Received: 29 February 2020 | Revised: 15 May 2020 | Accepted: 5 June 2020

DOI: 10.1002/widm.1381

ADVANCED REVIEW

WIREs DATA MINING AND KNOWLEDGE DISCOVERY WILEY

An overview of unsupervised drift detection methods

Rosana Noronha Gemaque[1,2] | **Albert França Josuá Costa**[1,3] | **Rafael Giusti**[1] | **Eulanda Miranda dos Santos**[1]

[1]Institute of Computing, Federal University of Amazonas, Manaus, Amazonas, Brazil

[2]National Institute for Amazonian Research, Manaus, Amazonas, Brazil

[3]Federal Institute of Amazonas - Zona Leste, Manaus, Amazonas, Brazil

Correspondence
Eulanda Miranda dos Santos, Institute of Computing, Federal University of Amazonas, Manaus, Amazonas 69077-000, Brazil.
Email: emsantos@icomp.ufam.edu.br

Funding information
Foundation for Research Support of the State of Amazonas (FAPEAM, Grant/Award Number: 009/2012, 009/2017 and 005/2019 PAPAC Project); Samsung Electronics of Amazonia Ltda, Grant/Award Number: 8.387/1991

Abstract

Practical applications involving big data, such as weather monitoring, identification of customer preferences, Internet log analysis, and sensors warnings require challenging data analysis, since these are examples of problems whose data are generated in streams and usually demand real-time analytics. Patterns in such data stream problems may change quickly. Consequently, machine learning models that operate in this context must be updated over time. This phenomenon is called concept drift in machine learning and data mining literature. Several different directions have been pursued to learn from data stream and to deal with concept drift. However, most drift detection methods consider that an instance's class label is available right after its prediction, since these methods work by monitoring the prediction results of a base classifier or an ensemble of classifiers. Nevertheless, this constraint is unrealistic in several practical problems. To cope with this constraint, some works are focused on proposing efficient unsupervised or semi-supervised concept drift detectors. While interesting and recent overview papers dedicated to supervised drift detectors have been published, the scenario is not the same in terms of unsupervised methods. Therefore, this work presents a comprehensive overview of approaches that tackle concept drift in classification problems in an unsupervised manner. Additional contribution includes a proposed taxonomy of state-of-the-art approaches for concept drift detection based on unsupervised strategies.

This article is categorized under:
Technologies > Classification
Technologies > Machine Learning

KEYWORDS

classification, concept drift detectors, data streams, machine learning, nonstationary environments, online learning, overview

Rosana Noronha Gemaque, Albert França Josuá Costa, Rafael Giusti, and Eulanda Miranda dos Santos contributed equally to this study.

1 | INTRODUCTION

In machine learning, the way data must be processed depends on specific characteristics determined by how data are accessed and their availability. In the case of data streams, they differ from other forms of data in the sense that instances arrive continuously and sequentially. Over time, the underlying data distribution of the data may change dynamically. This phenomenon is known as concept drift, and it represents a challenging issue that can impact several application domains. This fact may explain the high number of surveys published during the last decade, as summarized in Figure 1, focusing on studying approaches to cope with concept drift in different application areas, including health—surgery prediction Beyene, Welemariam, Persson, & Lavesson, 2015) and sleep quality assessment (Chiang & Wu, 2018), information security—spam filtering (Guzella & Caminhas, 2009), fraud detection (Abdallah, Maarof, & Zainal, 2016), and cybersecurity (Bobowska, Chorás, & Woźniak, 2018), and data-driven soft sensors (Kadlec, Grbić, & Gabrys, 2011). In general, methods described in these works are not exclusively machine learning-based, since they are problem-dependent methods.

Surveys on machine learning-based solutions have also attracted the interest of researchers (Gama, Zliobaite, Bifet, Pechenizkiy, & Bouchachia, 2014; Hu, Kantardzic, & Sethi, 2019; Lu et al., 2018). However, despite the increasing volume of concept drift survey publications in this context, the vast majority of them focus on supervised approaches. In fact, several supervised concept drift methods have been compiled in surveys such as (Lu et al., 2018) and (Barros & Santos, 2018). For instance, Barros and Santos (2018) present a large-scale comparison of 14 supervised concept drift detectors. They indicate that semi-supervised and/or unsupervised approaches could be evaluated in future studies. Krawczyk, Minku, Gama, Stefanowski, and Woźniak (2017) and Barros and Santos (2019) discuss concept drift in data stream mainly focused on ensemble-based approaches, therefore, they are not primarily concerned with unsupervised methods. Iwashita and Papa (2019) emphasize the booming concentration of publications on supervised approaches (roughly 85% supervised, 12% semi-supervised, and 5% unsupervised). Finally, according to Lu et al. (2018), which surveyed more than 130 publications on concept drift, unsupervised or semi-supervised drift detection and adaptation is a very unexplored research area.

On the other hand, considering real-world problems, it is more realistic to try to solve the concept drift problem in an unsupervised way, since having the ground truth of the arriving instances available immediately after their classification is most often not possible. To the best of our knowledge, there is no survey describing state-of-the-art unsupervised concept drift approaches and proposing a specific taxonomy to these methods. This survey aims to fill this literature gap.

This survey was designed to answer the following research questions: (RQ1) Which are the unsupervised drift detection approaches proposed in the literature? (RQ2) Were these approaches proposed for online or batch data streams? (RQ3) How can we categorize these approaches into a meaningful taxonomy? (RQ4) What is the contribution of the proposed taxonomy to existing taxonomies in the literature?

To address these research questions, we followed a systematic review methodology (Okoli, 2015). The selection of references in this survey paper was performed according to a search string in the Scopus metasearch engine (www.scopus.com). The first search string was designed to type overviews, surveys, and reviews addressing concept drift from

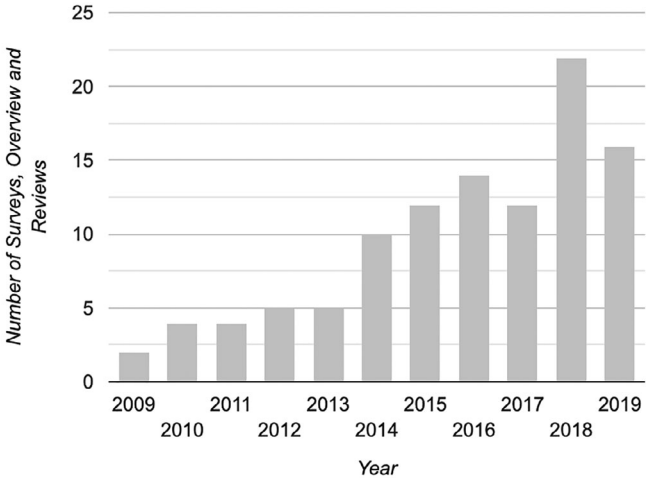

FIGURE 1 The increasing number of publications as surveys, overviews, and reviews on concept drift detection published in the last decade

2009 through 2019, which resulted in 106 documents. Then, the search was further refined to include only papers in English published in journals or conference proceedings. This new step led to 78 documents. A first verification in this group of documents demonstrated a higher concentration of papers published after 2014. Therefore, to review more recent advances in concept drift detection using unsupervised strategies, we limited the search to the period between 2015 and 2019. This search returned 55 documents. The result is that, in these articles, 16 unsupervised concept drift detection methods are described, as shown in Tables 1 and 2. These methods are detailed in Sections 3 and 4. The main steps of this methodology are described as follows.

- *Step 1*: Search engine. Both searches were conducted using Scopus meta search (www.scopus.com), which searches in Science Direct (www.sciencedirect.com), ACM Digital Library (dl.acm.org), IEEE Xplore (ieeexplore.ieee.org), and Springer Link (link.springer.com) databases.
- *Step 2*: Documents selection. Preliminary triage of articles: The first search was based on keywords. The articles were then selected as references if they (a) present new theory, algorithm, or methodology in the area of concept drift; or (b) report a concept drift application.
- *Step 3*: Result filtering for articles: The articles selected in Step 2 were divided into two groups: Batch and online concept drift detection. The references in each group were filtered again, based on the period of publishing (within the last 5 years).
- *Step 4*: Methods selection. The last step screens all documents to extract only papers that propose a semi-supervised or unsupervised drift detection component in its framework.

TABLE 1 Categorization of batch-based concept drift detection methods

Category	Subcategory	Method	References
Batch		NN-DVI	Liu, Lu, Liu, and Zhang (2018)
	Whole-batch detection	FAAD	Li, Wang, Yang, Li, and Ma (2019)
		UDetect	Bashir, Petrovski, and Doolan (2017)
		SQSI-IS	Maletzke, dos Reis, and Batista (2018)
		MD3	Sethi and Kantardzic (2015)
		MD3-EGM*	Sethi and Kantardzic (2017)
	Partial-batch detection	PDetect*	Sethi and Kantardzic (2018)
		DDAL	Costa, Albuquerque, and Santos (2018)

Note: Names with an asterisk were introduced in this survey because the methods were not given any name in the referenced works.

