitive accuracy results similar to those yielded by NB and displayed reduction rates. Time results do not show significant differences between examined methods.

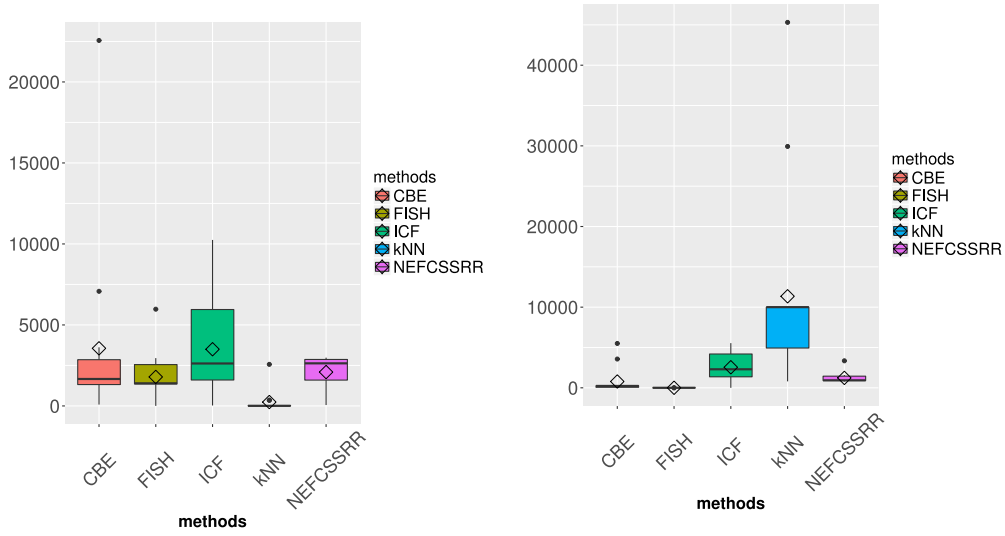
Detailed results on the entire data stream for *spam_data* and *usenet3* benchmarks with respect to obtained prequential accuracies, CPU usage and memory usage are depicted in Figs. 4 and 5.

5.3. Instance selection

Here, we evaluate how IS methods perform in non-stationary environments. As opposed to Section 4.1, in this experiment we have included datasets with both numerical and nominal attributes. In previous experiments [13] instance selectors were shown to be impractical when dealing with medium datasets. Because of that we have discarded those problems with a number of instances > 100, 000. Additionally, we have created new artificial datasets with a lower number of examples (10,000 instances).

Accuracy displayed by examined methods are given in Table 8. Table 9 shows results on accuracy for the Wilcoxon and Friedman–Holm test, following the same scheme presented in Section 4.1. From these results, we can conclude that:

- The best method on average is the updated kNN without selection (80.49%). The closest competitor (CBE) is five units below kNN. Other online methods, like FISH or ICF, do not respond well to concept drifts.
- No method is statistically better than updated kNN. Although kNN wins in each pairwise comparison in Wilcoxon tests, it only significantly overcomes ($\alpha = 0.05$) FISH and ICF according to Friedman–Holm tests.



(a) Selection time (in seconds)

(b) Reduction rate (in # instances selected)

Fig. 6. Box-plot representation for selection time and reduction (IS).

Table 8
Total test accuracy by method (IS).

	NEFCSSRR	ICF	CBE	FISH	kNN
elecNormNew	67.7652	43.7882	73.8105	60.0455	**84.0815**
powersupply	11.9400	4.2502	12.3800	5.4435	**15.1296**
spambase	81.6779	39.3827	**97.5440**	95.9139	95.1532
spam_data	88.8782	25.6113	91.1197	77.3488	**93.9833**
spam_nominal	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**
usenet1	**56.6667**	54.5333	54.0667	55.2000	56.4667
usenet2	61.2000	63.4667	48.4000	**69.8000**	68.2000
usenet_recurrent	**100.0000**	**100.0000**	**100.0000**	**100.0000**	**100.0000**
blips	90.8900	34.1300	94.2300	31.3200	**97.1800**
sudden_drift	76.5300	61.7300	74.2300	60.8700	**82.6600**
gradual_drift	68.2700	52.3600	74.4500	52.0200	**81.2700**
gradual_recurring_drift	87.5800	28.9400	92.6500	28.8400	**96.3300**
incremental_fast	65.8900	51.7700	68.4000	55.8300	**77.2800**
incremental_slow	72.4300	50.9800	68.7000	56.5800	**79.1000**
MEAN	73.5513	50.7816	74.9986	60.6580	**80.4882**

Table 9
Wilcoxon test results and average rankings of methods (Friedman Procedure & Adjusted p-value with Holm's Test) for accuracy.

Algorithms	Accuracy		Ranking	p_{Holm}
	+	±		
kNN	4	4	18.2143	–
CBE	2	3	26.1429	0.321710
NEFCSSRR	2	3	29	0.321710
FISH	0	1	47.5	0.000421
ICF	0	1	56.6429	0.000002

- Selection methods make decisions about the relevance or difficulty of a given instance without knowing the future state of the stream. It is normal that the no-selection option always performs better than others. It only depends on the amount of noise introduced by each problem and not by other factors like redundancy.

Regarding reduction and time, Fig. 3 depicts the distribution for both variables. From these plots, we can claim that CBE can be considered as the most accurate solution, and it also offers the highest reduction rates, at the cost of increased time complexity. NEFCSSRR also represents an interesting option as this method shows precise, and performs faster than CBE. The outstanding reduction rate of FISH is explained because it normally selects the kNN for each new example. This fact also explains its poor outcome on accuracy.

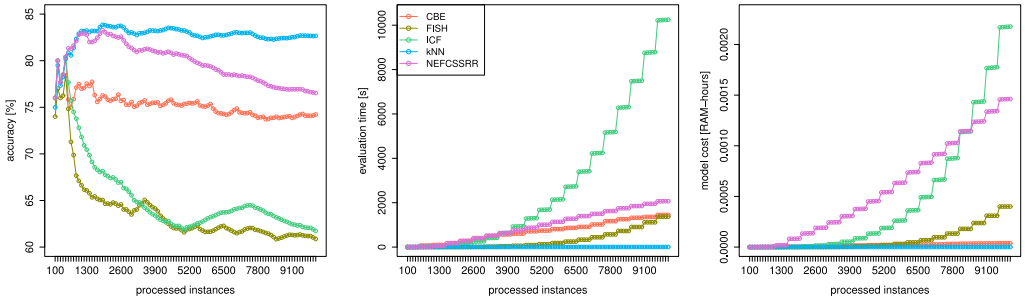
Detailed results on the entire data stream for *sudden_drift* and *gradual_drift* benchmarks with respect to obtained prequential accuracies, CPU usage and memory usage are depicted in Figs. 7 and 8.

