Received November 18, 2018, accepted November 25, 2018, date of publication December 10, 2018, date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2886026

An Overview on Concept Drift Learning

ADRIANA SAYURI IWASHITA[1] AND JOÃO PAULO PAPA[2], (Senior Member, IEEE)
[1]Department of Computing, Federal University of São Carlos, São Carlos 13565-905, Brazil
[2]Department of Computing, São Paulo State University, Bauru 17033-360, Brazil

Corresponding author: João Paulo Papa (joao.papa@unesp.br)

This work was supported in part by FAPESP Grant 2013/07375-0, Grant 2014/16250-9, Grant 2016/19403-6, and Grant 2014/12236-1, in part by CNPq Grant 307066/2017-7, and in part by the Coordenaçño de Aperfeiçooamento de Pessoal de Nível Superior - Brasil (CAPES)–Finance Code 001.

ABSTRACT Concept drift techniques aim at learning patterns from data streams that may change over time. Although such behavior is not usually expected in controlled environments, real-world scenarios can face changes in the data, such as new classes, clusters, and features. Traditional classifiers can be easily fooled in such situations, resulting in poor performances. Common concept drift domains include recommendation systems, energy consumption, artificial intelligence systems with dynamic environment interaction, and biomedical signal analysis (e.g., neurogenerative diseases). In this paper, we surveyed several works that deal with concept drift, as well as we presented a comprehensive study of public synthetic and real datasets that can be used to cope with such a problem. In addition, we considered a review of different types of drifts and approaches to handling such changes in the data. We considered different learners employed in classification tasks and the use of drift detection mechanisms, among other characteristics.

INDEX TERMS Concept drift, machine learning, pattern recognition.

I. INTRODUCTION

Traditional learning algorithms generally consider being in a static environment. However, data in the real-world environment may have a dynamic behavior, and the concept can change [1]. The term *Concept Drift* address the problem of learning in dynamic environments [2]. Therefore, concept drift occurs when the training and testing data diverge. In many areas, the dilemma of concept drift can be found including monitoring, planning, personal assistance, applications to handle diverse environments [3], among others. Additionally, many other environments may contain hidden concept drift as well [4]–[6].

The literature proposes several methods for dealing with the concept drift. Widmer and Kubat [7] presented the FLORA (FLOating Rough Approximation) framework: a family of methods that maintains a set of descriptors, employing dynamic size window to select the descriptors that correctly assimilate the current concept. Klinkenberg and Joachims [8] employed Support Vector Machines (SVM) to deal with concept drift, preserving training instances located within a window upon data and discarding irrelevant instances that minimize the generalization error. The authors employed the following data management approaches: "no memory," "full memory," "window of fixed size," and "window of adaptive size."

Stanley [9] proposed the Concept Drift Committee, a committee of hypotheses whose weighted vote make decisions on the current classification. A new member replaces an older member of the committee if the voting record of the older member falls below a threshold. Later on, Kolter and Maloof [10] proposed an ensemble algorithm to detect concept drift. Weighted experts are added and removed dynamically as changes in performance are noticed. With dynamic weighted majority vote for classification phase, the ensemble learners are weighted according to their performance, and the method deletes or creates new learners based on the global ensemble performance. Finally, Farid *et al.* [11] proposed an ensemble algorithm based on decision trees able to detect new classes.

Since concept drift is an important area that has gained the attention of the last years, this work presents a compilation of several works to foster the research in the area of knowledge. The main contribution of this survey is to study different techniques to detect and deal with concept drift, as well as to study public synthetic and real datasets in this area. The remainder of this paper is organized as follows. Section II presents the main theoretical background regarding concept drift, types of algorithms to deal with (Subsection II-A), and techniques in the literature that handles the concept drift issue (Section III). Section IV presents a summary of the

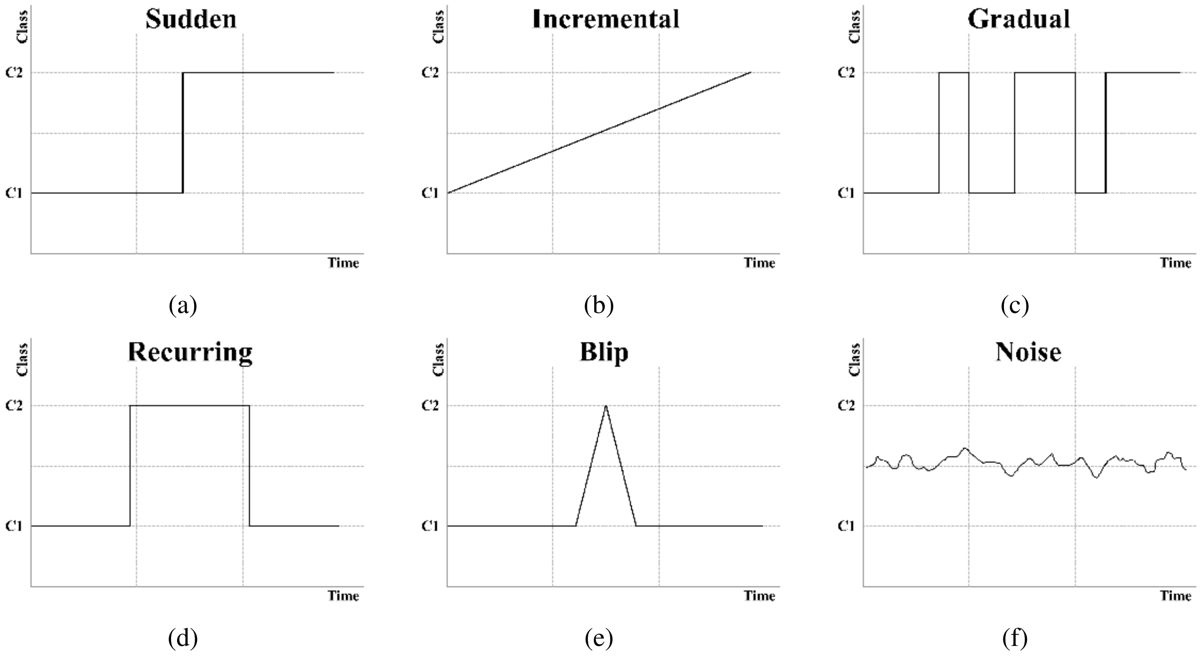

FIGURE 1. Types of concept drift, extracted from [12]: (a) sudden drift, (b) incremental drift, (c) gradual drift, (d) recurring drift, (e) outlier, and (f) noise.

articles studied, and Section V states conclusions and future directions of the area.

II. CONCEPT DRIFT

Concept drift happens when the target concept changes in a non-stationary environment. Let $C1$ and $C2$ be two target concepts, and $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ an ordered instance sequence. Instances prior to i_d have a stable concept $C1$ and does not change. After Δx instances, the concept stabilizes once more, but in another target concept $C2$. The concept among instances i_{d+1} and $i_{d+\Delta x}$ is *drifting* between $C1$ and $C2$ [9]. According to Δx length, the drift can be called gradual or abrupt. In gradual drift, the two concepts slowly swap; whereas in abrupt drift it sudden occurs.

Wadewale and Desai [12] classify the variations of the target concept drift in sudden, incremental, gradual, recurring, blip and noise drifts. Fig. 1 contains this different types of drifts. Fig. 1a shows a sudden drift that the data changes instantly and without alternation. In incremental and gradual drifts the changes occur slowly. Incremental drift (Fig. 1b) happens when the data values gradually change over time, whereas gradual drift (Fig. 1c) also includes changing in class distribution. Recurring drifts (Fig. 1d) occurs when instances of a concept temporary disappear and return after a while. Fig. 1e shows a rare event which in a static distribution can be considered as an outlier, and Fig. 1f shows random changes in instances (noise) that have to be filtered out [12].

Some authors [3] define the drifts in two types: real concept drift and virtual drift. Considering concept drift as changes in data distribution, the real concept drift occurs when the conditional distribution changes on the output whereas the input distribution remains unchanged. Virtual drift has different interpretations in the literature, as the changes in the distribution of incoming data, among others [3].

A. TYPES OF ALGORITHMS

The most usual ways to handle concept drift are the following three [11]: (a) instance selection or window-based approach, (b) weight-based approach, and (c) ensemble of classifiers.

Instance selection or window-based approaches select an appropriate set of previous data to train a classifier [13]. It selects instances inside of a fixed or dynamic sliding window [11]. These approaches assume older examples are incompatible with new data classification, so it has to forget old instances which are considered useless to handle concept drift [13]. Therefore, the training employs the last batch of information with the last training instances. The window of fixed size approach is the simplest rule and the window size is usually determined by the user. By having information on the time-scale of change, a window of fixed size approach is a good choice. Yet, the user is often caught in a trade-off: choosing a small window size (reflects the current distribution with fast adaptivity) or a large window size (having more instances in periods of stability with no concept drift may increase accuracy and have better generalization) [14]. The adaptive window approach adjusts the window size to the length of the drift [8]. It commonly maintains the examples up until the concept drift: in this case, it is not necessary to choose a priori and unknown parameter [14]. Other approaches maintain some of the examples in memory but select a training set of it for classification [13].

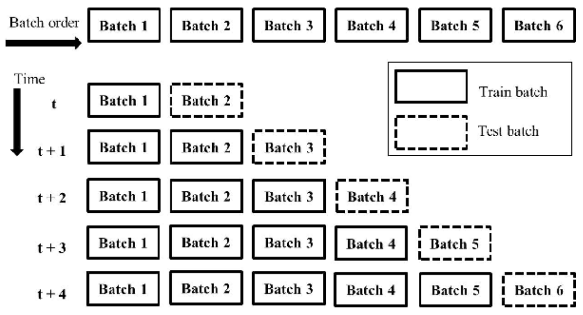

FIGURE 2. ***Full-memory*** **approach.**

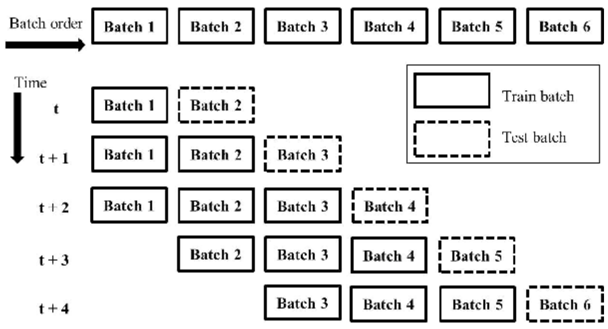

FIGURE 4. ***Window*****-of-size-3 approach.**

A weight-based approach weights instances and deletes the outdated training ones based on their weights [11]. As time passes, it considers old information increasingly irrelevant. Despite considering all instances of learning, new instances have more relevance. Within this framework, must be chosen an updateable classifier capable of weighted learning [13].