TABLE 2 Categorization of online-based concept drift detection methods

Category	Subcategory	Method	References
Online	Fixed reference window	IKS-bdd*	dos Reis, Flach, Matwin, and Batista (2016)
		CD-TDS	Koh (2016)
		OMV-PHT*	Lughofer, Weigl, Heidl, Eitzinger, and Radauer (2016)
		NM-DDM*	Mustafa et al. (2017)
	Sliding reference window	Plover	de Mello, Vaz, Grossi, and Bifet (2019)
		SAND	Haque, Khan, and Baron (2016)
		DSDD*	Pinagé, dos Santos, and Gama (2019)
	Multiple approaches	DbDDA*	Kim and Park (2017)

Note: Names with an asterisk were introduced in this survey because the methods were not given any name in the referenced works.

The main contribution of this survey is to propose a taxonomy of state-of-the-art approaches for concept drift detection based on unsupervised strategies, as well as to present a comprehensive overview of approaches that tackle concept drift in classification problems in an unsupervised manner.

The remainder of this article is organized as follows. Section 2 discusses taxonomies existing in the literature and presents the taxonomy proposed in this paper for unsupervised concept drift detection methods. Section 3 presents the first group of drift detectors, named batch-based methods. Section 4 presents the second group of our taxonomy, online-based methods. Section 5 aims to compare significant aspects of methods described throughout this work. Finally, Section 6 presents conclusion and future work.

2 | TAXONOMY

We previously mentioned that some authors have contributed to the scientific literature with surveys on the subject of drift detection, such as Hu et al. (2019), Khamassi, Sayed-Mouchaweh, Hammami, and Ghédira (2018), Lu et al. (2018), and Wares, Isaacs, and Elyan (2019). We note, however, that none of those works is specific to unsupervised drift detection, though Khamassi et al. (2018) take into account a category that includes unsupervised methods. This article is exclusively focused on unsupervised detection methods. Therefore, the taxonomy we are proposing is novel and differs from the previous ones in that it deals specifically with the properties of detectors designed to work in unsupervised environments.

Lu et al. (2018) proposed a categorization of automatic drift detection methods into three large classes. The first class includes methods which monitor error rates, The Drift Detection Method (DDM; Gama et al., 2014), Early Drift Detection Method (EDDM; Baena-García, del Campo Ávila, Bifet, Gavald, & Morales-Bueno, 2006), and others (Gomes et al., 2019; Xu & Wang, 2017). The second class is composed of methods that use distance measures to estimate the similarity between distributions of previous and current data. The last class includes methods that make use of multiple hypothesis tests to detect concept change. None of these classes is specific to unsupervised methods, however, the works of Haque et al. (2016), Chandra, Haque, Khan, and Aggarwal (2016), and Hosseini, Gholipour, and Beigy (2015) are presented as unsupervised or semi-supervised methods that belong to the last two classes. Therefore, if we take into account the taxonomy proposed in Lu et al. (2018), unsupervised detectors may fall under the two last categories.

Hu et al. (2019) proposed a categorization of concept drift detection methods into two large groups named performance-based and data distribution-based approaches. Methods based on performance continuously monitor some error-related metric, such as accuracy, F-measure, precision, and recall. In general terms, a drift is signaled when a significant decrease in the metric is observed. Because true labels are required to estimate errors, these methods are not applicable to unsupervised tasks. On the other hand, distribution-based approaches rely on distribution-monitoring measures, such as location, density, and range. Methods that belong to this category may be supervised or unsupervised. We note that this group includes methods from the two last classes from the categorization proposed in Lu et al. (2018).

Wares et al. (2019) present a categorization for supervised methods only. The methods are classified into four groups: Statistical methods, window-based methods, block-based ensembles, and incremental-based ensembles. The first class comprises detectors that use statistical tests, such as the Cumulative Sum and the Page–Hinckley Test (Page, 1954). This group includes DDM (Gama et al., 2014), EDDM (Baena-García et al., 2006), and the McDiarmid drift detection method (Pesaranghader, Viktor, & Paquet, 2018). The second class includes window-based methods, which, in general, monitor the accuracy of the classifier for instances in a window. The last two classes include methods that rely on monitoring the accuracy of ensembles of classifiers, and they differ on how data are processed during reaction to drift: Methods in the third class retrain classifiers on chunks or blocks of instances, while methods in the last class retrain incrementally at each arriving instance.

Khamassi et al. (2018) introduce a taxonomy guided by the questions: “How are data processed?”, “How is learning processed?”, “How is concept drift monitored?”, “How is concept drift handled?”, and “What are the performance criteria?”. With respect to the third question, methods are grouped according to the type of detection, specifically whether it is supervised, unsupervised, or semi-supervised. The authors further refine the classification of unsupervised methods into similarity in time, similarity in space, and model complexity measures. The first is related to how distribution evolves between two timestamps, and usually, differences are detected by means of hypothesis tests (Alippi & Roveri, 2008; Kuncheva & Žliobaitundefined, 2009; Shaker & Lughofer, 2014). The second group includes methods which monitor the evolution of data distribution in space by means of distance functions, such as Euclidean,

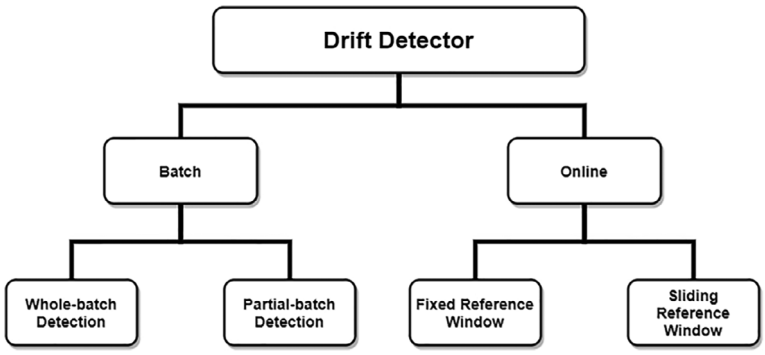

FIGURE 2 Proposed taxonomy of unsupervised concept drift detection methods

Heterogeneous Euclidean-overlap, and Mahalanobis distances—as observed in Tran (2019), Tsymbal and Puuronen (2000), and Gonçalves Jr and Barros (2013), respectively. The last group focuses on changes in the model structure and/or parameters.

In the taxonomy proposed by Khamassi et al. (2018), the similarity-in-time group is composed of papers published before our time limit (2015). In fact, as mentioned in Wares et al. (2019), there have been only a few works that take into account temporal dependence in concept drift detection. Since we focus on the analysis of more recent characteristics of unsupervised drift detection methods, this criterion is not considered in our taxonomy. When observing the similarity-in-space category, it is possible to note that all methods discussed in this article may be included in this group. Therefore, our taxonomy expands this category of methods. Finally, the last group is algorithm-dependent, while works described in this article may be generated using any machine learning algorithm.

Therefore, the main differences between the taxonomy we are proposing to the currently existing taxonomies are (a) we propose a taxonomy specific to unsupervised drift detection methods and (b) we categorize methods according to how the drift detection is performed, unlike other taxonomies that focus on how classifiers are trained or how they react to drift.

In our systematic literature review, we identified common attributes of unsupervised and semi-supervised drift detection methods, with the intent of providing a taxonomy. We briefly introduce the categories of the taxonomy in Figure 2. These categories will be explained in-depth in the remainder of this section.

While concept drift is not exclusive to data streams, all of the methods found in our survey handle instances by using some form of window on an instance stream. In general, at least one window contains the instances considered to belong to the most recent known concept, which were used to train or update the most recent predictor. And at least one window contains the instances which may have suffered a concept drift. Although each author may use their own terminology, we refer to these windows as the *reference window* and the *detection window*. All categories in our taxonomy can be explained by how these windows and the instances in these windows are handled by the algorithms.

At the first level of the category, we distinguish methods according to how the detection window is constructed. One may easily observe that unsupervised drift detectors rely on either accumulating a batch of instances or testing for drift at every instance, which reinforces the findings presented in (Khamassi et al., 2018). Therefore, methods in the first group were described as *batch drift detection methods*, whereas those in the second were categorized as *online drift detection methods*. It is important to remark that this classification takes into consideration only how the methods differ when *detecting* drift, not how they train, update, or retrain their models, or perform any other specific task once a drift has been detected.

At the second level of the taxonomy, we categorize batch-based drift detectors according to how they manipulate the data in the detection window. While the methods differ in many aspects, including whether the size of the batch is fixed or dynamic, the most remarkable difference among them is whether they use every instance in a batch or sample the batch to detect drifts. Therefore, the second-level division has the categories of *whole-batch detection* and *partial-batch detection*.