Table 10
Classification test accuracy after discretization.

	PiD	IDA	OC	Naïve Bayes
airlines	63.0057	64.1563	**65.0723**	64.5504
powersupply	2.9237	13.5793	11.2938	**16.1087**
elecNormNew	71.9522	**76.6905**	74.0731	73.3625
spambase	**98.0439**	97.8700	97.6744	82.8081
kddcup_10	**99.1474**	98.4644	98.1404	97.1908
poker-lsn	55.0335	59.4337	58.5465	**59.5528**
covtypeNorm	**66.6306**	62.7235	64.2254	60.5208
blips	**74.5680**	66.4494	64.2148	60.9060
sudden_drift	65.7736	81.3168	77.8808	**83.8144**
gradual_drift_med	60.8404	82.8908	80.1032	**84.7000**
gradual_recurring_drift	**65.1678**	58.5250	58.5612	56.7450
incremental_fast	73.9900	75.6472	75.6036	**76.3642**
incremental_slow	65.6074	76.9186	75.4316	**78.0688**
MEAN	66.3603	**70.3589**	69.2939	68.8225

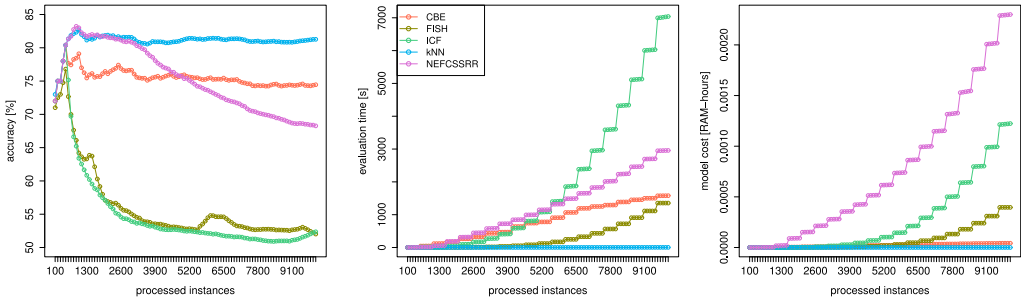
5.4. Discretization

To evaluate the ability of supervised discretizers to reduce the continuous feature space, we propose a new study with three discretization methods for data streams. NB and Incremental Discretization Algorithm (IDA) [11] have been elected as benchmark to assess the quality of supervised discretization schemes. The first one employs a gaussian estimation method, whereas the second one employs an unsupervised scheme based on quantile-estimation. In this experiment, only datasets with at least one numerical attribute have been considered. Email-based dataset used in Section 4.1 are thus discarded.



(a) Prequential accuracy. (b) CPU usage. (c) Memory usage.

Fig. 7. Plots of prequential accuracy (in %), CPU processing time (in s.) and memory usage (in RAM-hours) over the data stream progress (processed instances) for instance selection methods on *sudden_drift* benchmark.



(a) Prequential accuracy. (b) CPU usage. (c) Memory usage.

Fig. 8. Plots of prequential accuracy (in %), CPU processing time (in s.) and memory usage (in RAM-hours) over the data stream progress (processed instances) for instance selection methods on *gradual_drift* benchmark.

Table 11
Wilcoxon test results and average rankings of methods (Friedman Procedure & Adjusted p-value with Holm's Test) for accuracy.

Algorithms	Accuracy		Ranking	p_{Holm}
	+	±		
IDA	1	3	21.7692	–
OC	0	2	26	0.905821
Naïve Bayes	0	3	26.2308	0.905821
PiD	0	3	32	0.255677

Tables 10 and 11 contain test accuracy results for NB classification with and without explicit discretization. From these results, we can conclude the following statements:

- The most accurate method (in average) is IDA, an unsupervised method based on quantile-estimation and a sampling approach. However, its results are pretty close to those obtained by OC and NB. OC also outperforms the base solution, but with smaller margin than presented by IDA.
- According to the Wilcoxon test, we can statistically assert that IDA is only better than OC. Nevertheless this claim is rejected by Friedman's procedure, with a p-value far from the standard acceptance thresholds: 0.9 or 0.95. Although some improvement can be achieved by using supervised discretization, it can be deemed as superfluous and likely suboptimal.
- PiD represents the worst choice in this framework. Yet, it is specially remarkable that PiD is able to obtain the best accuracy mark in 5/13 datasets, with an outstanding mark in the blip dataset. This fact can be explained by the high number of parameters to be tuned in PiD and the high dependency on their values. Among the list of parameters, a global minimum and the maximum value need to be defined for the whole set of features, which is unfeasible in streaming environments. This parameter is essential as determines the expansion rate for new intervals, thus it may be possible to tailor it specifically for some datasets.

Apart from the previous deficiencies, Fig. 9a shows a high time-complexity of OC, as a result of a high number of data structures (binary tree, several queues, etc.) to be managed. IDA holds a similar time performance to NB.

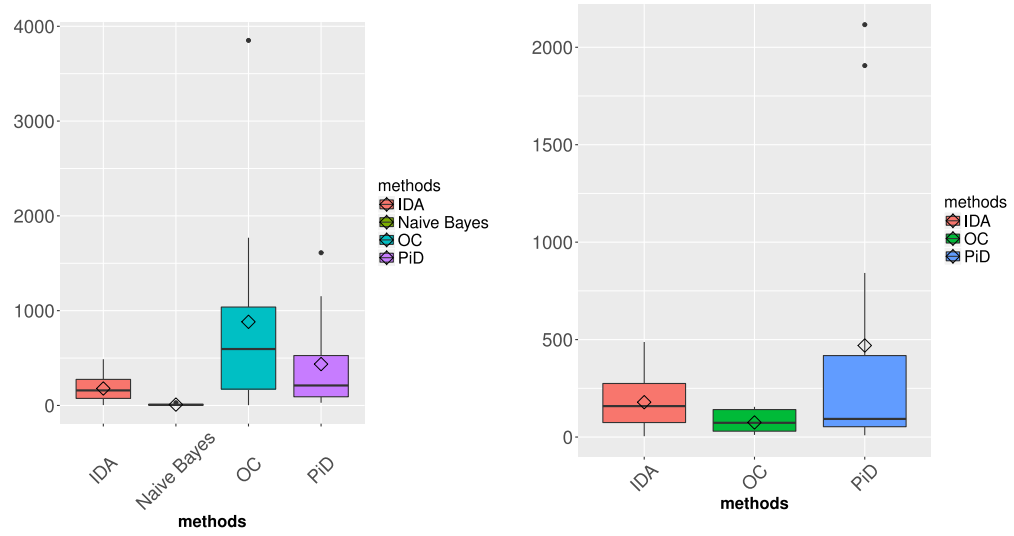
Fig. 9b illustrates the reduction performed by each method, represented as number of intervals generated per method. In this case, OC obtains the simplest solutions thanks to the control performed by χ^2. IDA defines the number of intervals before launching any process and PiD's inaccuracy is explained by a huge number of intervals generated initially ($\approx$ 500 per feature), as well as during the splitting process. Please notice that the subsequent merging process launched by the second layer is just not able to efficiently reduce such many input intervals.