The ensemble of classifier combines several outputs from different learners to define a final classification [11]. With a dynamic set of classifiers, performance (or another metric) is observed. If performance decreases, new classifiers replace the aged and poor performing classifiers on the ensemble. Outputs from learners are combined to classify instances on classification phase, commonly with a weighted-vote mechanism [13]. Benefits of weighted ensembles have been studied empirically and theoretically [13].

Data streams can be divided into batches in window-based approaches; thus, it continuously receives batches of data. Some examples are described below [8]:

- Full-memory: the learner employs all instances seen so far (batches), i.e., it can not "forget" old data (Fig. 2).
- No-memory: the learner employs a single batch on training, i.e., the most recent of the stream (Fig. 3).

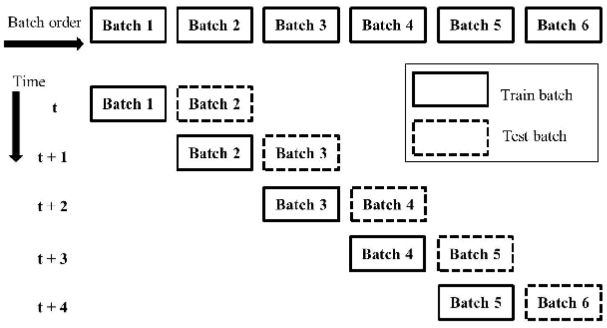

FIGURE 3. ***No-memory*** **approach.**

- Window of fixed size n: the learner employs n batches on training, i.e., a sliding window of size 3 is used with the most recent instances (Fig. 4).

Ditzler and Polikar [15] characterized various forms of concept drift handling algorithms:

- Online or Batch approaches: determined by the amount of instances considered at training phase;
- Single classifier or Ensemble-based approaches: determined by the number of learners employed to decide a classification;
- Incremental or Non-incremental approaches: determined by the reusing a data or not; and
- Active or Passive approaches: determined by the use of a drift detection mechanism or not.

After receiving an instance, the online approach updates the classifier; whereas the batch approach waits to receive plenty of instances to start learning [16]. Single classifiers use one learner to decide on classification phase; whereas ensemble learning combines the results of a set of concept learners with a single or weighted vote, or selecting the most significant result [16]. Incremental learning behaves like online learning with the model update as instances arrive; whereas non-incremental reuses data on learning phase. Active drift detection observes the stream to search for changes and determine whether and when a drift occurs: after a drift, it warns the learner to take the correct action. Passive drift detection considers drift may occur constantly or occasionally, therefore continually updates the learner as data arrive [15].

III. LITERATURE REVIEW

In 1986, Schlimmer and Granger [17] proposed the first technique to handle concept drift, a supervised learning method named STAGGER [10], which trains Boolean characterization via formula based on Bayesian statistics. It keeps a set of concept descriptions (produced by using feature construction and features from the method itself), and select those with better relevance to the present data [16]. It starts with its own features and trains new characterizations by middle-out beam search through the space of possible conjunctive, disjunctive, and negated characterizations. STAGGER is a binary classification method which detects concept drift using backtracking and Bayesian weighting measures to discriminate concept drift from noise [17].

Salganicoff [18], in 1993, proposed the DARLING method (Density-Adaptive Reinforcement Learning), a supervised density-adaptive forgetting technique that uses exponential weight-decay based on nearest neighbor criterium to exclude instances considered obsolete. The method excludes an example if its weight drops below a threshold and new examples take its place. In the space of attributes, the number of samples per unit volume is the local "density." DARLING builds a *kd-tree* structure for the nearest-neighbor forgetting method in a binary classification way [18].

Kubat and Widmer [19] use Radial Basis Functions (RBF) to learn in non-stationary numeric domains. FRANN algorithm (Floating Rough Approximation in Neural Networks) uses hill-climbing to search for the best "sliding window," and build an RBF network from it. This dynamic sliding window uses heuristics to delete older instances and to determine its window size [19].

Widmer and Kubat [7] proposed a supervised framework named FLORA (FLOating Rough Approximation), which consists of a family of algorithms where each version is an anterior extension. In the FLORA framework, three description sets represent a concept: the Accepted DEScriptors (ADES) used to classify new samples, represent the present (positive) hypothesis; the Negative DEScriptors (NDES) represent negative samples and employed to avoid over-generalization of ADES; and the Potential DEScriptors (PDES) which is a set that might become pertinent in the future, used as a reservoir of general hypotheses. FLORA keeps the relevant descriptions items of the most recent instances in the window. FLORA2 has a "forgetting" operator, which dynamically adjust the window size in the training phase. Two heuristics are used to detect drift: the predictive performance of old classifications, and some syntactic properties of the hypotheses. If performance drops significantly or a substantial difference appears in the number of items in ADES, the method signals a warning of concept drift. The window size decreases and the algorithm "forget" old instances if drift occurs. Otherwise, the window size enlarges to produce a stable concept. FLORA3 can deal with recurring contexts, storing old concepts for later reuse. When a stable concept reached, it stores the current hypothesis. When concept drifts, the algorithm verifies if there is any old descriptor that can describe the present instances. FLORA4 was proposed to deal with noise. Each description item has a classification record, and it builds statistical confidence intervals around these measures [7].

Klinkenberg and Joachims [8] use Support Vector Machines to deal with concept drift. The method keeps a window of examples, trying to minimize its generalization error discarding irrelevant data. The authors used the full-memory, no-memory, window of fixed and adaptive size data management approaches [8].

Street and Kim [20] proposed the SEA algorithm (Streaming Ensemble Algorithm), an ensemble of decision trees and each one created by one batch. It uses unweighted majority-vote, similar to bagging, to classify an instance. The ensemble has a maximum number of classifiers, and once this number achieved, those new classifiers that reach certain criteria replace old classifiers. Performance estimates are computed on the next batch using the new tree, and the ensembles are built with Quinlan's C4.5 [20].

Stanley [9] proposed the CDC (Concept Drift Committee) algorithm, a supervised method that employs a weighted committee of hypotheses. All committee members can access all features and when an older committee member's voting value falls below a threshold, a new member replaces it. Each member gives a (weighted) vote and keeps a hypothesis based on instances seen in its lifetime, each other having a different amount of instances considered. An implicit window considers only the latest examples, and each committee member uses a decision tree (but according to authors any supervised learning algorithm can be used [9]).

Scholz and Klinkenberg [21] proposed the KBS (Knowledge-Based Sampling) algorithm, a boosting-like ensemble method. This supervised algorithm employs KBS and SVM with the linear kernel or Decision Tree in a binary classification way ("relevant" or "irrelevant" class). It considers the latest instances batch, inducing and reweighting base models continuously [21].

Bifet and Gavalda [14] presented the ADWIN2 (ADaptive WINdowing) algorithm, an improved version of ADWIN algorithm. ADWIN2 has a variable sized window: it grows or shrinks when no change or concept drift is detected, respectively. This supervised method detects drifts using the average of the elements in the window [14].

Abdulsalam *et al.* [22] combine the ideas of streaming decision trees and Random Forests in an incremental multiclass algorithm. It can adjust its parameters to handle drifts and uses the difference in entropy between two-windows – the current and the reference window – to detect drift. Each attribute has counters and the probabilities of occurrences are calculated, being the differences averaged and used to calculate entropy changes. When the method builds a new tree; it adds or replaces an old one depending on the number of existing trees in the forest [22].

With minimal distance classifier and a sliding window, Kurlej and Wozniak [23] proposed an active learning approach that ponders if an outside expert has to label a new example as training instance or not. A heuristic algorithm defines if a new example is a good reference using two values obtained in this set: the distance to the closest point and the difference in the distance to two closest points belonging to distinct classes. If a new example achieves certain conditions, it replaces the oldest example of the reference set [23].

Vivekanandan and Nedunchezhian [24] proposed an online Genetic Algorithm (GA) that takes small snapshots of the training sample and creates rules for all classes separately. For each class, it uses multiple windows of fixed size to keep the examples. The window size of each class varies depends on his overall distribution. The oldest examples will be replaced by new ones if the window of the class becomes full [24].

Sun and Li [25] proposed the first study on Financial Distress Concept Drift (FDCD): if there is FDCD and in what way to discard it. To discard FDCD, they build a dynamic FDP model, which embraces instance selection, FDP modeling, and future prediction. To deal with FDCD, the algorithm uses methods like the ones based on windows (full memory, no memory, fixed and adaptative size), and batch selection. The authors also integrate Mahalanobis distance for feature selection and employed Fisher discriminant for classification [25].

Hegedus *et al.* [26] presented adaptative versions of GoLF (Gossip Learning Framework): ADAGoLF (with age-based drift handling) and CDDGoLF (with concept drift detection), to handle concept drift in large networks. Adaptivity in AdaGoLF uses the models in the network, employing age distribution manipulation. It offers both young and old models at any moment, providing diversity of varied ages. CDDGoLF is employed to detect concept drift. It can estimate and monitor performance drifts to forget data with a performance decay [26].

Bertini *et al.* [27] proposed the Ensemble of CPp-AbDG (Complete P-partite Attribute-based Decision Graph), being the data representation based on graph structures. It can handle missing attribute values. CPp-AbDG extends other data graph constructor: AbDG (Attribute-based Decision Graph). The AbDG can be theoretically designed to a graph in which one vertice represents a subrange of the range value of one attribute. Graph-based model is employed as classifiers in the ensemble, since the authors justify the use by the advantages of representing data topologically, with arbitrary shapes, and hierarchically [27].