In its turn, online detectors were categorized according to how they manipulate the reference window. In these methods, the detection window is always a sliding window that advances as new instances arrive, and refers to the current concept in the stream. On the other hand, the reference window may be fixed or sliding. Therefore these methods are categorized into *fixed reference window* and *sliding reference window* detectors.

We do not differentiate the methods with regard to how they behave once a drift has been detected. In general, all of them will train a new model/ensemble of models or update a model/ensemble of models with instances from the detection window when a drift is detected. Then, the detection window becomes the next reference window. Moreover,

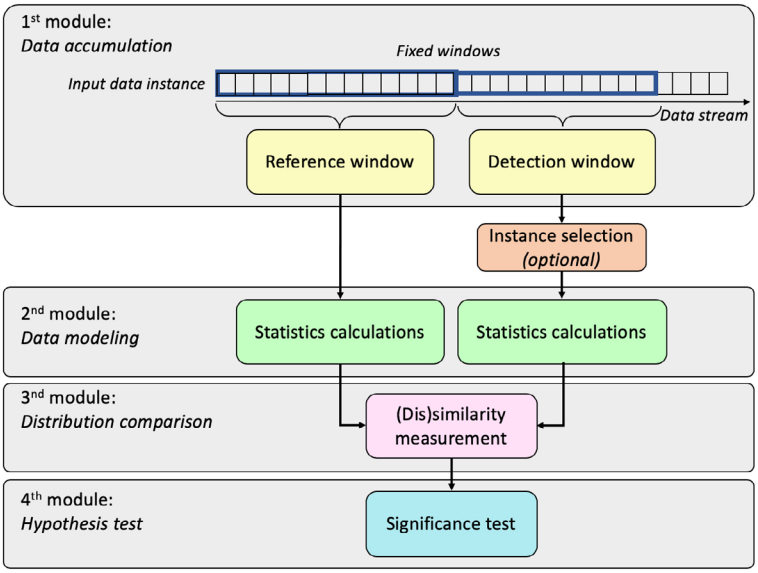

FIGURE 3 A general framework of unsupervised batch-based drift detection methods

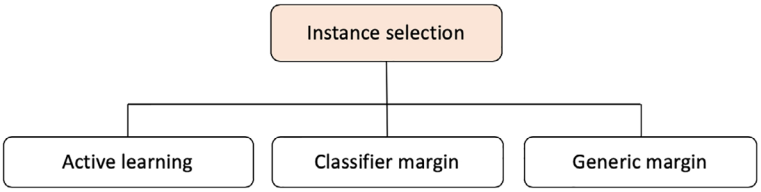

FIGURE 4 Instance selection strategies used in the partial-batch detection methods surveyed

these windows may or may not overlap, and in the case of online detectors, there may be a period of inactivity until a new detection window reaches the minimum number of samples required for drift detection.

3 | BATCH-BASED METHODS

In our proposed taxonomy, batch-based detectors differ on how the instances in the batch are used. We use the terms "batch" and "window" interchangeably. In the first group, *whole-batch detection*, the entire batch is used in the detection. In the second, *partial-batch detection*, only a subset of the batch is used.

A general framework for batch-based detection methods, adapted from Lu et al. (2018), is shown in Figure 3. It combines four modules: Data accumulation, data modeling, distribution comparison, and hypothesis test. The first module aims to accumulate the initial instances into a fixed-size reference window, and instances that arrive in the sequence into a fixed-size detection window, since instances arrive continuously and in a sequential manner. As an optional step, a subset of instances from the detection window can be selected by applying an instance selection approach, as summarized in Figure 4, to generate a smaller detection window for the partial-batch detection methods. Then, the second module aims to represent data from both windows by means of statistics and other information, as batch summary or quantification, to express data distribution. Next, these values are the input for the third module, which is designed to use similarity/dissimilarity measures to compare both windows. Finally, the last module uses hypothesis test to evaluate thresholds to decide when a drift must be signaled.

A list of the works that were taken into consideration for the definition of the taxonomy of batch-based concept drift methods is presented in Table 1.

3.1 | Whole-batch detection

The first method we describe in this subcategory is one of the most prominent batch-based detectors. The Nearest Neighbor-based Density Variation Identification (NN-DVI) algorithm Liu et al. (2018) was proposed to deal with

regional drifts. According to the authors, distribution-based drift detectors are global drift oriented and frequently miss drifts caused by regional density changes. NN-DVI is a distribution-based drift detection method based on regional-density estimation composed of three modules: (a) kNN-based space partitioning for data modeling; (b) distance function to accumulate density discrepancies; and (c) statistical significance test.

This method works by comparing the reference window to the detection window as follows. In its first module, NN-DVI focuses on creating data partitions according to the neighborhood of each example. This strategy results in each instance being grouped with its instance particle to form a partition. The particle of an instance x_i is composed by x_i and its neighbors. Then, instance particle groups are created by taking into account shared neighborhood, that is, each data instance is represented by a set of shared instance particles, and a multiplicity function is used to provide a multiset of instance particles for each instance so as to represent uniform weighted instances. The objective in this first module is to allow NN-DVI to be more sensitive to small discrepancies in data distribution. The density differences among partitions calculated considering instance particles are used to similarity measurement in the second module of NN-DVI. A set-related dissimilarity measure, also called distance measure, is applied to quantify the differences in the number of instance particles among pairs of multisets of instances. When comparing sample sets, the distance between two sets is calculated in terms of their accumulated differences in the number of instance particles. Finally, the last module involves using a tailored statistical significance test to indicate drift occurrence. The authors assume that instances in time windows are independent and show that the distance values of sample sets fit a normal distribution. As a result, they use max likelihood to estimate the mean and variance of sample sets distance distributions. Then, the cumulative distribution function is compared to a user-defined drift threshold to decide whether or not a drift has been observed. A Naive Bayes classifier is used as base learner. When a drift is detected, the current detection window is used to represent the new concept.

The second method is the unsupervised Fast and Accurate Anomaly Detection (FAAD) framework (Li et al., 2019), which was proposed to tackle unsupervised sequence-anomaly detection in the context of problems involving multi-dimensional sequences in data streams prone to concept drift. This kind of application refers to analyzing the ordering relationship in sequences so as to detect anomalies. To achieve such a challenging goal, FAAD is composed of three different algorithms. The first is a feature selection method based on mutual information and symmetric uncertainty information to reduce feature redundancy and better cope with multi-dimensional sequences. Then, the second algorithm is designed to perform anomaly detection by constructing models using random feature sampling and by calculating anomaly scores with these models. The anomaly scores are compared to a user-defined threshold to determine when a sequence is an anomaly. Finally, the anomaly buffer based on model dynamic adjustment (ABMDA) algorithm is proposed to detect drifts. The third component is detailed as follows.

The key issue in FAAD is distinguishing between true anomalies and normal sequences incorrectly detected as anomalies due to concept drift. Consequently, the ABMDA algorithm is divided into two anomaly detection steps. Only anomalies assigned in both steps are assumed to be true anomalies. The second step is performed using the second component (algorithm) of FAAD, while the drift detector is used in the first step. This detector works based on a score obtained considering (a) the distribution of anomalies and (b) statistics used to express data distribution. In both data modeling procedures, ABMDA compares the reference window to the detection one. The first procedure is designed to calculate the proportion of anomalies in the detection window, while this proportion in the reference window is assumed to obey the normal distribution—the average and variance of reference anomalies are previously calculated. Finally, the difference between both proportions is obtained. The standpoint is that the proportion of anomalies is low when the data stream is stable. As a result, the larger the proportion of anomalies in each time period deviates from that from reference anomalies, the higher the probability that false anomalies are detected due to concept drift.

In the second procedure, the algorithm compares two word frequency matrices M: One from reference data and the other from detection data. Each matrix M is calculated as follows: Given an attribute x_i, the frequency of the occurrence of the value v_j to x_i throughout its respective batch is computed and stored in $M_{i,j}$. The difference between the two matrices correlates with a higher chance of a drift having occurred. Next, a weighed sum is calculated between the aforementioned difference of proportion of anomalies and the 1-norm of matrix difference of frequency. If this sum is higher than a user-defined threshold, then current anomalies are stored in an anomaly buffer. When this buffer attains its maximum size, a new model is generated and added to the set of initial models—which are subject to weight updates and model removal to keep the number of models in check. Subsequently, the second algorithm of FAAD is used to detect anomalous sequences in the buffer. When sequences are assigned as anomalies in this second detection process, they are classified as true anomalies. Finally, the detection anomalies distribution and frequency matrix are used to

update the reference ones. It is important to mention that FAAD is an application-dependent approach whose focus is toward detecting data anomalies instead of detecting drift.

In the third method, Bashir et al. (2017) proposed a two-level distribution-based framework called Unsupervised Change Detection for Activity Recognition (UDetect). This framework is based on the premise that a drift is indicated by high variance from the instances classified to a certain label with respect to the training instances from that label. In the first level, UDetect trains a classifier and collects data to characterize each class. The second level is unsupervised and collects batches of data for each class, verifying whether the instances of those classes differ significantly from the data previously collected.