As discussed before, evolving intervals sharply affect streaming classification since new and deleted intervals normally imply dramatic changes in the learning process. New models and techniques with a better interaction between discretization and classification must be designed if we want to transform online discretization into a truly useful tool for data analytics.

Detailed results on the entire data stream for *sudden_drift* and *gradual_drift* benchmarks with respect to obtained prequential accuracies, CPU usage and memory usage are depicted in Figs. 10 and 11.

6. Data preprocessing for data stream mining: lessons learned and future directions

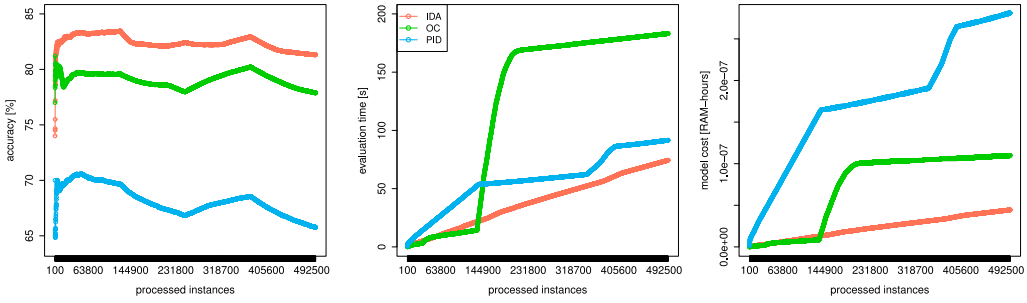
In this section we will discuss observations made on the basis of the presented survey of existing preprocessing methods for data streams, as well as the accompanying experimental study. Then, we will outline open challenges and future directions in this field.



(a) Selection time (in seconds)

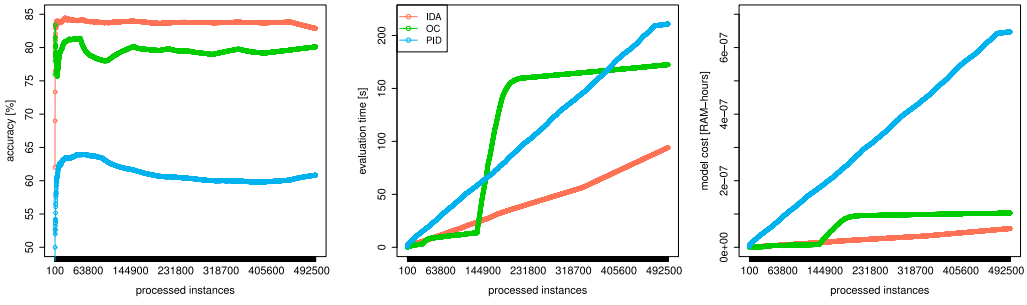
(b) Reduction rate (in # intervals generated)

Fig. 9. Box-plot representation for discretization time and reduction (discretization).



(a) Prequential accuracy. (b) CPU usage. (c) Memory usage.

Fig. 10. Plots of prequential accuracy (in %), CPU processing time (in s.) and memory usage (in RAM-hours) over the data stream progress (processed instances) for discretization methods on *sudden_drift* benchmark.



(a) Prequential accuracy. (b) CPU usage. (c) Memory usage.

Fig. 11. Plots of prequential accuracy (in %), CPU processing time (in s.) and memory usage (in RAM-hours) over the data stream progress (processed instances) for discretization methods on *gradual_drift* benchmark.

6.1. Lessons learned

Some important outcomes and guidelines can be inferred from the study, which we enumerate below:

- A wide range of phenomenons specific to data stream mining, ranging from concept-evolution to dynamic feature space, directly affects the features describing incoming instances. DXMiner is the only system that address all these problem through a combined strategy based on an information-based FS and an unsupervised selection method.
- As expected FS does not improve accuracy results presented by the option with the full set of features. Nevertheless FS solutions are able to yield simpler solutions with similar predictive performance, which is of crucial importance to stream mining frameworks. SU, the selector included in DXMiner, can be elected as the best method for FS because of its outstanding results in accuracy and its low complexity.

- Competence-based methods for IS tend to maintain case-bases polished in the highest degree, free of noise and redundancy. However they are characterized by a very high computational complexity. In distance-based solutions, this overhead is even bigger, while not being balanced by any gains in overall accuracy. In general, all instance selectors have shown an unfavorable behavior with respect to time and memory requirements, thus preventing their meaningful applications in high-speed data stream mining.
- CBE can be elected as best option for IS in terms of precision and reduction. NEFCSSRR also presents itself as an interesting option, with similar results to CBE, however requiring more computational resources.
- When selecting preprocessing methods for data stream mining we must consider not only the obtained accuracy, but also the computational costs that are associated with this method. As our study clearly showed some of the considered methods are characterized by bottlenecks in either CPU or memory usage, thus making them unsuitable for high-speed data streams.

6.2. Challenges and future directions

Here we outline the main challenges that should be addressed by the research community in order to obtain a meaningful progress in the area of preprocessing techniques for data stream problems:

- A scarce number of online and supervised discretizers have been proposed in the literature so far. Most of current methods are unsupervised techniques based on quantiles, using an adaptation strategy with smooth shifts in intervals' definition and the previous definition of intervals. Adding class information to the discretization process would allow to accommodate for local drifts, where properties of only some class changes. Additionally, we envision the potential of ensemble learning that will allow to use various discretization intervals to allow for training a diverse set of classifiers.
- Current online discretizers have shown to perform poorly as their adjustments tend to be more abrupt than those yielded by quantile-based techniques (see Section 4.3). However, this problem has been compensated by the inclusion of class information in the discretization process. Abrupt tweaks and labelings are two major concerns that must be addressed by further developments in this area. This shows that there is a need for combining discretization with active learning solutions. This would allow for selective labeling of only these samples that yield highest probability of influencing the intervals' definitions.
- No pure wrapper-based solutions have been proposed for online problems yet. Efficient implementations of these methods may be challenging die to their increased computational cost, but this may be compensated by the inherent discriminative ability of online learners and their adaptiveness to drifts. One potential solution would be to combine filter and wrapper approaches in order to reduce the number of times the more costly method will be used and to allow for continuous classification even during the wrapper computation. Another potential solution lies in using high-performance solutions based on GPU or distributed computing to reduce the computational load connected with this approach [136].
- There is a need for further research on feature and instance selection methods that can directly address the problem of concept drift. One way of approaching this would be to combine instance selection approaches with drift detection module that could directly influence the usability of prototypes. Whenever a strong drift is being detected, one may discard the previous prototypes and use only the incoming objects. After stream stabilizes, the instance selection can be repeated to adapt to the current concept. Another potential solution is to have weighted prototypes, where weight would reflect how long time ago they were created and how useful they are to mining current state of the stream. This would allow to smoothly forget outdated prototypes, while keeping in memory the ones still useful. Local drifts that occur only within a subset of classes should also be considered. In such a case only selected prototypes must be modified in the areas of drift presence. This would require class-based prototype pruning methods and a method to overlook the influence of these drifting prototypes on stationary classes.
- There is a need for further developing preprocessing methods characterized by a low computational requirements that would allow for a real-time decision making when dealing with big and high-speed data streams [137]. In case of data stream mining one must always balance the obtained accuracy with the amount of time spent on the computations. Therefore, developing approximate solutions with stopping criteria could be beneficial, especially in cases of sudden changes.
- There exist no solutions that directly take into account the possibility of recurrent concept drifts. Therefore, it seems promising to develop preprocessing methods that could accommodate the fact that previously used set of features / instances / discrete bins may become useful once again in the future. Simplest way of approaching this would be to create a secondary buffer storing these items for a certain amount of time, allowing to reuse them when necessary. To avoid unacceptable memory requirements this buffer should be flushed after a certain period of time with no action.
- There is a need to develop data preprocessing methods for more complex data stream types. Such techniques are crucial for imbalanced [138,139], multi-label [140] and multi-instance [141] problems and should be extended into the streaming framework.