Escovedo *et al.* [28] presented the NEVE (Neuro-EVolutionary Ensemble) algorithm, a supervised ensemble of weighted classifiers (neural networks), which employs QIEA (Quantum-Inspired Evolutionary Algorithm) to train. QIEAs can estimates class distribution, providing good performance. It also determines weights for each ensemble's classifier when a new batch arrives. A new classifier is added to the ensemble once a new batch arrives, and all weights are updated to improve the performance [28].

Li *et al.* [29] proposed the EDTC (Ensemble Decision Trees for Concept drifting data streams), an incremental method that defines cut-points in the growing tree with three different random feature selection. When an instance arrives, each growing node split-features randomly to avoid producing unnecessary branches. EDTC employs two thresholds and uses local data distributions to detect drift [29].

Loeffel *et al.* [30] proposed an online algorithm (the Droplets algorithm) that can opt to abstain from predicting to handling drifts. Each instance is considered as a droplet falling on a plan (feature space), i.e., the "Droplets' map." Classes can be considered as a "chemical" composition of the droplets which are reciprocally repellent. Thus, two droplets from different classes will not mutual coverage parts of the map. The method does not employ a fixed threshold. According to authors, it is the former "algorithm to handle concept drift" proposed to abstaining from prediction [30].

Chen *et al.* [31] proposed a Genetic Algorithm to obtain fuzzy concept drift patterns. CDGFM (Concept Drift Genetic-Fuzzy Mining) handles drift with instance selection in a multiclass classification way. Membership functions of items are transformed into chromosomes and obtained on GA process. Each example has a fitness value assessed by the amount of concept drift patterns (new concept, drift and added/expired rules) and the membership function. The membership functions search for satisfying sets of fuzzy association rules to define drift [31].

ZareMoodi *et al.* [32] proposed LOCE (LOcal Classifier Ensemble), an algorithm capable to detect the appearance of new classes in streams. Each class has an ensemble of classifiers, which are updated by its own metrics and by a pruning phase (cutting classifiers to system update). If one class no longer exists, all classifiers belonging to the ensemble will be eliminated. It classifies existing classes' examples and discerns between the novel and existing classes with local patterns. Using a neighborhood graph to detect new classes, it stores new class candidates to its nodes using cohesion and separation to determine the components of it [32].

Diaz *et al.* [33] proposed the FAE (Fast Adapting Ensemble) algorithm, a multiclass ensemble algorithm which can deal with recurring concepts. It employs a batch scheme, but it does not need to wait for the entire batch to start classification. A drift detector (often Drift Detection Method - DDM) decides when to increase the number of classifiers to the ensemble. To deal with recurring concepts, it keeps several aged classifiers (former concepts) which are awakened if these concepts reoccur, avoiding unnecessarily inclusion of new classifiers. To make a global decision, it employs a weighted majority ensemble vote. It dynamically adjusts the base classifiers weights, which permits classifiers to stay longer on the ensemble [33].

Mirza *et al.* [34] proposed the ESOS-ELM (Ensemble of Subset Online Sequential Extreme Learning Machine), a drift detector which can deal with class imbalance on stream of data. The main ensemble represents the short-term memory, in which each classifier trains with a balanced selection of the original imbalanced data. It also has a concept drift detector, and the long-term memory keeps information. 'm' classifiers process a minority class instance, being 'm' the imbalance ratio; while a single classifier is employed to a majority class instance, which is processed in a round-robin fashion. According to the authors, the method can do both online and batch learning [34].

ZareMoodi *et al.* [35] proposed the SVSCLASS (Support Vector-based Stream CLASSifier), a support vector-based method that can deal with the appearance of new classes. Using the support vector domain description, it maintains classes' boundaries with spheres in the kernel space, which is used to label instances of the incoming batches. Instances located outside of the spheres may stand for an emergence of a new class. To detect a new class, it builds a neighborhood graph to analyze the cohesion together and separated from existing classes. Using support vector clustering, the instances will be partitioned into clusters and analyzed to be labeled (the method acquires the true labels and the sphere is updated). To handle concept drift, the spheres boundaries shrink, enlarge and merge during learning [35].

Klinkenberg and Renz [36] presented a method to detected concept drift using windows approach with fixed or adaptive size. The subject addressed in this paper was information filtering, having two concepts to deal with: relevant or irrelevant. Using some indicators (recall, precision, and accuracy, being the latter considered less important to detect drift according to authors), it detects concept drift and calculates heuristics to adapt the window size. If occurs an abrupt change, the window reaches its minimal size (one batch); whereas if occurs a gradual change, its size is decreased according to a reduction rate defined by the user. If no drift is detected, instances are stored until a maximum training set size to build a stable learner [36].

Gama *et al.* [1] proposed the DDM (Drift Detection Method), a drift detector that works with probability distribution using the online error-rate. It defines two levels: the warning and the drift level. If the error reaches respectively the warning level at instance i_w and the drift level at instance i_d, it is considered the occurrence of a drift, and the method starts the training process with data from i_w. According to the authors, it can operate with online and incremental methods, and also as a wrapper to batch classifiers [1].

Yang *et al.* [37] proposed a RePro system (REactive plus PROactive). RePro can conduct reactive prediction: it detects the concept drift and learns with new data; and also can be proactive: predict the next concept given the present one. They generate a concept history from the stream (which is more compact than raw data) to learn patterns of concept transitions. RePro employs a sliding window structure which starts with a misclassified example. Once the window is filled and its error rate stands above a threshold, the first example is considered a trigger; otherwise, the window pass to the next misclassified example, excluding previous examples. The proactive mode works after trigger detection and does not employ window warning. When a new trigger is identified, the predicted concept takes the lead. The reactive mode builds a model only after drift detection, using trigger examples. According to the authors, RePro can detect the emergence of a new concept and the reappearance of an old one [37].

García *et al.* [38] presented EDDM (Early Drift Detection Method), a drift detector that works with distance-error-rate (estimated distribution of the distance among classification errors) rather than errors classifications. EDDM has two thresholds: the Warning level – exceeded this level, instances are kept; and the Drift level – the method considers concept has drifted and builds a new model with data from warning level. EDDM starts to search for a concept drift after 30 errors occurred. Otherwise, if the system has a rise of similarity after warning threshold, the instances are deleted and the system reverts to an ‘‘in-control level’’ [38].

Da Silva *et al.* [39] proposed the RL-CD (Reinforcement Learning with Context Detection), which evaluates the prediction quality of partial model. It manages several partial models, creating, updating and selecting them. The method only select the partial model with the highest quality, each one specialized in a particular environment dynamics. A model replacement means a drift detection in the environment. A new model is created if the best model quality is below a threshold. The new model produced will consider dynamics predictor and the corresponding optimal policy [39].

Kolter and Maloof [10] proposed the DWM (Dynamic Weighted Majority), an ensemble algorithm to deal with concept drift that creates and removes weighted experts according to global performance. If the main procedure gives a wrong weighted majority answer, the method includes an expert to the ensemble. Otherwise, if an expert gives a wrong answer, its weight is decreased. If an expert gets his weight greatly decreased, it is deleted from the ensemble [10].

Katakis *et al.* [13] proposed the CCP (Conceptual Clustering & Prediction) framework, an incremental ensemble to deal with recurring contexts which uses clusters and a transformation function to map batches into conceptual representation models. Each classifier in the ensemble represents a concept, and when a new batch comes from the stream, the clustering algorithm identifies its concept and CCP employs the corresponding classifier to label the instances [13].

On the area of face detection, Susnjak *et al.* [40] proposed an adaptive learning method for cascades of boosted ensembles. The algorithm employs a hybrid of active and passive updating approach. During training, classifiers belonging to the same layer are combined into ensemble-clusters in which its results are weighted based on their performance. During classification, the algorithm uses the deliberations of ensemble-clusters per layer to calculate a collective value, which is compared with the learned layer thresholds to give a decisive answer. To detect drift it uses trigger mechanism with classification error rate [40].

Minku and Yao [41] proposed the DDD (Diversity for Dealing with Drifts) online ensemble algorithm. The authors also analyzed the combination of low and high diversity ensembles, concluding that each diversity level reaches better prequential accuracy according to the drift type. The paper further reveals the possibility of old concept information benefit the training process of a new concept by using different levels of diversity on ensemble learning, which is a measure that minimizes the error (i.e., discordance) among classifiers. So, DDD keeps ensembles with different diversity levels to deal with drifts, taking old concept information to assist the learning [41].

Farid *et al.* [11] proposed an adaptive Ensemble Model (EM) able to detect new class. EM uses automatic decision trees with clustering, which classify an instance by majority weighted voting. Some instances are labeled to train a new classifier, and when it becomes competitive, it can replace an older one with the smallest weight (which means it has the minimum classification accuracy rate). To detect new class, the assumption is that instances should be closer to each other if they belong to the same class, otherwise should be distant from instances of other classes. If an instance is distant from the present clusters, it is considered as a new class example [11].

Harel *et al.* [42] proposed a method that analyzes the empirical loss distribution, whose statistics are acquired by reusing the data multiple times via resampling. The method employs random permutations to detect drift, creating multiple train-test data from the stream. If no drift occurred, the ordered data prediction should not quite differ from the shuffled data when the algorithm achieves stability. Receiving time indices when concept drift, the method uses it to modify the windows size and initiate a training phase, and also providing information to ensemble learners. Instances suppose to have temporal independence, but according to authors, it can apply tactics to maintain exchangeability [42].

Raza *et al.* [43] proposed the ALCSD (Adaptive Learning with Covariate Shift-Detection), an adaptive algorithm that employs dataset shift-detection with exponential-weighted moving average (EWMA) model. An EWMA model-based shift-detection test supervises the covariate shift and initiates adaptation with the shift-detection point: it retrains classifiers with the updated knowledge base and different adaptation methods [43].

Lu *et al.* [44] proposed a method to detect drift in case-based reasoning (CBR) environment. They presented a competence model to detect drifts (using two sliding windows, it compares the data distribution through competence instead of the feature space). Competence measures the success of CBR system to accomplish its goals. Prior knowledge of case distribution is not necessary. It presents secure statistical of the drift detected and also maintains records and drift quantifications [44].