More specifically, during training, the data are divided into chunks of fixed size such that no chunk contains instances that belong to more than one class. Each chunk is summarized into a single parameter, which is the average distance from all its instances to its centroid. Then, the parameters for each class are calculated into thresholds using three different heuristics. The framework does not impose any restriction on the classifier, and the entirety of the data may be used to train a model. The result of this phase is a classifier, and a set of thresholds for each class.

The second phase is involved with the employment of the model with the arriving batch of data. For each class that was known in the first phase, the framework designates a buffer with capacity identical to the size of the chunks from the first phase. The goal of that buffer is to accumulate a batch of instances that are assigned to a specific label. When a sufficient amount of data is collected for that class, the chunk is summarized with the same heuristics used in the first phase. The resulting parameter is compared to the thresholds using Shewart charts (Tobias, 2001) and peaks are identified to signal that a drift has occurred. The proposed framework was evaluated with two activity recognition data sets, where subjects are performing activities such as walking, sitting, standing, and so on. Despite its name, there is no clear indication from the article that the method is specific for activity recognition.

Finally, the last method (Maletzke, dos Reis, & Batista, 2018) is an improved version of Stream Quantification by Score Inspection (SQSI; Maletzke, dos Reis, & Batista, 2017), methodology for quantification in data streams that uses a distribution-based drift detector. The problem of quantification is closely related to classification in that the quantifier's task is to estimate the number of instances of each class in a sample. In fact, a common approach, known as classify-then-count, is to first classify a set of instances and then count the labels.

SQSI detects drift with a two-step test to verify if the distribution of the classifier errors has changed (Maletzke et al., 2017). As new instances arrive, they are collected into the detection window of user-defined size. When the window is full, a classifier provides scores for each instance in an unsupervised fashion, which may be any statistics that represent the classifier's confidence in that instance, such as the distance from the instance to the hyperplane in Support Vector Machines (SVMs) classifier. A statistical test is used to verify whether the current scores come from the same distribution as the scores estimated from the reference data—initially, the training data. If that null hypothesis is rejected, then a linear transformation is applied to the reference data such that the instances in the window have the same mean and standard deviation as the detection data (the current concept). Next, new scores are calculated and the statistical test is performed again. If the null hypothesis is still rejected, then SQSI signals a drift. Once a drift has been detected, SQSI requests true labels for all instances in the detection window. However, if either test does not reject the null hypothesis, then a new detection window is created with no overlapping instances with the previous one, that is, a new batch.

"SQSI" with Instance Selection (SQSI-IS; Maletzke, dos Reis, & Batista, 2018) improves the initial version of SQSI (Maletzke, dos Reis, & Batista, 2018). Both SQSI and SQSI-IS perform the aforementioned two-step check, and both train a new classifier after a drift has been detected, using the detection window as reference data to detect the next drift. However, while SQSI requests true label for all instances, SQSI-IS uses instance selection and self-learning to request true labels for only a fraction of the instances in the batch.

The test used in both algorithms was the Kolmogorov–Smirnov (KS), which is a nonparametric test to identify whether two samples come from the same distribution (Lovric, 2011). Because KS is very sensitive to changes in the proportion of the classes (Maletzke, dos Reis, Cherman, & Batista, 2018), the authors suggest using a very strict confidence level (at most 0.001), lest a significant amount of false alarms tends to be raised. For problems where the proportion of the classes does not change as significantly, for example, classification, a different test or confidence level might be used.

3.2 | Partial-batch detection

In this subcategory, the drift detectors involve selecting a subset of instances from the detection window to check if a drift has occurred. In general, this module may be undertaken using any instance selection strategy, ranging from SVMs

margin (Sethi & Kantardzic, 2015) to Active Learning (Costa et al., 2018). All methods discussed in this section use one of the instance selection strategies summarized in Figure 4.

In Sethi and Kantardzic (2015), the main objective of the proposed Margin Density Drift Detection (MD3) method is to monitor changes in a classifier's margin—region of the decision space where predictions are highly uncertain. The assumption is that variation's density higher than a user-defined threshold indicates drift. The number of instances that are in the region delimited by the margin is used to calculate the margin density. This first version of MD3 is an unsupervised drift detector that uses SVM to obtain a classifier's margin. An SVM classifier is generated using the initial training dataset. Next, two reference measures are obtained: The maximum and the minimum historical densities. Then, the density is calculated for the arriving batch. When the number of unlabeled samples increases within the margin, this indicates a gradual drift. On the other hand, when the number of unlabeled samples decreases, a sudden drift is signaled. MD3 uses a sliding window to process data stream. After drift detection, this method uses labeled data to replace the SVM classifier and also to update the reference density values. The main problem of this method is to work only with SVM as learning algorithm.

This limitation inspired Sethi and Kantardzic (2017) to propose an improved version of MD3 to avoid the aforementioned drawback of the first version. In this new version, which shall be referred to as MD3 Ensemble Generic Margin (MD3-EGM) in this survey, an ensemble of classifiers is used to produce a generic margin, which is defined as the set of unknown instances with the highest uncertainty. The uncertainty is measured as the disagreement among member classifiers when assigning a label for each instance. The ensemble of classifiers is created using the random subspace approach, that is, it randomly chooses different n subspaces from the original feature space to train n classifiers. The drift detection is based on monitoring the diversity among member classifiers so that a drift is expected to occur whether the ensemble's disagreement increases when assigning classes to the unknown samples contained in the detection window. However, since this new version of MD3 may result in high false-positive rates, a supervised drift confirmation module is added to the method. Therefore, we consider this method as a semi-supervised drift detector.

The tracking of the disagreement is conducted in the following way. Four reference measures are obtained from the reference window using a 10-fold cross-validation training and test procedure: Disagreement and prediction performance, along with their standard deviation. Next, the disagreement between the classifier members is calculated considering the instances of the detection window. Then, considering the instances within the margin, the drift detector is divided into two significant levels: Unsupervised and supervised drift indicators. These two levels use a threshold weighted in terms of the standard deviation of the reference measures. In the first detection level, the current accumulated disagreement is compared to the reference agreement. If the difference between both values is higher than the threshold, the second drift indicator is triggered. The second level works by selecting a group of instances from the current batch to be labeled and to be included in a dataset, which is used to calculate the current prediction performance. This value is compared to the reference prediction performance. It is assumed that a concept drift takes place when the difference between both values is higher than the threshold. The instance selection performed is a procedure where a fixed number of instances that arrive after the first drift indication are labeled. Finally, the ensemble of classifiers is retrained using the dataset created after the second drift indicator. It is worth noting that this method may handle classifiers without explicit notions of margin, however, it needs a supervised drift confirmation to take a final decision.

The Predict–Detect streaming classification framework proposed by Sethi and Kantardzic (2018), which shall be referred to as PDetect in this survey, was designed to cope with a specific application, precisely adversarial attacks. The assumption is that attacks lead to changes in the distribution of the data and to decreasing the prediction capabilities. When this application problem is dealt with in the context of streaming data mining, these changes are called adversarial drifts. According to the authors, proposing an application-dependent method is a key issue in this kind of problem, since the application-independent nature of distribution-based drift detectors may be a drawback when facing adversarial drifts. The problem arises because the detectors may be vulnerable to adversarial manipulation at test time. To overcome this drawback, they propose an adversarial-aware drift detection method developed as an ensemble-based and classifier-type independent framework.

The proposed framework splits the feature space of the training dataset into two disjoint subsets to train two classification models: (a) predict and (b) detect. The division of the feature space is performed by ranking all available features according to their F-value from ANOVA and distributing the features between both classifiers in a round-robin strategy to obtain high predictive classifiers. The prediction classifier is designed to assign label to the unknown instances of the detection window, while the detect classifier is responsible for indicating adversarial activity. The former is prone to get attacked at some point. The latter is expected to be invulnerable to adversarial manipulation at test time, since it is a hidden component in the prediction process. These classifiers form a self-monitoring scheme. When the disagreement between their predictions significantly reduces over time, this may indicate a possible drift.

The drift detector works as in (Sethi & Kantardzic, 2017). Here, however, only two classifiers are taken into account. Consequently, this method uses a semi-supervised drift detector. Once a drift has been detected, the two classifiers are retrained using the dataset created after the first drift indication. In terms of feature partitioning, the feature split may be kept the same or a new one can be generated.

Costa et al. (2018) also proposed a drift detection method based on the hypothesis that the density variation of the most significant instances may indicate drift. In this work, the instance selection procedure is based on Active Learning. The proposed method, named Drift Detection Method Based on Active Learning (DDAL), is divided into two phases. The first phase involves generating a classifier using instances contained in the reference window (first batch of data), while the second is subdivided into three modules: Drift detection, reaction, and classification. In the second phase, DDAL monitors the occurrence of concept drift for each new batch of unlabeled data and depending on the result of drift detection, it also performs the reaction and the classification modules.