7. Concluding remarks

We have presented a thorough survey of data reduction methods applied to data stream mining. Basic concepts, existing works, and present and future challenges have been analyzed in this work. Based on a number of relevant characteristics, we have proposed a simple, yet useful taxonomy of current developments in the online data preprocessing.

Most relevant methods have also been analyzed empirically through a conscious experimental framework, which includes a long and diverse list of artificial and real datasets with different types of drift. A statistical analysis based on non-parametric tests have been conveyed to support the resulting conclusions.

Concluding this work, we can claim that data preprocessing for data streams is still in its early days. New and more sophisticated methods that deal with previously unsolved challenges need to be designed in the years to follow. Great progress has been made in instance and feature selection, but other tasks like discretization remains yet to be properly addressed.

Acknowledgments

This work is supported by the Spanish National Research Project TIN2014-57251-P, the Foundation BBVA project 75/2016 BigDaPTOOLS, the Andalusian Research Plan P11-TIC-7765, and the Polish National Science Center under the grant no. DEC-2013/09/B/ST6/02264. S. Ramírez-Gallego holds a FPU scholarship from the Spanish Ministry of Education and Science (FPU13/00047).

References

[1] S. García, J. Luengo, F. Herrera, Data Preprocessing in Data Mining, Springer, 2015.
[2] S. García, J. Luengo, F. Herrera, Tutorial on practical tips of the most influential data preprocessing algorithms in data mining, Knowl. Based Syst. 98 (2016) 1–29.
[3] D. Pyle, Data Preparation for Data Mining, Morgan Kaufmann Publishers Inc., 1999.
[4] V. Mayer-Schnberger, K. Cukier, Big Data: A Revolution That Will Transform How We Live, Work and Think., 2013.
[5] S. García, S. Ramírez-Gallego, J. Luengo, J.M. Benítez, F. Herrera, Big data preprocessing: methods and prospects, Big Data Anal. 1 (1) (2016) 9.
[6] J.a. Gama, Knowledge Discovery from Data Streams, Chapman & Hall/CRC, 2010.
[7] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014) 44:1–44:37.
[8] I. Zliobaite, B. Gabrys, Adaptive preprocessing for streaming data, IEEE Trans. Knowl. Data Eng. 26 (2) (2014) 309–321.
[9] M.M. Masud, Q. Chen, J. Gao, L. Khan, J. Han, B. Thuraisingham, Classification and novel class detection of data streams in a dynamic feature space, in: Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part II, in: ECML PKDD'10, 2010, pp. 337–352.
[10] J.P. Barddal, H.M. Gomes, F. Enembreck, B. Pfahringer, A. Bifet, On dynamic feature weighting for feature drifting data streams, in: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19–23, 2016, Proceedings, Part II, 2016, pp. 129–144.
[11] G. Webb, Contrary to popular belief incremental discretization can be sound, computationally efficient and extremely useful for streaming data, in: IEEE International Conference on Data Mining (ICDM), 2014, pp. 1031–1036.
[12] V. Bolón-Canedo, N.S.-M. no, A. Alonso-Betanzos, Recent advances and emerging challenges of feature selection in the context of big data, Knowl. Based Syst. 86 (2015) 33–45.
[13] N. Lu, J. Lu, G. Zhang, R.L. de Mantaras, A concept drift-tolerant case-base editing technique, Artif. Intell. 230 (2016) 108–133.
[14] M.M. Gaber, Advances in data stream mining, Wiley Interdisc. Rew.: Data Min. Knowl. Discov. 2 (1) (2012) 79–85.
[15] E. Lughofer, P.P. Angelov, Handling drifts and shifts in on-line data streams with evolving fuzzy systems, Appl. Soft Comput. 11 (2) (2011) 2057–2068.
[16] L.I. Kuncheva, Classifier ensembles for detecting concept change in streaming data: overview and perspectives, in: 2nd Workshop SUEMA 2008 (ECAI 2008), 2008, pp. 5–10.
[17] D. Brzezinski, Block-based and Online Ensembles for Concept-drifting Data Streams, Poznan University of Technology, 2015 Ph.D. thesis.
[18] L.L. Minku, X. Yao, A.P. White, The impact of diversity on online ensemble learning in the presence of concept drift, IEEE Trans. Knowl. Data Eng. 22 (2009) 730–742.
[19] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, K. Ghédira, Self-adaptive windowing approach for handling complex concept drift, Cogn. Comput. 7 (6) (2015) 772–790, doi:10.1007/s12559-015-9341-0.
[20] J. Gama, P. Medas, G. Castillo, P.P. Rodrigues, Learning with drift detection, in: Advances in Artificial Intelligence - SBIA 2004, 17th Brazilian Symposium on Artificial Intelligence, São Luis, Maranhão, Brazil, 29 - October 1, 2004, Proceedings, 2004, pp. 286–295.
[21] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the Seventh SIAM International Conference on Data Mining, April 26–28, 2007, Minneapolis, Minnesota, USA, 2007, pp. 443–448.
[22] P. Sobolewski, M. Wońiak, Concept drift detection and model selection with simulated recurrence and ensembles of statistical detectors, J. Univ. Comput. Sci. 19 (4) (2013) 462–483.
[23] R.M.M. Vallim, R.F. de Mello, Proposal of a new stability concept to detect changes in unsupervised data streams, Expert Syst. Appl. 41 (16) (2014) 7350–7360.
[24] B.I.F. Maciel, S.G.T. de Carvalho Santos, R.S.M. de Barros, A lightweight concept drift detection ensemble, in: 27th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2015, Vietri sul Mare, Italy, 9–11, 2015, 2015, pp. 1061–1068.
[25] M. Wońiak, P. Ksieniewicz, B. Cyganek, K. Walkowiak, Ensembles of heterogeneous concept drift detectors - experimental study, in: Computer Information Systems and Industrial Management - 15th IFIP TC8 International Conference, CISIM 2016, Vilnius, Lithuania, 14–16, 2016, Proceedings, 2016, pp. 538–549.