Wang and Abraham [45] proposed the LFR (Linear Four Rates) framework, a drift detector that can deal with batch and stream approaches, binary class problems and imbalanced data. LFR requires user-specific parameters but does not depend on the underlying statistical-model, and the drift detector is not dependent on the classifier [45].

Khamassi *et al.* [46] proposed the EDIST2, a drift detector with self-adaptive windowing to deal with different types of drift. EDIST2 supervises performance and detects drift with two sliding windows: a global and a current one. The global window (GW) increases in stable environments and shrinks if a drift is detected. The current window (CW) comprises only the current batch instances. EDIST2 computes the error distance distribution of GW and CW, and compares the difference between their error distance averages. It has three thresholds: In-Control level – assumes that there are no changes, therefore CW's instances are added to GW; Warning level – starts storing instances in an auxiliary window for a potential change; and Drift level – drift is detected and only instances kept from Warning level remains in GW. Drift threshold is automatically calculated with statistical proofs [46].

Gao *et al.* [47] argued that descriptive model similar to posterior probability is preferable for real-stream classification. They proposed the UCB (Uncorrelated Bagging), a framework to deal with imbalanced data that apply sampling and ensemble methods to skewed stream mining problem. It makes a balanced training data by maintaining all minority instances and under sampled-majority instances (minority class is assumed as stationary) [15]. The averaged probability calculated on application data by several models are the final outputs [47].

Yalcin *et al.* [48] employed Support Vector Machines in an ensemble-based incremental learning algorithm. In previous works, they integrate SVM and an ensemble framework with Learn++ to create an incremental learning algorithm (SVMLearn++). A forgetting approach is employed on SVMLearn++ to eliminate the effects of redundant data, and in this work SVMLearn++ was assessed in the following learning approaches: Learn++ without pruning, with top N highest performance classifiers, and with replacing the looser device. The authors use SVM with RBF kernel in two-class problems [48].

Chen and He [49] proposed the SERA (SElectively Recursive Approach) algorithm to handle imbalanced data. It selectively retains minority instances in the current batch and it assigns the sampling probabilities proportionally to the majority and minority instances to increase minority instances performance. The amount of minority instances is limited by the authors to be proportional to the size of majority data, and a Mahalanobis distance decides the priority order of acceptance. They also elaborated a biased bagging approach (BBagging) to boost the performance of a single classifier focusing on minority instances on imbalanced datasets [49].

Elwell and Polikar [50] presented the Learn++.NSE algorithm, an incremental ensemble algorithm depicted by Non-Stationary Environments (NSEs). Learn++.NSE receives the batches and incrementally learn from them without requesting access to previous data and handles concept drift with a passive approach. One classifier is trained at each batch and the results are obtained by combining the ensemble dynamically-weighted-majority vote based on their time-adjusted errors. Learn++.NSE can handle the appearance of a new class and the deletion of an old one [50].

Ditzler *et al.* [51] proposed the Learn++.SMOTE, a hybrid algorithm containing Learn++.NSE and SMOTE approaches to deal with class imbalance. This hybrid algorithm can boost the recall of the minority class, and any supervised learning algorithm can be used as base classifier [51].

Ditzler and Polikar [52] proposed a method based on Learn++.NSE algorithm: the Learn++.NIE (Nonstationary and Imbalanced Environments). Differences in NSE and NIE are i) for each batch NIE generates a sub-ensemble (rather than an individual classifier), and ii) another metric (not a classification error) is employed as an evaluation measure. Compared to Learn++.NSE, Learn++.NIE has slow recovery but can boost minority class performance. It can separately weight the average error of the majority and minority class recall and can also reward classifiers with good performance not only in the majority class but both minority and majority classes [52].

Ditzler and Polikar [15] proposed two incremental ensemble algorithm to deal with imbalanced data. The first method (Learn++.CDS) is a logical union of Learn++.NSE algorithm and the Synthetic Minority class Oversampling TEchnique (SMOTE) to learn in imbalanced environments. The second method is Learn++.NIE. Learn++.CDS employs SMOTE to readjust the class balance with synthetic minority class examples, then uses Learn++.NSE on this balanced data. Both Learn++.CDS and Learn++.NIE reveal to perform better on fixed ensemble size [15].

Klinkenberg [53] employed Support Vector Machines to handle concept drift. Exploiting the work in [8], they reduce the need for labeled data using unlabeled instances in a transductive way. It discards irrelevant data minimizing the generalization error with the use of automatically adjusted size windows. The method employs some properties of SVMs, adapting $\varepsilon\alpha$-estimates (a particular process to assess SVM performance based on the idea of leave-one-out estimation) to select the window size [53].

Widyantoro [54] proposed a framework that employs unlabeled data on information-filtering domains. A concept hierarchy is incrementally built in an unsupervised way to be used on classification. To deduce concepts is employed a persistence assumption in temporal reasoning. The method permits to classifier be tailored to target applications. The performance depends on the method for identifying classes, the method to build the hierarchy, and the chosen classifier [54].

Spinosa *et al.* [55] proposed the OLINDDA (OnLIne Novelty and Drift Detection Algorithm), a cluster-based algorithm that can detect new concepts and assessed in intrusion detection domain. Initially, it creates a normal profile of a single class, and to identify a new concept it uses cohesive sets of clusters, merging similar concepts during learning. OLINDDA stores information in three hypersphere-based models, which is about normal profile, extended concepts of normal profile and new concepts. The normal model is static and used as a reference. The extension and new concepts are continuously created and updated. It has both supervised and unsupervised learning, where the first occurs building the normal model and the latter treating unlabeled instances and detecting novel concepts [55].

Masud *et al.* [56] proposed the ECSMiner (Enhanced Classifier for data Streams with novel class Miner), a method to detect the appearance of a new class. To handle drift, it uses M classifiers in an ensemble and the majority vote, which is continuously updated: when a new model is trained, it can replace an older one with the highest error. Each classifier has a class detector and if all of them declare a novel class, a new class is determined. To do such thing, it verifies cohesion and separation, and if an instance is isolated from training examples, it is recognized as an $Foutlier$. If an instance is not an $Foutlier$, ECSMiner uses ensemble voting to classify it. If a number of cohesive $Foutlier$ is reached, a new class is determined [56].

Masud *et al.* [57] proposed the SCANR (Stream Classifier And Novel and Recurring class detector), a method to deal with recurring class and to detect new ones in multi-class problems. It maintains a primary and auxiliary ensembles to store old classification models. After building a new primary model, it substitutes the model in primary ensemble with the worst prediction error. The primary ensemble analyses instances to detect outliers and starts classification if the instance is not an outlier. If it is classified as an outlier (primary outlier), the auxiliary ensemble verifies it again. If it is not classified as an outlier, it is assumed to be a recurring class and classified by the auxiliary ensemble. Otherwise, it is named as secondary outlier, and it is provisionally added into a buffer. The new class detection module starts if buffer instances reach a certain number [57].

Li *et al.* [58] proposed the SUN method, a Semi-supervised classification algorithm for data streams with concept drifts and UNlabeled data. It creates a decision tree generating concept clusters at leaves with k-Modes. It detects drifts from noise with a bottom-up search and the deviation among history concepts and new ones. The concept clusters classify instances, and the labeled examples are used again trying to reduce the drift rate [58].

Katakis *et al.* [59] proposed the CCP (Conceptual Clustering and Prediction) framework, a probabilistic representation model for stream learning employing incremental cluster algorithms. It maps batches into ''Conceptual Vectors'' (CV) containing conceptual information, and those vectors geometrically close do always belong to the same conceptual theme. Clustering works in the stream of CV, summarizing batches into concepts. For each batch, CCP assigns the concept (cluster) and employs the specific classifier. One advantage of CPP is having to store only the clusters' centers and the classifiers for every cluster, with no need to store old batches or CV [59].

Sethi and Kantardzic [60] proposed the MD3 (Margin Density Drift Detection), a drift detector algorithm for unlabeled stream of data. The number of instances in the area of uncertainty of the classifier (margin) is used as a metric to detect drift. If a variation in margin density occurs, the classifier needs labeled instances to be retrained.

Tennant *et al.* [61] proposed the MC-NN (Micro-Cluster Nearest Neighbour) method that calculates statistics from the stream and employs nearest neighbour algorithm, which can characterize MC-NN in parallel. The serial MC-NN data has no need to be in memory and is incrementally processed. A statistical summary is built as a set of variance based Micro-Clusters (MC). MC handles concept drift through statistics update of new instances. The parallel MC-NN distribute MC to computational nodes in a computer cluster. Each node of parallel version adapts to drift likewise serial version with a voting mechanism to classify instances [61].

Liu *et al.* [62] proposed a drift detector in sensor network domain based on angle optimized global embedding (AOGE) and principal component analysis (PCA). The drift detector decreases time processing due to the compatibility between the detector and data processing and also improves performance through dimension reduction (with PCA). PCA and AOGE examine the projection variance and angle, respectively. Combined, they are used to identify changes on objective function. The authors used Extreme Learning Machine (ELM) and SVM as classifiers [62].

Silva *et al.* [63] proposed the FEAC-Stream (Fast Evolutionary Algorithm for Clustering data Streams) algorithm, which uses k-means clustering with k automatically estimated from the stream. FEAC-Stream uses the Page-Hinkley Test to identify decreases in clusters quality to start the re-estimation of k with an evolutionary algorithm. It considers that partially unknown data can afford valid stream knowledge [63].

Xu and Wang [64] proposed the DELM (Dynamic Extreme Learning Machine) to classify online data stream employing ELM as classifier. With the use of thresholds to detect drift, it employs a double hidden layer structure to train and improve the performance: when an alert of drift is issued, additional hidden layer nodes are included on neural network; once the drift is detected, a new classifier replaces an older classifier with low performance [64].

Arabmakki and Kantardzic [65] proposed the RLS-SOM (Reduced labeled Samples-Self Organizing Map) framework for imbalanced stream. An ensemble classifies with DWM and retrains a new model using partial labeled samples when a drift is detected. The method uses the global answer and each one of individual answers: if an individual model has higher performance than the ensemble's performance, it is chosen instead of the others. After drift detection, the method selects majority and minority class instances to train a new model. If it is a conditional drift, SVM is employed to choose the closest instances of decision boundary (margin). However, if no minority instance is found on decision boundary, SOM algorithm maps the batch to search for minority instances in the whole feature space [65].