The detection module works by calculating the density of the most significant instances chosen from the detection window using the Fixed Uncertainty Active Learning strategy (Žliobaitė, 2014) as follows. Virtual margins are created as the projection of hyperplanes, which are set according to a user-defined uncertainty threshold, that are equidistant to the separating hyperplane. For each instance in the detection window, the classifier maximum posterior probability is calculated and compared to the uncertainty threshold. All instances whose confidence value is lower than the threshold are assumed to be inside the subspace delimited by virtual margins and, consequently, are selected for the density calculation. The density value is obtained computing the number of instances that fall into the subspace delimited by virtual margins divided by the number of instances in the detection window. Then, this density value is compared to the reference maximum and minimum density values. The new values replace the reference ones when it is greater than the reference maximum or lower than the reference minimum. Finally, if the difference between these two values is greater than a user-defined drift threshold, then a drift is signaled and the reaction module is triggered. At the reaction module, a new classifier is generated using the detection window data in replacement to the current classifier. Only at this point, the instance's true labels are required, since a new training set must be provided. On the other hand, if no drift is detected, DDAL reaches its classification module, where samples of the detection window are classified. Then, DDAL returns to the detection module to monitor the next batch of unlabeled instances.

4 | ONLINE-BASED METHODS

As discussed in Section 3, batch-based methods work by accumulating instances into a batch of data, and when the batch is full, one or more algorithms are triggered to detect if a drift has occurred since the last predictor was trained. In contrast, online-based methods check for drift at every arriving instance—possibly after an initialization period to accumulate instances before the first check may be performed. All online-based methods analyzed in this survey use a sliding detection window. However, some of them differ in that the reference window is fixed over the training instances, while others slide the reference window into the unlabeled samples arriving from the drift (possibly overlapping with the detection window). Therefore, our proposed taxonomy refine the online category into methods that use fixed and sliding reference windows. The works analyzed in each category are discussed in the next subsections, and Kim and Park (2017) is discussed in Section 4.3, as the method proposed by the authors has variants for both fixed and sliding reference windows.

A general framework of unsupervised online-based drift detection methods is presented in Figure 5. The framework is divided into three modules: Data retrieval, (dis)similarity measurements, and significance test. The first module is devoted to receive incoming data instances to generate the reference window, which, as mentioned before, can be fixed or sliding. This first module is also responsible for manipulating the detection window, which is always a sliding window. The second and third modules are similar to the third and fourth modules of the framework of batch-based methods, respectively. A list of the works that were taken into consideration for the definition of the taxonomy of online-based concept drift methods is presented in Table 2.

4.1 | Fixed reference window

The Incremental Kolmogorov–Smirnov test (IKS) is an online variant of the KS test proposed in dos Reis et al. (2016). It uses a treap data structure to insert and remove observations into a dynamic set that allows updating the p-value for

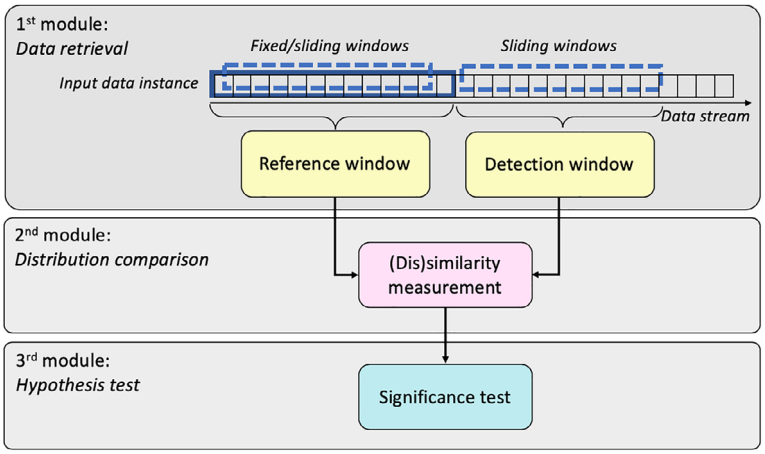

FIGURE 5 A general framework of unsupervised online-based drift detection methods

the KS test without revisiting entire batches of data. The advantage of IKS is that it behaves exactly as the traditional KS test, but with much lower overall time complexity when applied multiple times as a sample is updated with new objects.

Dos Reis et al. (2016) also proposed an online unsupervised drift detection method using the IKS, which shall be referred to as IKS-based Drift Detector (IKS-bdd) in this survey. The principle behind IKS-bdd is to apply the IKS test individually to each attribute. This removes the necessity of using multivariate tests or mapping functions, thus reducing the required amount of data. The detection of a change in a single attribute may be sufficient to trigger the detection of concept drift. The downside of this approach is that some drifts may not be detectable when features are analyzed individually, although a multivariate detector might work for the same data.

For each attribute, IKS-bdd keeps two windows of size W. The reference window is fixed and contains the data that was used to train the most recent model. The detection window is sliding and contains data from the stream. As each instance arrives, it is classified and the IKS test is performed, comparing the distribution of the data in the reference window against the detection window. If the null hypothesis that the distributions are identical is rejected, then a drift is signaled. Otherwise, the window is shifted and a new instance is taken from the stream.

The authors propose three reactions to drift. The first is Model Replacement (MR), which consists of requesting true labels for all instances in the detection window and then training a new classifier. The second approach is AB transformation, and it involves (a) applying a linear transformation to the reference data, such that the attributes involved in the drift will have mean and standard deviation identical to those of the instances in the detection data; (b) performing the KS test again, resorting to MR if the null hypothesis is still rejected. The third approach is a modified decision tree. When a drift is detected, true labels are requested only for instances that reach the leaves with a decision on the attribute that triggered the drift.

Koh (2016) proposed an unsupervised drift detection method specifically for transactional data streams—problems involving representing interactions between entities or items. The method is named Change Detection in Transactional Data Stream for Frequent Pattern Mining (CD-TDS) and can be divided into two parts: (a) Local Drift Detector and (b) Global Drift Detector. When there are changes in some items but no changes in the connections among items are observed, this behavior may indicate a local drift, more precisely a virtual concept drift. In terms of global drift detection, the connections among the items must change to indicate a drift due to new concept generation or old concept disappearing.

The main procedure of the local drift detection involves comparing two windows W_0 and W_1 of a stream S using Hoeffding Bound. The difference in terms of sample means between both windows is compared to a value ϵ defined by the Hoeffding bound. Then, a drift is signaled when this difference is greater than ϵ. It is important to highlight that this local drift monitoring process is similar to that used in *ADaptive sliding WINdow* (ADWIN; Bifet and Gavaldà (2007).

To detect global drift, a tree structure is used to represent the connections between items in S. In this way, the two windows S_0 and S_1 are represented by graphs. To determine whether or not the two windows present high dissimilarity, a pairwise statistical test is used for testing the disagreement between the two tree structures. Several disagreement metrics reported in the literature could be used to measure this diversity, such as edit distance, largest common subtree, and smallest common supertree. In Koh (2016), the authors have worked with the Levenshtein edit distance (Gilleland, 2020).

4.2 | Sliding reference window

Lughofer et al. (2016) proposed two techniques to handle concept drift with few labeled instances and no labeled instances at all: The first technique uses single-pass active learning filters to select the most interesting samples for supervised classification, and the second considers the overlap between two classes in a classifier's output certainty distribution. In both cases, an online modified version of the Page–Hinkley test (PHT; Mouss, Mouss, Mouss, & Sefouhi, 2004) is performed to detect drifts, which shall be referred to as Online Modified Version of the Page–Hinkley Test (OMV-PHT) in this article. The PHT is used to detect abrupt changes in a Gaussian signal x. It uses the difference between the $x(t)$ and the previous average $\bar{x}_{t-1}$.

That signal is the performance indicator of the classifier. In the semi-supervised scenario, active learning is used to select the most significant instances to estimate the classifier error. The unsupervised technique for drift detection is based on the idea proposed by (Donmez, Lebanon, & Balasubramanian, 2010) to estimate classification errors without labels. It assumes that the uncertainty estimates of the two most probable classes correspond to a bimodal distribution, and that this distribution can be assumed to be two distinct Gaussian distributions. The overlap of these distributions can be used as an estimate of the error. Thus, the degree of variation of overlap is used as the signal for the PHT.

Mustafa et al. (2017) proposed a novel class detector, referred in this article as Nonparametric Multidimensional Drift Detection Method (NM-DDM), which is based on denoising autoencoders and an NM-DDM based on log-likelihood random walk. We are interested in the former. That drift detection method relies on calculating the log-likelihood ratio between data from two windows, one prior to the potential drift point, and the second after. The log-likelihood is estimated for each dimension (attribute) and if the highest ratio between the two windows is greater than a threshold, then a drift is signaled. After a drift is signaled, the samples in the last window are used to train a new classifier. The instances with low confidence values are labeled by an external component, while the samples with high confidence values are labeled using the predicted label.