[26] G. Hulten, L. Spencer, P.M. Domingos, Mining time-changing data streams, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, 26–29, 2001, 2001, pp. 97–106.
[27] J. Shan, J. Luo, G. Ni, Z. Wu, W. Duan, CVS: fast cardinality estimation for large-scale data streams over sliding windows, Neurocomputing 194 (2016) 107–116.
[28] B. Krawczyk, M. Wońiak, One-class classifiers with incremental learning and forgetting for data streams with concept drift, Soft Comput. 19 (12) (2015) 3387–3400.
[29] L. Du, Q. Song, X. Jia, Detecting concept drift: an information entropy based method using an adaptive sliding window, Intell. Data Anal. 18 (3) (2014) 337–364.
[30] O. Mimran, A. Even, Data stream mining with multiple sliding windows for continuous prediction, in: 22st European Conference on Information Systems, ECIS 2014, Tel Aviv, Israel, 9–11, 2014, 2014.
[31] P. Domingos, G. Hulten, Mining high-speed data streams, in: I. Parsa, R. Ramakrishnan, S. Stolfo (Eds.), Proceedings of the ACM Sixth International Conference on Knowledge Discovery and Data Mining, ACM Press, Boston, USA, 2000, pp. 71–80.
[32] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F.E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26.
[33] W.M. Czarnecki, J. Tabor, Online extreme entropy machines for streams classification and active learning, in: Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, Wroclaw, Poland, 25–27 May 2015, 2015, pp. 371–381.
[34] B. Lakshminarayanan, D.M. Roy, Y.W. Teh, Mondrian forests: efficient online random forests, in: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, 8–13 2014, Montreal, Quebec, Canada, 2014, pp. 3140–3148.
[35] M. Wońiak, Application of combined classifiers to data stream classification, in: Computer Information Systems and Industrial Management - 12th IFIP TC8 International Conference, CISIM 2013, Krakow, Poland, 25–27, 2013. Proceedings, 2013, pp. 13–23.
[36] M. Wońiak, M. Graña, E. Corchado, A survey of multiple classifier systems as hybrid systems, Inf. Fusion 16 (2014) 3–17.
[37] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.
[38] Y. Sun, K. Tang, L.L. Minku, S. Wang, X. Yao, Online ensemble learning of data streams with gradually evolved classes, IEEE Trans. Knowl. Data Eng. 28 (6) (2016) 1532–1545.
[39] G. Song, Y. Ye, H. Zhang, X. Xu, R.Y. Lau, F. Liu, Dynamic clustering forest: an ensemble framework to efficiently classify textual data stream with concept drift, Inf. Sci. 357 (2016) 125–143.
[40] L. Canzian, Y. Zhang, M. van der Schaar, Ensemble of distributed learners for online classification of dynamic data streams, IEEE Trans. Signal Inf. Process. Netw. 1 (3) (2015) 180–194.
[41] L.L. Minku, X. Yao, DDD: a new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. 24 (4) (2012) 619–633.
[42] L.L. Minku, A.P. White, X. Yao, The impact of diversity on online ensemble learning in the presence of concept drift, IEEE Trans. Knowl. Data Eng. 22 (5) (2010) 730–742.
[43] A. Bifet, G. Holmes, B. Pfahringer, Leveraging bagging for evolving data streams, in: Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, 20–24, 2010, Proceedings, Part I, 2010, pp. 135–150.
[44] D. Brzezinski, J. Stefanowski, Combining block-based and online methods in learning ensembles from concept drifting data streams, Inf. Sci. 265 (2014) 50–67.
[45] N. Japkowicz, M. Shah, Evaluating learning algorithms: a classification perspective, Cambridge University Press, 2011.
[46] A. Shaker, E. Hüllermeier, Recovery analysis for adaptive learning from non-stationary data streams: experimental design and case study, Neurocomputing 150 (2015) 250–264.
[47] J. Gama, R. Sebastião, P.P. Rodrigues, On evaluating stream learning algorithms, Mach. Learn. 90 (3) (2013) 317–346.
[48] D. Brzezinski, J. Stefanowski, Prequential AUC for classifier evaluation and drift detection in evolving data streams, in: New Frontiers in Mining Complex Patterns - Third International Workshop, NFMCP 2014, Held in Conjunction with ECML-PKDD 2014, Nancy, France, 19, 2014, Revised Selected Papers, 2014, pp. 87–101.
[49] M. Salehi, C. Leckie, J.C. Bezdek, T. Vaithianathan, X. Zhang, Fast memory efficient local outlier detection in data streams, IEEE Trans. Knowl. Data Eng. 28 (12) (2016) 3246–3260.
[50] I. Zliobaite, M. Budka, F.T. Stahl, Towards cost-sensitive adaptation: When is it worth updating your predictive model? Neurocomputing 150 (2015) 240–249.
[51] A. Bifet, G.D.F. Morales, J. Read, G. Holmes, B. Pfahringer, Efficient online evaluation of big data stream classifiers, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10–13, 2015, 2015, pp. 59–68.
[52] M. Wońiak, P. Ksieniewicz, B. Cyganek, A. Kasprzak, K. Walkowiak, Active learning classification of drifted streaming data, in: International Conference on Computational Science 2016, ICCS 2016, 6–8 June 2016, San Diego, California, USA, 2016, pp. 1724–1733.
[53] I. Zliobaite, A. Bifet, B. Pfahringer, G. Holmes, Active learning with drifting streaming data, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 27–39.
[54] Y. Dong, N. Japkowicz, Threaded ensembles of supervised and unsupervised neural networks for stream learning, in: Advances in Artificial Intelligence - 29th Canadian Conference on Artificial Intelligence, Canadian AI 2016, Victoria, BC, Canada, May 31 - 3, 2016. Proceedings, 2016, pp. 304–315.
[55] M.J. Hosseini, A. Gholipour, H. Beigy, An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams, Knowl. Inf. Syst. 46 (3) (2016) 567–597.

[56] B.S. Parker, L. Khan, Detecting and tracking concept class drift and emergence in non-stationary fast data streams, in: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, 25–30, 2015, Austin, Texas, USA., 2015, pp. 2908–2913.