Zhang *et al.* [66] proposed a three-layered drift detection technique in text data streams domain, where each layer denotes, respectively: the layer of label space, the layer of feature space, and the layer of the mapping relationships between labels and features. According to authors, it can employ any classifier and can measure changes in each layer to detect different types of drift [66].

The methods are generally evaluated with metrics as accuracy, precision, recall, and error. The metrics less used include time, little detection delay, detection delay, false alarm, prequential error, the total number of changes detected, prequential accuracy, Monte Carlo error, predictive accuracy, prediction errors, Gmean, among others.

Table 1 categorizes the techniques according to the drift detector mechanism, and Table 2 presents the classifiers employed in the articles considered in this work. Table 3 depicts the type of concept drift handling used: instance selection (i.e., a window of fixed size, window with automatically adjustment size, etc), instance weighting, ensemble learning, clustering or sampling. Table 4 categorizes the techniques according to the learning approach, i.e., supervised, unsupervised or semi-supervised.

TABLE 1. Drift detector mechanism.

Drift detector	[1], [7], [13], [14], [17], [22], [26], [29], [33], [34], [36]–[46], [58]–[60], [62]–[66]
No drift detector	[8]–[11], [15], [18]–[21], [23]–[28], [30]–[32], [35], [47]–[57], [61]

TABLE 2. Classifiers.

SVM	[8], [21], [35], [36], [43], [45], [48], [50], [53], [60], [62], [65], [66]
Decision trees	[1], [9], [11], [20]–[22], [29], [33], [36]–[38], [41], [46], [47], [52], [56], [58], [60], [65], [66]
Naive Bayes	[10], [11], [13], [14], [29], [36], [41], [43], [47], [50], [59], [66]
k-means clusterer	[14], [32], [55], [57], [63]
knn	[36], [42]–[44], [51], [56]
Logistic Regression	[26], [47]
Neural Networks	[1], [28], [41], [49], [51], [52], [61]
Regression tree	[15], [50]
ELM	[34], [62], [64]
Others	[7], [10], [17]–[19], [23]–[25], [27], [30], [31], [36], [38]–[40], [54], [65]

TABLE 3. Concept drift handling.

Instance selection	[1], [7]–[9], [14], [19], [23]–[26], [30], [31], [35]–[39], [42]–[46], [53], [60]–[62], [64]
Instance weighting	[17], [18]
Ensemble learning	[9]–[11], [13], [15], [20]–[22], [27]–[29], [32]–[34], [40], [41], [47], [48], [50]–[52], [56], [57], [65], [66]
Clustering	[54], [55], [58], [59], [63]
Sampling	[47], [49]

TABLE 4. Learning approaches.

Supervised	[1], [7]–[11], [13]–[15], [17]–[52], [61], [62], [64]–[66]
Unsupervised	[60], [63]
Semisupervised	[53]–[59]

A. DATASETS USED IN THE LITERATURE

In this subsection, we summarize some real-world and synthetic datasets used in the literature to test and simulate concept drift environments. Synthetic datasets are very significant as we can affirm that concept drift really exist and specify which type of change is (i.e., gradual or abrupt).

1) SYNTHETIC DATASETS

Below, some synthetic datasets used in literature:

- SINE1: abrupt concept drift, noise-free instances. Two relevant attributes, each one with a uniformly distributed value in [0, 1]. In the first concept, it classifies as positive if a value stands below the curve $y = sin(x)$; is classified as negative otherwise. After drift, the classification is reversed [1].
- SINE2: has two relevant attributes like SINE1. Classification function is $y < 0.5 + 0.3 \times sin(3\pi x)$. After drift, the concept is inverted [1].

- SINIRREL1: has classification function as SINE1, the instances, however, present two irrelevant attributes [1].
- SINIRREL2: has classification function as SINE2, but like SINIRREL1, instances present two irrelevant attributes [1].
- CIRCLES: presents gradual drift and instances without noise. It has four classification function determined by four circles (four concepts). Instances are classified according to its location: if it is inside the circle defined by circular function, it is classified as positive; otherwise is negative. The gradual change occurs by modifying the center and radius size of the circle [1].
- GAUSS: has abrupt drift and instances with noise. Domain $R \times R$ with two relevant attributes. Positive instances are located as a normal distribution with center [0, 0] and standard deviation as of 1. Negative instances have center [2, 0] and standard deviation as of 4. After drift, the classification is reversed [1].
- SINE1G: presents very slow gradual drift and instances without noise. Similar to Sine1, but the gradual drift is reached by selecting instances from the past and the current concept (transition time among concepts). To select an instance from the past concept and the new one has, respectively, gradually lower probability and gradually higher probability as time passes [38].
- STAGGER: has abrupt drift and instances without noise. Instances have three symbolic attributes – *size* (*small*, *medium*, *large*), *color* (*red*, *green*), and *shape* (*circular*, *non-circular*). In the first concept, instances are positive if $size = small \wedge color = red$. In second concept the instances are defined by $color = green \vee shape = circular$. In third concept the instances are defined by $size = medium \vee size = large$ [1].
- MIXED: has abrupt drift and instances without noise. Instances have four attributes: two boolean v, w and two numeric between [0, 1]. If an instance has the following two of three conditions satisfied, it is classified as positive: $v, w, y < 0.5 + 0.3 \times sin(3\pi x)$. After drift, classification is reversed [1].
- Rotating Hyperplane Dataset: designed by [67]. The (k, t) pairs details of each concept and all files are available in the Internet[1] [45].
- Usenet1 and Usenet2: used in [68], are available in the Internet.[2] They collected reports from several newsgroups (e.g., medicine, space, baseball) of a user. The distinction among datasets is the drift dimension: the Usenet1 dataset has a sharper topic drift [45].
- Usenet: text dataset inspired by Katakis *et al.* [13], available in the Internet.[3] Usenet simulates a news filtering from 20 Newsgroups with change of interest of a user (concept drift). There are six topics, the user is interested in two, but is subscribed in four. It also simulates recurring concepts repeating topics of interest (three concepts in training data and three recurring concepts in testing data). The dataset has 5,931 instances and 659 attributes, which are binary values (presence or absence of the respective word).
- SEA Concepts Dataset: proposed by Street and Kim [20], this dataset has 60,000 instances, 3 attributes, and 3 classes; with 10% of noise. The numeric attributes are between 0 and 10 with two relevant attributes. Instances are divided into groups of 15,000 into four concepts. Each concept has different thresholds values (8, 9, 7, and 9.5), and the concept function to determine 0 to a class instance is $relevant\_feature1 + relevant\_feature2 > Threshold$. Dataset is available in the Internet,[4] and is quite used by concept drift handling algorithms [45].

2) REAL-WORLD DATASETS

In regard to the real-world datasets, UCI machine learning repository [69] are also cited in concept drift literature:

- KDD Cup 1999: the Knowledge Discovery and Data mining 1999 (KDD99) competition data contains simulated invasions in a military network domain. The complete dataset has 5000,000 instances, and the dataset available in the Internet[5] contains only 10% of the size. The original dataset has 24 attack types, and to simplify it into a binary-class problem, it changes the labels of attack types to abnormal and normal. To detect intrusion, it has to differentiate between attacks and normal connections. It includes a wide type of intrusions, so the attack is not a minority class. The dataset has 49,4020 instances, where each one stands for a connection with 41 attributes (i.e., connection length, protocol type, network service on the destination, etc). This dataset is tested in many concept drift handling algorithm [41].
- NSL-KDD Database: it is a KDD99 dataset version that solves issues of the anterior one: it does not include redundant and duplicate instances. Each instance has 41 attributes, and the 25,192 instances are distributed into 23 classes, consisting of normal classes (13,449 instances) and intrusion classes, which can be types of: Denial of service - DoS (9,234), Remote to user - R2U (209), User to root - U2R (11) or Probing (2,289) [11].
- Large Soybean Database: there are 683 instances and 19 classes, each instance consisting of 35 attributes, some nominal and some ordered [69] (some authors uses all attributes nominalized, e.g., employing string rather numerical values [11]).
- Image Segmentation Database: this dataset emphasizes image segmentation and boundary detection domain.

[1]http://www.win.tue.nl/m̃pechen/data/ DriftSets/
[2]http://mlkd.csd.auth.gr/concept_drift.html
[3]http://www.liaad.up.pt/kdus/products/datasets-for-concept-drift
[4]http://www.liaad.up.pt/kdus/products/datasets-for-concept-drift
[5]http://www.liaad.up.pt/kdus/products/datasets-for-concept-drift

There are 2,310 instances with 19 attributes and 7 classes (outdoor images) including brickface, sky, foliage, etc [11]. It creates a classification for every pixel with image handsegmention, and each instance is a 3x3 region [69].

- Adult: extracted from U.S. Census Bureau with the aim to predict if a person achieves an amount of around \$50,000 per year by using 14 demographic features (i.e., age, level of education, marital status, occupation, gender). This dataset has 44,848 people and 29.3% of them belongs to "over 50k" class [20].
- SEER Breast Cancer: used in [20], it contains 44,000 breast cancer patients accompaniment from the Surveillance, Epidemiology, and End Results (SEER) program of the National Institutes of Health. They consider that patients of class 1 died of breast cancer in between five years of surgery and patients of class 2 have lived at least five years. The resulting dataset has 37,715 instances, 25.7% of it classified as class 1 [20].
- Covertype: comprises types of cover forest from US Forest Service. It has 581,000 instances, 54 attributes, and 7 classes. ZareMoodi *et al.* 32] and Masud *et al.* [56] normalized the data to have two or three classes in each batch, with the appearance of new random classes in some of them.
- Poker: each instance represents a hand, which is five cards from a deck of 52. An instance has 10 attributes, in which each card having two attributes (suit and rank). "Poker Hand" is a class attribute. The order of cards matters and it provides 480 Royal Flush hands in contrast to 4 (one for each suit) [69].