Plover is a proof-of-concept algorithm proposed in de Mello, Vaz, Grossi, and Bifet (2019). The authors performed an analysis of theoretical foundations and learning guarantees in unsupervised concept drift detection. They conclude that, while supervised methods are supported by statistical learning theory, unsupervised approaches depend on internal or external metrics that do not rely on similar guarantees. Internal metrics, such as cluster compactness and centroid distance, are calculated from the structure of the data, while external metrics depend on supervised labels—which cannot be realistically expected to be available in truly unsupervised tasks.

The concept of Algorithm Stability (Bousquet & Elisseeff, 2002) establishes conditions for any arbitrary function to converge with its expected value. Plover is a uniformly-stable concept drift detection algorithm that derives from that concept. Similar to other online methods discussed in this section, it computes the divergence between a sliding detection window and a reference window, and if this difference is greater than a threshold, a drift is signaled. According to the authors, two premises are required to use the proposed theory: (a) the function must be selected independently of the input data and (b) input data must be independent and identically distributed. Such function should be adequate to the problem at hand, for example, mean, variance, spectral power, and so on.

Haque et al. (2016) proposed a framework for detection of drift and novel class on data streams called SAND (Semi-supervised Adaptive Novel Class Detection and Classification over Data Stream). SAND uses an ensemble of k-nearest-neighbor (kNN) classifiers and is divided into two modules: Outlier detection and concept change detection. In addition, it is also focused on performing data self-annotation. As the framework observes more and more outliers, a clustering-based algorithm checks if these outliers are instances of a novel class to further train the ensemble with samples of the new class.

When an unknown example x_i arrives, SAND predicts its label by majority voting and estimates the confidence of each classifier using two heuristics: Association and purity. The association heuristic estimates the confidence of a classifier by measuring the distance between x_i and its closest pseudopoint—which the authors define as a structure that contains the centroid, radius, and the number of data points belonging to each class. The second heuristic is mainly focused on monitoring the most frequent class in the pseudopoint associated to x_i. Subsequently, a variable-size window stores x_i, its class prediction and the confidence scores.

After the instance has been classified, SAND monitors the confidence scores to detect drifts. The drift detection process is performed by assuming that the confidence values follow a *beta* distribution. Values stored in the variable-size window are used to estimate the distribution parameters and the result is compared to a threshold. If a drift is detected, SAND requests only true labels for instances with low confidence scores. These instances are incorporated into the new training dataset, alongside with self-annotated ones, and the ensemble is then updated using the new training dataset.

In line with the idea developed in Haque et al. (2016), Pinagé et al. (2019) proposed an unsupervised drift detection method that also allows self-annotation. This method, called here Dynamic Selection Drift Detector (DSDD), performs dynamic classifier selection from an initial ensemble of classifiers and uses a configurable auxiliary drift detector guided by a pseudo-error rate as metric to detect drifts. It is divided into three modules: (a) ensemble creation; (b) dynamic selection; and (c) detection. The first module works by partitioning an initial batch of labeled instances—composed of the first m incoming instances—into training and validation datasets.

The first dataset is used to generate an ensemble of classifiers using a modified version of the Online Bagging, Minku, White, and Yao (2010) algorithm, that is, each member classifier is trained with n copies of each instance, where n is defined by the Poisson distribution. The authors highlight that low n values define high diversity among ensemble members. They set values high enough to result in significant high diversity so as to generate different member classifiers. Instances from the second dataset are used as reference regions of competence in the dynamic selection module, which assumes that each member classifier is an expert in a region of competence. This second module works as follows: When an unknown instance x_i arrives, a region is defined in terms of its neighborhood in the validation dataset. Then, the competence of each member classifier is calculated. The classifier with the highest competence is chosen to assign a label to x_i. This decision is assumed to be the pseudo true label of x_i.

Finally, the detection module applies a drift detector to each member classifier. Since the idea proposed is not to use the instance's true labels, the key issue in this method is to compare the prediction provided by each member classifier to x_i to its pseudo true label in a pseudo error monitoring process. This pseudo-error can be used as a metric in several supervised drifts detectors available in the literature. The authors have used both the DDM (Gama, Medas, Castillo, & Rodrigues, 2004) and the EDDM (Baena-García et al., 2006). When a fixed number of member classifiers indicate a drift, a drift is signaled and the validation dataset is updated using new labeled samples. Otherwise, the ensemble of classifiers online learning proceeds using the self-annotated labels.

4.3 | Multiple approaches

Kim and Park (2017) proposed a distribution-based drift detection approach, which shall be referred to as Distribution-based Drift Detection Approach (DbDDA) here. DbDDA signals drifts when the classifier uncertainty becomes too high. Initially, a classifier is trained using labeled instances. Then, for each incoming instance x_i, a posterior probability vector $f(x_i)$ is estimated. A second vector $\bar{y}(x_i)$ is also determined. This vector is such that there are only components with value 1 for the class predicted to x_i, while the others are 0. The random variable X is drawn from the distances $X(x_i) = \| f(x_i) - \bar{y}(x_i) \|^2$ and is used to estimate the classifier uncertainty.

The proposed method monitors the differences $X(x_i)$ for the instances in the reference window and in the detection window. Since the classifier is trained with data from the reference window, instances in that window come from the current concept, while the instances in the detection window might have been generated from new concept. Before classification begins, the algorithm finds a confidence interval for X from the data in the reference window. As each new example arrives, the average value of $X(x_i)$ is calculated from the instances in the detection window. If the difference is higher than the upper limit of the confidence interval, then a drift is assumed to have occurred. In this case, true labels are requested for the instances in the detection window and these instances are used to train a new classifier.

The detection window is always a sliding time window. But the authors propose three different strategies for the reference window. The first two are classic fixed time window and sliding window. In both cases, the reference window does not overlap with the detection window. A third approach, which the authors named Ensemble of Reference Windows (ERW), is based on the use of Control Chart of means and standard deviations. Theses charts observe change in distribution using the mean of the windows means and the mean of windows standard deviations. While in the previous methods a single reference window is used, in ERW a new pair of reference and detection window is introduced as each instance arrives, until a drift is detected.

Additionally, Kim and Park (2017) propose an approach for the situation when no labeled data is available. In this case, it is not possible to build an initial model. The authors thus suggest applying k-means to obtain a cluster structure, resulting in a virtual classifier. Then the classification and drift detection procedure is carried out as previously explained.

TABLE 3 Compilation of some of the characteristics of unsupervised drift detectors reported in the literature highlighting the type of drift dealt with; whether there is application dependency; whether self-annotation is conducted; and the dependency on labeled data

Method	Drift type	Application	Self-annotation	Need label
NN-DVI	Local	Independent		
FAAD	Local/global	Sequence anomaly		
UDetect	Global	Independent		
SQSI-IS	Global	Independent	✓	
MD3	Global	Independent		
MD3-EGM	Global	Independent		✓
PDetect	Global	Adversarial attacks		✓
DDAL	Global	Independent		✓
IKS-bdd	Global	Independent		
CD-TDS	Local/global	Transactional data		
OMV-PHT	Global	Independent		
NM-DDM	Global	Independent	✓	
Plover	Global	Independent		
SAND	Global	Independent	✓	✓
DSDD	Global	Independent	✓	✓
DbDDA	Global	Independent		✓

5 | DISCUSSION

Concept drift is a challenging problem for learning in data stream scenarios. One way to handle this issue is by monitoring data streams and updating the current prediction model(s) with new concepts when significant changes occur. On the one hand, it is widely accepted that, due to the massive volume of incoming data from the streams, true labels are usually not immediately available. In addition, depending on the task to be solved, labeling these data might be a costly process that sometimes may also involve manual labor from multiple domain experts. On the other, it is easily observable a predominance of methods that address the concept drift problem by monitoring the evolution of indicators or measures based on the error rate of the current prediction model(s). For instance, DDM (Gama et al., 2004)—indicated by Lu et al. (2018) as one of the most-referenced concept drift detectors—works by counting the number of errors. Another example is EDDM (Baena-García et al., 2006), based on monitoring the distance between two classification errors.

In view of the difficulties in supervised approaches, drift detectors that can operate in unsupervised mode have been proposed. These approaches are expected to significantly lower the burden of label dependency in the same way that they could make better use of the available data, since unlabeled data are more often accessible. Besides, they may help handling virtual concept drift, that is, when the decision boundaries are not affected by incurring changes (Khamassi et al., 2018). According to the survey presented in this article, the unsupervised methods described fall into two basic categories: Batch-based and online-based approaches. However, both groups address the concept drift problem from the same viewpoint: The underlying data distribution must be monitored so as to identify the points at which the data distribution experiences a significant change. Even so, each approach uses strategies to cope with this issue that differ in their details and tend to suffer from different limitations.