[57] P. Sobolewski, M. Woźniak, Ldcnet: minimizing the cost of supervision for various types of concept drift, in: Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments, CIDUE 2013, IEEE Symposium Series on Computational Intelligence (SSCI), 16–19 April 2013, Singapore, 2013, pp. 68–75.

[58] G. Shikkenawis, S.K. Mitra, 2D orthogonal locality preserving projection for image denoising, IEEE Trans. Image Process. 25 (1) (2016) 262–273.

[59] A.A. Mohamad AL-Shiha, W. Woo, S. Dlay, Multi-linear neighborhood preserving projection for face recognition, Pattern Recogn. 47 (2) (2014) 544–555.

[60] H. Zhang, Q.M. Jonathan Wu, T.W.S. Chow, M. Zhao, A two-dimensional neighborhood preserving projection for appearance-based face recognition, Pattern Recogn. 45 (5) (2012) 1866–1876.

[61] G. Doquire, M. Verleysen, Feature selection with missing data using mutual information estimators, Neurocomputing 90 (2012) 3–11.

[62] V. Lopez, I. Triguero, C.J. Carmona, S. Garcia, F. Herrera, Addressing imbalanced classification with instance generation techniques: ipade-id, Neurocomputing 126 (2014) 15–28.

[63] A. Ferreira, M. Figueiredo, Incremental filter and wrapper approaches for feature discretization, Neurocomputing 123 (2014) 60–74.

[64] Y. Yang, G.I. Webb, Discretization for Naive–Bayes learning: managing discretization bias and variance, Mach. Learn. 74 (1) (2009) 39–74.

[65] H.-W. Hu, Y.-L. Chen, K. Tang, A dynamic discretization approach for constructing decision trees with a continuous label, IEEE Trans. Knowl. Data Eng. 21 (11) (2009) 1505–1514.

[66] A. Cano, D.T. Nguyen, S. Ventura, K.J. Cios, ur-caim: improved CAIM discretization for unbalanced and balanced data, Soft Comput. 20 (1) (2016) 173–188.

[67] A. Cano, J.M. Luna, E.L.G. Galindo, S. Ventura, LAIM discretization for multi--label data, Inf. Sci. 330 (2016b) 370–384.

[68] X. Wu, K. Yu, H. Wang, W. Ding, Online streaming feature selection, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 1159–1166.

[69] S. Eskandari, M. Javidi, Online streaming feature selection using rough sets, Int. J. Approx. Reason. 69 (C) (2016) 35–57.

[70] I. Katakis, G. Tsoumakas, I.P. Vlahavas, On the utility of incremental feature selection for the classification of textual data streams, in: Advances in Informatics, 10th Panhellenic Conference on Informatics, PCI 2005, Volos, Greece, November 11–13, 2005, Proceedings, 2005, pp. 338–348.

[71] J.P. Barddal, H.M. Gomes, F. Enembreck, A survey on feature drift adaptation, in: IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI), 2015, pp. 1053–1060.

[72] H.-L. Nguyen, Y.-K. Woon, W.-K. Ng, L. Wan, Heterogeneous ensemble for feature drifts in data streams, in: Proceedings of the 16th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part II, in: PAKDD'12, 2012, pp. 1–12.

[73] V.R. Carvalho, W.W. Cohen, Single-pass online learning: performance, voting schemes and online feature selection, in: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, in: KDD '06, 2006, pp. 548–553.

[74] J. Gomes, M. Gaber, P. Sousa, E. Menasalvas, Mining recurring concepts in a dynamic feature space, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 95–110.

[75] X. Wu, K. Yu, W. Ding, H. Wang, X. Zhu, Online feature selection with streaming features, IEEE Trans. Pattern Anal. Mach. Intell. 35 (5) (2013) 1178–1192.

[76] S.C.H. Hoi, J. Wang, P. Zhao, R. Jin, Online feature selection for mining big data, in: Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications, BigMine 2012, Beijing, China, 12, 2012, 2012, pp. 93–100.

[77] J. Wang, P. Zhao, S. Hoi, R. Jin, Online feature selection and its applications, IEEE Trans. Knowl. Data Eng. 26 (3) (2014) 698–710.

[78] J. Wang, M. Wang, P. Li, L. Liu, Z. Zhao, X. Hu, X. Wu, Online feature selection with group structure analysis, IEEE Trans. Knowl. Data Eng. 27 (11) (2015) 3029–3041.

[79] H. Li, X. Wu, Z. Li, W. Ding, Online group feature selection from feature streams, in: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, 14–18, 2013, Bellevue, Washington, USA., 2013.

[80] J. Yan, B. Zhang, N. Liu, S. Yan, Q. Cheng, W. Fan, Q. Yang, W. Xi, Z. Chen, Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing, IEEE Trans. Knowl. Data Eng. 18 (2) (2006) 320–333.

[81] Y. Tadeuchi, R. Oshima, K. Nishida, K. Yamauchi, T. Omori, Quick online feature selection method for regression -a feature selection method inspired by human behavior-, in: Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Montréal, Canada, 7–10 2007, 2007, pp. 1895–1900.

[82] Y. Cai, Y. Sun, J. Li, S. Goodison, Online feature selection algorithm with bayesian l1 regularization, in: Advances in Knowledge Discovery and Data Mining, 13th Pacific-Asia Conference, PAKDD 2009, Bangkok, Thailand, 27–30, 2009, Proceedings, 2009, pp. 401–413.

[83] K. Ooi, T. Ninomiya, Efficient online feature selection based on l1-regularized logistic regression, in: ICAART 2013 - Proceedings of the 5th International Conference on Agents and Artificial Intelligence, Volume 2, Barcelona, Spain, 15–18, 2013, 2013, pp. 277–282.

[84] W. Fan, N. Bouguila, Online learning of a dirichlet process mixture of generalized dirichlet distributions for simultaneous clustering and localized feature selection, in: Proceedings of the 4th Asian Conference on Machine Learning, ACML 2012, Singapore, Singapore, 4–6, 2012, 2012, pp. 113–128.

[85] W. Fan, N. Bouguila, Online variational learning of generalized dirichlet mixture models with feature selection, Neurocomputing 126 (2014) 166–179.

[86] O. Amayri, N. Bouguila, On online high-dimensional spherical data clustering and feature selection, Eng. Appl. AI 26 (4) (2013) 1386–1398.

[87] Z. Yao, W. Liu, Extracting robust distribution using adaptive gaussian mixture model and online feature selection, Neurocomputing 101 (2013) 258–274.

[88] H. Yang, M.R. Lyu, I. King, Efficient online learning for multitask feature selection, Trans. Knowl. Discov. Data 7 (2) (2013) 6.

[89] K. Yu, X. Wu, W. Ding, J. Pei, Towards scalable and accurate online feature selection for big data, in: 2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, 14–17, 2014, 2014, pp. 660–669.