Another real-world dataset frequently tested in literature is Electricity Market Dataset (ELEC2), first described by Harries [70]. The goal of this dataset is to recognize if the electricity price will increase or decrease [10]. Data was collected from TransGrid, an Australian New South Wales Electricity Market, in which the demand and supply of products affect its prices. Harries [70] presents the seasonality and the sensitivity of the price and short-term events (like weather variations), respectively. Electricity market was extended to nearby areas: the excess production of one area can be sold in the adjacent one, which can dampener the extreme prices. The ELEC2 dataset comprises 45,312 instances from 7 May 1996 to 5 December 1998. Each instance assigns to a 30 minutes duration, and has 5 attributes: the weekday (an integer between $[1, 7]$); the time stamp (a day period, a number between $[1, 48]$); the New South Wales electricity demand (numeric attribute); the Victoria electricity demand (numeric attribute); the programmed electricity transfer between states (numeric attribute) and the class label (a binary value between up or down that recognizes price changes of the last 24 hours). The dataset attractive is the real-world data characteristics: not knowing if there is a drift and when it occurs [38].

Real-world datasets less used can be referred as well:

- Calendar Apprentice (CAP): dataset used to predict user preferences for scheduling appointment in an academic institution [71]. The users preference to be predicted is the local of the appointment, duration, starting time, and weekday. An instance has 34 features – such as the type and scope of the meeting, kind of participants, and if it happens during lunchtime – with combinations of these features. There are 12 features for places, 11 for duration, 15 for start time, and 16 for the day of week [10].
- PAKDD 2009: consist of data from private label credit card operation on stable inflation condition of a major Brazilian retail chain. It has 50,000 instances of a one-year period, in which each instance represents a client by the use of 27 attributes, such as sex, age, marital status, profession, income, etc. Class identifies if the client is a "good" or a "bad" one, being the last a minority class composed by 19.5% of the data [41].

The next two high dimensional datasets were from e-mail filtering domain. The former depicts sudden drift and recurring contexts, and the latter depicts gradual drift. Both datasets are accessible in Weka (ARFF) format in Boolean bag-of-words vector representation athttp://mlkd.csd.auth.gr/concept drift.html [13]:

- Emailing List (elist) Dataset: consists of e-mail messages simulating some topics, labeled as interesting or junk depending on the user interest: the goal is to train and classify messages with user feedback. It collects messages from usenet posts of 20 Newsgroup collection [8]. The selected topics are: science/medicine, science/space, and recreation/sports/baseball. The dataset contains 1,500 instances and 913 attributes with words found at least 10 times on the message (boolean bag-of-words). 300 instances are assigned in five time periods: In the first period, medicine is the interesting topic, and the topic of interest at the end of each period changes to simulate concept drift [13].
- Spam Filtering Dataset: consists of e-mail messages from the Spam Assassin Collection. It has four segments: spam, spam2, ham (legitimate), and easy ham, which is quickly identified legitimate messages. Spam ratio of the original set is nearly 20%, and to transform it into a longitudinal data, the email sent date and time is extracted and converted into the format *yyyyMMddhh mmss* (*yyyy*: year, *MM*: month, *dd*: day, *hh*: hours, *mm*: minutes, *ss*: seconds). It maintains all copies, even if the user has more than one of the same e-mail, but the attachments are removed. It employs the boolean bag-of-words approach to represent e-mails. The dataset contains 9,324 instances and 500 attributes (words acquired by employing feature selection with the X^2 measure). This dataset has gradual concept drift [13]. In the Internet[6] is available a spam dataset consisting

[6]http://www.liaad.up.pt/kdus/products/datasets-for-concept-drift

of 9,324 instances and 40,000 attributes with gradual drift and binary class (legitimate and spam) with nearly 20% spam ratio.

IV. DISCUSSION

In this section, a summarization concerning all works are presented and further discussed. Fig. 5 shows the percentage of each learning method used in the articles separated by supervised, unsupervised or semi-supervised learning. Clearly, one can observe that supervised learning is by far the widest methodology employed in the context of concept drift. Such numbers can be explained by the preference for using supervised classifiers in most articles, and by the advantage of detecting changes in data distribution when one has the class information of a sample.

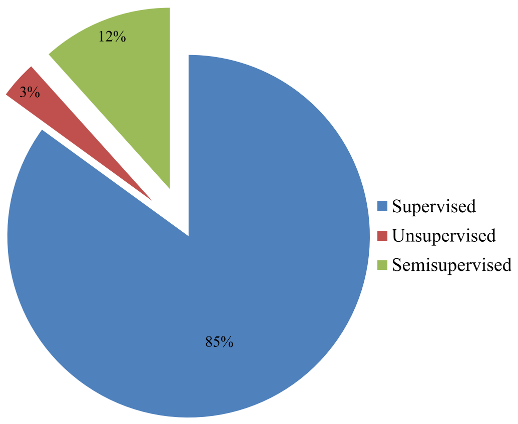

FIGURE 5. Percentage concerning different learning approaches used in concept drift.

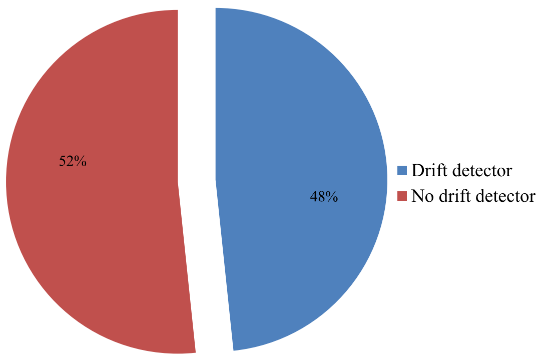

FIGURE 6. Percentage of drift detector mechanisms.

Fig. 6 shows the percentage of drift detection mechanism usage, i.e, whether the method is an active or a passive approach. If it is an active approach, it has a drift detection method that informs whether and when a drift occurs. If it is a passive approach, it does not have a drift detection mechanism and the algorithm assumes the drift may occur at any time and updates the model independently. The use of a drift detection mechanism requires a metric to determine that there is a drift, which can influence the performance. Therefore, the lack of adoption of a drift detection mechanism can influence the training time, i.e., the method has to determine when to update its model, which can be costly to do regularly. It is not clear to assume which is the best one since these methods were implemented in almost equivalent quantities, being the passive approach a little more used.

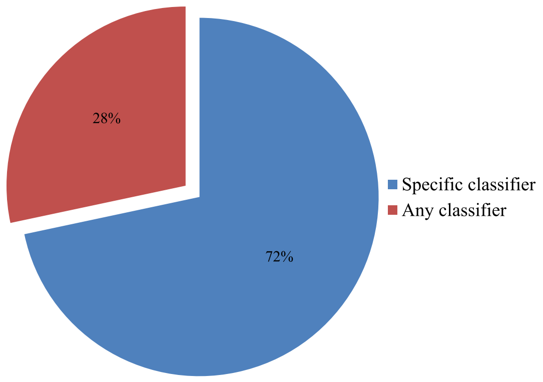

FIGURE 7. Percentage of specific or any-classifier approach that can be used.

Fig. 7 shows the percentage of classifiers the methods can handle, i.e., whether the method is specific for one classifier or it can be used with any other classifier. Notice the methods use a specific classifier to handle concept drift mostly, which means the techniques are usually designed with a specific technique in mind, and in some cases taking advantage of the characteristics of the classifier on the method.

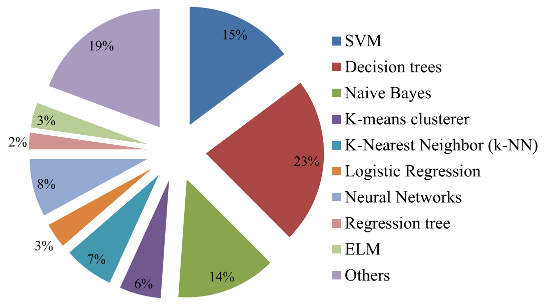

FIGURE 8. Percentage of the classifiers used in the works considered in this survey.

Fig. 8 shows the percentage of classifiers used in the articles, being the most used one the decision trees (DTs), followed by SVM and naïve Bayes. DTs are commonly used in ensemble learning due to their efficiency, thus turning out to be pretty much suitable for handling data streams efficiently. Since SVM and naïve Bayes are very popular classifiers in the community, it is expected they have been employed more regularly.

Fig. 9 shows the percentage of types of concept drift handling used in the articles. The most used approaches are the instance selection and ensemble learning. Instance selection can be easier to implement, i.e., it selects instances within a fixed or dynamic sliding window considering recent samples

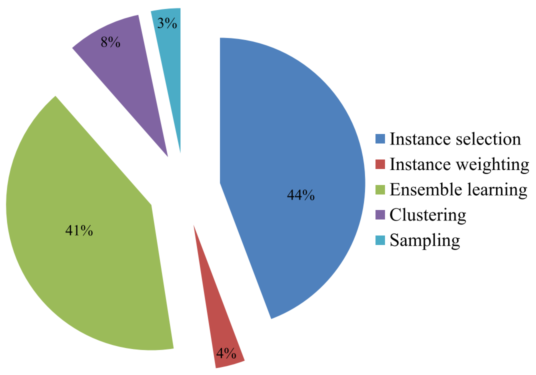

FIGURE 9. Concept drift handling.

that are more significant. Ensemble updating is easier to perform as well since it maintains a dynamic set of classifiers that are updated according to some criteria.

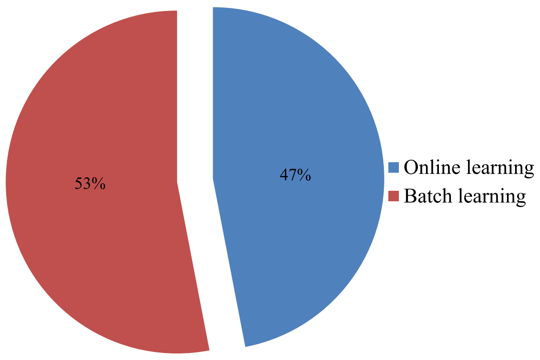

FIGURE 10. Online or Batch approaches.