The description provided in this article allows us to observe several properties discussed in this section and summarized in Table 3. First, most of unsupervised methods are global-drift oriented. Consequently, a major drawback is the possibility of missing regional drifts, leading to problems as poor or high sensitivity, and high false alarm rate, especially in detection of concepts that change very gradually. Considering the fact that online-based methods monitor each instance at a time upon arrival, these methods are less prone to missing local drifts, even when only focusing on detecting global drifts. The batch-based approaches, however, may be more severely impacted, since they monitor drifts considering the entire sample set, even the partial-batch ones. As a result of the attempts to overcome such a limitation, some strategies have been developed. For instance, NN-DVI (Liu et al., 2018) focuses on defining neighborhoods to cope with regional drifts, whereas FAAD (Li et al., 2019) performs feature subset selection and feature sampling.

From this table, it is also possible to observe that the studied methods are mainly application independent. They are usually tested in synthetic data and few real datasets. When real-world datasets are investigated, these are general, small-scale, and rather toy benchmark datasets compared to the recent large-scale real-world datasets. There are, however, works that use larger-scale and challenging real-world datasets, such as IKS-bdd (dos Reis et al., 2016), and works that investigate recent application problems, such as adversarial attacks, studied in PDetect (Sethi & Kantardzic, 2018). It is worth noting that the lack of benchmark datasets for evaluation is a limitation of the existing literature of concept drift in general (Wares et al., 2019). Nonetheless, a different scenario is observed for the supervised drift detectors, whose evaluation has been conducted using more realistic and recent datasets—for example, demand for taxi networks (Saadallah et al., 2020).

Despite this research being primarily concerned with unsupervised drift detection methods, some techniques reviewed involve labeled instances. In the partial-batch detection group, for instance, a main problem is that these approaches may be critically affected by the instance selection strategy used. Moreover, the reduction in the number of instances used to monitor data distribution may result in high false detection rates. Methods that focus on tackling this issue, such as MD3-EGM (Sethi & Kantardzic, 2017) and PDetect (Sethi & Kantardzic, 2018), include supervised drift confirmation to decrease false detection rates. Nevertheless, supervised confirmation adds dependency on real labels, as opposed to the label independence pursued by the unsupervised detectors. Some online-based methods may also suffer from this limitation, as DSDD (Pinagé et al., 2019), which involves with-holding a subset of labeled instances to be used as a validation set to allow dynamic classifier selection.

One distinct advantage of some unsupervised methods is the ability to find informative unlabeled samples to be self-annotated and added to the training set, which is especially important for detectors used in online learning scenarios. The criterion used for sample selection is basically the same: Class labels are only requested for instances whose confidence values are low, as seen in SAND (Haque et al., 2016), NM-DDM (Mustafa et al., 2017), and DSDD (Pinagé et al., 2019). Finally, there are unsupervised methods that do not aim only establishing a drift detector. In this case, their final objective is a different task, like anomaly detection, as in FAAD (Li et al., 2019), and novel class detection, as in NM-DDM (Mustafa et al., 2017).

There are additional properties that are worth mentioning. One of these is the need for accumulating a batch of instances into a detection window before attempting to detect drift on batch-based approaches. This may be pointed out as a limitation due to the following reasons: (a) drifts can occur at intervals that are not limited by the window size and (b) handling concept drift requires updating models on an ongoing basis. This problem is even more notable in approaches that assume more restrictive assumptions. For instance, in UDetect (Bashir et al., 2017), since a particular window size is fixed for each class, the drift detector needs to accumulate a fixed number of instances for each class before determining if a concept drift has occurred.

This last limitation is avoided in online-based methods, as they monitor each instance at a time upon arrival. However, while it can be argued that batch-based approaches have low memory and run-time overheads, given that they only focus on monitoring the measures extracted from the reference window, not the instances in the window itself, the opposite is observed in the online-based group. This group of approaches, especially the fixed reference window methods, such as the IKS-bdd (dos Reis et al., 2016), usually has to keep the reference window instances in memory to be able to detect the distribution changes caused by the incoming instances. One alternative to face this problem is using data representation, as the tree and the pseudopoint structures used in Koh (2016) and in Haque et al. (2016), respectively.

6 | CONCLUSION

In this work, we proposed a taxonomy for some state-of-the-art unsupervised concept drift approaches. We conducted a systematic literature review (Okoli, 2015) to provide two groups of recent scientific publications on concept drift detectors: (a) Articles surveying and analyzing different concept drift approaches, and (b) Articles proposing unsupervised concept drift methods. The first group allowed us to observe that very few work has been dedicated to summarizing unsupervised concept drift approaches. The second group, in its turn, led us to group unsupervised and semi-supervised drift detection methods into two main categories: Batch-based and online-based methods. Online and batch drift detection methods refer only to the detection component of the strategy. On the one hand, online methods indicate the occurrence of a drift on data stream one instance at a time. On the other hand, batch methods need to process a set of instances at a time to be able to sign a drift.

In this study, we analyzed eight batch-based methods observing that the main difference among these methods is whether or not significant data distribution changes are monitored on the entire arriving batch or only on a selected instance set. We also analyzed eight online-based methods. This group is subcategorized into two types according to the way they compare the two windows involved when detecting drift, called as reference window and detection window. One group differs from the other according to using fixed or sliding reference windows to detect concept drift. Future work may focus on extending this preliminary study to evaluate and compare highlighted methods by carrying out computational experiments as was done in Barros and Santos (2018) for supervised drift detectors.

ACKNOWLEDGMENT

This study was supported by the Foundation for Research Support of the State of Amazonas (FAPEAM; Grant Numbers 009/2012, 009/2017 and 005/2019 PAPAC Project) and Samsung Electronics of Amazonia Ltda, under the terms of Federal Law no 8.387/1991.

CONFLICT OF INTEREST

The authors have declared no conflicts of interest for this article.

AUTHOR CONTRIBUTIONS

Rosana Gemaque: Investigation; methodology; writing-original draft; writing-review and editing. **Albert Costa:** Investigation; methodology; writing-original draft; writing-review and editing. **Rafael Giusti:** Conceptualization; supervision; writing-review and editing. **Eulanda dos Santos:** Conceptualization; supervision; writing-review and editing.

ORCID

Rosana Noronha Gemaque https://orcid.org/0000-0002-3354-4640
Eulanda Miranda dos Santos https://orcid.org/0000-0002-5671-581X

RELATED WIREs ARTICLE

Classification systems in dynamic environments: An overview

REFERENCES

Abdallah, A., Maarof, M., & Zainal, A. (2016). Fraud detection system: A survey. *Journal of Network and Computer Applications*, *68*, 90–113.

Alippi, C., & Roveri, M. (2008). Just-in-time adaptive classifiers–Part i: Detecting nonstationary changes. *IEEE Transactions on Neural Networks*, *19*, 1145–1153.

Baena-García, M., del Campo Ávila, J., Bifet, A., Gavald, R., & Morales-Bueno, R. (2006) *Early drift detection method*. In Proceedings of the 4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams, Berlin, Germany, pp. 77–86.

Barros, R., & Santos, S. (2018). A large-scale comparison of concept drift detectors. *Information Sciences*, *451–452*, 348–370.

Barros, R., & Santos, S. (2019). An overview and comprehensive comparison of ensembles for concept drift. *Information Fusion*, *52*, 213–244.

Bashir, S., Petrovski, A., & Doolan, D. (2017). A framework for unsupervised change detection in activity recognition. *International Journal of Pervasive Computing and Communications*, *13*, 157–175.

Beyene, A., Welemariam, T., Persson, M., & Lavesson, N. (2015). Improved concept drift handling in surgery prediction and other applications. *Knowledge and Information Systems*, *44*, 177–196.

Bifet, A. and Gavaldà, R. (2007) *Learning from time-changing data with adaptive windowing*. In 7th SIAM International Conference on Data Mining (SDM'07). Mineapolis, Minnesota: Society for Industrial and Applied Mathematics.

Bobowska, B., Chorás, M., & Woźniak, M. (2018). Advanced analysis of data streams for critical infrastructures protection and cybersecurity. *Journal of Universal Computer Science*, *24*, 622–633.

Bousquet, O., & Elisseeff, A. (2002). Stability and generalization. *Journal of Machine Learning Research*, *2*, 499–526.

Chandra, S., Haque, A., Khan, L. and Aggarwal, C. (2016) *An adaptive framework for multistream classification*. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, CIKM'16. pp. 1181–1190.

Chiang, H.-S., & Wu, Z.-W. (2018). Online incremental learning for sleep quality assessment using associative petri net. *Applied Soft Computing Journal*, *68*, 774–783.