[90] A. Roy, Automated online feature selection and learning from high-dimensional streaming data using an ensemble of kohonen neurons, in: 2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12–17, 2015, 2015, pp. 1–8.

[91] H. Yang, R. Fujimaki, Y. Kusumura, J. Liu, Online feature selection: a limited-memory substitution algorithm and its asynchronous parallel variation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17, 2016, 2016, pp. 1945–1954.

[92] M. Hammoodi, F.T. Stahl, M. Tennant, Towards online concept drift detection with feature selection for data stream classification, in: ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016), 2016, pp. 1549–1550.

[93] S. Eskandari, M.M. Javidi, Online streaming feature selection using rough sets, Int. J. Approx. Reason. 69 (2016) 35–57.

[94] V. Bolón-Canedo, D. Fernández-Francos, D. Peteiro-Barral, A. Alonso-Betanzos, B. Guijarro-Berdiñas, N. Sánchez-Maroño, A unified pipeline for online feature selection and classification, Expert Syst. Appl. 55 (2016) 532–545.

[95] Y. Yeh, C. Hsu, Online selection of tracking features using adaboost, IEEE Trans. Circuits Syst. Video Technol. 19 (3) (2009) 442–446.

[96] J. Yang, K. Zhang, Q. Liu, Robust object tracking by online fisher discrimination boosting feature selection, Comput. Vis. Image Underst. 153 (2016) 100–108.

[97] K. Yu, W. Ding, X. Wu, LOFS: a library of online streaming feature selection, Knowl.-Based Syst. 113 (2016) 1–3.

[98] I. Jolliffe, Principal Component Analysis, Springer Verlag, 1986.

[99] J. Nie, W. Kotlowski, M.K. Warmuth, Online PCA with optimal regret, J. Mach. Learn. Res. 17 (173) (2016) 1–49.

[100] P. Jain, C. Jin, S.M. Kakade, P. Netrapalli, A. Sidford, Streaming PCA: matching matrix bernstein and near-optimal finite sample guarantees for oja's algorithm, in: Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23–26, 2016, 2016, pp. 1147–1164.

[101] E. Hazan, S. Kale, M.K. Warmuth, On-line variance minimization in $O(n^2)$ per trial, in: Proceedings of the 23rd Annual Conference on Learning Theory, in: COLT '10, 2010, pp. 314–315.

[102] A.A. Joseph, T. Tokumoto, S. Ozawa, Online feature extraction based on accelerated kernel principal component analysis for data stream, Evol. Syst. 7 (1) (2016) 15–27.

[103] M. Ghashami, D.J. Perry, J.M. Phillips, Streaming kernel principal component analysis, in: Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS 2016, Cadiz, Spain, 9–11, 2016, 2016, pp. 1365–1374.

[104] L.I. Kuncheva, W.J. Faithfull, PCA feature extraction for change detection in multidimensional unlabelled streaming data, in: Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012, Tsukuba, Japan, 11–15, 2012, 2012, pp. 1140–1143.

[105] A.A. Qahtan, B. Alharbi, S. Wang, X. Zhang, A pca-based change detection framework for multidimensional data streams: change detection in multidimensional data streams, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10–13, 2015, 2015, pp. 935–944.

[106] A. Allahyar, H.S. Yazdi, Online discriminative component analysis feature extraction from stream data with domain knowledge, Intell. Data Anal. 18 (5) (2014) 927–951.

[107] F. Sheikholeslami, D. Berberidis, G.B. Giannakis, Kernel-based low-rank feature extraction on a budget for big data streams, in: 2015 IEEE Global Conference on Signal and Information Processing, GlobalSIP 2015, Orlando, FL, USA, 14–16, 2015, 2015, pp. 928–932.

[108] W. Li, J. Yang, J. Zhang, Uncertain canonical correlation analysis for multi-view feature extraction from uncertain data streams, Neurocomputing 149 (2015) 1337–1347.

[109] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, IEEE Trans. Inf. Theory 13 (1) (1967) 21–27.

[110] S. García, J. Derrac, J. Cano, F. Herrera, Prototype selection for nearest neighbor classification: taxonomy and empirical study, IEEE Trans. Pattern Anal. Mach. Intell. 34 (3) (2012) 417–435.

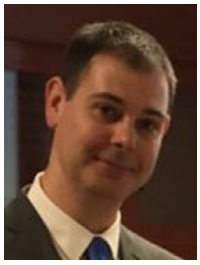
[111] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66.

[112] M. Salganicoff, Density-adaptive learning and forgetting, in: Machine Learning Proceedings 1993, Morgan Kaufmann, 1993, pp. 276–283.