Fig. 10 shows the percentage of methods using online learning (which evolves and updates a model as instances are processed) and batch learning (which learns by examining a collection of instances at once), being the batch learning approaches employed a little more in the evaluated methods. The approach selection depends on the classifier and the method implementation, among other details.

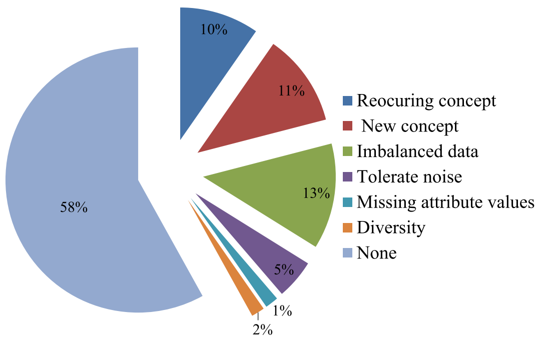

FIGURE 11. Other issues addressed in the articles.

Fig. 11 shows the percentage of other issues addressed in articles in addition to concept drift handling, being the most addressed drawbacks the imbalanced data (when the class distribution is imbalanced, i.e., having minority and majority classes), followed by new concept handling (or concept evolution – emergence of a new concept in the environment mainly in unsupervised learning), and reoccurring concept (when old concepts may re-appear in the future).

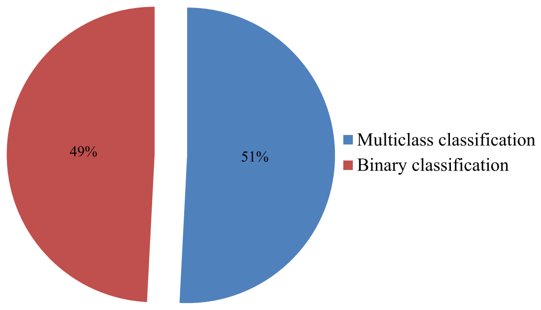

FIGURE 12. Number of classes recognized by the methods.

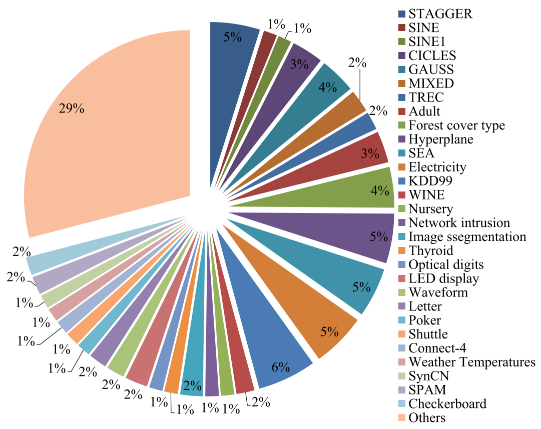

FIGURE 13. Datasets used in the articles.

Fig. 12 shows the percentage of binary classifiers (when the problem has only two classes, i.e., relevant or irrelevant class, positive or negative class, among others) and multiclass classifiers used in the articles, being both nearly equally used (multiclass classifiers are a little more used). Fig. 13 shows the percentage of each dataset used in the articles, being the most used dataset the ''KDD Cup 1999"''; followed by ''STAGGER,'' ''Electricity,'' ''Hyperplane'' and ''SEA'' datasets; and then ''Gauss'' and ''Forest Cover type'' datasets.

Fig. 14 shows the percentage of methods compared in the articles, being the most used methods the Window-of-fixed-size, DDM, Learn++ family, and methods that do not handle concept drift; followed by Full-memory methods, EDDM, CVFDT, and DWM. The Window-of-fixed-size is widely used due to its implementation simplicity for new classifiers in nonstationary environments; the DDM method is a popular drift detector method for active approaches; and the Learn++ family has different techniques to deal with concept drift.

Finally, Fig. 15 shows the percentage of articles published by year. Notice that 2015 had more articles published,

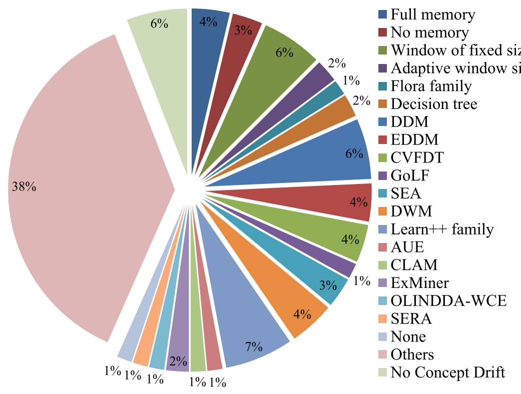

FIGURE 14. Methods compared in the articles.

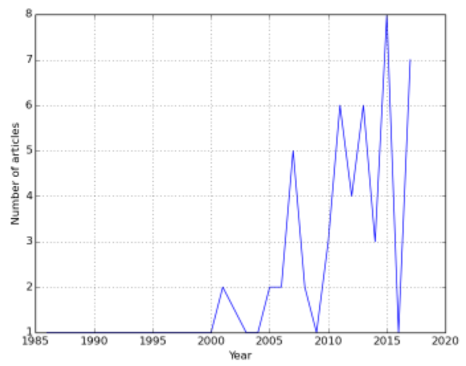

FIGURE 15. Articles separated by years.

followed by the year of 2017. However, one can observe that such area of research has been focused even more yearly, thus showing the increasing interest by the scientific community.

V. CONCLUSIONS

In this work, we presented the concept drift problem, classifying the variants of target concept in different forms. We also named some types of algorithms to deal with concept drift, including instance selection or window-based approaches, weight-based approaches, and ensemble of classifiers. The window-based approaches can be full-memory, no-memory, window-of-fixed-size, or window-of-adapting-size, depending on the treatment of the batches. In [15], they characterized concept drift algorithms in others ways such as online vs. batch approaches; single classifier vs. ensemble-based approaches; incremental vs. non-incremental approaches; and active vs. passive approaches.

We also summarized some techniques available in the literature that detect or deal with concept drift, like Learn++ family, DDM, DWM; in addition to some real-world and synthetics datasets used in the literature to test and simulate concept drift environment like Electricity, KDD Cup 1999, STAGGER, Hyperplane, and SEA datasets. Finally, we summarized in percentage charts the articles considered in this overview.

Future directions concerning the area may be related to other issues in addition to concept drift handling like imbalanced data, new concept handling, and reoccurring concept, as well as more studies in unsupervised environments.

REFERENCES

[1] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Proc. SBIA Brazilian Symp. Artif. Intell.*, vol. 3171. New York, NY, USA: Springer-Verlag, 2004, pp. 286–295.
[2] I. liobaitè. (Oct. 2010). "Learning under concept drift: An overview." [Online]. Available: https://arxiv.org/abs/1010.4784
[3] J. Gama, I. liobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, Apr. 2014
[4] H. Ishiwara, Y. Aoyama, S. Okada, C. Shimamura, and E. Tokumitsu, "Ferroelectric neuron circuits with adaptive-learning function," *Comput. Elect. Eng.*, vol. 23, no. 6, pp. 431–438, 1997.
[5] Y. Z. Tsypkin and B. T. Poljak, "Optimal recurrent algorithms for identification of nonstationary plants," *Comput. Elect. Eng.*, vol. 18, no. 5, pp. 365–371, 1992.
[6] M. Törngren and J. Wikander, "Real-time control of physically distributed systems: Application: Motion control," *Comput. Elect. Eng.*, vol. 18, no. 1, pp. 51–72, 1992.
[7] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, 1996.
[8] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines," in *Proc. 17th Int. Conf. Mach. Learn.* Burlington, MA, USA: Morgan Kaufmann, 2000, pp. 487–494.
[9] K. O. Stanley, "Learning concept drift with a committee of decision trees," Univ. Texas Austin, Dept. Comput. Sci., Tech. Rep. UT-AI-TR-03-302, 2003.
[10] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.
[11] D. Farid *et al.*, "An adaptive ensemble classifier for mining concept drifting data streams," *Expert Syst. Appl.*, vol. 40, no. 15, pp. 5895–5906, 2013.
[12] K. Wadewale and S. Desai, "Survey on method of drift detection and classification for time varying data set," *Int. Res. J. Eng. Technol.*, vol. 2, no. 9, pp. 709–713, 2015.
[13] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: An application to email filtering," *Knowl. Inf. Syst.*, vol. 22, no. 3, pp. 371–391, 2010.
[14] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*, 2007, pp. 443–448.
[15] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, Oct. 2013.
[16] A. Tsymbal, "The problem of concept drift: Definitions and related work," Comput. Sci. Dept., Trinity College Dublin, Dublin, Ireland, Tech. Rep. TCD-CS-2004-15, 2004.
[17] J. Schlimmer and R. Granger, Jr., "Beyond incremental processing: Tracking concept drift," in *Proc. 5th Nat. Conf. Artif. Intell.*, T. Kehler and S. Rosenschein, Eds., 1986, pp. 502–507.
[18] M. Salganicoff, "Density-adaptive learning and forgetting," in *Proc. 10th Int. Conf. Mach. Learn.*, 1993, pp. 276–283.
[19] M. Kubat and G. Widmer, "Adapting to drift in continuous domains," in *Proc. 8th Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 1995, pp. 307–310.
[20] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2001, pp. 377–382.
[21] M. Scholz and R. Klinkenberg, "An ensemble classifier for drifting concepts," in *Proc. 2nd Int. Workshop Knowl. Discovery Data Streams*, 2005, pp. 53–64.
[22] H. Abdulsalam, D. B. Skillicorn, and P. Martin, "Classification using streaming random forests," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 1, pp. 22–36, Jan. 2011.
[23] B. Kurlej and M. Wozniak, "Active learning approach to concept drift problem," *Logic J. IGPL*, vol. 20, no. 3, pp. 550–559, 2012.
[24] P. Vivekanandan and R. Nedunchezhian, "Mining data streams with concept drifts using genetic algorithm," *Artif. Intell. Rev.*, vol. 36, no. 3, pp. 163–178, 2011.