Costa, A. F. J., Albuquerque, R. A. S. and dos Santos, E. M. (2018) *A drift detection method based on active learning*. In 2018 International Joint Conference on Neural Networks, IJCNN 2018, Rio de Janeiro, Brazil, 1–8.

de Mello, R., Vaz, Y., Grossi, C., & Bifet, A. (2019). On learning guarantees to unsupervised concept drift detection on data streams. *Expert Systems with Applications*, *117*, 90–102.

Donmez, P., Lebanon, G., & Balasubramanian, K. (2010). Unsupervised supervised learning i: Estimating classification and regression errors without labels. *Journal of Machine Learning Research*, *11*, 1323–1351.

dos Reis, D. M., Flach, P., Matwin, S. and Batista, G. (2016) *Fast unsupervised online drift detection using incremental Kolmogorov-Smirnov test*. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'16. pp. 1545–1554.

Gama, J., Medas, P., Castillo, G. and Rodrigues, P. (2004) *Learning with drift detection*. In Advances in Artificial Intelligence, SBIA 2004. pp. 286–295.

Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, *46* (4), 1–37.

Gilleland, M. (2020) Levenshtein distance, in three flavors. Retrieved from http://www.merriampark.com/ld.htm.

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfahringer, B., ... Abdessalem, T. (2019). Correction to: Adaptive random forests for evolving data stream classification. *Machine Learning*, *108*, 1877–1878.

Gonçalves, P. M., Jr., & Barros, R. S. M. D. (2013). Rcd: A recurring concept drift framework. *Pattern Recognition Letters*, *34*, 1018–1025.

Guzella, T., & Caminhas, W. (2009). A review of machine learning approaches to spam filtering. *Expert Systems with Applications*, *36*, 10206–10222.

Haque, A., Khan, L. and Baron, M. (2016) *Sand: Semi-supervised adaptive novel class detection and classification over data stream*. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI'16. pp. 1652–1658.

Hosseini, M. J., Gholipour, A., & Beigy, H. (2015). An ensemble of cluster-based classifiers for semi-supervised classification of non-stationary data streams. *Knowledge and Information Systems*, *46*, 567–597.

Hu, H., Kantardzic, M., & Sethi, T. (2019). No free lunch theorem for concept drift detection in streaming data classification: A review. *WIREs Data Mining and Knowledge Discovery*, *10*, 1–25.

Iwashita, A., & Papa, J. (2019). An overview on concept drift learning. *IEEE Access*, *7*, 1532–1547.

Kadlec, P., Grbić, R., & Gabrys, B. (2011). Review of adaptation mechanisms for data-driven soft sensors. *Computers and Chemical Engineering*, *35*, 1–24.

Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., & Ghédira, K. (2018). Discussion and review on evolving data streams and concept drift adapting. *Evolving Systems*, *9*, 1–23.

Kim, Y., & Park, C. H. (2017). An efficient concept drift detection method for streaming data under limited labeling. *IEICE Transactions on Information and Systems*, *E100D*, 2537–2546.

Koh, Y. S. (2016) *Cd-tds: Change detection in transactional data streams for frequent pattern mining*. In 2016 International Joint Conference on Neural Networks (IJCNN). pp. 1554–1561.

Krawczyk, B., Minku, L., Gama, J., Stefanowski, J., & Woźniak, M. (2017). Ensemble learning for data stream analysis: A survey. *Information Fusion*, *37*, 132–156.

Kuncheva, L. I., & Žliobaitundefined, I. (2009). On the window size for classification in changing environments. *Intelligent Data Analysis*, *13*, 861–872.

Li, B., Wang, Y.-J., Yang, D.-S., Li, Y.-M., & Ma, X.-K. (2019). Faad: An unsupervised fast and accurate anomaly detection method for a multi-dimensional sequence over data stream. *Frontiers of Information Technology and Electronic Engineering*, *20*, 388–404.

Liu, A., Lu, J., Liu, F., & Zhang, G. (2018). Accumulating regional density dissimilarity for concept drift detection in data streams. *Pattern Recognition*, *76*, 256–272.

Lovric, M. (2011). *International Encyclopedia of statistical science* (pp. 718–720). Berlin: Springer.

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2018). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*, 2346–2363.

Lughofer, E., Weigl, E., Heidl, W., Eitzinger, C., & Radauer, T. (2016). Recognizing input space and target concept drifts in data streams with scarcely labeled and unlabelled instances. *Information Sciences*, *355–356*, 127–151.

Maletzke, A., dos Reis, D., & Batista, G. (2018). Combining instance selection and self-training to improve data stream quantification. *Journal of the Brazilian Computer Society*, *24*, 12.

Maletzke, A. G., dos Reis, D. M., & Batista, G. E. (2017) *Quantification in data streams: Initial results*. In 2017 Brazilian Conference on Intelligent Systems (BRACIS). pp. 43–48.

Maletzke, A. G., dos Reis, D. M., Cherman, E. A., & Batista, G. E. (2018) *On the need of class ratio insensitive drift tests for data streams*. In Second International Workshop on Learning with Imbalanced Domains: Theory and Applications, vol. 94 of Proceedings of Machine Learning Research, Dublin, Ireland. pp. 110–124.

Minku, L. L., White, A. P., & Yao, X. (2010). The impact of diversity on online ensemble learning in the presence of concept drift. *IEEE Transactions on Knowledge and Data Engineering*, *22*, 730–742.

Mouss, H., Mouss, D., Mouss, N. and Sefouhi, L. (2004) Test of page-hinckley, an approach for fault detection in an agro-alimentary production system. In 2004 5th Asian Control Conference (IEEE Cat. No.04EX904), Melbourne, Victoria, Australia. vol. *2*, 815–818

Mustafa, A. M., Ayoade, G., Al-Naami, K., Khan, L., Hamlen, K. W., Thuraisingham, B. M. and Araujo, F. (2017) *Unsupervised deep embedding for novel class detection over data stream*. In 2017 IEEE International Conference on Big Data, BigData 2017. pp. 1830–1839.

Okoli, C. (2015). A guide to conducting a standalone systematic literature review. *Communications of the Association for Information Systems*, *37*, 879–910.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, *41*, 100–115.

Pesaranghader, A., Viktor, H. and Paquet, E. (2018) *Mcdiarmid drift detection methods for evolving data streams*. In Proceedings of the International Joint Conference on Neural Networks (IJCNN). pp. 1–9.

Pinagé, F., dos Santos, E. M., & Gama, J. (2019). A drift detection method based on dynamic classifier selection. *Data Mining and Knowledge Discovery*, *34*, 50–74.

Saadallah, A., Moreira-Matias, L., Sousa, R., Khiari, J., Jenelius, E., & Gama, J. (2020). Bright–drift-aware demand predictions for taxi networks. *IEEE Transactions on Knowledge and Data Engineering*, *32*, 234–245.

Sethi, T., & Kantardzic, M. (2015). Don't pay for validation: Detecting drifts from unlabeled data using margin density. *Procedia Computer Science*, *53*, 103–112.

Sethi, T., & Kantardzic, M. (2017). On the reliable detection of concept drift from streaming unlabeled data. *Expert Systems with Applications*, *82*, 77–99.

Sethi, T., & Kantardzic, M. (2018). Handling adversarial concept drift in streaming data. *Expert Systems with Applications*, *97*, 18–40.

Shaker, A., & Lughofer, E. (2014). Self-adaptive and local strategies for a smooth treatment of drifts in data streams. *Evolving Systems*, *5*, 239–257.

Tobias, P. (2001). Statistical process control and failure mode analysis. In K. J. Buschow, R. W. Cahn, M. C. Flemings, B. Ilschner, E. J. Kramer, S. Mahajan, & P. Veyssière (Eds.), *Encyclopedia of materials: Science and technology* (pp. 8816–8824). Oxford: Elsevier.

Tran, D. (2019) *Automated change detection and reactive clustering in multivariate streaming data*. In 2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF). pp. 1–6.

Tsymbal, A., & Puuronen, S. (2000). Bagging and boosting with dynamic integration of classifiers. In D. A. Zighed, J. Komorowski, & J. Żytkow (Eds.), *Principles of data mining and knowledge discovery. PKDD 2000. Lecture Notes in Computer Science* (Vol. 1910, pp. 116–125). Berlin, Heidelberg: Springer.

Wares, S., Isaacs, J., & Elyan, E. (2019). Data stream mining: Methods and challenges for handling concept drift. *SN Applied Sciences*, *1*, 1412.

Xu, S., & Wang, J. (2017). Dynamic extreme learning machine for data stream classification. *Neurocomputing*, *238*, 433–449.

Žliobaitė, I. (2014). Controlled permutations for testing adaptive learning models. *Knowledge and Information Systems*, *39*, 565–578.

How to cite this article: Gemaque RN, Costa AFJ, Giusti R, dos Santos EM. An overview of unsupervised drift detection methods. *WIREs Data Mining Knowl Discov*. 2020;e1381. https://doi.org/10.1002/widm.1381