[113] R. Klinkenberg, Learning drifting concepts: example selection vs. example weighting, Intell. Data Anal. 8 (3) (2004) 281–300.
[114] M. Salganicoff, Tolerating concept and sampling shift in lazy learning using prediction error context switching, Artif. Intell. Rev. 11 (1) (1997) 133–155.
[115] J. Beringer, E. Hüllermeier, Efficient instance-based learning on data streams, Intell. Data Anal. 11 (6) (2007) 627–650.
[116] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Min. Knowl. Discov. 6 (2) (2002) 153–172.
[117] I. Tomek, Two modifications of CNN, IEEE Trans. Syst., Man, Cybern. 6 (11) (1976) 769–772.
[118] S.J. Delany, P. Cunningham, A. Tsymbal, L. Coyle, A case-based technique for tracking concept drift in spam filtering, Knowl. Based Syst. 18 (45) (2005) 187–195.
[119] B. Smyth, M.T. Keane, Remembering to forget: a competence-preserving case deletion policy for case-based reasoning systems, in: Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1, in: IJCAI'95, 1995, pp. 377–382.
[120] A. Shaker, E. Hüllermeier, Iblstreams: a system for instance-based classification and regression on data streams, Evolv. Syst. 3 (4) (2012) 235–249.
[121] I. Žliobaitė, Combining similarity in time and space for training set formation under concept drift, Intell. Data Anal. 15 (4) (2011) 589–611.
[122] L. Zhao, L. Wang, Q. Xu, Data stream classification with artificial endocrine system, Appl. Intell. 37 (3) (2012) 390–404.
[123] K.B. Dyer, R. Capo, R. Polikar, Compose: a semisupervised learning framework for initially labeled nonstationary streaming data, IEEE Trans. Neural Netw. Learn. Syst. 25 (1) (2014) 12–26.
[124] D. Mena-Torres, J.S. Aguilar-Ruiz, A similarity-based approach for data stream classification, Expert Syst. Appl. 41 (9) (2014) 4224–4234.
[125] Y. Ben-Haim, E. Tom-Tov, A streaming parallel decision tree algorithm, J. Mach. Learn. Res. 11 (2010) 849–872.
[126] A. Gupta, F.X. Zane, Counting inversions in lists, in: Proceedings of the 14th Annual ACM-SIAM Symp. on Discrete Algorithms, 2003, pp. 253–254.
[127] S. Guha, A. McGregor, Stream order and order statistics: quantile estimation in random-order streams, SIAM J. Comput. 38 (5) (2009) 2044–2059.
[128] J. Lu, Y. Yang, G.I. Webb, Incremental discretization for Naïve-bayes classifier, in: Proceedings of the Second International Conference on Advanced Data Mining and Applications, in: ADMA'06, 2006, pp. 223–238.
[129] J. Gama, C. Pinto, Discretization from data streams: applications to histograms and data mining, in: Proceedings of the 2006 ACM Symposium on Applied Computing, in: SAC '06, 2006, pp. 662–667.
[130] P. Lehtinen, M. Saarela, T. Elomaa, Online ChiMerge Algorithm, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 199–216.
[131] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, MOA: massive online analysis, J. Mach. Learn. Res. 11 (2010) 1601–1604.
[132] M. Lichman, UCI machine learning repository, 2013, [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.
[133] A. Bifet, R. Kirkby, Data stream mining: a practical approach, Technical Report, The University of Waikato, 2009.
[134] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2009) 959–977.
[135] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evolut. Comput. 1 (1) (2011) 3–18.
[136] A. Cano, A. Zafra, S. Ventura, Solving classification problems using genetic programming algorithms on gpus, in: Hybrid Artificial Intelligence Systems, 5th International Conference, HAIS 2010, San Sebastián, Spain, 23–25, 2010. Proceedings, Part II, 2010, pp. 17–26.
[137] S. García, S. Ramírez-Gallego, J. Luengo, J.M. Benítez, F. Herrera, Big data preprocessing: methods and prospects, Big Data Anal. 1 (1) (2016) 9. URL http://dx.doi.org/10.1186/s41044-016-0014-0.
[138] B. Krawczyk, Learning from imbalanced data: open challenges and future directions, Progr. Artif. Intell. 5 (4) (2016) 221–232.
[139] V. López, A. Fernández, S. García, V. Palade, F. Herrera, An insight into classification with imbalanced data: empirical results and current trends on using data intrinsic characteristics, Inf. Sci. 250 (2013) 113–141.
[140] F. Herrera, F. Charte, A.J. Rivera, M.J. del Jesús, Multilabel Classification - Problem Analysis, Metrics and Techniques, Springer, 2016.
[141] F. Herrera, S. Ventura, R. Bello, C. Cornelis, A. Zafra, D.S. Tarragó, S. Vluymans, Multiple Instance Learning - Foundations and Algorithms, Springer, 2016.

Sergio Ramırez-Gallego received the M.Sc. degree in Computer Science in 2012 from the University of Jaén, Spain. He is currently a Ph.D. student at the Department of Computer Science and Artificial Intelligence, University of Granada, Spain. His research interests include data mining, data preprocessing, big data and cloud computing.

Bartosz Krawczyk is an assistant professor in the Department of Computer Science, Virginia Commonwealth University, Richmond VA, USA, where he heads the Machine Learning and Stream Mining Lab. He obtained his MSc and PhD degrees from Wroclaw University of Science and Technology, Wroclaw, Poland, in 2012 and 2015 respectively. His research is focused on machine learning, data streams, ensemble learning, class imbalance, one-class classifiers, and interdisciplinary applications of these methods. He has authored 35+ international journal papers and 80+ contributions to conferences. Dr Krawczyk was awarded with numerous prestigious awards for his scientific achievements like IEEE Richard Merwin Scholarship and IEEE Outstanding Leadership Award among others. He served as a Guest Editor in four journal special issues and as a chair of ten special session and workshops. He is a member of Program Committee for over 40 international conferences and a reviewer for 30 journals.



Salvador Garcıa received the M.Sc. and Ph.D. degrees in Computer Science from the University of Granada, Granada, Spain, in 2004 and 2008, respectively. He is currently an Associate Professor in the Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain. He has published more than 45 papers in international journals. As edited activities, he has co-edited two special issues in international journals on different Data Mining topics and is a member of the editorial board of the Information Fusion journal. He is a co-author of the book entitled "Data Preprocessing in Data Mining" published in Springer. His research interests include data mining, data preprocessing, data complexity, imbalanced learning, semi-supervised learning, statistical inference, evolutionary algorithms and biometrics.

Michal Wozniak is a professor of computer science at the Department of Systems and Computer Networks, Wroclaw University of Science and Technology, Poland. He received M.Sc. degree in biomedical engineering from the Wroclaw University of Technology in 1992, and Ph.D. and D.Sc. (habilitation) degrees in computer science in 1996 and 2007, respectively, from the same university. In 2015 he was nominated as the professor by President of Poland. His research focuses on compound classification methods, hybrid artificial intelligence and medical informatics. Prof. Wozniak has published over 260 papers and three books. His recent one Hybrid classifiers: Method of Data, Knowledge, and Data Hybridization was published by Springer in 2014. He has been involved in research projects related to the abovementioned topics and has been a consultant of several commercial projects for well-known Polish companies and public administration. Prof. Wozniak is a senior member of the IEEE.

Francisco Herrera (SM'15) received his M.Sc. in Mathematics in 1988 and Ph.D. in Mathematics in 1991, both from the University of Granada, Spain. He is currently a Professor in the Department of Computer Science and Artificial Intelligence at the University of Granada.
He has been the supervisor of 40 Ph.D. students. He has published more than 300 journal papers that have received more than 49,000 citations (Scholar Google, H-index 112). He is coauthor of the books "Genetic Fuzzy Systems" (World Scientific, 2001) and "Data Preprocessing in Data Mining" (Springer, 2015), "The 2-tuple Linguistic Model. Computing with Words in Decision Making" (Springer, 2015), "Multilabel Classification. Problem analysis, metrics and techniques" (Springer, 2016), "Multiple Instance Learning. Foundations and Algorithms"(Springer, 2016). He currently acts as Editor in Chief of the international journals "Information Fusion" (Elsevier) and "Progress in Artificial Intelligence (Springer). He acts as editorial member of a dozen of journals.
He received the following honors and awards: ECCAI Fellow 2009, IFSA Fellow 2013, 2010 Spanish National Award on Computer Science ARITMEL to the "Spanish Engineer on Computer Science", International Cajastur "Mamdani" Prize for Soft Computing (Fourth Edition, 2010), IEEE Transactions on Fuzzy System Outstanding 2008 and 2012 Paper Award (bestowed in 2011 and 2015 respectively), 2011 Lotfi A. Zadeh Prize Best paper Award of the International Fuzzy Systems Association, 2013 AEPIA Award to a scientific career in Artificial Intelligence, and 2014 XV Andalucía Research Prize Maimónides (by the regional government of Andalucía).
His current research interests include among others, soft computing (including fuzzy modeling and evolutionary algorithms), information fusion, decision making, biometric, data preprocessing, data science and big data.