[25] J. Sun and H. Li, "Dynamic financial distress prediction using instance selection for the disposal of concept drift," *Expert Syst. Appl.*, vol. 38, no. 3, pp. 2566–2576, 2011.

[26] I. Hegedűs, R. Ormándi, and M. Jelasity, "Massively distributed concept drift handling in large networks," *Adv. Complex Syst.*, vol. 16, no. 04n05, p. 1350021, 2013.

[27] J. R. Bertini, M. de Carmo Nicoletti, and L. Zhao, "Ensemble of complete P-partite graph classifiers for non-stationary environments," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2013, pp. 1802–1809.

[28] T. Escovedo, A. V. A. da Cruz, M. M. B. R. Vellasco, and A. S. Koshiyama, "Learning under concept drift using a neuro-evolutionary ensemble," *Int. J. Comput. Intell. Appl.*, vol. 12, no. 4, p. 1340002, 2013.

[29] P. Li, X. Wu, X. Hu, and H. Wang, "Learning concept-drifting data streams with random ensemble decision trees," *Neurocomputing*, vol. 166, pp. 68–83, Oct. 2015.

[30] P.-X. Loeffel, C. Marsala, and M. Detyniecki, "Classification with a reject option under concept drift: The droplets algorithm," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2015, pp. 1–9.

[31] C.-H. Chen, Y. Li, T.-P. Hong, Y.-K. Li, and E. H.-C. Lu, "A GA-based approach for mining membership functions and concept-drift patterns," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 2961–2965.

[32] P. ZareMoodi, H. Beigy, and S. K. Siahroudi, "Novel class detection in data streams using local patterns and neighborhood graph," *Neurocomputing*, vol. 158, pp. 234–245, Jun. 2015.

[33] A. O. Díaz *et al.*, "Fast adapting ensemble: A new algorithm for mining data streams with concept drift," *Sci. World J.*, vol. 2015, Sep. 2014, Art. no. 235810.

[34] B. Mirza, Z. Lin, and N. Liu, "Ensemble of subset online sequential extreme learning machine for class imbalance and concept drift," *Neurocomputing*, vol. 149, pp. 316–329, Feb. 2015.

[35] P. ZareMoodi, S. K. Siahroudi, and H. Beigy, "A support vector based approach for classification beyond the learned label space in data streams," in *Proc. 31st Annu. ACM Symp. Appl. Comput. (SAC)*, New York, NY, USA, 2016, pp. 910–915.

[36] R. Klinkenberg and I. Renz, "Adaptive information filtering: Learning in the presence of concept drifts," in *Proc. Workshop Notes ICML/AAAI Workshop Learn. Text Categorization*. New Orleans, LA, USA: AAAI Press, 1998, pp. 33–40.

[37] Y. Yang, X. Wu, and X. Zhu, "Combining proactive and reactive predictions for data streams," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2005, pp. 710–715.

[38] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldá, and R. Morales-Bueno, "Early drift detection method," in *Proc. 4th Int. Workshop Knowl. Discovery Data Streams*, 2006, pp. 77–86.

[39] B. C. da Silva, E. W. Basso, A. L. Bazzan, and P. M. Engel, "Dealing with non-stationary environments using context detection," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 217–224.

[40] T. Susnjak, A. L. C. Barczak, and K. A. Hawick, "Adaptive cascade of boosted ensembles for face detection in concept drift," *Neural Comput. Appl.*, vol. 21, no. 4, pp. 671–682, 2012.

[41] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, Apr. 2012.

[42] M. Harel, S. Mannor, R. El-Yaniv, and K. Crammer, "Concept drift detection through resampling," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1009–1017.

[43] H. Raza, G. Prasad, and Y. Li, "Adaptive learning with covariate shift-detection for non-stationary environments," in *Proc. 14th UK Workshop Comput. Intell. (UKCI)*, Sep. 2014, pp. 1–8.

[44] N. Lu, G. Zhang, and J. Lu, "Concept drift detection via competence models," *Artif. Intell.*, vol. 209, pp. 11–28, Apr. 2014.

[45] H. Wang and Z. Abraham. (Apr. 2015). "Concept drift detection for imbalanced stream data." [Online]. Available: https://arxiv.org/abs/1504.01044

[46] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, and K. Ghédira, "Self-adaptive windowing approach for handling complex concept drift," *Cognit. Comput.*, vol. 7, no. 6, pp. 772–790, 2015.

[47] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting data streams with skewed distributions," in *Proc. SIAM Int. Conf. Data Mining*, 2007, pp. 3–14.

[48] A. Yalcin, Z. Erdem, and F. Gurgen, "Ensemble based incremental SVM classifiers for changing environments," in *Proc. 22nd Int. Symp. Comput. Inf. Sci. (ISCIS)*, Nov. 2007, pp. 1–5.

[49] S. Chen and H. He, "SERA: Selectively recursive approach towards non-stationary imbalanced stream data mining," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jun. 2009, pp. 522–529.

[50] R. Elwell and R. Polikar, "Incremental learning of concept drift in non-stationary environments," *IEEE Trans. Neural Netw.*, vol. 22, no. 10, pp. 1517–1531, Oct. 2011.

[51] G. Ditzler, R. Polikar, and N. Chawla, "An incremental learning algorithm for non-stationary environments and class imbalance," in *Proc. 20th Int. Conf. Pattern Recognit. (ICPR)*, Aug. 2010, pp. 2997–3000.

[52] G. Ditzler and R. Polikar, "An ensemble based incremental learning framework for concept drift and class imbalance," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2010, pp. 1–8.

[53] R. Klinkenberg, "Using labeled and unlabeled data to learn drifting concepts," in *Proc. Workshop Notes IJCAI Workshop Learn. Temporal Spatial Data*. New Orleans, LA, USA: AAAI Press, 2001, pp. 16–24.

[54] D. H. Widyantoro, "Exploiting unlabeled data in concept drift learning," *Jurnal Informatika*, vol. 8, no. 1, pp. 54–62, 2007.

[55] E. J. Spinosa, F. de Leon, A. Ponce, and J. Gama, "Cluster-based novel concept detection in data streams applied to intrusion detection in computer networks," in *Proc. ACM Symp. Appl. Comput.*, 2008, pp. 976–980.

[56] M. Masud, J. Gao, L. Khan, J. Han, and B. M. Thuraisingham, "Classification and novel class detection in concept-drifting data streams under time constraints," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 6, pp. 859–874, Jun. 2011.

[57] M. M. Masud *et al.*, "Detecting recurring and novel classes in concept-drifting data streams," in *Proc. IEEE 11th Int. Conf. Data Mining*, Dec. 2011, pp. 1176–1181.

[58] X. Wu, P. Li, and X. Hu, "Learning from concept drifting data streams with unlabeled data," *Neurocomputing*, vol. 92, pp. 145–155, Sep. 2012.

[59] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Incremental clustering for the classification of concept-drifting data streams," Tech. Rep., Dec. 2008.

[60] T. S. Sethi and M. Kantardzic, "On the reliable detection of concept drift from streaming unlabeled data," *Expert Syst. Appl.*, vol. 82, pp. 77–99, Oct. 2017.

[61] M. Tennant, F. Stahl, O. Rana, and J. B. Gomes, "Scalable real-time classification of data streams with concept drift," *Future Gener. Comput. Syst.*, vol. 75, pp. 187–199, Oct. 2017.

[62] S. Liu, L. Feng, J. Wu, G. Hou, and G. Han, "Concept drift detection for data stream learning based on angle optimized global embedding and principal component analysis in sensor networks," *Comput. Elect. Eng.*, vol. 58, pp. 327–336, Feb. 2017.

[63] J. de A. Silva, E. R. Hruschka, and J. Gama, "An evolutionary algorithm for clustering data streams with a variable number of clusters," *Expert Syst. Appl.*, vol. 67, pp. 228–238, Jan. 2017.

[64] S. Xu and J. Wang, "Dynamic extreme learning machine for data stream classification," *Neurocomputing*, vol. 238, pp. 433–449, May 2017.

[65] E. Arabmakki and M. Kantardzic, "SOM-based partial labeling of imbalanced data stream," *Neurocomputing*, vol. 262, pp. 120–133, Nov. 2017.

[66] Y. Zhang, G. Chu, P. Li, X. Hu, and X. Wu, "Three-layer concept drifting detection in text data streams," *Neurocomputing*, vol. 260, pp. 393–403, Oct. 2017.

[67] W. Fan, "Systematic data selection to mine concept-drifting data streams," in *Proc. 10th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2004, pp. 128–137.

[68] I. Katakis, G. Tsoumakas, and I. P. Vlahavas, "An ensemble of classifiers for coping with recurring contexts in data streams," in *Proc. 18th Eur. Conf. Artif. Intell. (ECAI)*. Amsterdam, The Netherlands: IOS Press, 2008, pp. 763–764.

[69] M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[70] M. Harries, "SPLICE-2 comparative evaluation: Electricity pricing," Univ. South Wales, Newport, U.K., Tech. Rep., 1999.

[71] T. M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski, "Experience with a learning personal assistant," *Commun. ACM*, vol. 37, no. 7, pp. 80–91, Jul. 1994.

ADRIANA SAYURI IWASHITA was born in São Paulo, Brazil. She received the bachelor's and master's degrees in computer science from São Paulo State University, São Paulo, in 2010 and 2013, respectively. She is currently pursuing the Ph.D. degree with the Federal University of São Carlos, São Paulo.

Her research interests include machine learning, pattern recognition, and concept drift detection.

JOÃO PAULO PAPA received the B.Sc. degree in information systems from São Paulo State University, São Paulo, Brazil, the M.Sc. degree in computer science from the Federal University of São Carlos, São Carlos, in 2005, and the Ph.D. degree in computer science from the University of Campinas, Campinas, Brazil, in 2008.

From 2008 2009, he was a Post-Doctoral Researcher with the University of Campinas, and from 2014 to 2015, he was a Visiting Scholar with the Center for Brain Sciences, Harvard University.

He has been a Professor with the Computer Science Department, São Paulo State University, since 2009. His research interests include machine learning, pattern recognition, and image processing. He was elected as a Research Fellow by the Alexander von Humboldt Foundation.